UNIVERSITY OF WARWICK

DEPARTMENT OF ECONOMICS

Searching For Nash At Monte Carlo:

Approximating Optimal Strategies For Five-Card Draw Poker

BY

BRYAN PEK ZONG EE

University ID: 2100397

—Word Count: 4996 —

April 25, 2024

Abstract

Historically, the fields of game theory and AI have primarily concentrated on solving perfect-information games where all relevant data is available to all players at all times. In contrast, real-world strategic interactions often involve elements of bluffing, deception, and the anticipation of an opponent's intentions, making imperfect-information games a more accurate representation of real-world decision-making. Until recently, Texas Hold'em poker, has emerged as the leading benchmark for evaluating game solving algorithms in imperfect information games. This paper aims to contribute to the existing poker literature by extending the modelling approach to Five-Card Draw poker. It describes the steps taken towards building an optimal poker-playing program and is evaluated against models of human poker playing accounting for factors such as agent-modelling, deception, risk management and unreliable information. The result is indicative of a program capable of playing Five Card Draw Poker against a player who acts randomly. However, there remains considerable work to be done to play at a superhuman level, primarily due to computational limitations on running the program on a personal laptop. Nonetheless, the expansion of the methodology in this paper can serve as a foundation for future research.

Contents

1 INTRODUCTION

The study of strategic games has long been an interesting domain for research due to their well-defined set of rules, specific objectives and complex decision making. Additionally , having well-defined metrics for progress allows for comparisons of different approaches based on novel ideas. These games are often characterised by the level of information available to players (perfect-information vs imperfect-information) as well as the influence of chance on outcomes (deterministic vs stochastic).

Historically, the fields of game theory and AI have primarily concentrated on solving deterministic perfect-information games. This is evidenced in the successes of Deep Blue in the game of Chess [1], Chinook in the game of Checkers [2] and Alpha Go [3] in the game of Go. However, as John von Neumann, pioneer of modern game theory, aptly noted, real-world strategic interactions often involve elements of bluffing, deception, and the anticipation of an opponent's intentions, making imperfect-information games a more accurate representation of real-world decision-making [4].

Five Card Draw Poker is an example of such an imperfect-information game, capturing the intricacies of real-world decision-making, where players grapple with varying levels of information, make decisions under uncertainty, and employ strategies that encompass bluffing and deception [5]. For example, in evaluating alternative courses of action, agents have to account for the plausible (albeit not necessarily rational) actions of others, similar to how a poker player has to account for how the opponent might react to a certain style of play. Similarly, agents infer probable subsequent states of the world given previous histories and information, similar to how a poker player would identify patterns in an opponent's play and attempt to exploit them. As poker is a microcosm of real world decision making, the goal is then to develop a model of best-response strategies that would play optimally against

any human, thereby gaining insights into reasoning with imperfect information.

The research in this paper focuses on a specific variant of poker called Five-Card Draw Poker. In contrast to its popular counterpart Texas Hold Em, which serves as the foundation for much of the existing poker literature, literature on Five Card Draw Poker is almost nonexistent despite its widespread recognition worldwide. However, Five Card Draw introduces a distinctive game mechanic called “Drawing” which leads to a deeper level of complexity. In particular, the strategic action of ”false carding”, whereby a player deliberately chooses not to draw the maximum number of cards needed to improve his hand in attempt to conceal the strength or weakness of his hand acts as a bluffing mechanism [6]. This dual utility of “Drawing” as both a strategic action used to improve one’s hand and a bluffing mechanism creates the potential for players to convey mixed signals, adding complexity to the interpretation of an opponent’s intentions.

2 LITERATURE REVIEW

Previous literature on building poker programs is divided into two main streams: heuristic-based and game-theoretic approaches.

In heuristic-based approaches, poker decisions are made using a rule-based expert system, integrating various game information such as a player's position and hand history. These strategies can be crafted manually by domain experts or derived through trial and error either by manual play or by simulations. The exploration of this approach started with Findler in the 1970s who simulated the human cognitive process to develop a poker-playing program for Five-Card Draw Poker [5] [7]. However, Findler's program fell short of defeating strong players. Similarly, Waterman attempted to learn heuristics for Five-Card Draw Poker using machine learning to create a betting system for the game and achieved a skill level comparable to that of an experienced human player [8]. Korb et al. utilised a Bayesian network, incorporating opponent information through increasing game trials, resulting in a heuristic betting strategy for the game of Five-Card Stud Poker that performed reasonably against test opponent programs but fell short against experienced amateur poker player [9]. The University of Alberta Poker Research Group created heuristic-based programs Loki and Poki with opponent modelling for the game of Texas Hold'Em Poker and achieved an intermediate level of play as well as consistently win in games of full-ring Texas Hold'Em for which they are designed for but could not compete at a world-class level [10] [11]. The lack of performance of heuristic-based approaches may be attributed to the complexity of the game, which contains too many different decision scenarios. The vast scope of possibilities surpasses the capabilities of any rule-based system, leading to low performance.

Transitioning to game-theoretic approaches, AI systems that have achieved superhuman performance in various games like Checkers, Go and Texas Hold'em poker have done so

by attempting to approximate a Nash equilibrium strategy as opposed to a rule-based system. However, computing game-theoretic optimal strategies for full-scale poker is almost intractable due to the large branching factor, rendering the calculation and storage of optimal strategies prohibitively expensive. For example, the game tree of two-player Texas Hold'Em poker contains 10^{18} nodes.

Early attempts at applying game-theoretic betting principles to the game of Five-Card Draw Poker were made by Nesmith Ankeny, who broke down each phase of the game and analysed them separately using mathematical analysis [6]. This resulted in betting and calling rules that were supported by phase-specific tables of hand distributions but it overlooked valuable information by treating each phase in isolation.

In contrast, approximate game-theoretic optimal strategies were only found in the game of Texas Hold'Em Poker through abstracting the problem and computing an exact solution to the problem using Counterfactual Regret Minimization (CFR). This methodology underpinned the success of Cepheus in heads-up limit Texas Hold'Em [12], Libratus in heads-up no-limit Texas Hold'Em [13] and Pluribus in six-player no-limit Texas Hold'Em.

3 METHODOLOGY

3.1 Rules and Model Constraints

In Five-Card Draw poker, the game starts with each player being dealt 5 cards. Similar to Texas Hold'Em, the game is played with a dealer, small blind and big blind in which the player next to the dealer is the small blind and the player next to the small blind is the big blind. After cards are dealt, the first betting round begins, starting from the small blind, players are allowed to check/call, raise or fold. After the first betting round comes the drawing round, where each player may choose to discard and replace any of his current cards to make up a five card hand or Stand Pat (not discard any cards). The second betting round ensues, followed by a showdown if more than one player remains after the second betting round. The winner of the showdown is determined by the player with highest hand according to the universal poker hand rankings (in descending order: royal flush, straight flush, four of a kind, full house, flush, straight, three of a kind, two pair, high card).

The game environment is modelled after a commonly played cash game on Poker Stars with 2 players, a small blind of $1 and a big blind of $2. Both player always start the round with a stack of $40 and is allowed to raise any amount up to his stack. If a player chooses to raise, the minimum amount is twice the current largest bet.

3.2 Abstraction and Information Set Representation

Abstraction employs abstracted mathematical models to capture the essential properties of the actual domain, such that a solution to the abstracted domain provides a useful approximation of an optimal strategy for the actual domain [14]. This method of abstraction is used in solving large incomplete-information games as the original game surpasses current technological capabilities for equilibrium solving or requires simplification for computational

feasibility [15].

