Counterfactual Regret Minimization Algorithms in Imperfect Information Games

Student Name: Yosef Berezovskiy
Supervisor Name: Dr Trehan Amitabh
Submitted as part of the degree of BSc Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—This paper presents a comparative analysis of various Counterfactual Regret Minimization (CFR) algorithms in imperfect information games. The study focuses on the original CFR algorithm by Zinkevich and its modifications: Discounted CFR, CFR+, and Outcome Sampling Monte Carlo CFR, evaluating their performance in Leduc Hold'em and a newly introduced variant, Short Deck Big Leduc Hold'em. Exploitability is the primary metric to assess algorithm performance over 1000 iterations. The results confirm that Discounted CFR with alternating updates consistently outperforms other variants, and reveals that the benefits of quadratic weighted averaging in CFR+ diminish in more complex games. Furthermore, the research examines the impact of either applying alternating updates and different averaging schemes simultaneously or applying them independently on convergence rates. The results of the experimental setup of the study also highlight the limitations in comparing sampling-based methods with deterministic algorithms using iteration count alone. The research contributes to the field by offering empirical evidence of algorithm performance in scenarios of different complexity and identifying potential areas for future investigation in game-solving algorithms.

Index Terms—Games and infotainment; Applications and Expert Systems; Problem Solving, Control Methods, and Search; Graph and tree search strategies

1 Introduction

From the very beginning of computer science, games and AI have been deeply intertwined. As games continued to serve as testbeds, remarkable progress in game-playing AI was achieved. Deep Blue's triumph over Kasparov in chess [1] and Chinook's victory in checkers [2] stand as testaments to this advancement.

However, surpassing even the most skilled human players does not necessarily equate to solving a game in the theoretical sense [3]. Game theory, a field dedicated to studying strategic decision-making in interactive scenarios, provides a framework for understanding game solutions. This field is of significant importance, extending beyond recreational games to domains like economics, politics, and cybersecurity. The lack of comprehensive studies in this area has been cited as a contributing factor to financial crises, as argued by Petrosian and Zenkevich [4].

A central concept in game-theoretic solutions is the Nash equilibrium (NE), a state where no player can improve their outcome by unilaterally changing their strategy, assuming all other players' strategies remain the same. In this context, "game" denotes a structured interaction among agents with individual interests and is defined to be an extensive-form game if the interaction happens sequentially. Achieving, or approximating NE is considered as solving a game. While perfect-information games, where all players have complete knowledge of the game's state, have seen numerous game-theoretic solutions [5], [6], imperfect-information games (IIGs), characterised by hidden information, have presented far greater challenges [3].

This difficulty in solving IIGs underscores their research value. John von Neumann, a founding figure in game theory suggested that IIGs, with their inherent uncertainties and hidden information, offer a more realistic reflection of real-world decision-making than perfect-information games [7].

Poker is an extensive-form IIG exemplifying the complexities above with bluffing, betting, and hidden cards. The most popular variant of the game is Texas Hold'em; a variant of great complexity with its simplest competitive form played with just two players (heads-up) and with fixed bet sizes and a fixed number of raises (limit), Heads-up Limit Hold'em (HULHE) lacking solution until as late as 2015 [3]. This is due to the imperfect information nature of the game which makes it computationally challenging for computers to play due to the unseen events, such as cards dealt to the opposing players. This lack of visibility means that out of the $3.16 \cdot 10^{17}$ potential game states in HULHE, many are indistinguishable from the player making a decision. As a result, the game effectively has about $3.19 \cdot 10^{14}$ information sets - decision points where a player must act [3], [8]. Hence, solving a real poker game even in its simplest forms poses significant computational challenges - even highly optimised algorithms take 900 core years of computation to provide a solution for the game [8]. Consequently, simpler synthetic variants like Leduc hold'em (LH) [9] and a proposed variant of Big Leduc hold'em (SDBLH) [10] will be employed in this project to act as testbeds for developing and evaluating algorithms designed to tackle IIGs. LH, with its minimal complexity and as little as 288 information sets [3], serves as a proving ground for algorithm effectiveness in research [9], while proposed SDBLH has 31944 information sets and is aimed to assess scalability to more complicated problems. The proposed design of SDBLH aims to maintain the original aspects of the game while making the computational demands manageable within the project's scope by

reducing the deck size to 12 cards as the primary focus of the project is on comparing the relative performance and behaviour of algorithms rather than pushing or confirming the boundaries of game size solvability. Furthermore, by keeping the game size manageable, the project ensures that results can be reproduced and verified by other researchers without requiring access to high-performance computing clusters.

The development of algorithms for solving IIGs has been heavily influenced by the concept of regret matching [11]. Regret matching centres around adjusting future actions based on the regret of not having chosen seemingly better actions in the past. This approach has proven to be successful in facilitating convergence to Nash equilibrium in IIGs and by its widespread adaptation in top-performing IIG-playing algorithms, many of which have achieved impressive results in competitions like the Annual Computer Poker Competition (ACPC) [12].

Counterfactual regret minimization (CFR) is the most popular algorithm for solving two-player zero-sum games in extensive form with imperfect information, achieving state-of-the-art results by extending the regret matching approach [13]. In the context of poker, the structure of the game can be modelled as a tree. This tree details potential actions by players or outcomes determined by chance, such as cards drawn from the deck. The leaves of this tree mark the end of the game, with each leaf assigning a specific utility to each player. When the game involves exactly two players and the sum of utilities at every leaf equals zero, the game qualifies as a zero-sum game [3]. Consequently, CFR applies to variations of poker satisfying two-player zero-sum conditions, solving some of them successfully.

Several modifications of the CFR algorithm exist [14], [15], [16], [17], each aiming to improve its convergence speed and overall efficiency. The research aims to answer how these modifications differ in terms of convergence speed and to find out their relative performances when implemented and analysed comparatively. To address this question, the research evaluates the exploitability of each algorithm in both LH and SDBLH. Exploitability, a measure of how much a strategy can be exploited by a theoretically optimal opponent [13], serves as a proxy for the quality of the approximate solution obtained by each algorithm. By comparing the exploitability of the strategies produced by each algorithm over a range of iterations, we gain insights into their relative convergence speeds.

The goal of this research is to add value to the field by looking into and confirming the relative strengths and weaknesses of each algorithmic variant when applied to extensive-form zero-sum IIGs. In addition, by understanding the performance characteristics of these algorithms in simplified poker variants, LH, and how well the algorithms scale when applied to a more complex form of the game, SDBLH, additional value is extracted from evaluating how algorithms could potentially perform in even more complex environments.

Consequently, this paper presents a comprehensive analysis of various CFR algorithms in imperfect information games. The project successfully implements and performs experiments to compare the performance of the original CFR [13], Outcome Sampling Monte Carlo CFR (OS-MCCFR) [14], CFR+ [15] and Discounted CFR (DCFR) [16] in two poker variants: Leduc Hold'em and a newly introduced variant, Short Deck Big Leduc Hold'em.

2 Related Work

This section provides an overview of key developments in solving IIGs, focusing on the evolution of solution concepts from early approaches to state-of-the-art techniques.

2.1 Early Forms of Problems & Solutions

The challenge of solving imperfect information games (IIGs) has been a persistent focus in game theory research. Initial attempts to tackle extensive-form IIGs often involved transforming them into a simplified normal-form representation and solving using linear programming [18]. While this approach was theoretically sound, it struggled with scalability as game complexity increased. The exponential growth in the number of possible strategies made this method computationally infeasible for all but the smallest games like Kuhn Poker [3], [9].

Romanovskii [19] introduced sequence-form linear programming (SFLP) to utilise the sequential aspect of extensive-form games, representing strategies by sequences of actions instead of pure strategies, effectively avoiding the exponential growth seen in normal-form representations. This allowed for solving significantly larger games like Rhode Island Hold'em [20], [21].

To study imperfect information games (IIGs) without the complexities of full-scale poker, researchers developed simpler versions like Leduc Hold'em (or similarly referred to as Leduc poker) [10] and Big Leduc Hold'em [22], with both utilised in this project and the latter being simplified further following the methods described in [23].

SFLP was groundbreaking as the first algorithm to solve extensive-form IIGs with computation times that scale polynomially with the size of the game representation [3]. Billings et al. [24] showcased SFLP's efficacy in poker by solving simpler yet full-scale heads-up limit hold'em games and developing competitive poker agents.

2.2 Counterfactual regret minimization

The launch of the ACPC in 2006 [12] prompted quick advancements in algorithms for handling extensive form zero-sum IIGs. CFR [13] soon became prominent and underpinned many of the top-performing agents in the ACPC.

In his work [13] Zinkevich et al. proved that minimising counterfactual regret (what the payoff would have been had different actions been taken) in each information set is sufficient to minimise overall regret for the entire game. This allows CFR to decompose the complex task of finding an equilibrium in an extensive-form game into many smaller, manageable subproblems. The paper [13] finds CFR's theoretical bound on the number of iterations needed to reach an ϵ-equilibrium to be $O(1/\epsilon^2)$, where ϵ represents the solution's exploitability. A Nash equilibrium is an ϵ-equilibrium with $\epsilon = 0$. While this bound is not as tight as some alternative, primarily derivative-based, methods discussed further in the section, CFR's practical performance as suggested by [13] significantly outpaces its theoretical guarantees.

