“BLUFF” WITH AI

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Tina Philip

December 2017

The Designated Project Committee Approves the Project Titled

“Bluff” with AI

By

Tina Philip

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2017

Dr. Christopher Pollett,
Department of Computer Science Date

Dr. Philip Heller,
Department of Computer Science Date

Dr. Robert Chun,
Department of Computer Science Date

APPROVED FOR THE UNIVERSITY

Associate Dean
Office of Graduate Studies and Research Date

ABSTRACT

"Bluff" with AI

By Tina Philip

The goal of this project is to build multiple agents for the game Bluff and to conduct experiments as to which performs better. Bluff is a multi-player, non-deterministic card game where players try to get rid of all the cards in their hand. The process of bluffing involves making a move such that it misleads the opponent and thus prove to be of advantage to the player. The strategic complexity in the game arises due to the imperfect or hidden information which means that certain relevant details about the game are unknown to the players. Multiple agents followed different strategies to compete against each other. Two of the agents tried to play the game in offense mode where they tried to win by removing the cards from the hand efficiently and two other agents in defense mode where they try to prevent or delay other players from winning by calling Bluff on them when they have few cards left.

In the experiments that we conducted with all four agents competing against each other, we found that the best strategy was to not Bluff and play truthfully. Playing the right cards, gave the most wins to any player. Also we found out that calling Bluff on a player even if we have more than one card of the same rank would prove risky, since there is a chance that the player was actually playing the correct cards and we could lose the bet as shown by the Anxious AI. We conducted an interesting experiment to find out the best defense strategy and which agent would catch the most number of bluffs correctly. The Anxious AI was the winner. We also try to "teach" an agent how to play the game effectively and experiments show that the agent did learn the strategy very well. We also found that the Smart AI was the evolutionary stable strategy among the four agents.

ACKNOWLEDGEMENT

I would like to thank my mother Reena Mary Philip and my husband Sujith Koshy for making my dream of pursuing a master’s degree in Computer Science come true. I would like to give my deepest gratitude to Dr. Christopher Pollett for being so patient with me, coming up with suggestions that would help me improve my project and for being so cool throughout the project which spanned for a year. I would also like to extend my gratitude to my committee members, Dr. Robert Chun and Dr. Philip Heller for their valuable suggestions, support and time. I would also like to thank my friends especially Priyatha and Roshni.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Bluff is a multi-player card game in which each player tries to empty their hand first. The goal of this project is to build four different agents that play Bluff and find out how they perform over thousands of games. Artificial Intelligence (AI) simulates the decision making capability of humans using machines. We define the computer players as intelligent agents since they understand their environment and take actions to maximize their gain. Two of the agents would play Bluff in an offensive mode where they try to use policies to eliminate cards from their hand as quickly as possible while the other two agents play in a defensive or attacking mode where they try to prevent or delay the opponents from winning. Then we will conduct experiment on these agents to see how they perform in various scenarios. One such scenario is self play where we check if playing in the first position would give any advantage when compared to playing in the last position with the same strategy. We also conduct experiments between the agents to see which strategy would fare better when played a large number of times. Another experiment is to find the evolutionarily stable strategy among the agents. The AIs aim to replicate themselves by culling the weakest player and thus defeating the competitive strategy. The Smart AI was the evolutionarily stable strategy among the four agents.

We could not establish any prior work that conducted research on the game Bluff or experimented with different strategies for agents, but we came across various implementations of Bluff as an online multiplayer game [1]. Some of the agents that we encountered were studied in detail to know more about useful strategies in the game. One such strategy was the truthful agent who plays an honest game and chooses the nearest neighbor heuristics when he does not have the correct card. Nearest neighbor heuristics means that he plays the card that is closest in rank to the card to be played in this turn. After much consideration we came up with a better strategy than nearest neighbor for the Smart AI, which was to play the farthest card in future. Another agent that we came across employed a defense strategy which was to call Bluff on players who had very limited cards in their hand. We modified this strategy slightly to have the Anxious AI who calls Bluff on opponents that have less than three cards in their hand.