One common form of game abstraction includes information abstraction, where similar information sets are grouped together. This paper considers two types of information abstraction "No Suits" and "Highest Suit Mini". In "No Suits", the suit of the cards drawn is removed, leaving the player with only knowledge of the numeric values of his cards. A solution to this abstracted game may still be a solution to the actual game as the only information loss to the player is whether he currently has or potentially has a flush. As a player draws to a flush $\frac{\binom{13}{5}\binom{4}{1}}{\binom{52}{5}} = 0.1965\%$ of the time, the probability is low enough to ignore it in exchange for faster convergence of the model to the optimal strategy. In "Highest Suit Mini", the abstracted game attempts to include information of flushes and potential flushes through binary encoding of the cards with the same suit, if the number of cards with the same suit is greater than 3. Binary encoding is used instead of representing each suit separately as the suit itself provides no additional information. For example, a hand of $3\clubsuit 5\clubsuit 7\clubsuit 8\clubsuit 9\clubsuit$ has the same value as $3\spadesuit 5\spadesuit 7\spadesuit 8\spadesuit 9\spadesuit$ and hence should be played using the same strategy.

To summarise, for the hand $3\clubsuit 5\diamondsuit 7\clubsuit 8\clubsuit 9\clubsuit$, the "No Suit" information abstraction would represent the cards as "35789" whereas the "Highest Suit Mini" information abstraction would represent the cards as "10111|35789"

Similarly, game abstraction could also include action abstraction where similar actions are bucketed together. In this paper, all raises are assigned a value equal to the minimum raise possible.

The information state representation of the game thus would include the player's current card, the sequence of betting action in the first round, the sequence of drawing action and

the sequence of betting action in the second round. Betting actions are denoted by c for calls or checks, and r[value raised] for raises. Folds are ignored as a terminal node is reached where the game ends. As for drawing actions, a player has up to $2^5 = 32$ possible drawing combinations. Hence, each number from 0 to 31 is mapped to a corresponding drawing combination. For example, 0 would map to the action stand pat, 15 maps to discarding the third and fourth card and 31 maps to discarding all five cards. These numbers are prefixed with the character "d", to indicate that the action is a discarding action. During the drawing round, a player also observes the number of cards burned by the opponent, hence it is denoted as d[number of cards burned].

To illustrate, the small blind's current information set may be "56789:r4c/d0b2/", which indicates that the small blind has a straight. He raised up to a value of 4 in the first betting round which was called by the big blind. During the drawing round, he did not dispose any cards but saw the big blind dispose two of his cards. It is now the start of the second betting round and is his turn to move.

3.3 Monte Carlo Counterfactual Regret Minimisation (MCCFR)

This paper attempts to use the MCCFR algorithm to approximate a Nash equilibrium for the game of Five-Card Draw poker. To fully understand how MCCFR functions, the following section elaborates on the interconnection of key concepts: regret, regret-matching, vanilla Counterfactual Regret Minimization (CFR), and their relationship to MCCFR.

Regret, in decision theory, measures the difference between the payoff of a chosen action and the optimal action in hindsight [16]. Suppose a player chose action X which gave him a payoff of 5 whereas optimal action Y would have given him a payoff of 10, he would thus have a regret of -5 (5-10) for choosing X.

In normal form games, regret matching algorithms leverage regret to inform subsequent

strategies, updating strategies to favour actions regretted in the past. As both player apply regret matching, expected regret is minimised through self play as players choose actions in proportion to their positive regrets. Over time, this process converges to a correlated equilibrium [17]. The process goes as follows, all players are initialized with cumulative regrets and strategy profile sum equal to 0. For some number of T iterations, a regret-matching strategy profile is computed. Suppose action A has a regret of 5 and action B has a regret profile of 20, player would play action A with probability $\frac{5}{5+20}$ and B with probability $\frac{20}{5+20}$). The strategy profile is added to the strategy profile sum and an action is taken by the player based on the strategy profile. Player regrets are then computed and added to the cumulative regrets. After T iterations, the average strategy as computed by the dividing strategy profile sum by the T number of iterations make up the correlated equilibrium.

In sequential games, vanilla CFR uses the regret matching algorithm but also accounts for the probabilities of reaching certain information sets in the game tree. Hence, there is a forward pass of game state information and probability of player actions as well as backward pass of utility information. To summarise, CFR recursively runs through a game tree, computes a reward when a terminal node is reached, calculates the total regret and updates the strategy after each iteration according to the probability of reaching the node. For two player zero sum games, the average strategy eventually converges to a Nash Equilibrium. Pseudocode for Vanilla CFR can be found in Appendix A.

MCCFR is a variant of CFR that uses Monte Carlo sampling to estimate the counterfactual regrets. Instead of traversing the entire game tree, a subset of the game tree is sampled and traversed, making it particularly useful for solving large games when traversing the entire game tree is computationally infeasible. In solving Five Card Draw Poker, this pa-

per uses external sampling by considering a single action for both chance nodes as well as opponent decision nodes, where the opponent's actions are sampled according to their current strategy profile. For each game iteration, a player (traverser) will explore every possible action and update his regrets whereas the opponent will play according the strategy determined by his current regrets. The traverser role is switched after each iteration.

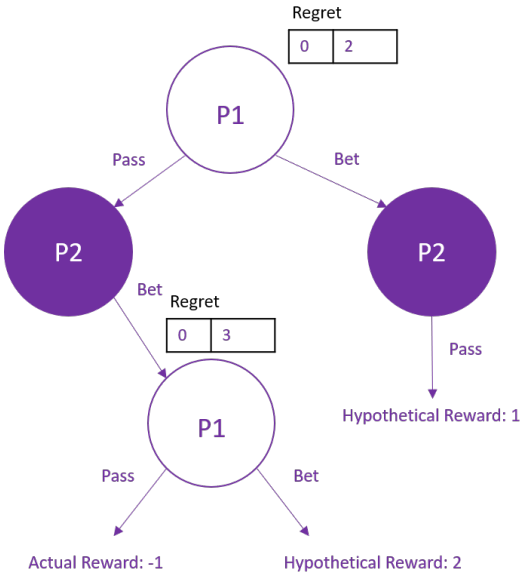

Figure 1: Game Tree for Kuhn Poker

Figure 6 shows the first iteration of MCCFR for player 1 in the game Kuhn Poker. Suppose at the initial chance node, player 1 is assigned the King card and player 2 is assigned the Jack card. At the start, the accumulated regrets for actions pass and bet is initialised to a value of 0. Hence, the player takes a random action, suppose he plays pass. He observes that player 2 plays bet and reaches the bottom node. As before, he takes a random action, suppose he plays pass, giving him a payoff of -1. Player 1 then considers his payoff had he chosen to bet instead of playing pass at this bottom node, which would have given him a hypothetical reward of 2. The regrets at this node is thus -1-(-1) = 0 for playing pass and 2-(-1) = 3 for playing bet. This actual reward is then passed backwards, therefore at the top node, he gains a reward of -1 by playing pass. He then considers what

he would have gotten had he played bet instead of pass at the top node and computes regrets accordingly. The strategy of player 1 is then updated corresponding to the positive regrets in each node (in this case, plays bet with p=1 at the top node and bottom node). Pseudocode for MCCFR can be found in Appendix B.