This project heavily relies on the theory presented in [32] for exploring and implementing the CFR algorithms. The paper provides a detailed introduction to the field of Counterfactual Regret Minimization, presenting detailed explanations of the core concepts and mathematical foundations underlying the CFR algorithm, including the formulation of counterfactual regret, the regret matching procedure, and the convergence properties of CFR. Hence this work by Neller and Lancot served as a solid foundation for understanding and implementing the various CFR variants explored in this study.

The majority of works solving large IIGs such as poker [20], [21], [24] have employed different levels of abstraction and the most popular approaches are discussed in detail in the work by Billings [23]. Common practice is to simplify the extensive game's state space by combining similar game states or decisions. However, this method may limit the capture of complex strategic detail, since abstraction by its nature involves balancing computational efficiency with strategic accuracy [23]. Abstraction is not a fundamental part of CFR algorithms but rather an additional strategy to manage the computational demands of large state spaces as [23] confirms. Consequently, this paper intentionally avoids a detailed discussion on abstractions, focusing on assessing the performance of CFR algorithms in solving problems in their pure form. This approach seeks to solely evaluate the inherent effectiveness of CFR, without the possible biases introduced by abstraction techniques.

2.3 CFR Modifications and Advancements

CFR's success has spawned numerous extensions and variations aimed at enhancing its convergence speed and reducing hardware requirements. One significant development was Monte Carlo CFR (MCCFR) [14], which introduced the concept of sampling to the CFR framework. While MCCFR increases variance in the algorithm's updates, it significantly reduces the computational burden as [14] proves. Although it requires a higher number of iterations to converge, the cost per iteration is smaller, potentially allowing for computationally faster convergence overall. MCCFR's ability to handle large decision spaces without exhaustive computation makes it particularly well-suited for complex games like large-scale poker. In these environments, the trade-off of increased variance and more iterations for reduced computational load per iteration tends to be highly beneficial as suggested by [14].

Empirical results [14] comparing External Sampling MCCFR (ES-MCCFR) and Outcome Sampling (OS-MCCFR) have shown that ES-MCCFR outperforms the variant utilising different sampling techniques on the majority of the testbeds used in the research, however, OS-MCCFR has shown better convergence in the poker variant testbed. Consequently, since this project focuses on comparing algorithms for solving synthetic poker IIGs, OS-MCCFR is the focus of this project.

As M. Lancot proceeded with the work on sampling techniques in conjunction with CFR he introduced the notion of alternating player updates form of Outcome Sampling in his work [25], showing that the technique improves convergence in practice when compared to OS-MCCFR variant with simultaneous updates. This project utilises this technique in the implementation of OS-MCCFR.

Building on these ideas, the development of the CFR+ algorithm by Oskari Tammelin [15] represented a significant advancement in the domain. CFR+ incorporated alternating updates among other improvements, an idea first applied in the context of MCCFR. The key innovation was the introduction of "regret matching +", a variant of the regret-matching strategy update rule that ensures non-negative regrets, significantly impacting convergence by preventing the algorithm from becoming trapped in local optima. Furthermore, the convergence of CFR+ is enhanced by the weighted averaging technique applied during the computation of the average strategy - the solution of the CFR algorithm. In the context of CFR+ as per [15] and [16] this technique linearly increases the importance of the most recent iterations during the computation of the average strategy. In many instances, CFR+ operates at least an order of magnitude faster than the original CFR (referred to as Vanilla CFR throughout this project), providing a more efficient computational framework. The application of CFR+ allowed it to produce weak solutions to complex variants of poker, including Heads-Up Limit Texas Hold'em [3]. Theoretically, CFR+ maintains the same $O(1/\epsilon^2)$ bound as original CFR, but in practice, it often converges much faster, sometimes even faster than $O(1/\epsilon)$ in many games [16].

It's worth noting that while the original CFR+ paper asserts the necessity of a vector-form implementation, this project demonstrates that CFR+ can be effectively implemented without vector-form optimizations. The vector form is a common improvement but is not essential for CFR+ implementation.

Brown and Sandholm [16] introduced Discounted CFR (DCFR), which assigns greater weight to recent iterations when updating regrets and strategies, effectively discounting the impact of past iterations. By carefully calibrating the discount factors, DCFR algorithms can achieve markedly faster convergence than both Vanilla CFR and CFR+, particularly in games with extended horizons or complex dynamics. However, the tests and comparisons in the DCFR paper are based on subgames of Heads-Up No-Limit Hold'em (HUNL), potentially overlooking certain game dynamics. Our research aims to provide a more consistent benchmark by evaluating performance in simpler yet full games.

The DCFR paper similarly to the aforementioned papers posits that alternating updates in CFR yield superior performance in practice. However, explicit comparisons between non-alternating variants of the algorithm have been lacking. Our research addresses this gap by providing these comparisons.

DCFR similarly retains the same as that of standard CFR bound, which is $O(1/\epsilon^2)$ to reach an ϵ-equilibrium. However, the constant factors in this bound are different as further explained in the original paper [16], leading to significant practical differences in performance, often converging faster than CFR and CFR+.

Because DCFR is an extension to CFR+ with more advanced discounting techniques Brown and Sandholm in [16] show that an instance of DCFR with specific parameters is equivalent to CFR+. Furthermore, they claim that they have observed a faster convergence of CFR+ with a quadratic

weighting averaging rather than with linear as in the original CFR+ implementation, however, no empirical evidence of this was shown.

Farina et al. [17] introduced Predictive Counterfactual Regret Minimization Plus (PCFR+), which combines the strengths of CFR+ with predictive techniques to achieve faster convergence. PCFR+ builds upon the idea of using predictions of future utilities to compute strategy updates. The algorithm incorporates a prediction step where, at each iteration, it estimates the next counterfactual value based on the observed values from previous iterations. A key innovation in PCFR+ is its use of stable-predictivity, a concept that ensures the algorithm's regret bound depends on the quality of its predictions. This approach allows PCFR+ to potentially converge faster than CFR+ when good predictions are available, while still maintaining the worst-case guarantees of CFR+ when predictions are poor.

PCFR+ modifies the regret update rule of CFR+ to incorporate these predictions. Specifically, it uses a predictive variant of regret matching+ that adjusts the cumulative regrets based on both the observed and predicted counterfactual values. This modification allows the algorithm to more quickly adapt to changing game dynamics and potentially discover optimal strategies faster.

Empirical results presented by Farina et al. [17] demonstrate that PCFR+ consistently outperforms CFR+ across a wide range of imperfect information games, including variants of LH and Goofspiel [26]. In some instances, PCFR+ achieved convergence rates orders of magnitude faster than CFR+, particularly in games where good predictions could be made.

While PCFR+ represents a significant advancement in CFR algorithms, the theoretical foundations presented in [17] behind its predictive capabilities are highly complex. Due to this complexity and the specialized knowledge required to fully implement and analyze PCFR+, this algorithm will not be explored in the current project. The focus will remain on the aforementioned CFR variants whose theoretical underpinnings can be reproduced accurately in the technical implementation and hence provide greater value during comparisons.

2.4 Further Modifications, Alternative Approaches, Hybrid Methods

In addition to the major variants of CFR discussed earlier, researchers have proposed several modifications and optimizations to improve the computational efficiency and convergence speed of CFR-based algorithms.

Regret-based pruning (RBP), introduced by Brown and Sandholm [27], is an optimization technique that allows CFR to avoid traversing paths in the game tree if either player takes actions leading to that path with zero probability. The number of CFR iterations during which a sequence can be skipped depends on how poorly the sequence has performed in previous iterations. Empirical results in [27] show significant improvements in computational efficiency, especially in games where many actions are suboptimal but not immediately apparent as such, such as poker and its variations.

Hyperparameter Schedules (HS), introduced by Zhang et al. [22], propose a simple method to dynamically adjust the hyperparameters of CFR variants across iterations. By using schedules with large initial values for certain parameters and adjusting across iterations, these methods have shown improvements in convergence speed, improving the performance of DCFR and CFR+ by multiple orders of magnitude.

From the experimental results, it is apparent that RBP, HS and other less notable optimisations show great potential for improving the performance of CFR-based algorithms and future work may constitute further in-depth analysis of algorithms employing these optimizations. However, the current project focuses on comparing fundamental, distinct algorithms to establish a baseline understanding of their relative performances without the distorting effects of any supplementary enhancements that could potentially introduce biases.

While CFR and its variants have dominated recent advancements in solving zero-sum extensive form IIGs, alternative approaches and hybrid methods continue to be explored.

One such alternative is the excessive gap technique (EGT) [28], adapted for extensive-form games by Gilpin et al. and others [29], [30]. EGT exhibits favourable asymptotic time complexity compared to CFR, $O(1/\epsilon)$, but its practical performance has often lagged, particularly in games with imperfect recall, where players may not have complete information about their past actions.

