Research and Development of Automated Poker Playing Software (Counterfactual Regret Minimization)

by

Tamara Shmavonyan

Bachelor in Applied Mathematics and Informatics, Yerevan State University, 2016

A thesis submitted in partial satisfaction of
the requirements for the degree of
Master of Science

in

Computer and Information Science

in the

COLLEGE OF SCIENCE AND ENGINEERING

of the

AMERICAN UNIVERSITY OF ARMENIA

Supervisor: ______________________________

Signature: ________________ Date:____________

Committee Member: _______________________

Signature: ________________ Date:____________

Committee Member: _______________________

Signature: ________________ Date:____________

Committee Member: _______________________

Signature: ________________ Date:____________

Abstract

Research and development of automated poker playing software (Counterfactual Regret Minimization)

Author: Tamara Shmavonyan

Poker is an imperfect information game, i.e. the player possesses some information about the state of the game, but not all of it (each player can see his/her own cards and the public cards, but not the cards of the opponent). Inversely, chess is a perfect information game, as the knowledge about other players is available to all players (everyone sees all of the pieces on the board). This makes poker a substantially different game, making impossible to automatically play poker using conventional Machine Learning algorithms. University of Alberta is leading the research in the field of creating computer programs that play poker better than any human being. [1]

By being occasional poker players interested in game theory, we decided to analyze already existing algorithms, define problems and try to build an AI-based strategy that will play the most popular variation of the poker game, 2 player limit Texas Hold'em Poker, without human interaction.

One way of building a poker playing strategy is to build a program which is playing at Nash Equilibrium. One way of reaching the Equilibrium is using the algorithm called "Counterfactual Regret Minimization" (CFR). CFR is a self-play algorithm: it learns to play a game by repeatedly playing against itself. There are many modifications of this algorithm such as vanilla CFR, Chance Sampling (CS) CFR, Outcome Sampling (CS) CFR, Public Chance Sampling (PCS) CFR. Our goal was to implement the Pure CFR.

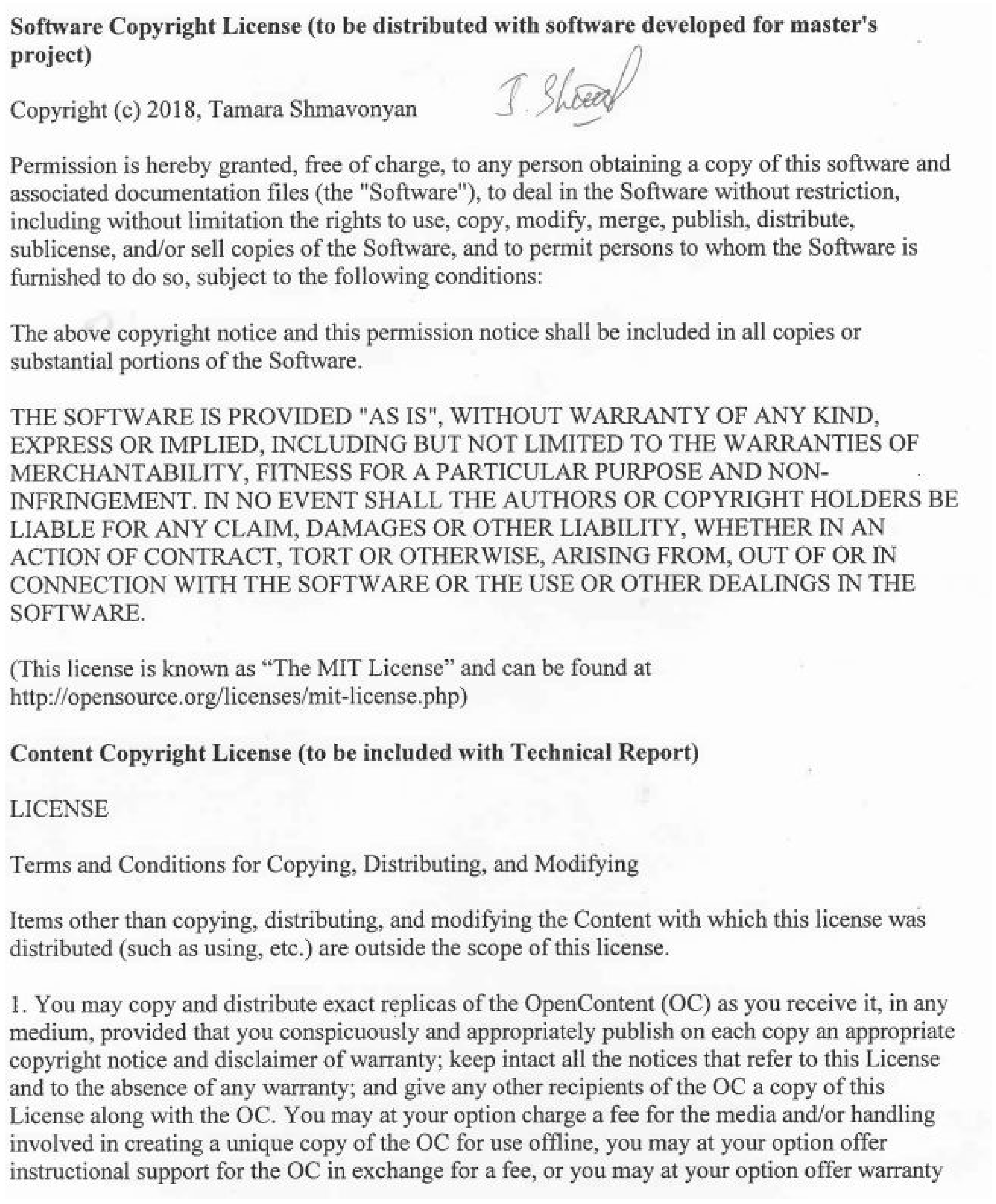
We implement the CFR algorithm to solve abstractions of limit Texas Hold'em with 10^{12} states. Game states are the number of possible sequences of actions by the players. In the poker setting, this would include all of the ways that the players' private and public cards can be dealt and all of the possible betting sequences [9]. This number allows us to compare a game against other games such as chess or backgammon, which have 10^{47} and 10^{20} distinct game states respectively.