In vanilla CFR and MCCFR, the first iteration is weighted equally as the last iteration (first iteration regrets and strategies decay at a rate of 1/T where T=number of iterations). However, in early iterations the strategy is still developing and the model is likely to act randomly. For example, in the first iteration, the model plays randomly over all information sets [18]. Hence, the player taking the average strategy will be weakened by these early iteration strategies, given that T is not large enough. Hence, to discount early iterations and speed up convergence, a weight of X is given to the regret and strategy contributions every N number of iterations, where X=1 and is incremented every N number of iterations.

3.4 Evaluation

The models created using MCCFR will be evaluated using self-play simulations, where the strategy from the ”No Suit” abstraction is played against the strategy from ”Highest Suit Mini” to quantify the contribution of ”flush information” set.

In addition to self-play simulations, comparisons against heuristic-based models are conducted to benchmark the proposed models against established strategies in poker. These heuristic models are constructed based on key characteristics essential for successful poker play, including hand strength, hand potential, bluffing, betting strategy, opponent modeling, and unpredictability [19]. Each heuristic model is designed with varying levels of complexity, incorporating different combinations of these characteristics to emulate diverse player strategies. Additionally, a rule-based model is built using insights from an article that summarises strategies from poker books written by world class poker players as well as game

theorists [20]. Descriptions of implementations for each heuristic model can be found in Appendix C.

Simulation between models would use duplicate matches to reduce the influence of luck. As poker simulations are used, each hand can be played with no memory of preceding hands, hence it is possible to play the same round twice with the initial cards dealt to both models having swapped in the second round. This would mean that neither models could get systematically lucky, since each model will have the same number of good and bad hands in expectation. The differences between the performance of models can therefore be attributed more strongly to the quality of the decisions made in each state. Performance is measured using the average number of small blinds won per hand (sbb/h) of poker. Each match between models would involve 200,000 different poker hands being played. The MCCFR models are than tested at the 99% CI using a one-tailed t test to determine whether it is profitable. By definition, the Nash equilibrium solution concept is where both players do not have an incentive to unilaterally deviate from their current strategy. In two player zero sum games, playing the Nash Equilibrium would mean that is guaranteed not to lose in expectation, regardless of the skill level of the opponent. For example, in a game of scissors-paper-stone, the Nash equilibrium is for both players to play each with 1/3 probability. Hence, regardless of the heuristic-based model applied, a fully converged MCCFR model should show that it is at least breaking-even against it.

4 RESULTS

4.1 Self-Play Simulations

The strategy for the ”No Suit” abstraction as well as ”Highest Suit Mini” abstraction was each computed on a 16GB personal laptop over 3 weeks with approximately 10^7 MCCFR iterations. For the ”No Suit” abstraction, over 88.3 million information sets and behavioral strategies were reached and calculated, whereas the ”Highest Suit Mini” abstraction involved over 116.2 million information sets and behavioral strategies.

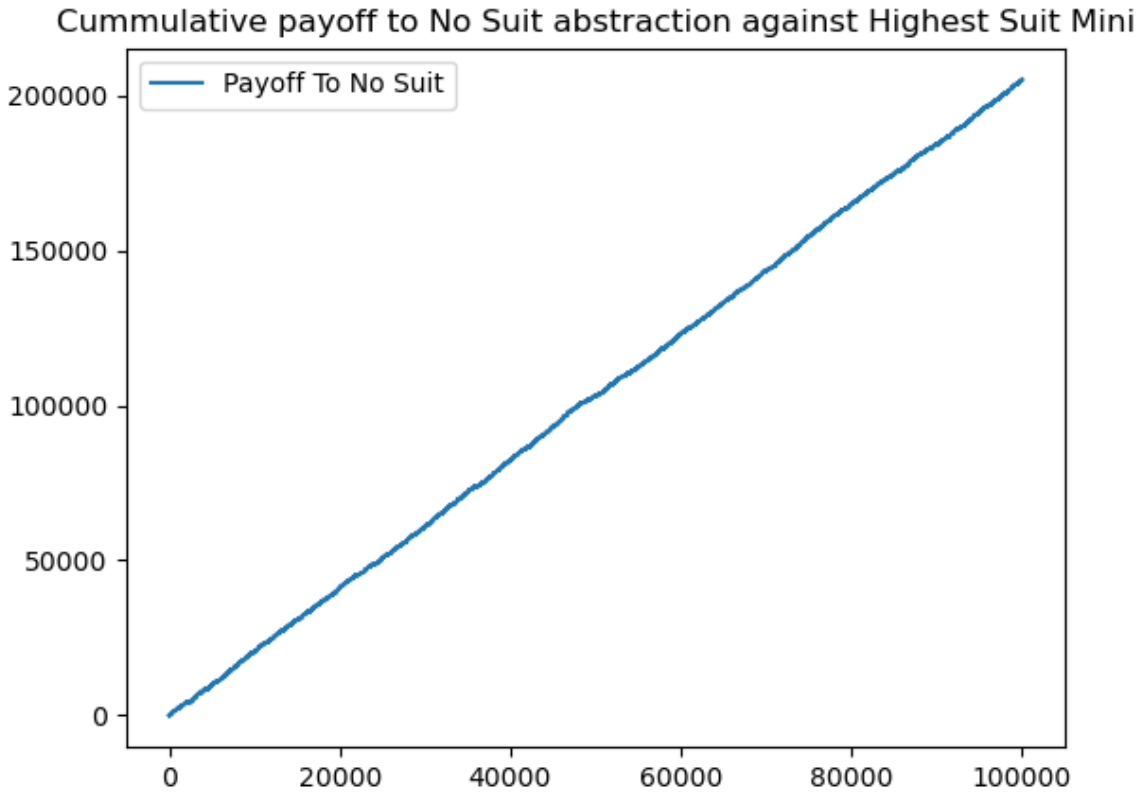

Figure 2: Cumulative payoff of ”No Suit” abstraction against ”Highest Suit Mini”

Contrary to prior belief, additional information has seen the ”Highest Suit Mini” abstraction performing significantly worse than ”No Suit”, with ”No Suit” winning 2.05 sb-

b/h. Running a t-test on the null hypothesis where the average payoff to ”No Suit” is equal to 0 against the alternative that the average payoff is positive indicates that we reject the null hypothesis of payoff being equal to 0. In hindsight, this is likely due to ”No Suit” abstraction converging to it’s Nash equilibrium at a faster rate as compared to the ”Highest Suit Mini abstraction” due to the smaller sized game tree. As MCCFR converges to a Nash, the expected value of a strategy will start to stabilise. Both strategies can thus be played against copies of itself and have it’s level of variance in payoffs compared in an f-test to check for evidence in stabilisation. Despite ”No Suit” having a lower variance in payoffs, it is not statistically significant (p-value of 0.59). One plausible explanation is that ”No Suit” itself is still far away from converging to a Nash Equilibrium despite having faced with a smaller game tree.