Another notable line of research involves combining elements of CFR with other techniques, such as deep learning, to leverage the strengths of each approach. These hybrid methods aim to achieve faster convergence or handle specific game structures more efficiently, pushing the boundaries of what is computationally feasible in the field of IIG solving.

Deep Counterfactual Regret Minimization (Deep CFR) [31] represents a step towards addressing the limitations of abstraction-based approaches. By utilising the strengths of the deep neural networks to generalise across information states, Deep CFR can retain strategic detail that might be lost by using traditional abstraction techniques. While not the focus of this paper, Deep CFR stands as a promising direction for future research in solving large-scale IIGs without relying on explicit abstractions.

This section of the project testaments remarkable progress in solving IIGs, from early normal-form representations to CFR variants and hybrid approaches. Further work in this research aims to contribute to this ongoing evolution of the field by analysing and comparing CFR algorithms' performance in conventional and proposed benchmark games.

3 Methodology

This section presents the methodological approach employed in the project. It begins by introducing the theoretical framework underpinning CFR and its variants, followed by the technical implementation of the discussed theory. The complexity of the underlying concepts necessitates a foundational understanding to effectively present the subsequent practical work.

3.1 Theoretical Framework

3.1.1 Notions and definitions

A normal form game is defined as a triplet (N, A, u), where $N = \{1, \ldots, n\}$ represents a finite set of n players, S_i denotes the finite set of actions available to player i, $A = S_1 \times \cdots \times S_n$ represents the set of all possible combinations of simultaneous actions by all players (each combination is referred to as an action profile), u is a function assigning a utility vector to each player for every action profile [32].

A sequential game, such as poker, consists of a sequence of actions.

Extensive-form games naturally model sequential decision-making. An extensive form game specifies possible moves, available information, and payoff sizes at game termination. Information sets contain information about the active player and all available game state information the player possesses.

A zero-sum game is one in which the sum of the utilities for each player across all outcomes is zero. For two players, this can be expressed as $u_1 = -u_2$ [32].

The project focuses on imperfect information zero-sum games in extensive form and represents them by a tuple $(N, H, P, \sigma_c, u, I)$, where N is the set of players, H is the set of histories (sequences of actions), $P : H \to N \cup \{c\}$ is the player function (with c denoting chance), σ_c is the chance probability function, $u : Z \to \mathbb{R}^{|N|}$ is the utility function for terminal histories $Z \subset H$, and $I = (I_1, \ldots, I_{|N|})$ is the information partition for each player. While the tuple defines the overall game, a game state is a snapshot of the game at any given moment - it is a specific instance of the game as defined by the tuple. For a given game state, the set of legal actions, a set of actions the player is allowed to make at the current game iteration, is denoted as $A(I)$, where I is the current information set [32].

A player employs a pure strategy if they choose a particular action with probability 1 [32].

A player has a mixed strategy if they select between two or more actions, each with a positive probability. We denote a mixed strategy by σ, where $\sigma_i(s)$ is the probability that player i chooses action $s \in S$ [32].

To compute the expected utility for a player, the following equation is used:

$$u_i(\sigma_i, \sigma_{-i}) = \sum_{s \in S_i} \sum_{s' \in S_{-i}} \sigma_i(s)\sigma_{-i}(s')u_i(s, s').$$

The best response strategy for player i is to maximize their expected utility, knowing the strategies of the other players. When each player plays their best response strategy, the set of strategies forms a Nash equilibrium, where no player can unilaterally improve their payoff by changing strategies alone [32]. Hence, the CFR algorithm wants to find or approximate the best response strategy.

A strategy profile is a set of strategies for all players. It fully specifies all actions in a game. A strategy profile must include one and only one strategy for every player [32].

Regret is the difference between the utility of an action not chosen by the player and the utility of the action that was chosen [32].

3.1.2 Counterfactual Regret Minimization

This subsection introduces the original Counterfactual Regret Minimization (CFR) algorithm, often referred to as Vanilla CFR [15], [16], initially developed by Zinkevich. Vanilla CFR lays the groundwork for various modifications that will be discussed in later sections. Introducing Vanilla CFR first provides a clear reference point, as its core principles are preserved in subsequent adaptations, facilitating a better understanding of their comparative analysis. The formulation and notation used in [32] serve as the foundation for describing the algorithm's key components in this project.

Vanilla CFR operates iteratively, with t representing the current time step or iteration, and T denoting the total number of iterations. In the implementation, t is specific to each information set and increments with each visit to that set.

In Vanilla CFR, a strategy σ_i^t for player i maps each player i information set I_i and legal player i action $a \in A(I_i)$ to the probability $\sigma_i^t(I_i, a)$ that the player will choose a in I_i at time t. All player strategies together at time t form a strategy profile σ^t, while σ_{-i}^t refers to the strategy profile excluding player i's strategy. Let $\sigma_{I \to a}^t$ denote a profile equivalent to σ^t, except that action a is always chosen at information set I.

The progression of a game is represented by its history $h \in H$, which is a sequence of actions from the game's start to a particular point. The reach probability of history h under strategy profile σ is defined by $\pi^\sigma(h)$. The probability of reaching information set I through any possible history is denoted as $\pi^\sigma(I)$ and is defined as:

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

Whereas $\pi_{-i}^\sigma(I)$ is the probability of reaching I with strategy profile σ except that, treating current player i actions to reach the state as having probability 1.

The expected value for player i given history h under strategy profile σ is denoted as $v_i^\sigma(h)$. For an information set I, $v_i^\sigma(I)$ is defined as the expected value for player i, considering all histories $h \in I$.

The counterfactual value $v_i^\sigma(I)$ is hence defined as:

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

where z ranges over all terminal histories reachable from h, and $u_i(z)$ is the utility for player i at terminal history z.

The immediate counterfactual regret for not taking action a at information set I in iteration t is defined as:

$$r_i^t(I, a) = v_i^{\sigma_{I \to a}^t}(I) - v_i^{\sigma^t}(I) \tag{1}$$

where $\sigma_{I \to a}^t$ is the strategy profile identical to σ^t except that player i always chooses action a at information set I.

The cumulative counterfactual regret $R_i^T(I, a)$ after T iterations is the sum of immediate regrets:

$$R_i^T(I, a) = \sum_{t=1}^{T} r_i^t(I, a) \tag{2}$$

If cumulative counterfactual regret is defined as $R_i^{T,+}(I, a) = \max(R_i^T(I, a), 0)$, then, by applying the

regret-matching technique to accumulated regret, a new strategy can be obtained:

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a' \in A(I)} R_i^{T,+}(I,a')} & \text{if } \sum_{a' \in A(I)} R_i^{T,+}(I,a') > 0, \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \tag{3}$$

For each information set, the equation uses the probability of actions proportionally to the accumulated regrets. For each action, CFR considers the next stage of the game and recursively calculates the utility of each action. The solution concept in CFR, particularly for approaching a Nash equilibrium in two-player zero-sum games, relies on computing the average strategy over all iterations. This average strategy $\overline{\sigma}_i^T$ is defined as:

$$\overline{\sigma}_i^T(I,a) = \frac{\sum_{t=1}^T \pi_i^{\sigma^t}(I) \cdot \sigma_i^t(I,a)}{\sum_{t=1}^T \pi_i^{\sigma^t}(I)} \tag{4}$$

By definition, as $T \to \infty$, $\overline{\sigma}_T$ converges to a Nash equilibrium.

Equations presented serve as foundational elements common across all CFR variants, and specific adaptations in these equations for different CFR versions will be detailed in the subsequent sections.

Further subsections will detail the technical framework of three Counterfactual Regret Minimization variants in the project: CFR+, DCFR and OS-MCCFR. Each variant builds upon the core CFR algorithm, introducing specific modifications to enhance performance or address particular challenges in different game scenarios.

3.1.3 CFR+

In CFR+, negative regrets are treated as zero during strategy computation - after each iteration, any action with negative regret is set to have a regret of zero [15]. This modification is changing the cumulative regret update:

$$R_i^{+,T}(I,a) = \begin{cases} \max\left\{r_i^T(I,a), 0\right\} & \text{if } T = 1, \\ \max\left\{R_i^{+,T-1}(I,a) + r_i^T(I,a), 0\right\} & \text{if } T > 1. \end{cases} \tag{5}$$

This approach, known as Regret Matching+ [15], prevents actions with negative cumulative regret from being ignored for too long. In Vanilla CFR, actions with large negative regret might take many iterations to become positive again (Equation 2), potentially missing out on good strategies. By resetting negative regrets to zero, CFR+ allows these actions to be reconsidered more quickly if they start performing well.

The strategy update remains similar to Vanilla CFR (Equation 3) but uses the non-negative regrets from Equation 5.

In addition, CFR+ introduces a linear weighting scheme for the average strategy computation [16]:

$$\overline{\sigma}_i^T(I,a) = \frac{\sum_{t=1}^T t \cdot \pi_i^{\sigma^t}(I) \cdot \sigma_i^t(I,a)}{\sum_{t=1}^T t \cdot \pi_i^{\sigma^t}(I)}$$

As the equation shows, this linear weighting gives more importance to later iterations, which tend to have better strategies. As mentioned earlier in the Related Work section, this modification helps to speed up convergence by reducing the impact of early, potentially poor strategies.