We train the CFR algorithm by playing million rounds with itself and get the approximate Nash Equilibrium of the game (the probabilities of each action in the tree which leads to profit). The software is implemented using C++ programming language.

Most universities who do research in this area have their own implementations of different bots, but the source code is not available. This creates a barrier for new students trying to join the research. For that reason we decided to implement the "basic" poker playing algorithm as well as some AI algorithms, which can serve as a basis for any future research. After a few more optimizations, when we believe our software is mature enough, we will make it open-source, allowing other people to do their research starting not from scratch anymore.

Keywords: Nash Equilibrium, CFR in poker, cards isomorphism, bucketing, regret minimization, imperfect information games, game tree, backward induction.

Licenses for Software and Content



Software Copyright License (to be distributed with software developed for master's project)

Content Copyright License (to be included with Technical Report)

in exchange for a fee. You may not charge a fee for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY

4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU. SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY

Table of contents

1 Introduction

This paper has the following structure: next chapter (Theoretical Background) aims to acquaint the reader with the rules of Texas Hold'em and some terms from the game itself and game theory, which are necessary for further understanding the thesis. It also defines extensive-form games and Nash Equilibrium for incomplete information extensive-form games. The glossary in the last part of the chapter lets the reader quickly lookup the term associated with the game if needed. Chapter 3 covers the Counterfactual regret minimization algorithm which helps to find the Nash Equilibrium of the Poker game. The algorithm and the implementation are described in that chapter. The results and further plans regarding the project are described in chapter 4.

2 Theoretical Background

2.1 The game of Poker

Poker is the most popular card game which includes strategy, gambling and skill. Some of its properties like incomplete information and clearly defined rules make poker an interesting and challenging target for AI experiments. Poker has never been as widespread nor as quickly growing as it is right now. The rules are simple. The game is played with a standard deck of 52 playing cards. Four suits: hearts, spades, clubs, diamonds. Thirteen cards of each suit. The number of players can vary from two up to ten, playing clockwise around a table. Each player is dealt two cards, called hole cards. Five more public cards are subsequently revealed in four round gameplays. The rounds are called: preflop, flop, turn, river. On each round a player can choose among 5 actions: fold, check, call, bet, raise. You can look up the explanation of the actions in the glossary. Finally, after making a decision in each round the winner is determined by the best combination picked from his hole cards and 5 public cards. There are many game variations and we will discuss 2-player Limit Texas Hold'em Poker game variation in this paper.