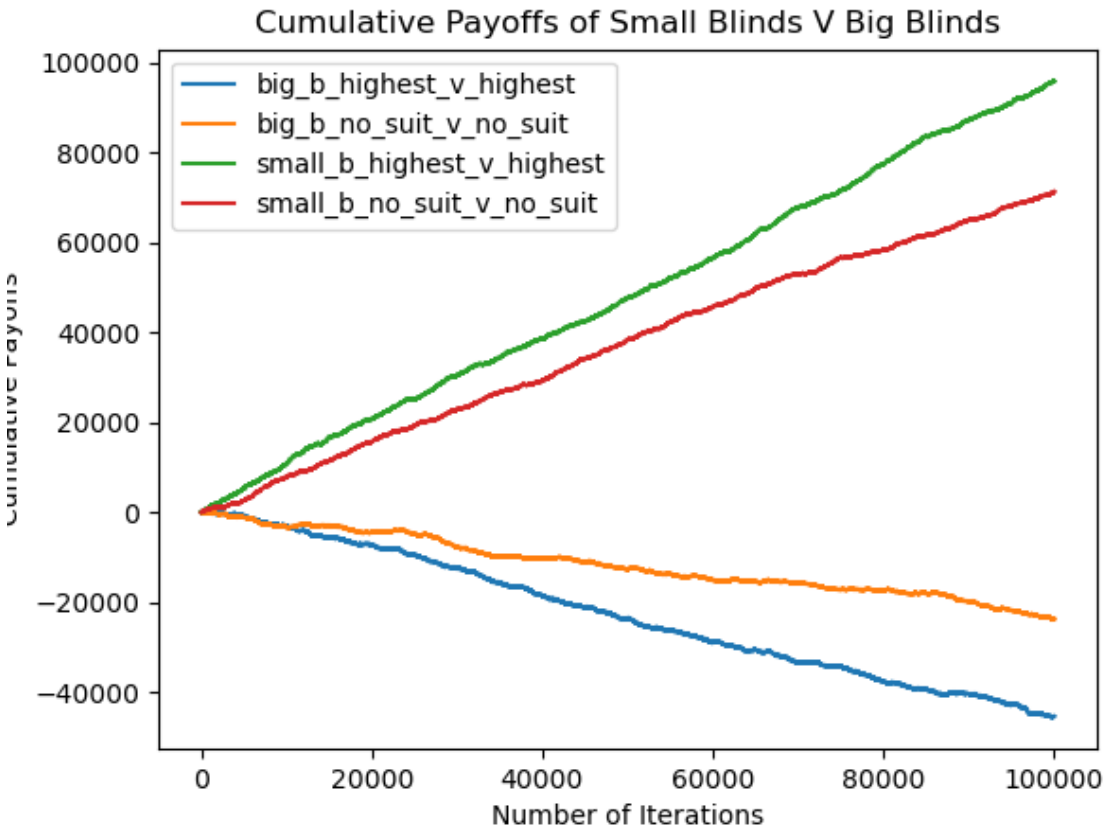

Figure 3: Cumulative payoff of Small Blinds and Big Blinds

By examining the payoff history of the models against copies of itself, it is observed that the small blind consistently profits from the big blind regardless of what abstraction was used, with the small blind of ”No Suit” earning 0.71343 sbb/h and ”Highest Suit Mini” earning 0.95902 sbb/h. This is analogous to the ”first-mover advantage” often found in sequential games where the first player to act receives the highest payoff. In this case, the small blind pays a premium of 1 unit in exchange for the privilege to look at his current cards and chooses an action which acts as a signal to the opponent. The small blind is in an offensive position if he chooses to play anything but fold, signalling that he current has a strong hand and the big-blind has to respond defensively.

4.2 Simulations against heuristic-based models

As the ”No Suit” abstraction strategy plays profitably against the ”High Suit Mini” abstraction strategy, it is used as the representative MCCFR model to be compared against other heuristic-based models.

Initial simulations against weak benchmark strategies reveal that it is statistically significantly profitable in defeating a model than randomises between folding,raising and calling (3.371 sbb/h) as well as a model based on the game theoretic solution of a simplified Five Card Draw game (0.4124 sbb/h) [6] whilst being unable to defeat a model that always raises (-3.82807 sbb/h) as well as a model that always calls (-0.90916 sbb/h). Upon further investigation, it is revealed that many information sets in the simulation were reached where the information set itself was either not in the set of information sets in the ”No Suit” abstraction or were visited infrequently during the training process. This resulted in the ”No Suit” model playing a random action over all valid actions against the benchmark models. The absence of certain information sets in the ”No Suit” blueprint strategy occurs when the opponent performs an unexpected action such as drawing 0 cards, in which case playing a

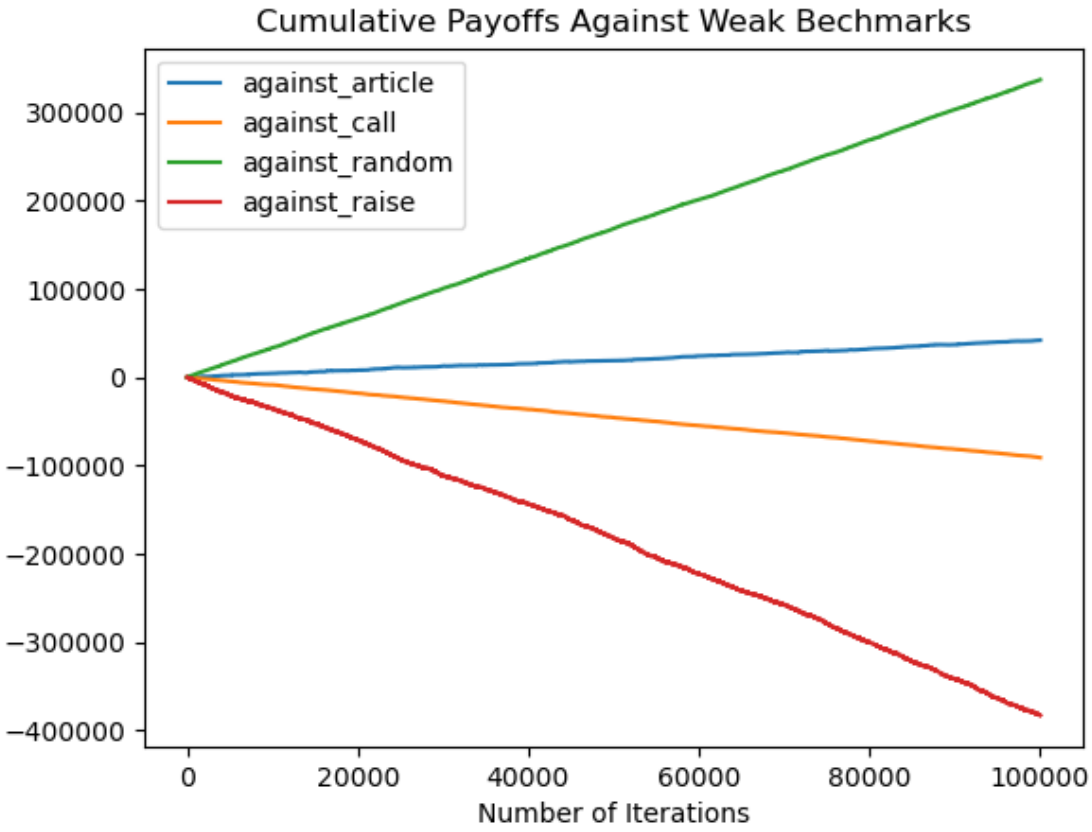

Figure 4: Cumulative payoff against Weak Benchmarks

random action performs badly as an opponent who does not discard any cards is likely to have a high hand. Moreover, during the training process of MCCFR, five cards are sampled out of the standard 52C5 possible five card combinations. However, this results in many high-value hands being sampled less frequently, thereby reducing the frequency of visits and updates to information sets which start from the high-value hand, in which case playing randomly would be sub-optimal as the model would be folding when it posses a high hand.