Similarly, as was mentioned earlier, like other CFR variants discussed further, CFR+ employs alternating updates.

The alternating update process works as follows:

- In odd-numbered iterations, only Player 1's strategy is updated.
- In even-numbered iterations, only Player 2's strategy is updated.

This technique allows each player to immediately utilize the most recent information from the opponent's update and improves convergence in practice.

3.1.4 Discounted CFR

DCFR extends the Vanilla CFR and CFR+ by introducing discounting factors for regrets and strategy updates. It uses three parameters: α, β, and γ, which control the discounting of positive regrets, negative regrets, and average strategy updates, respectively [16]. This is done by multiplying positive cumulative counterfactual regrets by coefficient $c_t^\alpha = \frac{t^\alpha}{t^\alpha+1}$, negative cumulative counterfactual regrets by $c_t^\beta = \frac{t^\beta}{t^\beta+1}$ and contributions to the average strategy by $c_t^\gamma = \frac{t^\gamma}{t^\gamma+1}$ on every iteration t.

DCFR modifies the cumulative regret update as follows:

$$R_i^T(I,a) = \begin{cases} r_i^T(I,a) & \text{if } T = 1, \\ R_i^{T-1}(I,a) \cdot c_T^\alpha + r_i^T(I,a), & \text{if } R_i^{T-1}(I,a) > 0 \\ R_i^{T-1}(I,a) \cdot c_T^\beta + r_i^T(I,a), & \text{otherwise} \end{cases} \tag{6}$$

This modification allows DCFR to discount the impact of past regrets differently for positive and negative regrets. By setting $\alpha > \beta$, DCFR can 'forget' negative regrets faster than positive ones, allowing the algorithm to adapt more quickly to changing game dynamics, or to consider only positive regrets, treating negative regrets as zero, consequentially reproducing regret-matching+.

DCFR modifies average strategy computation as follows:

$$\overline{\sigma}_i^T(I,a) = \frac{\sum_{t=1}^T c_t^\gamma \cdot \pi_i^{\sigma^t}(I) \cdot \sigma_i^t(I,a)}{\sum_{t=1}^T c_t^\gamma \cdot \pi_i^{\sigma^t}(I)}$$

This change allows DCFR to discount the contribution of earlier iterations to the average strategy. By setting $\gamma > 0$, DCFR gives more weight to recent iterations, potentially speeding up convergence.

DCFR similarly to CFR+ still uses the same core principle of minimizing counterfactual regret: the immediate regret calculation (Equation 1) remains unchanged and the strategy computation from regrets (Equation 3) is the same.

From the definition of DCFR, it is observable that an instance of DCFR with parameters $\alpha = \infty$, $\beta = -\infty$, $\gamma = 1$ is equivalent to CFR+.

3.1.5 Outcome-Sampling MCCFR

In OS-MCCFR, instead of traversing the entire game tree, the algorithm samples a single path from the root to a terminal node in each iteration. This sampling is performed according to a sampling policy π, which is typically a

mixture of the current strategy σ and a uniform random policy [14], [25]. Formally, for each information set I and action a, the sampling policy is defined as:

$$q(I,a) = (1-\mu)\sigma_i(I,a) + \frac{\mu}{|A(I)|}$$

where μ is an exploration parameter, typically set to 0.6 [25], and $|A(I)|$ is the number of legal actions in information set I.

For a sampled path z, the sampled counterfactual value for player i at information set I is defined as:

$$\tilde{v}_i^{\sigma}(I,z) = \frac{\pi_{-i}^{\sigma}(z[I])}{q(z[I])} \cdot u_i(z)$$

where z[I] is the prefix of z from the root to information set I, $\pi_{-i}^{\sigma}(z[I])$ is the probability of opponents reaching z[I] under strategy σ, q(z[I]) is the sampling probability of reaching z[I], and $u_i(z)$ is player i's utility at terminal history z.

The sampled immediate counterfactual regret for not taking action a at the information set I in iteration t is then calculated as:

$$\tilde{r}_t(I,a) = \tilde{v}_i^{\sigma_{I\to a}^t}(I,z) - \tilde{v}_i^{\sigma^t}(I,z) \tag{7}$$

where $I \cdot a$ denotes the information set reached after taking action a from I.

The cumulative counterfactual regrets are updated using these sampled immediate regrets:

$$R_T(I,a) = \sum_{t=1}^{T} \tilde{r}_t(I,a)$$

The strategy update remains similar to Vanilla CFR but uses the updated method for calculating counterfactual regrets from Equation 7.

OS-MCCFR also employs an important sampling technique to update the average strategy [14]. The contribution to the average strategy at iteration t is weighted by the ratio of the probability of reaching the information set under the current strategy to the probability under the sampling policy:

$$w_t(I,a) = \frac{\pi_i^{\sigma_t}(I)}{q(I,a)}$$

The average strategy is then updated as:

$$\overline{\sigma}_T(I,a) = \frac{\sum_{t=1}^{T} w_t(I,a)\sigma_i^t(I,a)}{\sum_{t=1}^{T} w_t(I)}$$

This ensures that the average strategy converges to the same limit as in Vanilla CFR, despite the sampling approach.

The alternating update scheme, as employed in CFR+ and DCFR, is also utilised for OS-MCCFR. In the context of OS-MCCFR updating only one player's strategy in each iteration allows for more efficient use of the sampled information as empirical results show in [25].

3.2 Technical Framework

3.2.1 Solution Design Overview

The theoretical framework presented earlier demonstrates that the various modifications of CFR (CFR+, DCFR, and OS-MCCFR) share significant structural similarities with Vanilla CFR, particularly in methods for computing new strategies and updating regrets. To leverage these commonalities and ensure consistency across implementations, this project implements a hierarchical structure using inheritance principles from object-oriented programming.

Below, the implementation structural design of the regret matching techniques and algorithms is described.

Regret matching techniques:

- Abstract regret matching - This is the base class for all regret matchings. It defines the interface for core regret-matching operations:
- Vanilla CFR regret matching - Inherits from abstract regret matching and implements the standard regret matching algorithm used in Vanilla CFR. Accumulates regrets as per Equation 2 in the theoretical framework. Implements regret matching as per Equation 3. Computes the average strategy as per Equation 4.
- DCFR regret matching - Inherits from Vanilla CFR regret matching and implements the discounted regret matching used in DCFR. Overrides to implement Equation 6, applying discount factors to positive and negative regrets.

Algorithms:

- Abstract CFR - This class implements the basic structure of CFR algorithms, including tree traversal and the general flow of the algorithm. It provides abstract methods for specific operations that vary between CFR variants.
- Vanilla CFR - This class inherits from the abstract CFR class and implements the standard CFR algorithm, serving as a foundation for other variants. It provides concrete implementations for regret updates and strategy computation as described in the theoretical framework. Uses Vanilla CFR regret matching.
- DCFR - This class inherits from the Vanilla CFR class and introduces discounting factors for regrets and strategies. It overrides the regret update and strategy averaging methods, implementing the discounting logic described in the theoretical framework. Utilises DCFR regrets matching.
- CFR+ - Inherits from DCFR class, as DCFR with parameters $\alpha = \infty, \beta = -\infty, \gamma = 1$ is equivalent to CFR+.
- OS-MCCFR - While OS-MCCFR shares the basic CFR structure, its sampling approach requires changes to the tree traversal method. Therefore, it directly inherits from the Abstract CFR class and implements its own traversal logic. Instead of having a distinct regret matching class in the hierarchy, OS-MCCFR implements its sampling approach at the algorithm level. This means that the sampling logic is handled in the traversal and update methods of the OS-MCCFR class itself, rather than in the regret matching, so Vanilla CFR regret matching is utilised.

This hierarchical structure allows us to encapsulate common functionality in base classes and override specific methods in derived classes as needed. For instance, all variants share the same high-level iteration structure but differ in how they update regrets and compute strategies.

3.2.2 Solution Implementation Overview

By presenting the solution as a series of high-level steps, the project wants to avoid the complexity of the source code and maintain an appropriate level of abstraction for the implementation independent reproduction of the approach.

The Vanilla CFR and its modifications (CFR+, DCFR, OS-MCCFR) are implemented to operate through a structured iterative process. The overview of the functional implementation of this core procedure is presented below:

1. Initialization

- Create data structures indexed by information sets to store regrets and average strategy profiles for each player.
- Set the initial strategy profile to a uniform value for each information set.