The main challenge of this game is that, unlike popular games like chess and backgammon, in which players have full knowledge of each other's game state, Bluff has imperfect information and stochastic outcomes [2]. Imperfect information stems from the lack of knowledge about the other players' cards and thus introduces uncertainty due to unreliable information which provides a chance for deception. The fact that the hand is dealt completely at random produces more uncertainty and a higher degree of variance in results which explains the lack of generous study by computer scientists in this area until recently. Partial observability means that at any time, there is some information hidden from a player and certain information that is known only to the player. Bluff is a multi-player game with non-cooperation among players which reduces the complexity due to players cooperating among each other to target other players and win at the game. Thus Bluff falls into the category of one of the hardest problems in computer science – stochastic, partially observable game with imperfect information.

In the beginning of 2017, a research team from Carnegie Mellon University developed a system called Libratus, which could beat professional players in the card game Poker. This was a significant milestone in Artificial Intelligence for games and sparked the interest for many papers in the field. Poker strategies were not studied in detail prior to the early nineties and pose many uncertainties due to imperfect information [3]. The study of board games, card games and other adversarial models present the following features: well-defined rules, complex strategies, specific and achievable goals and measurable results.

This report is structured as follows. In Chapter 2 we discuss the game terminology and the rules. Then in Chapter 3 we show the game design for the program. Chapters 4 through 8 discuss the strategies used by computer agents to win the game. In chapter 9 we talk about the sampling plan to conduct the experiments and in Chapter 10, we report the experiments where the intelligent agents compete against each other and identify the strongest opponent in the game of Bluff. Chapter 11 concludes the research with some details on the future work for this project.

2. GAME RULES

The card game Bluff is a game of deception and is generally called 'Cheat' in Britain, 'I doubt it' in the USA and Bluff in Asia. Normally, Bluff is played with a standard pack of 52 cards (excluding Jokers) as shown in Fig. 1. The deck is shuffled and each player gets the same number of cards to begin with. The goal of each player is to be the first one to empty their hand. In this game, all cards have equal weight and there is no point system involved. The first player has to start the game by playing Aces, the next player plays Twos and so on. After Kings the next player has to start again from Aces.

Player 1 starts the game by placing some cards face down on the middle of the table and declaring what the rank of the card is and how many there are. Since the cards are played face down, players can lie or bluff about the cards they actually put down. In his or her turn, a player is not allowed to pass, which means that players would have to bluff at some point in the game, if they do not have the actual card to be played. Once a player plays his cards, each of the other players gets a chance to call Bluff on the player. If a challenger calls Bluff and the player bluffed, the player gets all the cards from the discard pile. If the player did not bluff the challenger gets the pile.

One of the strategies in this game is to keep the opponents clueless whether you are playing the right cards or not. The act of bluffing confounds players and game designers alike and implementing agents that can bluff to effectively maximize gains is by no means an easy task. Game strategy can be very complex and depend on various parameters such as the hand dealt, number of players, opponent’s strategy for offense and defense and also luck to a great extent. In the case of human players it depends on the mentality of the players which is very difficult to quantify.

