INDIAN INSTITUTE OF MANAGEMENT KOZHIKODE

Working Paper

IIMK/WPS/536/ITS/2022/07

March 2022

Games with strategic decision making: Poker algorithms and an application to Tic-tac-toe with accelerating modifications

Lija Chandran T V[1]
Mohammed Shahid Abdulla[2]

[1]Independent Researcher, Information Technology and Systems, Email: lijachandran@gmail.com

[2]Associate Professor, Information Technology and Systems, Indian Institute of Management, Kozhikode, IIMK Campus PO, Kunnamangalam, Kozhikode, Kerala 673570, India; Email: shahid@iimk.ac.in, Phone Number (+91) 495 – 2809254

Games with strategic decision making: Poker algorithms and an application to Tic-tac-toe with accelerating modifications

Abstract

This paper focuses on the games that aid in making strategic decisions, especially the game of *Poker* and different computer poker programs/bots. We have here an overview of some of the significant Poker computer programs which have solved or have *essentially solved* different variants of the game of Poker till date. As a part of comprehending the algorithms used to solve these games programmatically, we have experimented and attempted to implement Counterfactual Regret Minimization (CFR) equilibrium finding algorithm to solve the two player zero sum game *3x3 Tic-tac-toe*. A variant of the algorithm has a speedup upto a factor of 20. The source code of our CFR implementation is available to download from a public code repository.

1. Introduction

Games like *Chess*, *Poker*, *Bridge* etc claim to improve our decision making skill sets that aid in our personal and professional life. There are courses being offered in business schools like *Game of Poker* and *Game Theory* that teaches the strategy involved in playing these games, so that these strategies can be used when a real life business decision is to be made. Compared to a game like Poker, however, a game like *Chess* has a situation where the players can see the board and moves of each player: indeed these kinds of games where all the information about the game is known to all the players are categorized as perfect information games. Whereas, games like *Poker* are categorized as *imperfect information games* since it involves hidden cards and thus all the players do not get to know all the information about the game[1]. The decision making involved in these two categories of games are also dependent on these known and hidden information sets. Imperfect information game like *Poker* is considered more relevant when it comes to decision making in real life, as we are not given all the facts and information in real life when a decision is to be made[2]. Imperfect information games can be translated into many real world scenarios like negotiations, auctions, business strategy and security interactions[18].

Solving perfect and imperfect information games has been an interesting and challenging area of research for computer science and the emerging field of artificial intelligence also. Two player zero-sum games with perfect information like *Chess*, *Go*, *Tic-tac-toe* and

Checkers are some of the best studied games and are solved using exhaustive search techniques and algorithms. A computer program called *Chinook* developed by researchers at University of Alberta to play the game of *Checkers* became the first computer program to win the official world championship for a game of skill in 1994[3]. Optimal action or equilibrium in sequential perfect information games can be found using a method called backward induction[4]. Unlike perfect information games, solving imperfect information games is much more challenging - an early fact pointed out by computing and algorithms pioneer John von Neumann. More advanced algorithms like Counterfactual Regret Minimization (CFR), Excessive Gap Technique (EGT), Depth Limit etc. are used in solving the imperfect information games, wherever a solution is possible. The game of Poker has also been a part of this study by various universities and the poker bot called *Cepheus* developed by Computer Poker Research Group at the University of Alberta in 2015, became the first program to "essentially weakly solve" the *Heads-up Limit Texas Hold'em* poker[5]. Computer poker player *Pluribus* developed by Facebook's AI Lab and Carnegie Mellon University in 2019 plays the *No-limit Texas Hold'em poker* and became the first bot to beat humans in a complex multiplayer competition[6].

2. Poker and Managerial Skills

Poker is said to have an influence on the decision making skills of people, as Poker is an imperfect information game where the players have to make decisions based on both the information that is available and hidden from them. This more or less resembles the decision making involved in our real life and also business, with its attendant threat of multi-factor competition. Let's see some of the skills that product/project managers can learn from the game of poker[7][8][9]:

➢ Risk Management - As Poker is an imperfect information game, there is always a risk associated with every move you are going to make. A good player should always be aware when to *raise*, *fold* and when to go *all-in*. Likewise, managers should also have clarity on risks they might encounter while planning for the project or release.

➢ Observing Patterns - Poker involves multiple rounds of betting and each player in the table has their own style of playing. The players observe the moves of other players on the table and learn the pattern of each player's behavior. Reading pattern of each player can help us to make our own moves. In the same way, if the manager can deduce the pattern from the available metrics like performance of a product in the market, most and least used features in a product or competitors' moves can help to plan the next product release better.

- ➢ Resource Management - In poker, we know our private cards, community cards and the amount of money we have to play the hand. We should be able to calculate and plan while calling, raising the bet or going *all-in* in each round of betting. Similarly, a product manager should plan and allocate the resources (team members, fund, time, assets) for each sprint, release or feature accordingly.