Despite the sub-optimal performance of the MCCFR model, there exists an opportunity to leverage domain knowledge and how the information set is defined to create a fallback strategy. In particular, certain information sets is either infrequently visited during gameplay or entirely absent from the blueprint strategy, which signals that either the opponent

possesses a strong hand or our hand is particularly strong. Adjustments should also be made for ”zero-betting” to prevent the model from folding when it does not cost the model anything to stay in the game by checking. Hence, we define a fallback strategy to bet if the current hand is strong in the event that the information set is not in the blueprint strategy or visited less than 1000 times during training.

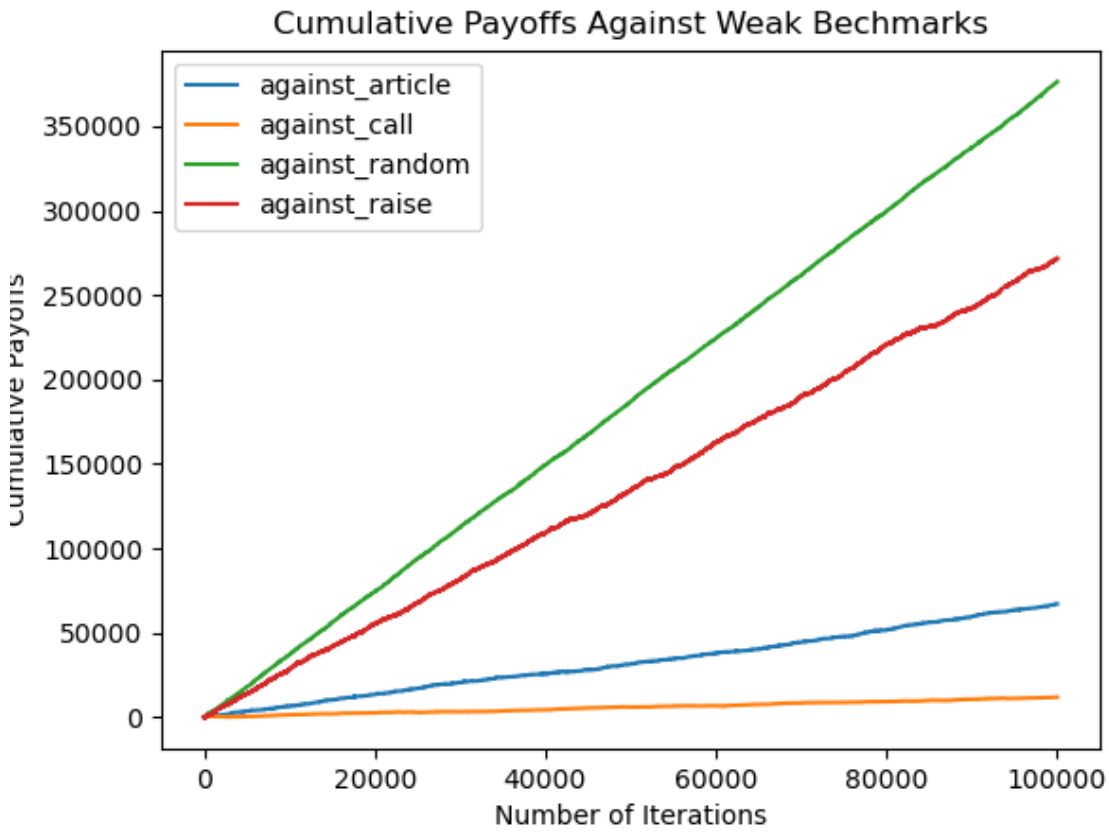

Figure 5: Cumulative payoff against Weak Benchmarks (Fallback)

The model with the fallback strategy now shows profitability against all weak benchmark strategies at the 99% confidence level with an average payoff of 0.67 against the simple game theoretic model, 0.118 against the always call model, 3.76 against the random model and 2.72 against the always raise model.

Aside from the simplified game-theoretic model, all the other weak benchmarks followed a fixed approach regardless of the specific cards being held. However, by incorporat-

ing card information into the opponent's model, it becomes evident that the MCCFR model with the fallback strategy is unsuccessful even against an honest and myopic player who plays according to the probability of currently holding highest hand, with the model losing 1.224 sbb/h in expectation. Upon closer examination, the under-performance of the MC-

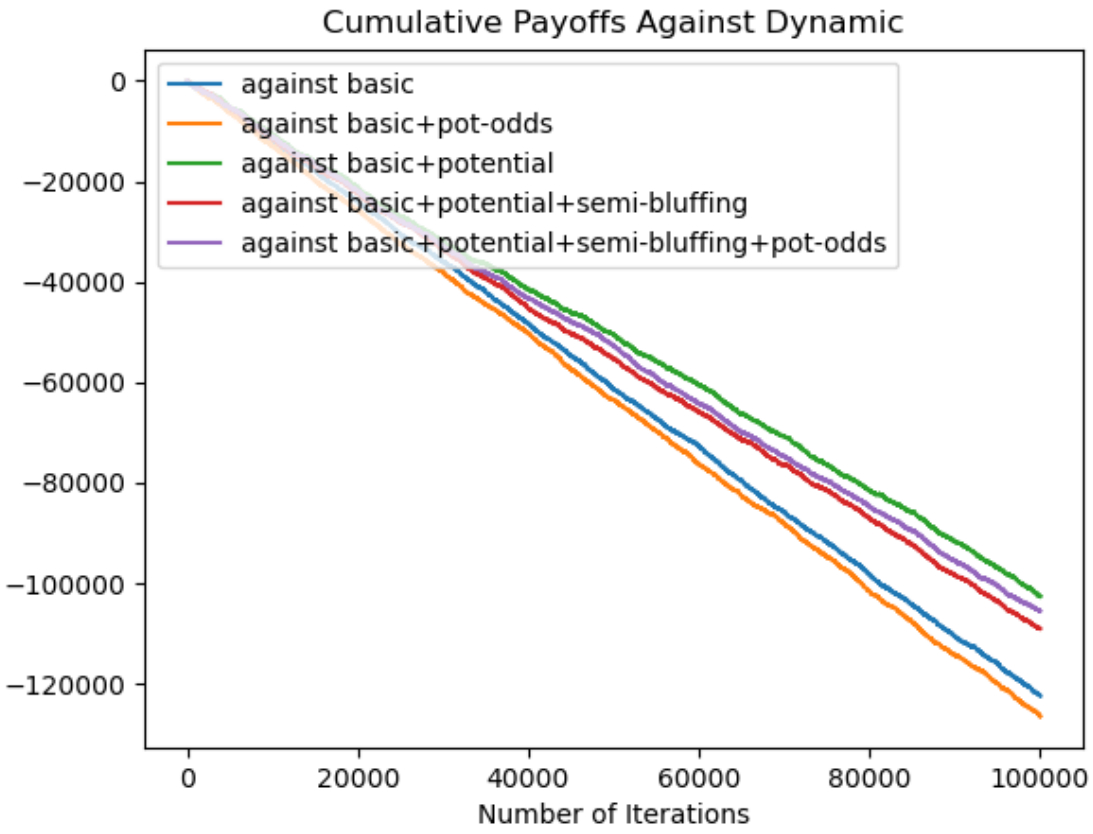

Figure 6: Cumulative payoff against Dynamic Heuristic Models

CFR model can be attributed to a tendency towards calling despite having weak hands. Out of the 52C5 possible poker hands, the blueprint strategy considers folding with a probability greater than 0.5 in only 8 instances. In contrast, the myopic player folds approximately 50% of the time across all possible hands. One plausible explanation for this systematic bias may be attributed to how the abstraction was defined. As a "no suit" information abstraction was employed, the model fails to distinguish between flush hands from high card hands. This oversight means that flush hands, which are high in value, are treated equivalently to high

card hands. Consequently, the model may be mislead into overestimating the strength of high card hands as the inherent strength of the flush hands is implicitly embedded within them, leading to an increased propensity towards calling.