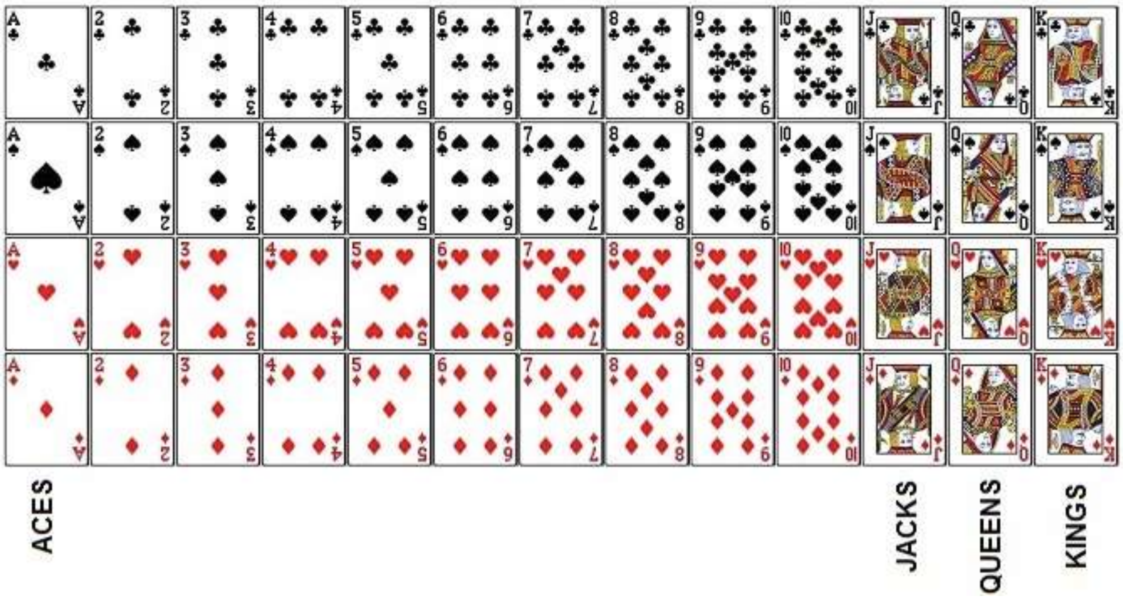


Fig. 1. A standard deck of 52 cards

2.1 Terminology

Deck: A set of 52 playing cards

Hand: The cards assigned to one player

Challenger: The player who calls “Bluff” on the opponent

Rank: The type of card, e.g. Ace, Two, Three, etc.

Turn: The time a player is allowed to play his cards

Discard pile: The set of face down cards in the middle, to which each player adds the cards removed from his hand

Round: A set of turns by all the players completes a round

Agent: The computer player

3. GAME DESIGN

Our game of Bluff can be played by humans and computer agents. We have formulated four AI players with different strategies which can play between themselves or with humans. The game can be played any number of trials and this is especially useful for battling AI players and to analyze their results.

The game was written from scratch in Java and has the following structure:
The cards are displayed to the user by their name and are represented internally as numbers from 0 to 51. As shown in Fig. 2, the Driver class is the main class from where the game begins. It controls the mode of game, number of players and type of players. The CardManagement class shuffles the deck and assigns the hand of each player. ComputerPlayers class is the super class of all the AI players. The *play*() method in each of the AI then handles the logic of the game depending on the strategy employed by each player. First the hand is displayed to the player out of which he can choose the card to play, based on the logic. Next the chosen card is removed from hand and moved to discard pile in *removeCards*() method.

The *callBluff*() method then asks all the remaining players whether they want to challenge the current player in a clockwise manner. It returns a Boolean value “True” if a challenger wants to challenge the current player, “False” otherwise. This decision is made based on the logic of the agent. In the BluffVerifier class, the cards just played by the current player are verified against the rank of the card to be played in that turn returned by the *getCurrentCard*().