- ➢ Agile thinking - A player has to make his move quickly when it is his turn, his decision in the current betting round is dependent on the moves of opponents, the community cards and the money left with him. If the previous player raises the bet amount, the player can choose to fold the cards, choose to continue or even raise the bet amount. A manager should be capable of making flexible decisions depending on the changing requirements and should be able to solve a problem quickly. These decisions must satisfy the optimal policy test for at least in the medium term.

- ➢ Emotional Intelligence - *Poker face* is something that the Poker players use to hide their exact emotions to deceive the opponents and it has become a quite popular phrase. Managers can learn from this to control their emotions and have a graceful attitude that leads to a healthy atmosphere in the workplace.

3. Planning Poker and Business Value Game

Planning Poker and Business Value Game/Poker are two estimation techniques used in an agile software development environment. Planning Poker is used while estimating a user story point for an upcoming sprint. The product owner will explain the user story to the technical team which consists of developers and testers. The team members can get their queries on the user story clarified during this meeting. Then the team members are asked to estimate the user story in terms of the overall efforts that need to be put in to implement the user story. Story points are relative estimations based on the complexity, risks, uncertainty and duration and points are represented using the Fibonacci series.

In Planning Poker, a deck of cards with Fibonacci sequence numbers (0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89) is given to all the team members and team members should pick the card with the number equivalent to the story point and place the card face-down on the table. All the team members will be asked to reveal their cards at the same time and then justify their estimates. This is done to avoid anchoring that happens in the traditional way of story point estimation where the estimations are said aloud[10].

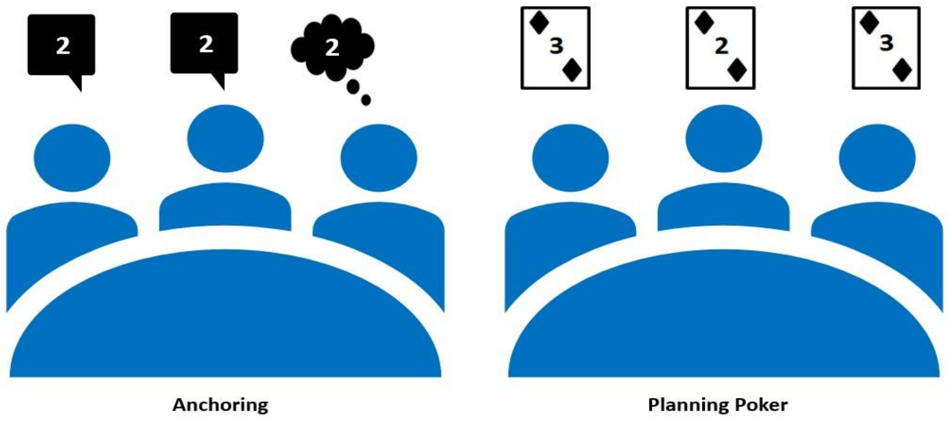


Figure 1: Estimation said aloud leading to anchoring vs planning poker

Business Value Poker or Business Value Game is similar to Planning Poker, but in value poker the relative value of a feature is estimated instead of the relative effort. This is a time-boxed meeting involving the product management board i.e, the product owner and all the stakeholders. Similar to Planning Poker, product owner will give an overview about the feature to the stakeholders and the stakeholders can get their queries clarified. The stakeholders will be given a deck of cards with the numbers 100, 200, 300, 500, 800, 1200, 2000 and 3000 and they have to estimate the value of the feature. Select the card which denotes the estimated value and keep it face down. Everyone has to show their cards simultaneously and then justify their estimations[11][12]. Both Planning Poker and Business value Poker use decks of cards with certain values and *showdown* happens after each round of estimation.

4. Poker Solvers and Poker Bots

Poker solvers are applications that help to find the optimal move for a given hand. These programs are mainly used for poker training. Various input parameters like preflop ranges of players, community cards, bet size, raise size, stack size and pot size have to be entered and an optimal solution will be recommended by the solvers accordingly. Real time assistance using a poker solver is not allowed in poker rooms or online poker[13]. Examples for Poker solvers are *GTO+*, *Simple PostFlop* and *PioSolver*.

Poker bots are computer programs that use artificial intelligence to play against human poker players. From 2004, there were many competitions held between computer poker programs and also between humans and bots. *Libratus* and *Pluribus* are examples of poker bots which beat top professional poker players in two-player heads up no limit Texas hold'em poker and six-player heads up no limit Texas hold'em poker respectively. The core of solving a game is finding the optimal solution or equilibrium in the game. As Poker is an imperfect information game it is difficult to find the optimal solution. The methods like Bayes Theorem, Nash Equilibrium, Monte Carlo simulation, Neural Networks etc have to be implemented to find the optimal solution[14]. Table1 summarizes the five significant Poker Bots till date that were able to beat human professional Poker players.