Table 1: Performance of MCCFR against Basic (Myopic)

Hand	Win Percentage	Expected Payoff	p-value (Profitable)	p-value (Greater)
High Card	0.437	-1.128	1.0	1.0
One Pair	0.56	-1.33	1.0	1.0
Four of a Kind	1.0	10.539	0.0	0.9998
Three of a Kind	0.892	6.268	0.0	1.0
Full House	0.985	11.01	0.0	0.9999
Two Pairs	0.715	0.225	0.012	1.0
Straight	0.977	10.649	0.0	0.06997
Flush	0.484	-0.989	0.999	0.0031
Straight Flush	1.0	7	0.1	0.088

Table 2: Performance of Basic (Myopic) against MCCFR

Hand	Win Percentage	Expected Payoff	p-value (Profitable)
High Card	0.154	-0.96	1.0
One Pair	0.629	0.624	0.0
Four of a Kind	0.826	3.6939	0.0
Three of a Kind	0.929	8.476	0.0
Full House	0.986	12.79	0.0
Two Pairs	0.826	3.694	0.0
Straight	0.981	9.931	0.0
Flush	0.519	-2.509	0.999
Straight Flush	0.6	-2	0.687

An in-depth comparison between how MCCFR and the myopic model play certain hands reveal how the bias in playing high card hands affect the overall payoff in MCCFR. Despite having a higher win percentage as more hands are played, MCCFR generated a lower expected payoff from these hands as compared to the myopic model. In contrast, the myopic model is a lot more exploitative in playing medium hands such as one pair, two pairs and three of a kind, having achieved a higher win percentage and expected payoff per hand. Notably, MCCFR performs well on high rank hands such as straight, flush and straight flush. This performance is attributed partly to the fallback strategy discussed

earlier. Additionally, MCCFR's strategic framework also accounts for potential straight as well as potential straight flushes, which is not accounted for in the myopic model which solely considers the strength of its current hand.

The comparisons between cumulative payoffs against the dynamic heuristic models reveal that adding complexity to different versions of the program does not necessarily lead to improved performance against the MCCFR model. In particular, the additional information of whether a hand can be promoted after the draw appears to lead to a worsening of performance. However, it should be noted that the hand potential value being used are approximations rather than precise measurement. This limitation arises from the sheer magnitude of possible cases considered. For instance, consider a scenario involving a three of a kind hand where 2 cards are to be discarded. In such a case, there are potentially 47C2 additional hands to consider. Moreover, for each of these hands in the 47C2 subset, we must then account for the (52-5-2)C5 potential hands held by opponents, resulting in an exponentially growing number of possibilities to evaluate.

An alternative approach to address this complexity could involve using dummy variables in a regression framework where the outcome variable could have been the precise simulated potential value of each hand. Another solution is to pre-compute the value beforehand and use it as a lookup table during the simulations, which was done for the calculation of hand-strength values.

5 Discussions and Limitations

One significant limitation of the current MCCFR model lies in the immense branching factor inherent in the game of poker. This limitation is particularly pronounced when considering high-value hands, which are often less frequently visited during training due to their rarity in game-play. Additionally, the adoption of an action abstraction where all raises

are treated equally exacerbates this issue. These factors contribute to an off-tree problem, where an action taken by the opponent may not be in the solved blueprint strategy.

Previous solutions to off-tree problems include action translation, which involves mapping an opponents' action outside of the abstraction back to the model [21]. For example, suppose the action abstraction only includes bet sizes in increments of $100, translating an opponent's $530 bet would map it to a nearby bet size, such as $500. The effectiveness of such translation relies on the quality of the abstraction. However, relevant information may be overlooked, leading to decisions based on an incomplete understanding of the game state.

Alternatively, after solving the abstracted version of the game offline, a table lookup can be performed in real time to implement the strategy for the initial stages of the game [22]. However, strategies for endgames are recomputed in real-time using Bayes' rule to compute the distribution of players' private information entering the endgames. This assumes opponents will play according to the blueprint strategy, making it vulnerable to exploitation, known as "unsafe subgame solving."

Libratus' solution to the "unsafe subgame solving" problem in the game No Limit Texas Hold'Em was nested and safe subgame solving. In nested and safe subgame solving, a new strategy is computed to ensure opponents are not made better off relative to the blueprint strategy [23]. As the opponent plays an off-tree action, a new subgame is solved via subgame solving to form a new strategy. The new strategy is then merged with the original blueprint strategy to create an extended abstraction that contains the off-tree action.

6 Concluding Remarks

This paper has used a MCCFR algorithm on two different game abstractions in an attempt to approximate a Nash Equilibrium in the game of Five-Card Draw Poker. Despite the

MCCFR model's relative poor performance against the heuristic models, this study maintains that over time, the model will converge to a Nash equilibrium and ultimately be able to at least break even against all other models. This assertion is supported by observations of the model's behavioral strategy, which exhibit rational decision-making tendencies. For instance, during the drawing round, the model with a 4-card straight consistently discards the odd cards that do not contribute to forming a straight, demonstrating a logical approach to hand evaluation. Additionally, the model displays strategic complexity by incorporating elements of deception. For example, when faced with a situation where the 4-card straight is not promoted to a straight, the model adopts a mixed strategy between calling and raising, indicating an attempt to mislead the opponent. These insights suggest that while the MCCFR model may currently lag behind heuristic-based approaches due to computational and time constraints, its underlying strategic framework exhibits promising signs of rationality and adaptability, providing a solid foundation for continued refinement and eventual convergence to equilibrium. Additionally, even though the blueprint strategy derived from the MCCFR has yet to converge to a Nash equilibrium, it can still be insightful in informing optimal decision making when integrated with additional strategies, such as the fallback strategy previously discussed earlier. Furthermore, this paper has shown that periodic testing of the models against heuristic-based counterparts serves as a valuable benchmarking method to gauge progress towards convergence.