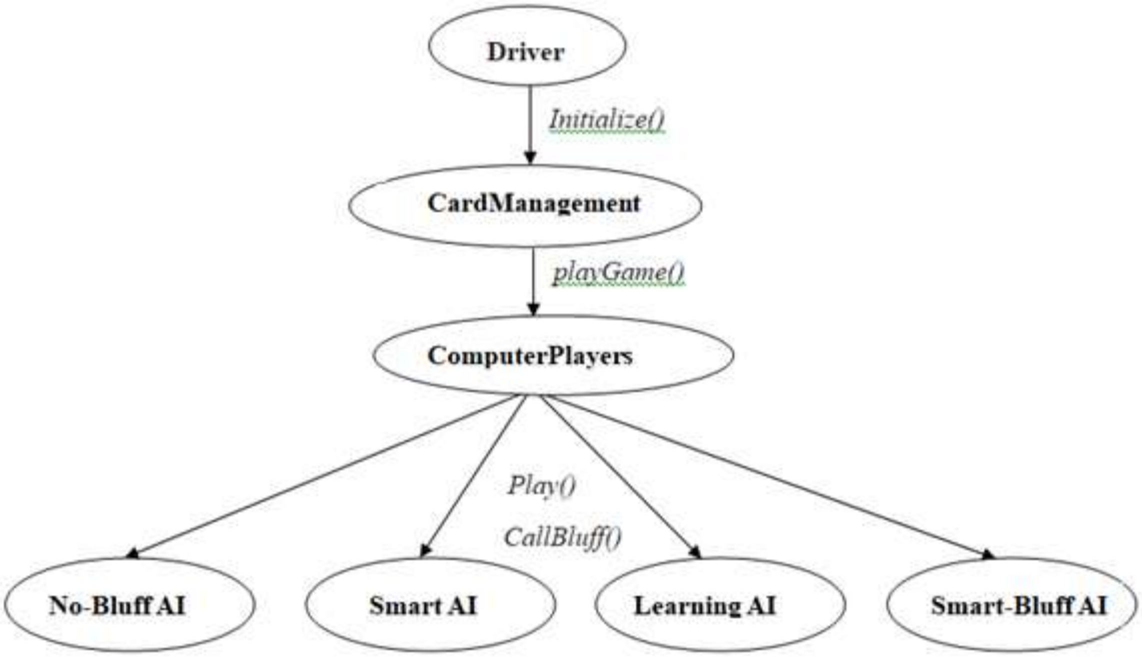


Fig. 2. Game Flow

The *bluffVerifier*() method compares the cards and returns a Boolean value *verdict* which is set to “True” if the player cheated and “False” otherwise. Based on this verdict, the variable *loser* is set to either current player or challenger, and the discard pile is added to the loser’s hand by the method *addDiscardPileToPlayerHands*(). The turn then goes to the next player and the game continues.

4. AGENTS

A number of decisions are to be made by the agent to play the game of Bluff efficiently. The type of card to play, the number of cards to play and when to call bluff on an opponent all these parameters can affect the outcome of the game. We observe each trial of the game with a given hand as an independent stochastic event, and the agent would have information only about his current hand, and nothing more. The agent then will have to make decisions based on this information and not from any previous events [2].

The game of Bluff has two main elements:

i. Which cards to play in the current turn - Offense
ii. When to call Bluff on your opponents - Defense

The answers to these problems depend on the type of AI player, as each of our four AI players have a different strategy. In a given turn there are hundreds of possible actions that can be taken as per the game rule. But we try to limit this by applying constraints in order to produce results faster. When we have multiple cards of the same rank to be played in that turn, we can safely venture to play them, but otherwise the safer strategy is to play one card to reduce suspicion.

The 4 agents we use in our game are:

1. No-Bluff AI
2. Smart AI
3. Reinforcement Learning AI
4. Anxious AI

While No-Bluff AI and Reinforcement Learning AI try to play an offensive game, the other two agents play a defensive game. When we say offensive game, we mean that these players try to avoid getting caught and win the game by effectively removing cards from the hand. Defensive game means that the player not only plays the correct card, it also tries to actively accuse the other players and prevent them from winning.

All the agents except the No-Bluff AI use their chance to call bluff on other players in the hope that other players might get caught playing the wrong cards. The first and a no-brainer decision to call Bluff on an opponent would be if he plays more than four cards. There are only four cards of the same rank and playing cards more than four would mean he is cheating. The next decision to call a Bluff would be when an opponent plays a card of the rank for which we have more than one in our hand. If we have all the four cards of that rank in our hand, we would definitely call Bluff. If we have 3 cards we would call bluff with a very high probability and if we have two cards, we would call Bluff with a lesser probability. An additional defense mechanism is to call Bluff on the opponent if he has less than three cards in hand. This is because, towards the end of the game, it is very rare for players to have the actual cards in hand, forcing them to Bluff. For this we maintain an info-table on each of the players.

5. NO-BLUFF AI

The No-Bluff AI (NBAI) is an offensive player and the simplest agent of the four. It plays the game truthfully. This agent was modeled so that we could understand the importance of bluffing to win the game. No-Bluff AI tries to play as many cards as possible truthfully and when the correct card to play is not in his hand, resorts to playing the first card in his hand. This agent does not suspect other players and never calls Bluff on them. The flowchart for the game logic is as shown in Fig. 3.