2. Iterative Full Tree Traversal

- For a predetermined number of iterations, repeatedly traverse the game tree:
  - Begin at the root node (initial game state) and recursively explore all possible actions at each decision point.
  - At each node, retrieve the current strategy profile for the acting player from the regret-matching instance associated with the corresponding information set.
  - Continue recursive exploration until a terminal state (end of the game) is reached.
  - Once a terminal state is encountered, access the utilities for each player at that endpoint.
  - Backpropagate these utilities up the recursion tree to calculate counterfactual values for each action at each decision point.
  - At each information set encountered during the backpropagation, perform the following:
    - **Regret Update:** - Calculate the regret for each action based on the differences between the counterfactual value of that action and the expected value of playing according to the current strategy at that information set. Update the regrets stored using the specific regret update rule for the chosen CFR variant (Vanilla CFR, CFR+, DCFR).
    - **Average Strategy Update:** - Compute a new recommended strategy. Incorporate into the average strategy profile being maintained for the current iteration.

3. Convergence and Output

- After the specified number of iterations, the algorithm outputs the average strategy profile. This profile represents an approximation of a Nash equilibrium.

The aforementioned summary assumes full tree traversal. However, OS-MCCFR, in contrast to a full tree traversal, samples a single trajectory (path from root to leaf) in each iteration. Implementation of OS-MCCFR tree traversal is summarised below:

- **Path Sampling** - Instead of exploring all actions at a decision point, sample a single action based on a sampling policy:
  - **Counterfactual Value Estimation** - Once a terminal state is reached by following the sampled trajectory, estimate the counterfactual values for actions along the sampled path.
  - **Importance-Weighted Updates:**
    - **Regret Update** - Similarly to the full tree traversal, calculate regrets based on estimated counterfactual values. Weight these regret updates by the inverse probability of the sampled path.
    - **Average Strategy Update** - update the average strategy using importance sampling weights.

As with the full tree traversal, repeat path sampling for a predetermined number of iterations.

3.2.3 Implementation Environment

The project was implemented using Python 3.11.5, a high-level programming language. Python was chosen to allow for rapid prototyping and iterative development, which the project necessitated for exploring and comparing different CFR variants. The available range of libraries has positively contributed to the solution prototyping as well.

OpenSpiel [33], an open-source framework available in Python (as pyspiel library) was used for game representation and basic utility functions. It provided an interface to select and define new games like SDBLH, and manipulate their states, allowing the successful implementation and validation of the algorithms across different games with minimal changes to the core code.

3.2.4 Implementation Format

The project's implementation of the CFR variants follows a scalar format, primarily due to the design constraints of the OpenSpiel framework. Although Oskari Tammelin [15] specifies that CFR+ is a vector-form algorithm, in theory, scalar and vector implementations should produce identical results given the same inputs and random seed as the underlying mathematics remains the same. Furthermore, scalar implementation offers greater transparency in the algorithm's operations. Each step of the algorithm can be more easily traced and understood, as instead of performing condensed vector operations on collections of elements the implementation consists of for-loops presenting the computations on each element individually, broken into more steps, making scalar implementation easier to produce and experiment with which is potentially beneficial for analysis and comparison of different CFR variants.

3.2.5 Implementation Challenges

The implementation approach chosen for this project, while offering benefits in terms of ease of experimentation, presents several significant challenges that impact the scope and performance of the research. The use of Python, selected for its rapid prototyping capabilities and readability, introduces performance overhead compared to lower-level

languages like C++. This performance impact is further amplified by the scalar format implementation rather than efficient vector operations. Consequently, implementation exhibits slower execution times, particularly for larger game trees or when running a high number of iterations. This trade-off between ease of implementation and computational efficiency is acceptable for the current research goals but limits the size and complexity of games the project can feasibly analyze within reasonable time frames.

Another limitation arising from these performance constraints is the project's inability to exhaustively compare many variants of CFR. For instance, in the case of MCCFR, the project had to focus on a specific sampling method (Outcome Sampling) rather than comparing multiple sampling techniques alongside other CFR variants. This restriction narrowed the breadth of the comparative analysis.

The combination of Python, scalar implementation, and limited computational resources also constrained the scalability of the experiments. This restricted the ability to fully explore the performance characteristics of different CFR variants on very large game trees or over extremely high numbers of iterations.

Furthermore, the scalar implementation does not easily lend itself to parallelization, which could have potentially offset some of the performance limitations. This restricted the project's ability to utilise multi-core processors or distributed computing environments.

3.2.6 Implementation Validation

Implementation convergence was validated using Kuhn poker, a well-understood game with a known optimal solution. The project compared the average expected utility of the computed strategies to the known optimal average expected utility value of $\pm\frac{1}{18} \approx \pm 0.0556$ [**?**].

Expected utility in this context refers to the average payoff a player can expect to receive when both players are using their respective strategies.

The optimal expected utility is the expected payoff when both players are using optimal strategies. In Kuhn poker, this value is $\pm\frac{1}{18}$, where the first player expects to lose $1/18$ on average, and the second player expects to win $1/18$ on average when both are playing optimally.

For each CFR variant implemented in this research (Vanilla CFR, CFR+, DCFR, and OS-MCCFR), the project tracked the average expected utility over a series of iterations. The resulting convergence graphs for all algorithms are presented in Appendix A. These graphs visually demonstrate how each algorithm's computed strategy approaches the theoretical optimal value of ± 0.0556 as the number of iterations increases.

Validation shows that all implemented algorithms successfully converged to strategies very close to the optimal value after a sufficient number of iterations. This convergence was observed within a small epsilon (ϵ) of the theoretical optimum, typically within 10^{-4} to 10^{-6} depending on the specific algorithm and the number of iterations run.

This validation step provided confidence in the correctness of the implementations, ensuring that any experimental observations were due to the inherent properties of the algorithms rather than implementation errors.

4 Results

4.1 Experimental Setup

In the experiment the number of iterations is set to 1,000 to keep the experiment within reasonable time frames supported by Tuomas Sandholm's claim in [22] that it is a sufficient number of iterations to reach low exploitability in CFR practice. The experiment involves two variants of Vanilla CFR and DCFR with both simultaneous and alternating updates (referred to as CFR, CFR(Alt.) and DCFR, DCFR(Alt.) throughout the experiment). DCFR discounting parameters are set to $\alpha = 1.5, \beta = 0, \gamma = 2$ as this set of parameters has shown consistently strong performance throughout the literature [22], [16]. The project is interested in finding the particular influence alternating updates have on these two variants as existing literature is lacking these observations. The rest of the algorithms involved (CFR+, OS-MCCFR) are implemented with alternating updates only. Furthermore, the project experiments with CFR+ with quadratic weighted averaging (referred to as CFR+ (QWA) throughout the experiment) to assess the claim made by Brown and Sandholm in [16] of it having superior performance to linear weighted averaging variant.

For evaluation, we use two extensive-form IIGs one of which is widely used as a benchmark in computational equilibrium finding.

4.1.1 Leduc Hold'em

Leduc Hold'em is a poker benchmark from the AI community with six cards in the deck divided into two suits. There are two betting rounds and a maximum of two raises per round. Leduc poker follows the same betting dynamics as Kuhn poker but has two betting rounds: players may bet two chips in the first round and four in the second. Following the first round, a public card is revealed. A player wins the pot by pairing their private card with the public card or, if no pair is made, by holding the highest-ranking private card [10].

4.1.2 Short-deck Big Leduc Hold'em

To evaluate the performance of our CFR variants on a more complex game while maintaining computational feasibility, we introduced a new poker variant called Short-deck Big Leduc Hold'em. It was designed to bridge a narrow yet noticeable gap between the simplicity of the original variant of Leduc Hold'em and a more complex variant of it - Big Leduc Hold'em introduced in [22].

Staying in the project scope, SDBLH is a two-player synthetic poker game. The game uses a reduced deck of 12 cards, consisting of 6 ranks (2 through Ace) and 2 suits. This "short-deck" approach substantially decreases the game size compared to the original variant with 12 ranks [22], as supported by Billings in [23]. Identically to the original Big Leduc Hold'em, SDBLH allows for up to 6 raises per betting round per player and the rest of the game definitions mirror the simpler variant, Leduc Hold'em.

The larger deck and extended betting structure provide a more challenging environment for CFR variants, allowing the project to better differentiate their performance. While more complex than Leduc Hold'em, SDBLH remains more

tractable for the implementations, allowing for more experimentation despite the tight computational constraints.

The implementation of SDBLH was facilitated by OpenSpiel's game definition framework. We extended the existing LH implementation in OpenSpiel, modifying the deck size, betting structure, and hand evaluation logic to match our SDBLH specifications.

4.1.3 Evaluation Metrics

To evaluate and compare the performance of the different CFR variants, we used exploitability as our primary metric. Exploitability measures how much a strategy can be exploited by an optimal counter-strategy [13].

For a two-player zero-sum game, the exploitability $e(\sigma)$ of a strategy profile $\sigma = (\sigma_1, \sigma_2)$ is defined as:

$$e(\sigma) = \max_{\sigma_1' \in \Sigma}(u_1(\sigma_1', \sigma_2) - u_1(\sigma_1, \sigma_2)) + \max_{\sigma_2' \in \Sigma}(u_2(\sigma_1, \sigma_2') - u_2(\sigma_1, \sigma_2))$$

where $u_i(\sigma_1, \sigma_2)$ is the utility for player i given strategy profile (σ_1, σ_2), and Σ is the set of all possible strategies. In our implementation, we used the exploitability function from OpenSpiel's Python API to compute this metric.