Appendices

Vanilla CFR algorithm

Algorithm 1 Counterfactual Regret Minimization (Vanilla CFR) [75]

Initialize regret: $\forall I, a \in A(I) : R(I, a) \leftarrow 0$
Initialize cumulative profile: $\forall I, a \in A(I) : s(I, a) \leftarrow 0$
Initialize current profile: $\forall I, a \in A(I) : \sigma(I, a) = 1/|A(I)|$
for $t \in \{1, 2, ..., T\}$ **do**
    for $i \in N$ **do**
        for $I \in \mathcal{I}_i$ **do**
            $\sigma_i(I, \cdot) \leftarrow$ RegretMatching($R(I, \cdot)$)
            for $a \in A(I)$ **do**
                $R(I, a) \leftarrow R(I, a) + v_i(I, \sigma_{(I \to a)}) - v_i(I, \sigma)$
                $s(I, a) \leftarrow s(I, a) + \pi_i^{\sigma}(I)\sigma_i(I, a)$
            end for
        end for
    end for
end for

Figure 7: Vanilla CFR [24]

Monte Carlo CFR algorithm

Algorithm 2 External Sampling [54]

if $n > 2$ **then require:** Update cumulative profile probability, $p \in (0, 1]$
Initialize regret: $\forall I, \forall a \in A(I) : R(I, a) \leftarrow 0$
Initialize cumulative profile: $\forall I, \forall a \in A(I) : s(I, a) \leftarrow 0$

WalkTree(history h, player i):
  if $h \in Z$ **then return** $u_i(h)$ **end if**
  if $P(h) = c$ **then** Sample action $a \sim \sigma_c(h, \cdot)$, **return** WalkTree($ha$, i) **end if**
  $I \leftarrow I(h)$, $\sigma(I, \cdot) \leftarrow$ RegretMatching($R(I, \cdot)$)
  if $P(h) \neq i$ **then**
    if $n = 2$ **then**
      for $a \in A(I)$ **do** $s(I, a) \leftarrow s(I, a) + \sigma(I, a)$ **end for**
    end if
    Sample action $a \sim \sigma(I, \cdot)$, **return** WalkTree(ha, i)
  end if
  for $a \in A(I)$ **do** $\tilde{v}(a) \leftarrow$ WalkTree(ha, i) **end for**
  for $a \in A(I)$ **do** $R(I, a) \leftarrow R(I, a) + \tilde{v}(a) - \sum_{a \in A(I)} \sigma(I, a)\tilde{v}(a)$ **end for**
  return $\sum_{a \in A(I)} \sigma(I, a)\tilde{v}(a)$

UpdateCumulativeProfile(history h, player i)
  if $P(h) = c$ **then** Sample action $a \sim \sigma_c(h, \cdot)$, UpdateCumulativeProfile(ha, i)
  else if $h \notin Z$ **and** $P(h) \neq i$ **then**
    for $a \in A(h)$ **do** UpdateCumulativeProfile(ha, i) **end for**
  else if $P(h) = i$ **then**
    $I \leftarrow I(h)$, $\sigma(I, \cdot) \leftarrow$ RegretMatching($R(I, \cdot)$)
    for $a \in A(I)$ **do** $s(I, a) \leftarrow s(I, a) + \sigma(I, a)$ **end for**
    Sample action $a \sim \sigma(I, \cdot)$, UpdateCumulativeProfile(ha, i)
  end if

Solve(iterations T):
  for $t \in \{1, 2, ..., T\}$ **do**
    for $i \in N$ **do** WalkTree($\emptyset$, i) **end for**
    if $n > 2$ **and** $Random(0, 1) < p$ **then**
      for $i \in N$ **do** UpdateCumulativeProfile($\emptyset$, i) **end for**
    end if
  end for

Figure 8: MCCFR [25]

Implementation Descriptions For Heuristic Models

Always Call

A weak benchmark strategy. Model always calls and draws to its current highest hand. For example, suppose the player currently has a pair with the hand 2♣2♦5♠8♥9♦, the model would retain the pair 2♣2♦ and dispose the rest of the cards 5♠8♥9♦.

Always Raise

A weak benchmark strategy. Model always raises and draws to its current highest hand.

Random

Model plays call, raise and fold with a uniform distribution and draws to its current highest hand.

Basic

Betting decisions are made solely on the immediate strength of the model's hand. That is, the probability that the current hand held by the model is the highest at the table. Hence, for the current hand that the model holds, we iterate through the remaining 47C5 cards in the first betting round and count the number of hands that are ahead, the number of hands that tied and the number of hands that are behind. Therefore, we define Hand Strength as:

$$HS = \frac{ahead + \frac{tied}{2}}{ahead + tied + behind}$$

Agent Modelling

In the prior hand strength calculation, each potential poker hand among the remaining 47C5 was weighted equally, however in practical poker scenarios this is highly improbable. For instance, conservative players are more inclined to fold weaker hands early in the game,

suggesting that hands that make it to the second round are likely to be strong. By leveraging information about the game state as well as the actions taken by the opponent under that state, a probability distribution can be calculated over the potential hands that an opponent may hold.

For example, suppose the model observes a pattern where after every call from the small blind, the big blind raises with probability 20%, checks with probability 70% and folds with a probability of 10%. In this scenario, if the opponent suddenly raises, the model can infer that the opponent is likely playing with the top 20% of their hand range. Utilizing this insight, we can reweigh each possible hand based on the inferred probability distribution, assigning higher weights to hands that are higher rank and penalising weaker ones.

Hence a threshold is defined as μ, which corresponds to the inferred strength of a hand. In this case, $\mu = 0.8$ as the player plays the top 20% of his hand. Iterating over each possible hand combination, the hand strength of each hand is compared against the μ accounting for some level of standard deviation. Hence if a hand has a high hand strength and μ is low, that hand will be weighed more highly relative to the others.

Potential

In addition to current hand strength also considers to probability of the hand improving during the drawing round and the probability of having the best hand after the draw. Consider the current high-card hand $2\clubsuit 4\spadesuit 5\spadesuit 6\spadesuit 7\spadesuit$, the probability of this hand being the strongest at the table is very low. However, there is a high probability of this hand improving by drawing a 3,8 or any spade cards. Hence, for each current hand, we calculate the probability of coming out ahead after the draw given that the current hand is behind the

opponent's before the draw. Therefore, we define Positive Potential as:

$$PPOT = \frac{T_{B,A} + \frac{T_{B,T}}{2} + \frac{T_{T,A}}{2}}{T_{B,S} + \frac{T_{T,S}}{2}}$$

where B=Behind, A=Ahead, T=Tied and S=Sum. Positive potential can then be combined with hand strength to represent an effective hand strength, defined as:

$$EHS = HS + (1 - HS) * PPOT$$

Semi-Bluffing

If in the first round, the model faced 0 bets and have a high enough positive potential to call a raise, will start raising itself. Then, in the second round, even without sufficient immediate hand-strength, will continue to raise to represent a strong hand.

Semi-bluffing is used when PPOT>=Pot_Odds_2, where Pot_Odds_2 is defined as:

$$Pot\_Odds\_2 = \frac{2 * BetSize}{(potsize + 4 * betsize) + 2 * betsize}$$

Calling with Pot Odds

Pot Odds represent the ratio of the amount in the pot against the cost to call.

$$Pot\_Odds = \frac{cost}{pot\_size + cost}$$

In the final round, this represents the required probability of winning. Hence, in order to call, we weight the cost of calling against the probability of winning and call when PPOT>=Pot_Odds in the first betting round and HS>=Pot_Odds in the second round.

Betting Strategies for Basic, Potential, Semi-Bluffing, Pot Odds

This paper defines thresholds to be used with different betting strategies. The strategy will Raise at most to 2 bets if HS/EHS>=0.85 and Call if HS/EHS>= 0.50. Below 0.5, semi-bluffing, calling with pot odds and calling with showdown odds will first be considered (if implemented) before choosing to fold.

Article Strategy

1. For all hands that do not contain the following, play check/fold: a pair or better, a 4 card flush (4 cards of the same suit), a 4 card open ended straight (suppose a hand with 2♠5♦6♦7♥8♥, disposing the 2, a player can expect to create a straight with probability 8/47 (4/47 from drawing a 4 and 4/47 from drawing a 9)), a royal cat-hoop (holding 3 cards towards a Royal Flush)

2. If holding 4 card flush as well as a pair, for example 2♠2♦6♦7♦8♦, choose to break up the pair if they are 10's or lower to pursue a flush hand.