	Pluribus	Libratus	DeepStack	Cepheus	Polaris
Year Of Release	2019	2017	2016	2015	2008
Developed By	Facebook's AI Lab and Carnegie Mellon University	Carnegie Mellon University	Charles University, The Czech Technical University, University of Alberta	University of Alberta	University of Alberta
Poker Variant	No limit Texas Hold'em	Heads-up no-limit Texas Hold'em	Heads-up no-limit Texas Hold'em	Heads-up limit Texas Hold'em	Heads-up Limit Texas Hold'em
Equilibrium Finding Algorithm	Monte Carlo CFR, Linear CFR	Monte Carlo CFR with RBP (Regret Based Pruning), CFR+	Depth limited continual re-solving using CFR-D, Vanilla CFR, CFR+	CFR+	CFR
Information Set		After abstraction 10^{12}	After applying depth-limit 10^{17}	10^{14}	10^{12}
Computation Nodes used for solving	1 server, 64 core, 512 GB local disk	196 nodes, 128GB RAM per node, 28 core per node		200 nodes, 24 2.1GHz core per node, 32GB RAM, 1TB	

				local disk.	
Solving Duration	12,400 CPU core hours	25 million CPU core hours	~175 CPU core years	900 CPU core years	

Table 1: Poker Bots Summary

Polaris is the first computer poker program to win a meaningful match against top professional Poker players[15]. *Cepheus* is the first program to essentially weakly solve heads-up limit Texas hold'em poker[16]. *Deepstack* is the first computer program to beat professional players in heads-up no-limit Texas hold'em poker[17]. *Libratus* is another poker program which plays the heads-up no-limit Texas hold'em poker and the first to beat top poker human professionals[18]. The first bot to beat human players in a six-player no-limit Texas hold'em poker competition is *Pluribus*[19].

5. Equilibrium finding Algorithms

As mentioned in Table 1, the equilibrium state of a game is calculated using different algorithms. Choosing the right algorithm is dependent on the number of information sets or the nodes in the game tree. Figure 2 is an illustration of translating two different hands and actions of two players in the *Kuhn Poker* game into a game tree. Linear programming can be used to find equilibrium for two player zero-sum imperfect information games with around 10^8 nodes in the game tree. Algorithms like Counterfactual Regret Minimization (CFR) and Excessive Gap Technique (EGT) can be used to find equilibrium in large games (10^{14} nodes). Large games are converted into abstract games i.e all possible paths in the game won't be considered for finding the equilibrium[20]. The most common algorithm used to find the optimal solution in Poker is CFR and it's variants (refer *Table 1*).

CFR is an iterative algorithm which finds the optimal actions by playing against itself and improves the outcome with each iteration. Regret is the difference between the payoff of a possible action (for eg. in *Poker,* for the given hand a player can choose to fold or call or raise the bet amount) and the payoff (player can lose, win or retain money) of the action taken [21] ie.