4.2 Experimental Results

The graphs provided in Figure 1 and Figure 2 show the exploitability of various CFR algorithms over 1000 iterations in the LH and BLH poker variants. The y-axis represents exploitability on a logarithmic scale, while the x-axis shows the number of iterations. Lower exploitability indicates a closer approximation to Nash equilibrium.

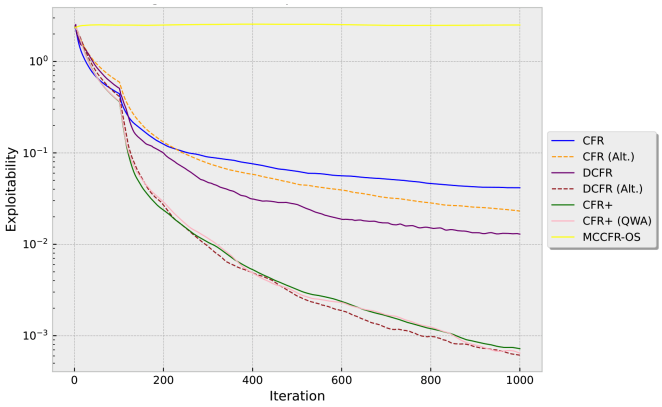

Figure 1.Convergence to Nash Equilibrium in Leduc Hold'em

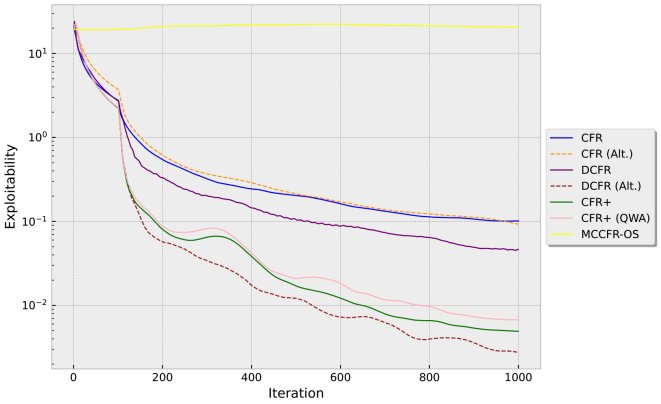

Figure 2.Convergence to Nash Equilibrium in Short Deck Leduc Hold'em

4.2.1 Convergence Analysis in Leduc Hold'em

Vanilla CFR (CFR) shows a consistent but comparatively slow decline in exploitability, indicating effective convergence towards a Nash equilibrium as iterations increase. The alternating variant of Vanilla CFR (CFR Alt.) exhibits a similar trajectory, yet it does reach a slightly lower exploitability compared to the non-alternating, however, the difference is less than an order of magnitude.

The non-alternating Discounted CFR (DCFR) converges at a rate competitive with standard CFR by 1000 iterations, improving convergence compared to Vanilla CFR by less than an order of magnitude. The alternating variant of DCFR (DCFR Alt.) shows significantly faster convergence compared to earlier mentioned methods and its non-alternating variant, outperforming them by multiple orders of magnitude. This difference in performance is noteworthy as it shows that applying alternating updates or weighted averaging separately yields a substantially lower convergence rate rather than when applied simultaneously. While DCFR (Alt.) shows the fastest convergence, it closely competes with CFR+ instances. This close competitiveness may contradict Brown and Sandholm's claim [16] that DCFR with discounting parameters (1.5, 0, 2) consistently outperforms CFR+ as algorithms exhibited nearly equivalent performance.

Both CFR+ and CFR+ with Quadratic Weighted Averaging (CFR+ QWA) show fast convergence rates. Although CFR+ (QWA) reaches lower exploitability after 1000 iterations than CFR+, the difference is insignificant, suggesting that quadratic weighted averaging does not substantially improve upon CFR+ with linear averaging in this context, potentially contradicting the claims made in [16]. Monte Carlo CFR with Outcome Sampling (MCCFR-OS) shows no signs of convergence compared to the other strategies.

4.2.2 Convergence Analysis in Short Deck Big Leduc Hold'em

It is observed that all algorithms achieved higher exploitability than in Leduc Hold'em by an order of magnitude, reflecting the increased complexity of the game. Standard CFR and CFR (Alt.) maintain close performance throughout, similar to their behaviour in Leduc Hold'em.

In the more complex environment, the differences in convergence between DCFR(Alt.), CFR+, and CFR+ (QWA) become more apparent. DCFR(Alt.) has reached comparatively

lower convergence, although the difference is less than an order of magnitude. CFR+ has notably outperformed CFR+ (QWA), showing a consistently faster convergence rate. This is in contrast to their performance in Leduc Hold'em, where the difference was negligible.

Similarly to Leduc Hold'em, MCCFR-OS in SDBLH shows no signs of convergence compared to the other strategies. This consistent behaviour across both games suggests that MCCFR-OS may require a significantly higher number of iterations to demonstrate its convergence properties effectively. These results provide empirical evidence for the varied performance of CFR variants across different game complexities. They highlight the superior performance of DCFR with alternating updates, while also revealing interesting dynamics in the relative performance of CFR+ variants as game complexity increases. The consistent behaviour of MCCFR-OS across both games suggests that MCCFR-OS requires more iterations to achieve low exploitability.

4.2.3 Scalability Analysis

This analysis aims to extrapolate CFR and its variant's performance in even larger, more complex games.

The uniform scaling of the exploitability increase by approximately an order of magnitude when moving from LH to SDBLH suggests that the increased complexity of SDBLH poses similar challenges to all the tested algorithms.

The gap between standard CFR and CFR (Alt.) narrowed in SDBLH, suggesting that the benefits of alternating updates may diminish in larger games. If this trend continues, the advantage of alternating updates becomes negligible in larger games, potentially due to the increased time between updates for any given information set. DCFR (Alt.) maintained its superior performance in both games, but its lead over other algorithms expanded in SDBLH. This could indicate that DCFR (Alt.) relative advantage may increase as game size increases.

The performance difference between CFR+ and CFR+ (QWA) increased in SDBLH, with standard CFR+ showing better performance. This is contrary to what was expected, as quadratic weighted averaging is often thought to be more beneficial in larger games [16]. This result suggests that the relationship between game size and the effectiveness of different averaging schemes is complex and depends on specific game characteristics.

Extrapolating to even larger games, we might expect the performance gap between algorithms to expand further, as the increased complexity may highlight the specific advantages of each variant.

4.2.4 Computational Resources

Each algorithm was allocated and executed on a single core of the M1 Max Apple Silicon processor [35], which operates at a clock speed of 3228 MHz. The training duration for the LH algorithm was notably brief, averaging just 10 minutes. In contrast, the training duration for the SDBLH algorithm was considerably longer, averaging 6.5 hours, with OS-MCCFR computing for 4.4 hours. The latter result shows the explosion of required computational power to compute the solution to a problem that is only slightly more complicated. This well justifies the proposal of the new game variant to bridge the gap between LH and BLH.

5 Evaluation

This research aimed to compare the performance of various Counterfactual Regret Minimization (CFR) algorithms in imperfect information games, focusing on their convergence rates and scalability. The primary research question was: How do different CFR variants differ in terms of convergence speed, and what are their relative performances when implemented and analyzed comparatively?

5.1 Answering the Research Question

To directly address our primary research question, the following subsection discusses the conclusions that can be drawn from the experimental results. Note that in the following discussion, algorithms are referenced by the names used in the experimental setup for the ease of cross-reference.

In terms of convergence speed, Discounted CFR with alternating updates (DCFR Alt.) consistently demonstrated the fastest convergence across both LH and SDBLH. CFR+ variants, both with linear and quadratic weighted averaging, showed rapid convergence rates, closely competing with DCFR Alt. Vanilla CFR, with and without alternating updates, exhibited the slowest convergence among the deterministic algorithms. OS-MCCFR showed no signs of convergence within the 1000 iterations, likely due to limitations in the project's evaluation methodology rather than inherent algorithm performance.

Regarding relative performances, the project's results show a clear hierarchy. In LH, the performance ranking from best to worst was: DCFR Alt. CFR+ CFR+ QWA $>$ DCFR $>$ CFR Alt. $>$ CFR $>$ OS-MCCFR. This order shifts in SDBLH: DCFR Alt. $>$ CFR+ $>$ CFR+ QWA $>$ DCFR $>$ CFR Alt. CFR $>$ OS-MCCFR.

The project effectively demonstrated the impact of game complexity on algorithm performance. All algorithms showed higher exploitability (by about an order of magnitude) in SDBLH compared to LH, reflecting the increased challenge of the more complex game. Interestingly, the performance gap between algorithms generally widened in SDBLH, with some exceptions such as CFR and CFR Alt. performing more similarly.

Contrary to the literature claims, quadratic weighted averaging in CFR+ showed minimal benefit in LH and actually performed worse than linear averaging in SDBLH.

