**Name: Manva Patel**
**Student ID: 20815074**
**Email: m329pate@uwaterloo.ca**

## Assignment 4 Question 1:

The agent starts off by cooperating. Then it checks the entire history of the other two players in order to predict their next move. If the number of times the player has cooperated is more than the number of times he/she has deflected, then the agent predicts that the player will most likely cooperate in the next round. And if the number of times the agent has deflected is more than the number of times the agent has cooperated, then the agent predicts that the player will mostly deflect in the next move. Whereas, if the number of times the player has cooperated is equal to the number of times the player has deflected, then the agent considers the worst case scenario and predicts that the player will deflect in the next round. By checking the history, the agent can predict properly even in the case of noise. Instead of checking just the previous move of players, which can be distorted due to noise, the agent checks all the previous moves made in the match and finds the tendency of the player.

After predicting the next move of the players, the agent looks at their combination. There are four possible combinations: CC, CD, DC, DD. Here C stands for cooperation and D stands for deflection. If both the players tend to cooperate (CC), then the agent will also cooperate. If one of the players deflects (CD or DC), then the agent will deflect in the next move. Similarly, if both the agents deflect (DD), then the agent will also deflect in the next move.

It is in the benefit of all the players to mutually cooperate in the long term. This would lead to equilibrium and all players will get high scores. So, if the two players cooperate, then our agent would also choose to cooperate with them to achieve a stable state. If all the players follow this policy, then they would all try to cooperate in the beginning and this would lead to equilibria, except if noise changes the outcome of one of the players. The change in outcome due to noise can be handled because the agent checks the entire history of the players. But if there is a mean player, then he/she would deflect even if the others cooperate and would get very high scores. So, the agent chooses to deflect if one of the other players deflects. This leads to a better score than if the agent cooperates, and also minimizes the score of the mean player. Also, if both the players deflect, then the agent has to deflect.

This strategy performs well if all the agents strive to cooperate. But if there is a mean player, then also this strategy performs well. Also, it takes into consideration the noise factor. So, I believe that it is the best option.

## Assignment 4 Question 2:

### (a) What are the motivations for this work? (5 pts)

Game playing has been one of the most important domains of AI. Games are one of the first areas where AI was used. This is because games provide a perfect platform to pit humans against computers. Also, games are an efficient way to measure the capacity and performance of AI systems. Games have definite rules, steps and outcomes. It is also very easy to evaluate the performance of an agent in games. Hence, there has been much interest in developing AI agents for games like chess, go, poker, etc. which can win against the experts of that game. But most of these agents are good at playing two-player games. They have not been able to beat humans in multi-player games. This is because it is very difficult to obtain Nash equilibrium in multi-player games. Further, even if the Nash equilibrium can be found, then it can be possible that different players calculate different Nash equilibria and act accordingly. If this occurs, then the result might not be a Nash Equilibrium. So, it is very difficult to use Nash Equilibrium for solving multi-player games.

Poker has always interested the AI community because it is a stochastic, zero-sum game. Systems have been made which can play poker, but they are only limited to two-player poker. But in real life, the game of poker usually has multiple players. So, the authors of this paper introduce a model which can play a six-player Texas hold'em poker without calculating the Nash equilibrium. This model was tested against human professional players and it managed to defeat them. The ability to win in multi-player poker opened up new possibilities where AI systems could play and win in multi-player, stochastic environments.

Also, this model is based on reinforcement learning. So, it requires no data for learning. It searches for an optimal strategy through self-play and search.

**(b) What is the proposed solution? (5 pts)**

In the recent years, many AI systems have been proposed which could solve more and more complex games of poker. But all these systems worked only in the two-player setting. Whereas traditionally, poker has always been played with more than 2 players. So the authors of this paper proposed an AI model called Pluribus which can successfully play 6-player poker and defeat its human opponents.

The systems which play 2-player games such as poker, chess, go, etc. usually use a Nash equilibrium strategy. This is a strategy where no player could improve their score by following some other strategy. A player following Nash equilibrium will never lose. There are algorithms for finding the Nash equilibrium in two-player zero-sum games. But finding the Nash equilibrium in games having more than two players is very complex and difficult. And even if Nash equilibrium is computed, it might be possible that different players compute different Nash equilibria. If different players follow different strategies, then the result would not be Nash equilibria.

So the authors of this paper have created a system which does not follow a specific theoretic game-theory solution. Instead they aim to create a system which can consistently defeat humans. Pluribus consists of the following steps:

- Before forming a strategy using self-play, Pluribus first performs two types of abstractions to reduce the number of decision points. The first is action abstraction, where it reduces the number of actions which are to be taken into consideration. And the second is information abstraction, in which the decision points which give similar information are treated as equal points. This reduces the complexity of the game.