2.1.1 Texas Hold’em Poker

In this game variation, designations of minimum and maximum bets are made. Texas Hold'Em game starts off the pot with two blind bets (bets made by players before they even see their cards). The small blind should be slightly less or equal to the minimum bet. The big blind should be twice as much as the small blind. Three cards are flipped after the first betting round (Flop), a fourth after the second betting round (Turn) and a fifth after the third betting round (River). After River round is finished, comes the showdown, where any remaining players will display their two cards and the player who has the best hand wins. If two or more players have the same hand then the pot is split equally between them.
There are three variations of Texas Hold ’em Poker: fixed-limit, pot-limit and no-limit. In Limit Texas Hold'em a maximum of four bets are allowed per player in a betting round - (1) bet, (2) raise, (3) re-raise and (4) cap. In No-Limit and Pot-Limit Texas Hold'em the number of times a player can raise is not limited. A player cannot raise themselves, i.e. if he was the last one to choose Raise action during a round, he would not get an option to re-raise. A player is declared All-In if they do not have enough chips to call [8].

2.1.2 Hand Strength

Here are the combinations sorted from strongest to weakest. The name and and example of the combination are provided. We will use A for Ace, K for King, Q for Queen, and J for Jack.

1. Straight Flush (5♥ 6♥ 7♥ 8♥ 9♥) - Five cards in a rank sequence of the same suit.
2. Four of a Kind (4♥ 4♠ 4♥ 4♣) - Four cards of the same rank and different suits.
3. Full House (Q♦ Q♠ 2♣ 2♠ 2♥) - Three cards of the same rank with a pair of another rank.
4. Flush (J♠ 2♠ 9♠ K♠ 7♠) - Five cards of the same suit.
5. Straight (7♠ 8♦ 9♦ 10♠ J♥) - Five cards in a rank sequence.
6. Three of a kind (3♠ 3♦ 3♥) - Three cards of the same rank.
7. Two Pair (A♥ A♦ 5♣ 5♠) - Two pair of cards.
8. Pair (8♠ 8♥) - A pair of cards of the same rank.
9. High card (K♠) - The highest ranked card in the current game[8].

2.1.3 Glossary

- **All-in:** Betting all of the chips a player has.
- **Big blind:** An obligatory bet on the initial round placed by the player sitting two places to the left of the dealer button, twice as big as the small blind.
- **Button:** The player in the position of the card dealer.
- **Flop:** Second round, the action of dealing the first three public cards. Sometimes refers to the cards themselves.
- **Hand:** The two cards that a player is holds. Sometimes refers to the best five cards of a player.
- **Kicker:** Also called a side card, is the card that may be used to break ties between the hands of the same rank.
- **Pot:** The overall amount of chips bet by all players in the current hand.
- **Pot odds:** The ratio of the actual pot and the amount of chips necessary to call a bet.
- **Pre-flop:** The first round in the game, before flop is dealt.
- **River:** The action of revealing the last fifth public card to the board, also refers to the card itself.
- **Small blind:** An obligatory bet on the initial round placed by the player to the left of the dealer button.
- **Turn:** The action of revealing the fourth public card to the board, also refers to the card itself.

2.2 Game Theory

Game theory studies strategic situations. Originally, it addressed zero-sum games, when for the first time Von Neumann and Morgenstern thoroughly characterized the optimal solutions to them in their book called *Theory of Games & Economic Behavior.* Zero-sum games are two-player games in which one player wins if and only if the other loses [Prajit K. Dutta]. Poker is a zero-sum game. Poker exemplifies a zero-sum game, because one wins exactly the amount one's opponent loses. In 1950, John Nash introduced the equilibrium concept which is the one of the most widely used concepts in modern game theory. We will define the Nash Equilibrium in the next subchapter [4].

2.2.1 Extensive Form Games

In game theory, a sequential game is defined as sequential moves of the players until reaching a terminal state. The next player has some information of the first's choice.