In terms of scalability, DCFR Alt. maintained its superior performance as game complexity increased, suggesting it may scale well to even larger games. Conversely, the relative advantage of alternating updates in Vanilla CFR decreased in the larger game, indicating this modification may be less effective in very large games.

5.2 Effectiveness of Research

The research largely achieves its primary objective of comparing the performance of different CFR variants. It successfully answers the research question by providing detailed comparisons of the relative convergence speeds of these algorithms in both a standard benchmark game (Leduc Hold'em) and a more complex variant (Short Deck Big

Leduc Hold'em). The use of exploitability as a metric effectively quantifies the quality of the approximate solutions obtained by each algorithm, allowing for direct comparisons.

The experimental setup is designed using appropriate metrics (exploitability, number of iterations) and game variants to test and compare the algorithms. The results successfully conclude the relative performance of different CFR variants and their scalability. The introduction of Short Deck Big Leduc Hold'em (SDBLH) as a new benchmark game was successful.

5.3 Suitability of Approach

The approach taken in this research is largely suitable for addressing the research question. The implementation of multiple CFR variants (Vanilla CFR, DCFR, CFR+, OS-MCCFR) allows for a comprehensive comparison of their performance. The use of two different game complexities (Leduc Hold'em and SDBLH) provides insights into how these algorithms scale with increasing game size.

The choice of exploitability as the primary metric for comparison is appropriate, as it directly measures how far a strategy is from a Nash equilibrium. This aligns well with the goal of assessing convergence speeds towards optimal play. The decision to run experiments for 1000 iterations, based on literature suggestions, provides a reasonable balance between computational feasibility and the ability to observe meaningful convergence trends. The implementation of algorithms in a scalar format, while not optimal for performance, offers greater transparency in the algorithm's operations. This choice aligns well with the research goals of understanding and comparing different CFR variants, rather than optimizing for raw performance.

However, there are areas where the research approach could have been improved:

- The comparison of OS-MCCFR with deterministic algorithms based solely on iteration count is a significant limitation. A more equitable comparison considering computational effort (using a "nodes touched" metric) would have allowed appropriate comparisons between OS-MCCFR and other CFR variants.
- The research could have benefited from a more in-depth analysis of why certain algorithms performed better or worse in different scenarios. For example, a deeper exploration of why CFR+ QWA underperformed in SDBLH compared to standard CFR+ could have provided additional theoretical insights.

Overall, the research approach was largely successful. The project achieved its main goal of implementing and comparing various CFR variants across different game complexities.

5.4 Project Organisation

The main deliverables of this project were:

- Implementation of multiple CFR variants (Vanilla CFR, DCFR, CFR+, OS-MCCFR) - Achieved
- Validation of the CFR in Kuhn poker with known optimal result to confirm the effectiveness of the implemented algorithms - Achieved
- Implementation of a new benchmark game - Achieved
- Comparative analysis of CFR variants - Achieved
- Scalability analysis of CFR variants - Achieved

All core deliverables were achieved, with the scalability analysis being somewhat limited in scope and comparative analysis of CFR variants to OS-MCCFR was not as informative as comparisons among other variants because of the evaluation techniques used.

The project's organization allowed for the successful completion of these deliverables within the given constraints. However, the project organization could have been improved in computational resource allocation to better plan for the computational demands of larger games like SDBLH. This could have allowed for more extensive experiments or the inclusion of additional game variants.

5.5 Evaluation of the Experiments

A significant limitation in the experiments taken in this project is comparing OS-MCCFR against deterministic algorithms purely on an iteration count basis overlooks a key advantage of sampling-based methods: their lower computational cost per iteration. In games where full tree traversal is computationally expensive, this reduced per-iteration cost allows sampling methods to perform many more iterations in the same amount of computational time, potentially leading to faster real-time convergence despite requiring more iterations.

To address this and provide a more equitable comparison, future experiments should consider making use of a number of nodes touched metric rather than using raw iteration counts, to measure algorithm convergence performance based on the number of nodes touched during tree traversal. This metric would provide a more accurate representation of computational effort, allowing for fairer comparison between sampling-based and deterministic methods. The current experimental setup overlooks this metric.

6 Conclusion

This research has provided valuable insights into the performance characteristics of various Counterfactual Regret Minimization algorithms in imperfect information games. Through experimentation and analysis using Leduc Hold'em and introduced Short Deck Big Leduc Hold'em, the project has addressed its primary research question regarding the convergence speeds and relative performances of different CFR variants.

The project's findings confirm that alternating updates generally improve convergence rates. However, the performance gap between standard CFR and CFR with alternating updates narrowed in the more complex SDBLH, suggesting that the benefits of alternating updates may diminish in larger games. Furthermore, the project confirms that DCFR with alternating updates and parameters $\alpha = 1.5, \beta = 0, \gamma = 2$ outperforms CFR+. Lastly, an important observation has been made on how the simultaneous application of alternating updates and discounting outperforms separate applications but multiple orders of magnitude.

Contrary to some claims in the literature, the research found that CFR+ with quadratic weighted averaging (CFR+ QWA) did not consistently outperform standard CFR+. While CFR+ QWA showed a slight advantage in Leduc Hold'em, standard CFR+ demonstrated better performance in the more complex SDBLH game. This result highlights the algorithm behaviour across different game scenarios and challenges existing assumptions about the benefits of quadratic weighted averaging in larger games.

The introduction of SDBLH as a new benchmark game allowed for more analysis of algorithm scalability while remaining computationally feasible. The results showed that all algorithms achieved higher exploitability in SDBLH compared to LH by approximately an order of magnitude, reflecting the increased complexity of the game.

The experiments also revealed limitations in the evaluation methodology, particularly for sampling-based methods. Outcome Sampling Monte Carlo CFR (OS-MCCFR) showed no signs of convergence within the 1000 iterations used in the experiments in both LH and SDBLH. This suggests that OS-MCCFR may require significantly more iterations to demonstrate its convergence properties effectively, emphasizing the need for different evaluation metrics when comparing sampling-based methods to deterministic algorithms.

These findings contribute to the field by providing empirical evidence for the relative strengths and weaknesses of each CFR variant in different game scenarios. The results challenge some existing assumptions about algorithm performance. Furthermore, these can contribute to establishing new trajectories in the research of the CFR algorithms.

Future work should focus on developing more equitable comparison methods between sampling-based and deterministic algorithms, potentially using metrics such as 'nodes touched' to better account for computational efficiency. Additionally, further investigation into the performance of CFR+ variants and the impact of different averaging schemes in larger games could yield valuable conclusions.

REFERENCES

[1] M. Campbell, A. J. Hoane, and F. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, Jan. 2002, doi: 10.1016/S0004-3702(01)00129-1.

[2] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "Chinook: The World Man-Machine Checkers Champion," *AI Magazine*, vol. 17, no. 1, pp. 21–29, 1996, doi: 10.1609/aimag.v17i1.1208.

[3] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145–149, Jan. 2015, doi: 10.1126/science.1259433.

[4] L. A. Petrosyan and N. A. Zenkevich, *Teoriya igr i social'no-ekonomicheskoe povedenie [Game Theory, Social and Economic Behaviour], Ekonomicheskaya shkola. Analiticheskoe prilozhenie [Economis School. Analytics Application]*, 2002, vol. 1, nr 1, pp. 119-131.

[5] J. Schaeffer et al., "Checkers Is Solved," *Science*, vol. 317, no. 5844, pp. 1518–1522, Sep. 2007, doi: 10.1126/science.1144079.

[6] V. Allis, "A Knowledge-Based Approach of Connect-Four: The Game Is Solved: White Wins," *ICG*, vol. 11, no. 4, pp. 165–165, Dec. 1988, doi: 10.3233/ICG-1988-11410.

[7] J. Bronowski, *The ascent of man*, Documentary, 1973.

[8] O. Tammelin, N. Burch, M. B. Johanson, and M. Bowling, "Solving Heads-Up Limit Texas Hold'em," presented at the International Joint Conference on Artificial Intelligence, Jul. 2015.

[9] H. W. Kuhn, "Simplified Two-Person Poker," in *Contributions to the Theory of Games*, vol. 1, H. W. Kuhn and A. W. Tucker, Eds., Princeton University Press, 1950, pp. 97–103.

[10] F. Southey et al., "Bayes' Bluff: Opponent Modelling in Poker," Jul. 04, 2012, arXiv: arXiv:1207.1411. doi: 10.48550/arXiv.1207.1411.

[11] S. Hart and A. Mas-Colell, "A Simple Adaptive Procedure Leading to Correlated Equilibrium," *Econometrica*, vol. 68, pp. 1127–1150, Sep. 2000, doi: 10.1111/1468-0262.00153.

[12] M. Littman and M. Zinkevich, "The 2006 AAAI Computer Poker Competition," *ICGA Journal*, vol. 29, Sep. 2006, doi: 10.3233/ICG-2006-29314.