- Next, Pluribus computes it's blueprint strategy using a type of CFR(counterfactual regret minimization). CFR is an algorithm which learns to play a game by repeatedly playing against itself. Here they are using Monte Carlo CFR. In each iteration, one player will be the traverser whose strategy is to be changed. The MCCFR first simulates the game on the basis of the current strategy of the players. Then the system checks the other actions it could have taken at all the decision points. The system then computes the difference between what it would have gotten if it had taken a particular action and what it actually received currently. This difference is known as the regret. At the end of the iteration, the system updates its strategy by choosing actions which have higher positive regret. As the regret is added up in each iteration instead of being replaced, the random strategy which the player plays in the first iteration effects it's strategies in the future. If the number of iterations played are T, then the effect decreases at the rate of 1/T. But to speed up this rate, the authors of the paper use a form of CFR called Linear CFR which assigns a weight of T to the regret contribution at iteration T. So the rate at which the effect of the first iteration decreases now becomes 2/T(T+1).

- After forming the blueprint strategy, Pluribus plays with a human player. It follows this blueprint strategy only in the first round, after that it uses search to obtain a better strategy. But in imperfect-information games, the value of the leaf nodes depends on the strategy which is chosen in the subgame. So the value of the leaf node is not fixed. So Pluribus assumes that each player may choose one of 4 different strategies when a leaf node is reached. Due to this, the searcher would find a balanced strategy. For forming a strategy in the subgame, Pluribus either uses Monte Carlo Linear CFR or Linear CFR. Further, to remain unpredictable, Pluribus first computes the strategy for every possible hand and then only will it perform an action for the hand it is holding.

This way, using a combination of self-play and search, Pluribus learns to defeat its human opponents in a 6-player poker game.

**(c) What is the evaluation of the proposed solution? (5 pts)**

The performance of Pluribus was evaluated in two ways: by playing one copy of Pluribus with five professional poker players and by playing five copies of Pluribus with one professional human. Each player was an expert in playing poker, having won more than 1 million in poker. The metric used was milli big blinds per game(mbb/game). This measures the number of blinds won by the player per thousand hands of poker. The variance reduction technique used was AIVAT. The profitability of Pluribus was obtained by measuring the statistical significance at 95% confidence level using a one-tailed t test. Evaluations were as follows:

- One AI agent and Five humans: Here, 10000 hands of poker were played over 12 days. Every day 5 players were selected out of 13 players. The participants did not know who else was participating. each player was given \$0.40 per hand for participating, but this could increase to as much as\$1.60 per hand on the basis of performance. Pluribus won an average of 48 mbb/game with a standard error of 25mbb/game. It was profitable with a p value of 0.028. This is a very high rate in poker and this indicates that Pluribus performs better than human players.

-One human and five AI agents: Here, two human players were used. Each of them played 5000 games of poker with five copies of Pluribus. Each human was given \$2000 for participating and \$2000 if he/she performed better against the AI than the other human player. Pluribus got an average score of 32 mbb/game with standard error of 15 mbb/game. It was profitable with a p value of 0.014.

By inspecting the strategy of Pluribus, we can come to know about the optimal play in poker. Pluribus confirmed the human perspective that limping is suboptimal for every player except for the one who has a small blind. However, Pluribus does not follow the human thought that donk betting is a mistake.

**(d) What are the contributions? (5 pts)**

Until now, all the game playing AI systems were mostly used in two-player games. But many games like poker are mostly played with multiple players. This paper came up with a system which can play the multi-player, stochastic game of Texas hold’em poker. This system was made using a unique combination of Monte Carlo CRF which was used for self-play and Linear CRF which was used for search. Further, most AI systems used for imperfect information games usually reason about the entire game and produce a complete strategy before playing. Here, the authors have presented a system which forms a coarse strategy in the beginning and then improves upon this strategy while playing. Also, this system proves that a complex multi-player game with hidden information can be won by an agent using self-play with search without generating the Nash-equilibrium. So, the development of Pluribus proved that despite the lack of theoretical guarantee, it is possible to solve a hidden, complex, multi-player game.

**(e) What are future directions for this research? (5 pts)**

Now that a system has been developed to play the game of poker for multiple players, it should also be developed for other multi-player stochastic games as well. While poker is a zero sum game, there are many games which are non-zero sum. Nash equilibrium cannot be obtained for non-zero-sum games. So, I believe that this technique would be suitable for solving non-zero-sum games as it does not try to find Nash equilibrium, but instead just tries to improve performance of the agent. Research should be conducted to extend this technique so that multiplayer non-zero sum games can also be solved. Further, Pluribus can be improved upon by using a better abstraction strategy or a better depth limited search.

Apart from this, Pluribus is a system which can now work in a multi-agent, stochastic environment and perform remarkably. Most of other real world scenarios are similar in nature. So, this technique can be applied to other areas which have imperfect-information and multi-agent space such as financial markets, traffic navigation, robots, etc. The use of Pluribus in these areas could lead to great success and could improve these systems.