$$regret = payoff(possible\ action) - payoff(action\ taken)$$

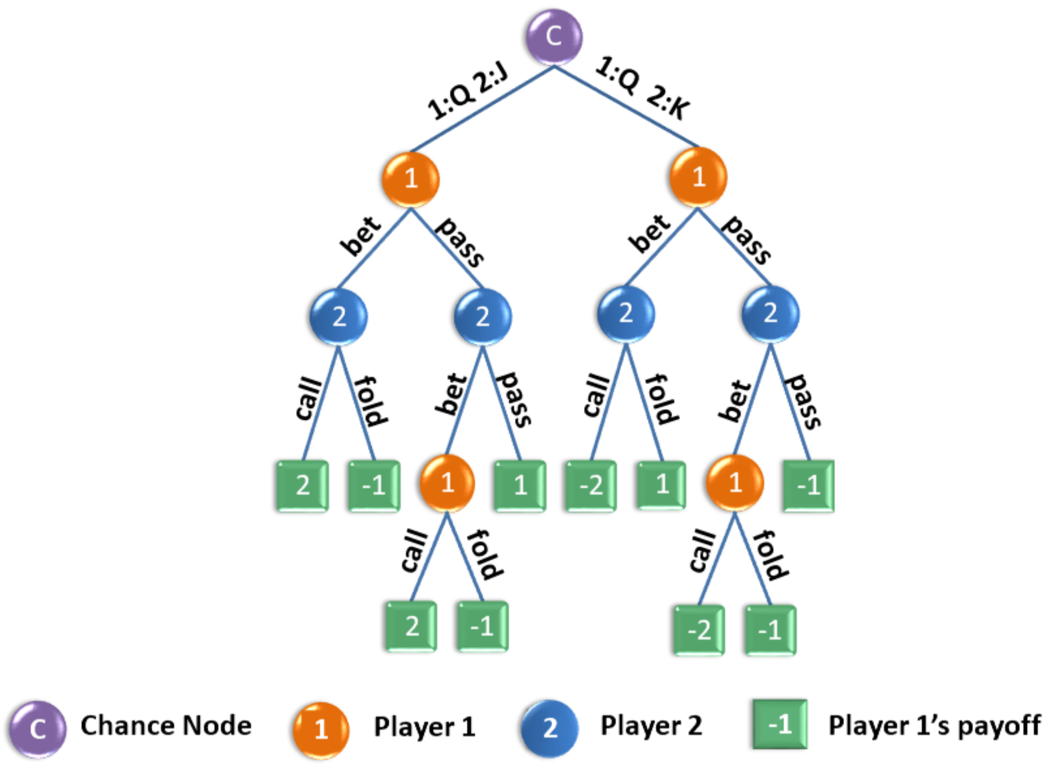
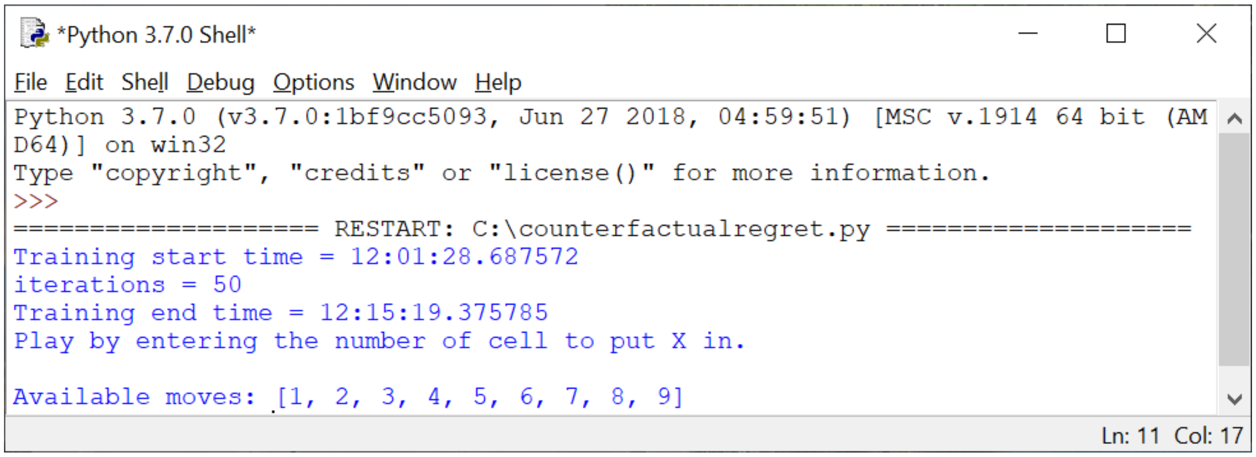


Figure 2: Kuhn Poker (3 card) game tree

The regret is calculated in each iteration of the game with different possible actions and at the end of each iteration the player's strategy is updated in such a way that the action with the higher regret value gets the higher probability[19].

5.1. Applying CFR for 3x3 Tic-Tac-Toe using recursion

An article written by a reinforcement learning enthusiast has shared a *Python* program to calculate the strategy for the *3x3 Tic-tac-toe* game using CFR[22]. Tic-tac-toe is a two player zero sum game which is usually played in 3x3 board, where each player has to put the symbols like 'X' or 'O'. The player who manages to put the symbol either horizontally, vertically or diagonally in each consecutive cell wins the game. The game could also end up in a tie, when both the players do not meet the winning criteria. The 3x3 tic-tac-toe game can have at most 3^9 = 19,683 states (total 9 cells in the board and 3 states for each cell), among which some states could be invalid. Also, the maximum number of games possible are 9! (362,880), however according to the sources; the exact number of games depending on various other factors results in 26,830 possible games[23].



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: C:\counterfactualregret.py ====================
Training start time = 12:01:28.687572
iterations = 50
Training end time = 12:15:19.375785
Play by entering the number of cell to put X in.

Available moves: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Figure 3: Output from the Python script

The program trains the algorithm to play *tic-tac-toe* for all valid information sets or the states on the tic-tac-toe board using a recursive method and stores the strategy in a dataset/dictionary. The program follows a top-down approach where the strategy is calculated for each state of the game from the first move on the tic-tac-toe board followed by the second move and so on. For each action/move on the tic-tac-toe board by the player, the payoff/utility is calculated recursively. The next state of the game and the probability of both the players taking that particular action is also passed to the function inorder to find the utility for a move. Then regret for each action/move is calculated by finding the difference between the utility calculated for the action in the previous iteration and utility calculated in the current iteration.