[13] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2007. [Online]. Available: https://proceedings.neurips.cc/paper/2007/hash/08d98638c6fcd194a4b1e6992063e944-Abstract.html

[14] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling, "Monte Carlo Sampling for Regret Minimization in Extensive Games," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2009. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2009/hash/00411460f7c92d2124a67ea0f4cb5f85-Abstract.html

[15] O. Tammelin, "Solving Large Imperfect Information Games Using CFR+," Jul. 18, 2014, arXiv: arXiv:1407.5042. doi: 10.48550/arXiv.1407.5042.

[16] N. Brown and T. Sandholm, "Solving Imperfect-Information Games via Discounted Regret Minimization," *AAAI*, vol. 33, no. 01, pp. 1829–1836, Jul. 2019, doi: 10.1609/aaai.v33i01.33011829.

[17] G. Farina, C. Kroer, and T. Sandholm, "Faster Game Solving via Predictive Blackwell Approachability: Connecting Regret Matching and Mirror Descent," Mar. 07, 2021, arXiv: arXiv:2007.14358. doi: 10.48550/arXiv.2007.14358.

[18] "AIPT Section 3.2: Solving Poker – Toy Poker Games," AI Poker Tutorial. [Online]. Available: https://aipokertutorial.com/toy-poker-games/

[19] I.V. Romanovskii, "Reduction of a game with complete memory to a matrix game," *Soviet Mathematics*, vol. 3, 1962, pp. 678–681.

[20] "Lossless abstraction of imperfect information games," *Journal of the ACM*. [Online]. Available: https://dl.acm.org/doi/10.1145/1284320.1284324

[21] J. Shi and M. L. Littman, "Abstraction Methods for Game Theoretic Poker," in *Computers and Games*, T. Marsland and I. Frank, Eds., Berlin, Heidelberg: Springer, 2001, pp. 333–345, doi: 10.1007/3-540-45579-5_22.

[22] N. Zhang, S. McAleer, and T. Sandholm, "Faster Game Solving via Hyperparameter Schedules," Apr. 13, 2024, arXiv: arXiv:2404.09097. [Online]. Available: http://arxiv.org/abs/2404.09097
[23] D. Billings, "Algorithms and assessment in computer poker," Ph.D. dissertation, University of Alberta, Canada, 2006.
[24] D. Billings et al., "Approximating Game-Theoretic Optimal Strategies for Full-scale Poker," presented at the International Joint Conference on Artificial Intelligence, Aug. 2003. [Online]. Available: https://www.semanticscholar.org/paper/Approximating-Game-Theoretic-Optimal-Strategies-for-Billings-Burc ce32b559b429f2549d38d67046409399a22bf91f
[25] M. Lanctot, "Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games," Ph.D. dissertation, University of Alberta Libraries, 2013, doi: 10.7939/R3K085.
[26] S. M. Ross, "Goofspiel — the game of pure strategy," *Journal of Applied Probability*, vol. 8, no. 3, pp. 621–625, Sep. 1971, doi: 10.2307/3212187.
[27] N. Brown and T. Sandholm, "Regret-Based Pruning in Extensive-Form Games," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. [Online]. Available: https://papers.nips.cc/paper_files/paper/2015/hash/c54e7837e0cd0ced286cb5995327d1ab-Abstract.html
[28] Yu. Nesterov, "Excessive Gap Technique in Nonsmooth Convex Minimization," *SIAM J. Optim.*, vol. 16, no. 1, pp. 235–249, Jan. 2005, doi: 10.1137/S1052623403422285.
[29] A. Gilpin, S. Hoda, J. Peña, and T. Sandholm, "Gradient-Based Algorithms for Finding Nash Equilibria in Extensive Form Games," in *Internet and Network Economics*, X. Deng and F. C. Graham, Eds., Berlin, Heidelberg: Springer, 2007, pp. 57–69, doi: 10.1007/978-3-540-77105-0_9.
[30] C. Kroer, G. Farina, and T. Sandholm, "Solving Large Sequential Games with the Excessive Gap Technique," ArXiv, Oct. 2018, Accessed: Aug. 01, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Solving-Large-Sequential-Games-with-the-Excessive-Kroer-Farina/80ed19d1ffa0586be91a56c424a470de7d548d25
[31] N. Brown, A. Lerer, S. Gross, and T. Sandholm, "Deep Counterfactual Regret Minimization," May 22, 2019, arXiv: arXiv:1811.00164. doi: 10.48550/arXiv.1811.00164.
[32] T. Neller and M. Lanctot, "An Introduction to Counterfactual Regret Minimization," 2013. [Online]. Available: https://www.semanticscholar.org/paper/An-Introduction-to-Counterfactual-Regret-Neller-Lanctot/0184855c7baafdbcadcab967d4bfa7d4f8b86285
[33] "What is OpenSpiel? — OpenSpiel documentation." [Online]. Available: https://openspiel.readthedocs.io/en/latest/intro.html

APPENDIX

A.

Below convergence graphs to optimal solution in Kuhn poker for Vanilla CFR and its 3 variants are provided:

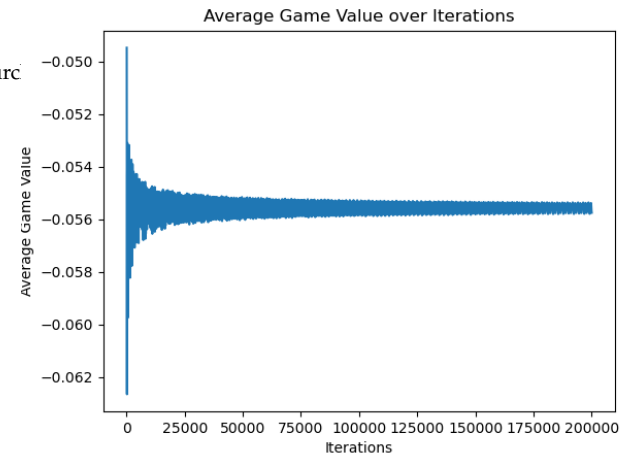

Figure 1. Convergence to Optimal Average Expected Utility in Kuhn Poker by CFR

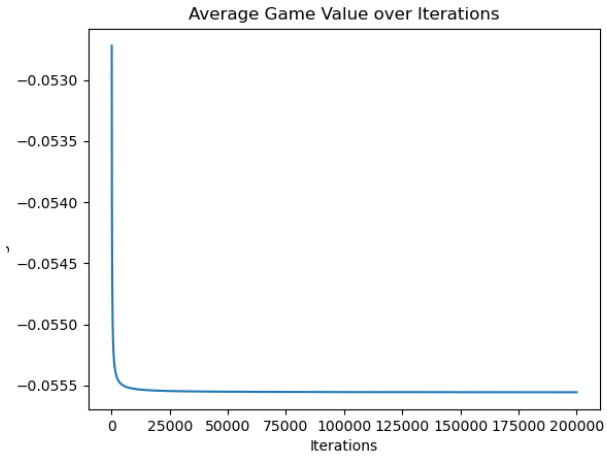

Figure 2. Convergence to Optimal Average Expected Utility in Kuhn Poker by DCFR(1.5, 0, 2)

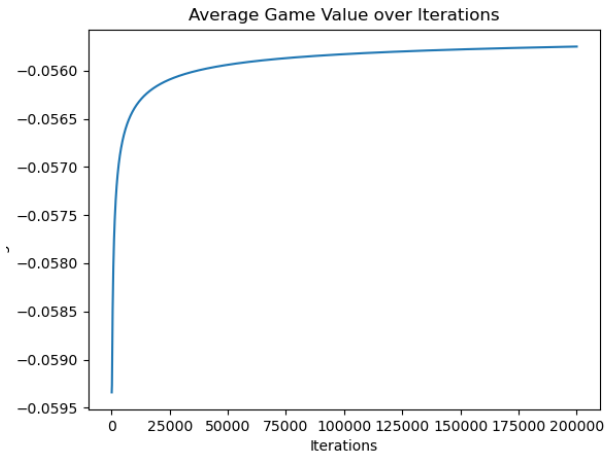

Figure 3. Convergence to Optimal Average Expected Utility in Kuhn Poker by CFR+ (linear weighted averaging)

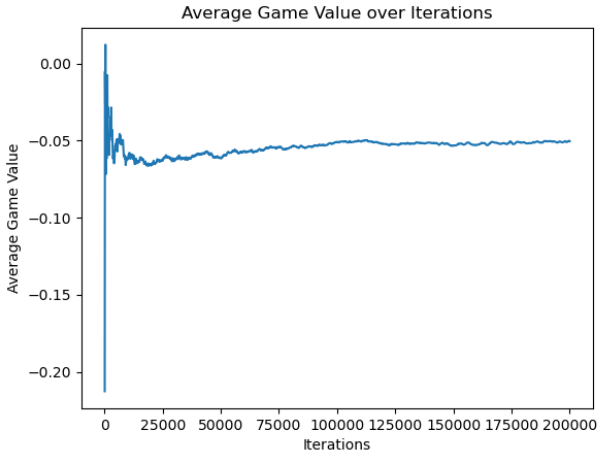

Figure 4. Convergence to Optimal Average Expected Utility in Kuhn Poker by OS-MCCFR