The extensive form is a way of describing a sequential game using a game tree. It’s simply a diagram that shows that choices are made at different points in time. N-person game in extensive form consists of a set N of n players, a rooted tree T called game tree, a partition of set of non-terminal nodes of T into n+1 subsets denoted P_0, P_1, P_2, …, P_n, the members of P_i are called the nodes of player i. Each terminal node of the game tree has an n-tuple of payoffs, meaning there is one payoff for each player at the end of every possible play.
Poker is an Extensive-Form Game. We’ll talk about the two-player poker game in this paper [5].

Games such as chess, backgammon, tic-tac-toe and Poker are examples of sequential extensive games. The size of the decision trees can vary according to game complexity, ranging from the small game tree of tic-tac-toe, to an immensely complex game tree of chess or poker so large that even computers cannot map it completely.

2.2.2 Nash Equilibrium

The *Nash Equilibrium* is a concept of game theory where the optimal outcome of a game is one where no player has an incentive to deviate from his chosen strategy after considering an opponent's choice. Overall, an individual can receive no incremental benefit from changing actions, assuming other players remain constant in their strategies. A game may have multiple

Nash Equilibria or none at all. If a game is *zero-sum* game, it must have an equilibrium. Computing this equilibrium for imperfect information games, where players have private, hidden information, is harder than solving perfect information games [4, 6].

In sequential games with perfect information, a subgame perfect equilibrium can be found by backward induction (will be described in algorithm implementation).

Poker is an imperfect information game as we don't know the cards of the opponent until the end of the game. That's why it's impossible to compute the exact Nash equilibrium during the game play.

3 Counterfactual Regret Minimization

3.1 Best strategy

Generally, the most profitable strategy to play is the strategy - Nash Equilibrium. That makes the opponent indifferent to our actions. If the opponent also plays with the Nash strategy, we become indifferent to his actions [4]. That allows both of us to maximize our benefits during the game play.

Let's denote by $\sigma_i \in \Sigma_i$ (i player's set of strategies) the i-th player's strategy. A strategy profile (strategies of all the players) is denoted by σ. σ_{-i} refers to all the strategies in σ except σ_i (i-th player's strategy). Our task is to find σ which is approximately the best strategy for all the players for the given card combination. By finding those σ strategy profile we find the Nash Equilibrium of the game.

3.2 Nash Equilibrium in imperfect information games

The term *regret* is used to describe how much more payoff would the player have over all the games so far if he deviated from his strategy.
If the regret is positive, the player would have done better if he had taken that action more often.
If it is negative, he would have done better by *not* taking that action at all.
Finding Nash Equilibrium of an imperfect information game is the same as *minimizing the regret* of the player for each action.
The correctness of the regret minimization formula that I use in my software, is proven in *Regret Minimization in Behaviorally-Constrained Zero-Sum Games (9 Nov 2017)* [2] (also in [3]). Let's include some notations from that paper.

- **N** - finite set of players. In our case N = 2.

- **H** - all the possible histories. Z ⊆ H is the terminal histories (when the game is over, i.e. when someone folds, or in the end of the round RIVER).

- **A(h)** = {a : (h, a) ∈ H} - actions available after non-terminal history h ∈ H.

- **P** is the player function (selects the next player). P(h) is the player who takes an action after the history h.

- For each player i ∈ N a partition Ψ_i of { h ∈ H : P(h) = i } with the property that A(h) = A(h_0) whenever h and h_0 are in the same member of the partition. For $I_i \in \Psi_i$ we denote by A(I_i) the set A(h) (all possible actions after history h) and by P(I_i) the player P(h) for any h ∈ I_i . I_i is the information partition of player i; a set Ii ∈ Ψ_i is an information set of player i.

- U_i is the **utility function** which shows the payoff of each action. It maps terminal states Z to the reals R . The non-terminal state payoffs are being computed recursively.