5.2. *Experiment 1*: building a 3x3 Tic-Tac-Toe using CFR and iteration vs recursion

As a part of understanding and learning the CFR algorithm and its implementation in a game, we also chose to solve the *3x3 Tic-tac-toe* game using the CFR algorithm. In our program, all the possible states on the 3x3 tic-tac-toe board for each player are validated and the invalid states are removed. Then the utility/payoff for both player's actions in all the valid states are calculated and stored in an array, also the regret value for each action in a state is calculated and stored in an array. The final strategy/policy to be taken in each state on the tic-tac-toe board after the completion of the training/iterations is also stored in an array. The algorithm has to refer to this array to make the optimal move for a state, when the human player plays against the algorithm.

Formulaic view of gameplay using CFR, would look as follows:

Step1) $\sigma^{T}(I, a) = \frac{(R^{T-1}(I,a))+}{\sum_{b\epsilon A(I)} (R^{t-1}(I,b))+}$

Step2) $\Delta R^{T}(I, a) = u_{T}(I, a) - \sum_{b\epsilon A(I)} \sigma^{T}(I, b)u_{T}(I, b)$

Where

$$u_{T}(I, a) = \sum_{b\epsilon A(I)} \sigma^{T}(I', b)u_{T}(I', b)$$

In the above formula:

$T \rightarrow$ Number of iterations/training

$I \rightarrow$ Information States on the tic-tac-toe board

$a \rightarrow$ actions/moves available at state I

$I' \rightarrow$ Next information state resulted from the move a

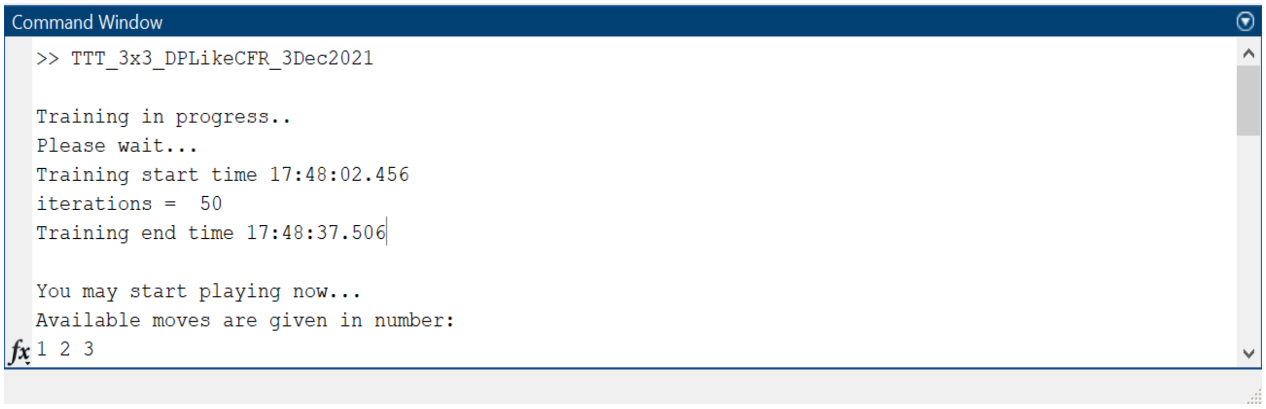
$b \rightarrow$ actions/moves available at state I'

Using a recursive method to solve the games with a large number of information sets will lead to more memory usage, insufficient RAM and slower execution[24]. This was observed by us, for instance, in MATLAB where the entire stream of recursion appears as a notification in the lower part of the edit screen - giving users an idea of how far the recursion has to proceed before termination. So, we solved the 3x3 Tic-tac-toe using an iterative method rather than a recursive method, and our code does not make use of each player's *reach* probability parameter to calculate the strategy. This was done in order to improve the time taken for the training by the algorithm, and also to create a useful codebase for the larger 5X5 TTT. The comparison of codes without using each player's reach probability parameter (*step 3*) and with using weighting parameter (*step 4*) can be represented as below:

Step3) $R^{T}(I, a) = \sum_{t=1}^{T} \Delta R^{T}(I, a)$

Step4) $R^{T}(I, a) = \sum_{t=1}^{T} P(I)\, \Delta R^{T}(I, a)$

The execution time required by our method to complete 50 iterations of training is approximately 35 seconds (refer *figure 4*) and the execution time required by the method mentioned in the related work (in section 5.1) to complete 50 iterations of training is approximately 831 seconds (refer *figure 3*): *a reduction of a factor of 20*. The Matlab script to solve the 3x3 Tic-tac-toe using CFR is uploaded in the repository[25].



```
>> TTT_3x3_DPLikeCFR_3Dec2021

Training in progress..
Please wait...
Training start time 17:48:02.456
iterations =   50
Training end time 17:48:37.506

You may start playing now...
Available moves are given in number:
fx 1 2 3
```

Figure 4: Training start and end time of Matlab CFR script

5.3. *Experiment 2*: 3x3 Tic-tac-toe using CFR and *divide-&-conquer*

We tried to solve 3x3 Tic-tac-toe by dividing and assigning the computation job to eight different 'assistants', which are in themselves computer programs. These assistants are responsible for the computation of regret values of the information sets related to; H1, H2, H3, V1, V2, V3, D1 and D2.