3. If holding 4 card open ended straight as well as a pair, for example 2♠5♠6♦7♦8♦, choose to break up the pair if they are 9's or lower to pursue a straight.

4. If holding 4 card straight flush, for example 2♠5♦6♦7♦8♦, always raise.

5. Pre-Draw and Drawing Playing rules for Pairs:

    - for pairs QQ's to AA's, discard 3 cards and generally raise
    - for pair JJ's discard 3 cards and generally call
    - for pair TT's only play if at least 1 of 3 discarded cards is higher than a 10, otherwise fold. Always discard 3 cards.

- for pair 99's play if at least 2 of 3 discarded cards is higher than a 9 or one of the discarded cards is an Ace, otherwise fold.
- for pair 88's, play if :
  - all 3 discards are higher than 8 (discard 3 cards)
  - holding an Ace and an additional card higher than an 8 (discard 3 cards)
  - holding an Ace with 2 cards lower or equal to an 8 (discard 2 cards, holding Ace as Kicker)
- for 22's to 77's, only play if holding an Ace as kicker (discard 2 cards), else fold. Play call

6. For 4 card open ended straights and 4 card flushes, call.
7. For high or medium triplets, draw only 1 card $\frac{1}{3}$ of the time and stand pat $\frac{1}{3}$ of the time, and raise $\frac{1}{2}$ of the time when drawing at most one cards, never raise if not drawing. For low triplets, draw only 1 card.
8. For quads, always draw 1 card, bet minimum in every round until the end
9. For Royal Cat-Hop, play call
10. Post-Draw Betting Strategy:
    - bet/call with:
      - 4 card flush
      - 4 card straight
      - original quads
      - low three of a kind

- high 2-Pairs
- Royal Cat-Hop

- raise with:
  - newly made quads
  - newly made straights
  - newly made flush
  - original high/medium three of a kind if stand pat pre draw or half the time drawed 1 card to triplets
  - medium 2 pairs
  - low 2 pairs (33's to 77's)
- fold unimproved pairs below 88

11. low cards<8, medium cards = 8 until J, high cards>J

Files and Descriptions

The following is a description of each file in the zip folder:

- simulation_results - a directory storing the simulated results of each match
- article_bot.py - the simplified game-theoretic solution based model
- comparisons.py - used to generate the 50% folding and folding 8 instances with probability greater than 0.5 statistic on page 17
- external_sampling_highest_suit_mini.py - used to train "Highest Suit Mini" abstraction
- external_sampling_no_suit.py - used to train "No Suit" abstraction, line by line walkthrough of the pseudocode is provided
- five_card_node_es.py - used during CFR, it represents a game tree node
- hand_evaluator.py - poker hand evaluator, used during simulations and training to determine payoffs
- hand_strength_probabilities.pkl - precomputed hand strength values of each hand, assumes each hand is weighted equally (no agent-modelling)
- HandRanks.dat - used together with hand_evaluator.py for poker hand evaluation
- heuristics.py - contains all the intermediate heuristic models (basic, agent modelling, potential, semi-bluffing, calling with pot odds)
- random_bot.py - contains all weak benchmark strategy (always call, always raise, random)

- rule.py - defines the rules and model constraints for training CFR
- sample_abstraction_highest_suit_mini.json - contains a small sample of the information sets and their corresponding cumulative regrets and strategy for the ”Highest Suit Mini” abstraction. Full dataset is 16.28GB and is available upon request
- sample_abstraction_no_suit.json - contains a small sample of the information sets and their corresponding cumulative regrets and strategy for the ”No Suit” abstraction. Full dataset is 11.57GB and is available upon request
- Self Play Simulation Results.ipynb - used for the Self-Play Simulations results section for the paper
- simulate_hand_strength.py - used to precompute the hand strength of each possible hand, outputs hand_strength_probabilities.pkl
- Simulations against heuristic-based models.ipynb - used for the Simulations against heuristic-based models results section for the paper
- simulator.py - used to simulate poker matches between models

References

[1] M. Campbell, A. Hoane, and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370201001291

[2] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "Chinook the world man-machine checkers champion," *AI Magazine*, vol. 17, no. 1, p. 21, Mar. 1996. [Online]. Available: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1208

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, 2016. [Online]. Available: https://www.nature.com/articles/nature16961

[4] J. Bronowski, "The ascent of man episode 13: The long childhood," apr 1 1975.

[5] N. V. Findler, H. Klein, W. Gould, A. Kowal, and J. Menig, "Studies on decision making using the game of poker," in *IFIP Congress*, 1971. [Online]. Available: https://api.semanticscholar.org/CorpusID:1443183

[6] N. C. Ankeny, *Poker strategy: Winning with Game Theory*. Perigee books, 1982.

[7] N. V. Findler, "Studies in machine cognition using the game of poker," *Commun. ACM*, vol. 20, no. 4, p. 230–245, apr 1977. [Online]. Available: https://doi.org/10.1145/359461.363617

[8] D. Waterman, "Generalization learning techniques for automating the learning of heuristics," *Artificial Intelligence*, vol. 1, no. 1, pp. 121–170, 1970. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370270900044

[9] K. B. Korb, A. E. Nicholson, and N. Jitnah, "Bayesian poker," in *Conference on Uncertainty in Artificial Intelligence*, 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID:3263755

[10] D. R. Papp, "Dealing with imperfect information in poker," Master's thesis, CAN, 1998, uML Order No. GAXMQ–34401.

[11] J. Davidson, "Opponent modeling in poker: Learning and acting in a hostile and uncertain environment," 11 2011.

[12] O. Tammelin, N. Burch, M. B. Johanson, and M. Bowling, "Solving heads-up limit texas hold'em," in *International Joint Conference on Artificial Intelligence*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:6556647

[13] N. Brown and T. Sandholm, "Superhuman ai for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aao1733

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

[15] T. Sandholm, "Abstraction for solving large incomplete-information games," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, p. 4127–4131.

[16] G. Loomes and R. Sugden, "Regret theory: An alternative theory of rational choice under uncertainty," *The Economic Journal*, vol. 92, no. 368, pp. 805–824, 1982. [Online]. Available: http://www.jstor.org/stable/2232669

[17] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000. [Online]. Available: http://www.jstor.org/stable/2999445

[18] N. Brown and T. Sandholm, "Solving imperfect-information games via discounted regret minimization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 1829–1836, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4007

[19] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Poker as a testbed for ai research," in *Advances in Artificial Intelligence*, R. E. Mercer and E. Neufeld, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 228–238.

[20] Semi-surreal. [Online]. Available: https://www.deviantart.com/semi-surreal/journal/5-card-draw-poker-complete-strategy-5CD-trategy-751874378

[21] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron, "Abstraction pathologies in extensive games," in *Proceedings of the Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009, pp. 781–788.

[22] S. Ganzfried and T. Sandholm, "Endgame solving in large imperfect-information games," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '15. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2015, p. 37–45.

[23] N. Brown and T. Sandholm, "Safe and nested subgame solving for imperfect-information games," *CoRR*, vol. abs/1705.02955, 2017. [Online]. Available: http://arxiv.org/abs/1705.02955

[24] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2007. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf

[25] 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*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22. Curran Associates, Inc., 2009. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2009/file/00411460f7c92d2124a67ea0f4cb5f85-Paper.pdf