- $\pi^{\sigma}(h)$ be the probability of history h occurring if players choose actions according to σ

Approximate (imperfect game) “Nash equilibrium” theoretical definition for each player is:

$$u_1(\sigma) \geq \max\ u_1(\sigma'_1, \sigma_2) \text{ for all } \sigma'_1 \in \Sigma_i$$

$$u_2(\sigma) \geq \max\ u_2(\sigma_1, \sigma'_2) \text{ for all } \sigma'_2 \in \Sigma_i$$

Our aim is to find the approximate Nash Equilibrium. An approximation of a Nash equilibrium or ε-Nash equilibrium is a strategy profile σ where:

$$u_1(\sigma) + \varepsilon \geq \max\ u_1(\sigma'_1, \sigma_2) \text{ for all } \sigma'_1 \in \Sigma_i$$

$$u_2(\sigma) + \varepsilon \geq \max\ u_2(\sigma_1, \sigma'_2) \text{ for all } \sigma'_2 \in \Sigma_i$$

In order to compute the Nash Equilibrium of the imperfect information game we should compute the regret of each action and minimize the regret by changing the player's strategy profile.

3.3 Computing Counterfactual Regret

Let σ_i^t be the strategy used by the player i on the round t.
The average overall regret of player i at time T is:

$$R_i^T = \frac{1}{T} \max \sum_{t=1}^{T} \left(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t) \right) \quad \text{for all } \sigma_i^* \in \Sigma_i$$

The most effective and understandable way of doing that is to divide this problem into subproblems and solve each using backtracking. Solving larger abstractions yields better approximate Nash equilibria in the original game.
The task is to programatically decompose overall regret into a set of additive regret terms, which can be minimized independently. This regret minimization concept is called *counterfactual regret*. Counterfactual regret exploits the degree of incomplete information in an extensive game. The overall regret is bounded by the sum of counterfactual regret, and the counterfactual regret can be minimized at each information set independently [1].

$$u_i(\sigma, I) = \frac{\sum_{h \in I, h' \in Z} \pi_{-i}^{\sigma}(h)\, \pi^{\sigma}(h, h')\, u_i(h')}{\pi_{-i}^{\sigma}(I)} \quad (1)$$

The most effective Regret minimization formula by the time we started our observations:

$$R_i^T(I, a) = \frac{1}{T} \sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I) \left(u_i(\sigma^t|I \to a, I) - u_i(\sigma^t, I) \right),$$

$$R_i^{T,+}(I, a) = \max(R_i^T(I, a), 0) \text{ for time T + 1,}$$

$$\sigma_i^{T+1}(I)(a) = \frac{R_i^{T,+}(I,a)}{\sum_{a\in A(I)} R_i^{T,+}(I,a)}, \text{ if } \sum_{a\in A(I)} R_i^{T,+}(I,\ a) > 0$$

$$\sigma_i^{T+1}(I)(a) = \frac{1}{|A(I)|}, \text{ otherwise}$$

3.4 Regret Minimization algorithm description

In this chapter the description of the applied algorithm will be illustrated on the poker tree.

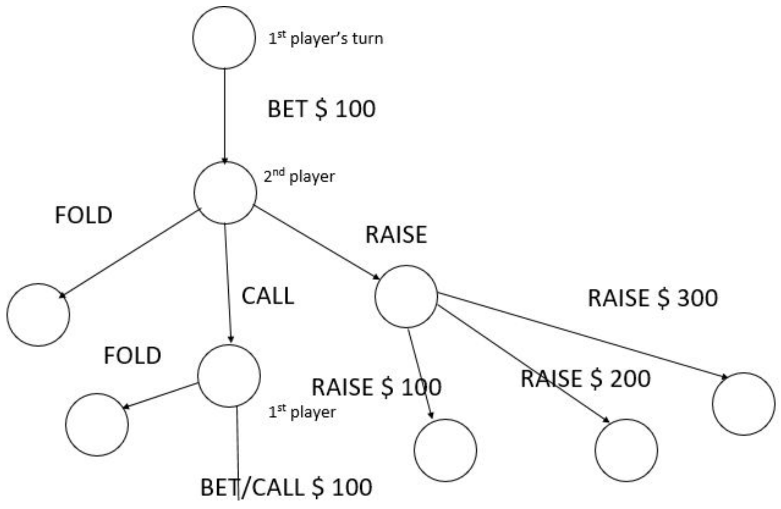
3.4.1 Open Pure CFR algorithm

Pure CFR is a time and memory-efficient variant of the Counterfactual Regret Minimization (CFR) algorithm for computing strategies for an extensive-form game with imperfect information.

The implementation of the algorithm requires a specific tree to be constructed.