Here as shown in *figure 3*:
→ H1, H2, H3 are the three rows on the tic-tac-toe board
→ V1, V2, V3 are the three columns on the tic-tac-toe board
→ D1, D2 are the two diagonals on the tic-tac-toe board

For each move, the regret value is calculated by eight different assistants and the strategy will be calculated based on the cumulative regret value calculated from the inputs of these eight assistants. The move which has the highest regret value will be suggested as the next optimal move.

This source code is uploaded in the repository[25], though we have implemented this method to solve the 3x3 tic-tac-toe; we would call this a partially successful experiment as there are discrepancies in the strategies suggested by this divide and conquer method and the method implemented in section 5.2.

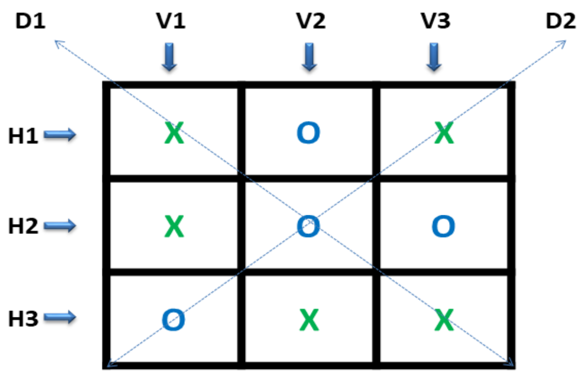


Figure 5: Eight assistants of 3x3 Tic-tac-toe divide and conquer method

For eg:

When the state on the tic-tac-toe board is "X - - - O - - - -" and it is Player 1's turn to make the next move this method of divide and conquer recommends the following strategy ie to put the symbol in the 3rd cell:

"0 0 1 0 0 0 0 0 0"

Whereas the method mentioned in section 5.2 recommends the below strategy ie put the symbol either on the 2nd cell or 4th cell:

"0 0.5000 0 0.5000 0 0 0 0 0"

5.4. *Experiment 3*: 3x3 Tic-tac-toe using CFR and *pseudo*- divide-&-conquer

In the experiments mentioned in the sections 5.2 and 5.3, the algorithm does the regret and strategy calculation as a part of training and creates a dataset with the strategy for all possible moves in the *3x3 tic-tac-toe* game. The algorithm has to refer to this dataset and return the strategy with maximum utility whenever a human plays against the algorithm. But, in this method; each state of *3x3 tic-tac-toe* game is divided into 8 subgames ie. I1, I2, I3, I4, I5, I6, I7, I8 and the strategy is calculated in real time when the human player makes a move. As a single move can change the states of multiple subgames (For eg: putting the symbol 'X' on the first cell will affect the state of subgames I1, I4 and I7), the strategy will be calculated for all the eight subgames and the strategy with the maximum utility (i.e strategy with the minimum utility for the opponent) will be considered by the algorithm as the next move.

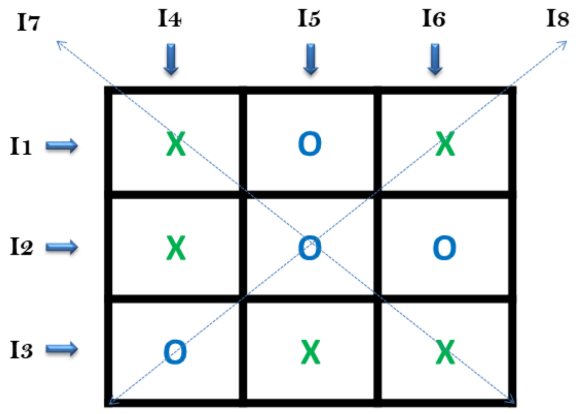


Figure 6: Eight sub-games of 3x3 Tic-tac-toe pseudo divide and conquer method

Formulaic representation of pseudo divide and conquer method is as follows:

Step 1) $$\sigma_t^l(I_l, a_l) = \frac{(R_{t-1}^l(I_l, a_l))+}{\sum_{b_l \epsilon A(I_l)} (R_{t-1}^l(I_l, b_l))+}$$

Step 2) $$\Delta R_t^l(I_l, a_l) = u_t^l(I_l, a_l) - \sum_{b_l \epsilon A(I_l)} \sigma_t^l(I_l, b_l)\, u_t^l(I_l, b_l)$$

Where

$$u_t^l(I_l, a_l) = min_{I'_m = S(I_m, a_l)} \sum_{b_m \epsilon A(I'_m)} \sigma_t^m(I'_m, b_m) u_t^m(I'_m, b_m)$$

Step 3) $$R_t^l(I_l, a_l) = \sum_{t=1}^{T} \Delta R_t^l(I_l, a_l)$$

Step 4) $$I_l^* = argmax_m \sum_{b_m \epsilon A(I_m)} \sigma_T^m(I_m, b_m)\, u_T^m(I_m, b_m)$$

Step 5) Thus $\sigma_T^*(I_l^*, b_l)$ is the maximum subgame strategy for a given state I

In the above formula:

→ I is state on the tic-tac-toe board, l=1 to 8 subgames $\left\{I_l\right\}_{l=1}^{8}$, a is the action

→ T is the number of iterations/training

→ I'_m are the subgame states affected by the action a_l

→ I_l is one of the I_m

→I_l^* is the subgame when action b is taken by the player in the state I

6. *Directions for future work:* solving 5x5 Tic-tac-toe

Solving the *5x5 Tic-tac-toe*, where the maximum number of states will be 3^{25} - more than 847 Billion states using CFR is a tedious task, as we discovered for ourselves when we engaged Amazon Web Services cloud computers to attempt to run the MATLAB code. Iterating through this huge information set and finding an optimal move is constrained by time and resources, we are recommending the following methods to solve the *5x5 Tic-tac-toe game* using CFR:

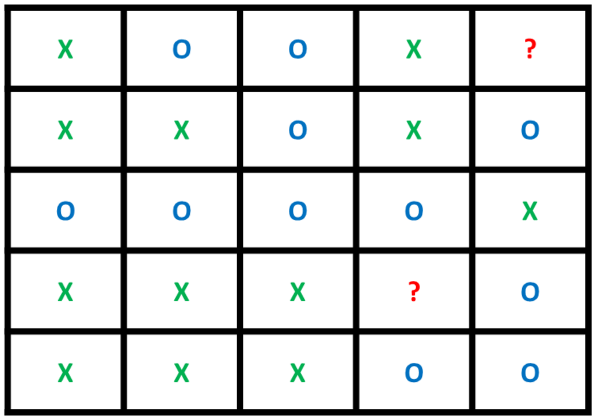
Method 1

Instead of iterating through all possible states in the *5x5 tic-tac-toe* board, solve only the endgame states using the same method used in the source code mentioned in section 5.2. For example, the endgame of *5x5 tic-tac-toe* would look similar to a state represented in Figure 4; where only two moves are left to finish the game. If we consider the state of each cell in the *5x5 tic-tac-toe* board as a bit; when combined it will form a 25 bit word. Eg: XOOX-XXOXOOOOOXOOO-OXXXOO. In an endgame, these two blanks can be in any two cells on the board and the number of all the possible combinations of moves/states that leads to two blank cells on the board can be calculated using the following formula[26][27]:

$$25!/(2!*23!) \; * \; 23!/(12!*11!) \; + \; 23!/(11!*12!)$$

In the above formula:

- ➢ 25 is the total number of cells in the 5x5 tic-tac-toe board
- ➢ 2 is the number of blank cells in the endgame of 5x5 tic-tac-toe board
- ➢ 23 is the number of cells with symbols ‘X’ or ‘O’ in the endgame
- ➢ 12 and 11 are the number of symbols ‘X’ or ‘O’ in the endgame



X	O	O	X	?
X	X	O	X	O
O	O	O	O	X
X	X	X	?	O
X	X	X	O	O

Figure 7: A sample 5x5 tic-tac-toe endgame state

Method 2

Solving 5x5 tic-tac-toe using the divide and conquer method, similar to the method mentioned in section 5.3. The number of assistants will be 12; for 5 rows, 5 columns and 2 diagonals.

Method 3

The pseudo divide and conquer method mentioned in section 5.4, where the game can be divided into 12 subgames and the strategy can be calculated in the real time by finding the least favorable move for the opponent/human player.

7. Conclusion

Games like Poker variants and tic-tac-toe can be solved programmatically using the Counterfactual Regret Minimization algorithm. CFR algorithm can be implemented using different intuitive modifications - even if one doesn't employ heuristics - and this can lead to the algorithm computing the regret values and strategies more efficiently.

Acknowledgement

We thank alumni of Indian Institute of Management Kozhikode; Mr. Arijit Goswami (Business Development Manager, Capgemini) and Mr. Suresh Gutti (Consultant, Optum) for their valuable inputs on Poker towards this paper.