The Poker tree initialization

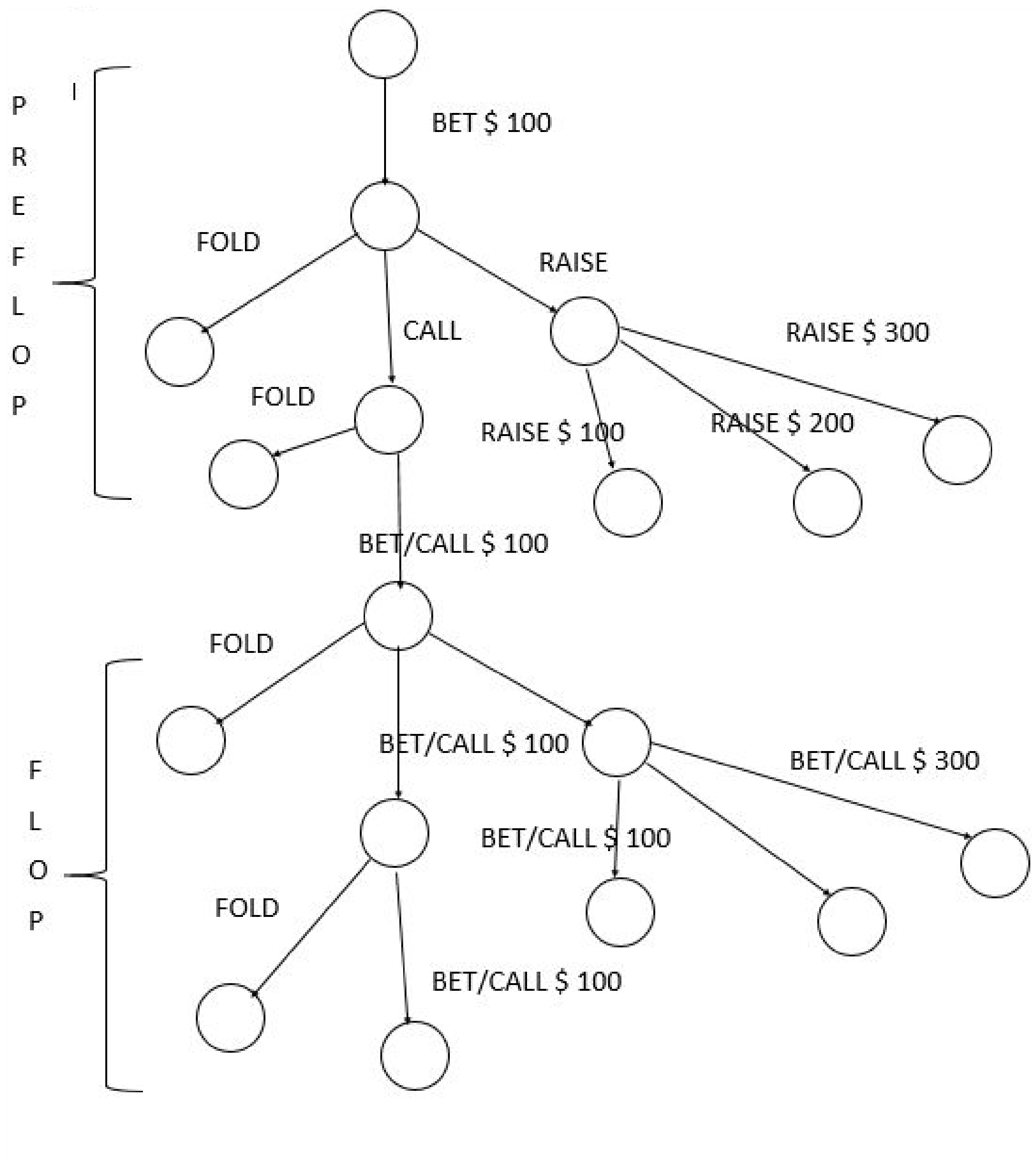
1. First we construct the poker tree with 4 iterations correspondingly for each round: preflop, flop, turn, river. Each iteration inserts 2 basic levels into the tree: one for the 1st player and the other for the 2nd one.