References

[1] *Perfect information*. (2003, September 19). Wikipedia. https://en.wikipedia.org/wiki/Perfect_information

[2] Holloway, D. (2016, July 31). *Mind games - Strategic thinking in chess and poker*. LinkedIn. https://www.linkedin.com/pulse/mind-games-strategic-thinking-chess-poker-daniel-holloway/

[3] Billings, D. (1995, August 30). *Computer Poker*. ERA:Education and Research Archive. https://era.library.ualberta.ca/items/a92e1a72-bf40-4ae8-b3fb-eb34723367be/view/5359642b-2e6d-43d3-81dc-9f36c321f752/TR95-22.pdf

[4] *Backward induction*. (2005, June 17). Wikipedia. https://en.wikipedia.org/wiki/Backward_induction

[5] *Cepheus (poker bot)*. (2015, January 17). Wikipedia. https://en.wikipedia.org/wiki/Cepheus_(poker_bot)

[6] *Pluribus (poker bot)*. (2019, November 27). Wikipedia. https://en.wikipedia.org/wiki/Pluribus_(poker_bot)

[7] Chu, B. (2016, January 3). *Product management is a lot like playing poker*. Medium. https://medium.com/@brandonmchu/product-management-is-a-lot-like-playing-poker-a1f3b00dcbe

[8] Nanji, L. (2018, June 5). *How poker helps you win at product management*. Medium. https://medium.com/product-to-product/how-poker-helps-you-win-at-product-management-7cde2521050d

[9] *What the art of poker can teach us about business management*. (2020, March 27). BOSS Magazine. https://thebossmagazine.com/poker-business-management/

[10] *Planning poker*. (2007, June 27). Wikipedia. https://en.wikipedia.org/wiki/Planning_poker

[11] Dahlgren, M. (2017, April 3). *Determining value using value poker*. Medium. https://medium.com/@MagnusDahlgren/determining-value-using-value-poker-980cb2a1e432

[12] *Business value game*. (2021, July 6). agile42. https://www.agile42.com/en/agile-community/agile-info-center/business-value-game/

[13] *What is real-time assistance (RTA)? Is it legal?* (2020, October 2). PokerNews. https://www.pokernews.com/news/2020/10/what-is-meant-by-real-time-assistance-rta-38054.htm

[14] *Computer poker player*. (2004, March 3). Wikipedia. https://en.wikipedia.org/wiki/Computer_poker_player

[15] Bowling, M. et al. (2009, May). *A Demonstration of the Polaris Poker System*. CiteSeerX. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.6204&rep=rep1&type=pdf

[16] Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2017). *Heads-up limit hold'em poker is solved*. Communications of the ACM, 60(11), 81-88. https://doi.org/10.1145/3131284

[17] Moravčík, M., et al. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 356(6337), 508-513. https://doi.org/10.1126/science.aam6960

[18] Brown, N., & Sandholm, T. (2017). *Libratus: The superhuman AI for no-limit poker*. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. https://doi.org/10.24963/ijcai.2017/772

[19] Brown, N., & Sandholm, T. (2019). *Superhuman AI for multiplayer poker*. Science, 365(6456), 885-890. https://doi.org/10.1126/science.aay2400

[20] Ganzfried, S., & Sandholm, T. (2015, May 4). *Endgame Solving in Large Imperfect-Information Games*. In the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). https://www.cs.cmu.edu/~sandholm/endgame.aamas15.fromACM.pdf

[21] SALLOUM, Z. (2020, May 24). *Introduction to regret in reinforcement learning.* Medium. https://towardsdatascience.com/introduction-to-regret-in-reinforcement-learning-f5b4a28953cd

[22] SALLOUM, Z. (2020, December 28). *Counterfactual regret minimization*. Medium. https://towardsdatascience.com/counterfactual-regret-minimization-ff4204bf4205

[23] *Game complexity*. (2003, September 13). Wikipedia. https://en.wikipedia.org/wiki/Game_complexity

[24] Dale, W. (2018, May 9). *Recursion: The pros and cons*. Medium. https://medium.com/@williambdale/recursion-the-pros-and-cons-76d32d75973a

[25] https://github.com/lijachandran/3x3Tic-Tac-Toe_with_CFR

[26] *How many bit strings of length 8 contain exactly 4 zeros?* (2018, May 6). Quora. https://www.quora.com/How-many-bit-strings-of-length-8-contain-exactly-4-zeros

[27] *Combination*. (2001, July 26). Wikipedia. https://en.wikipedia.org/wiki/Combination

Research Office

Indian Institute of Management Kozhikode

IIMK Campus P. O.,

Kozhikode, Kerala, India,

PIN - 673 570

Phone: +91-495-2809237/ 238

Email: research@iimk.ac.in

Web: https://iimk.ac.in/faculty/publicationmenu.php