1st iteration for the round Preflop

2. Initializing tree with actions. Each action forms a separate subtree.
3. Each action has a fixed amount of money to bet. We recursively traverse the tree by increasing the amount of pot in each state (node). So in the terminal states (leaves) it will be obvious what amount of money each of the players win/lose.
4. Each action of the tree has an associated σ value. It shows the probability of choosing exactly that action by having the previous history. Initially those values are determined according to this formula:

$\sigma = \frac{1}{size\ of\ subtrees}$ if size of subtrees > 0, otherwise $\sigma = 0.5$.



2 levels of the Poker tree

Next we start the main part of the algorithm.

We generate 9 random cards, pass them to the bucketing algorithm, and get bucketing numbers for both 2 players and all the rounds. We get 4 pairs. 1st pair contains the bucketing numbers (numbers showing in which bucket does the card combinations belong, there may be more than one card combinations having the same bucket number if the cards are equivalent) of 1st and 2nd player in the round preflop. 2nd pair contains the bucketing numbers of 1st and 2nd player in the round flop, etc.

In order to train the AI and find the approximate Nash values, CFR algorithm simulates a two-player game by playing against itself. After each game, it computes and updates new regret values for all of its actions it just made. Then the algorithm recomputes playing strategy so that it takes actions with probabilities proportional to their positive regret. If an action has been good in the past, then the CFR algorithm will choose it more often in the future.

Let's describe more details of the algorithm. For each bucketing number there's an associated probability value which is kept in each node of the tree in form of a matrix. There is an associated pot for every state in the tree. And there is a payoff value for each state. For the base case the payoff is u_i = pot/2.
For the other states it is being calculated recursively according to the formula 1 defined above. The second payoff is being calculated by using the probability values which are kept in each node of the tree.
Next step is to calculate the regret at time T. If it is positive, we should update the probability value corresponding to the bucket number of the node. This update affects the whole tree, as all the nodes are being recursively updated in the next steps.

3.4.2 The implementation

The software is implemented using C++ programming language. Poker tree, nodes and other structures are implemented using C++ stl data structures (set, map, vector). The Regret Minimization algorithm is implemented using backward induction which is the main technique of solving (finding the Nash Equilibrium of the game) sequential games. Backward induction programmatically is done by recursion. First we initialize the terminal nodes of the tree (leaves), then we go up level by level and update the regret associated values.

4 Results

As a result, we have a software which computes the approximate Nash Equilibrium for 2 player limit Texas Hold'em Poker by using counterfactual regret minimization. With the use of bucketing and card isomorphism algorithms, this modification of CFR runs a lot faster compared to other modifications. A very efficient algorithm, Monte Carlo CFR, is faster for no limit poker for many players [10], but this variant performs at its best in 2 player limit poker. After a few optimizations and refactoring, we will put the code open source and invite everyone to contribute.

5 References

1. Martin Zinkevich, Michael Johanson, Michael Bowling, Carmelo Piccione. *Regret Minimization in Games with Incomplete Information.* 2007, University of Alberta.
2. Todd W. Neller, Marc Lanctot. *An Introduction to Counterfactual Regret Minimization.* July 9, 2013.
3. Gabriele Farina, Christian Kroer, Tuomas Sandholm. *Regret Minimization in Behaviorally-Constrained Zero-Sum Games.* 9 Nov 2017. *(*Endorsed article by Cornell University).
4. Dutta, Prajit K. *Strategies and games: theory and practice*. 1999, MIT, Cambridge: The MIT Press.
5. Hart, Sergiu. *Games in extensive and strategic form*s. 1992, The Hebrew University of Jerusalem.
6. Joel Watson-Strategy *An Introduction to Game Theory. 2013,* W. Norton Company
7. Lanctot Marc, Waugh Kevin, Zinkevich Martin. *Monte Carlo Sampling for Regret Minimization in Extensive Games.* 2009.
8. Limit Poker rules. https://www.pokerstars.com/poker/games/texas-holdem/#/limit, PokerStars company.
9. Michael Johanson. *Measuring the Size of Large No-Limit Poker Games.* February 26, 2013.
10. Richard Gibson, Neil Burch, Marc Lanctot, and Duane Szafron. *Efficient Monte Carlo Counterfactual Regret Minimization in Games with Many Player Actions.* December 2012, University of Alberta.