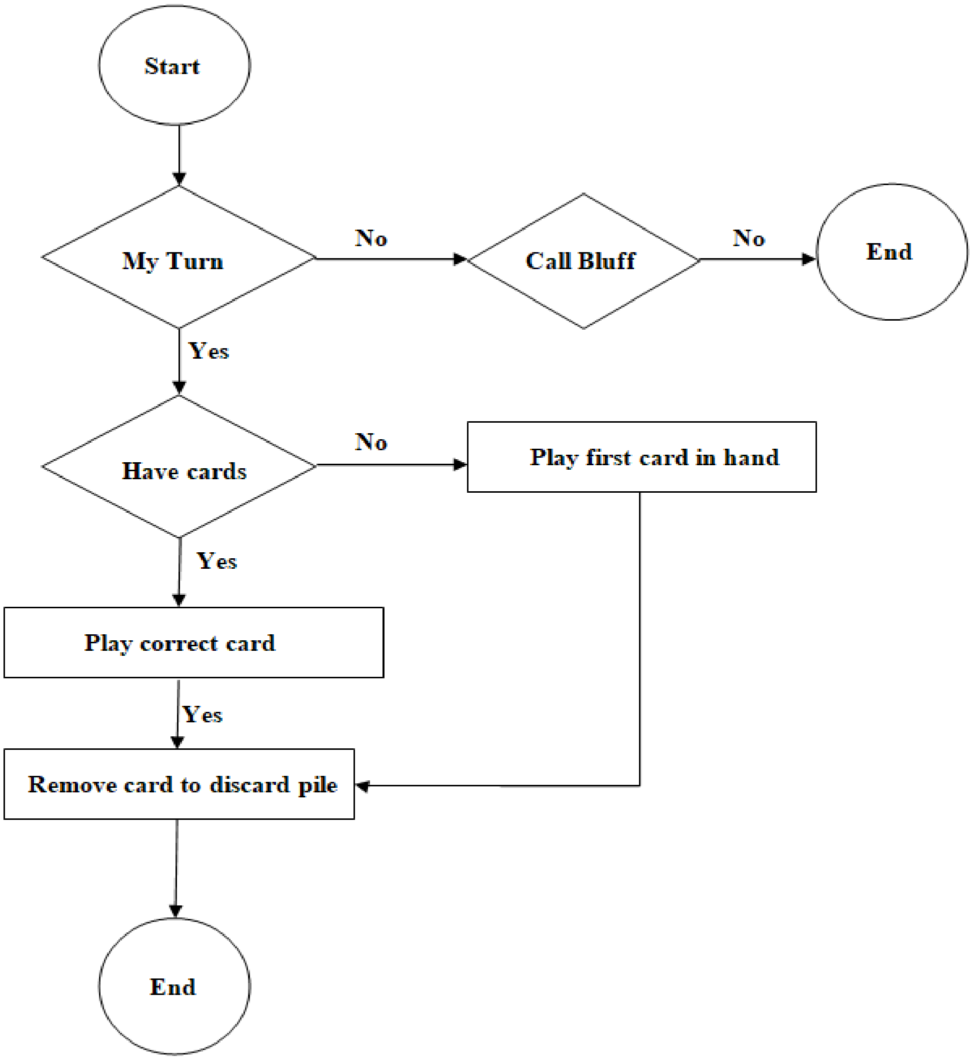


Fig. 3. Game flow of No-Bluff AI

6. SMART AI

The Smart AI (SAI) is a defensive player and has a more complex heuristic for deciding the card to play and when to call bluff on opponents, so as to win the game [4]. If the agent has the card to play, he chooses to play them, since it is the safest strategy and bound to bring reward anyway. Otherwise he plays the next safest strategy, which is to play the card which he would have to play only later on in the game as shown in Table 1. However, the cards to be played in the next four turns immediately after Ace would not be considered.

Table 1 Logic to find farthest card to play

Player 1	Player 2	Player 3	Player 4
ACE	TWO	THREE	FOUR
FIVE	SIX	SEVEN	EIGHT
NINE	TEN	JACK	QUEEN
KING	ACE	TWO	THREE
FOUR	FIVE	SIX	SEVEN
EIGHT	NINE	TEN	JACK
QUEEN	KING	ACE	TWO
THREE	FOUR	FIVE	SIX
SEVEN	EIGHT	NINE	TEN
JACK	QUEEN	KING	ACE
TWO	THREE	FOUR	FIVE
SIX	SEVEN	EIGHT	NINE
TEN	JACK	QUEEN	KING

In a four player game, Player 1 starts with Ace and after Players 2, 3 and 4 plays the ranks Two, Three and Four respectively, Player 1 then plays the rank Five, then Nine and so on, till Ten, before starting with Ace again. When Player 1 has to play Ace and if he does not have Ace in his hand, he could easily figure out that Ten would be the rank that he would have to play last, before starting with Ace again. If he has a card of rank Ten, he would play that, if not he would try the rank Six, Two, etc. up to Eight (leaving out the four cards after Ace).

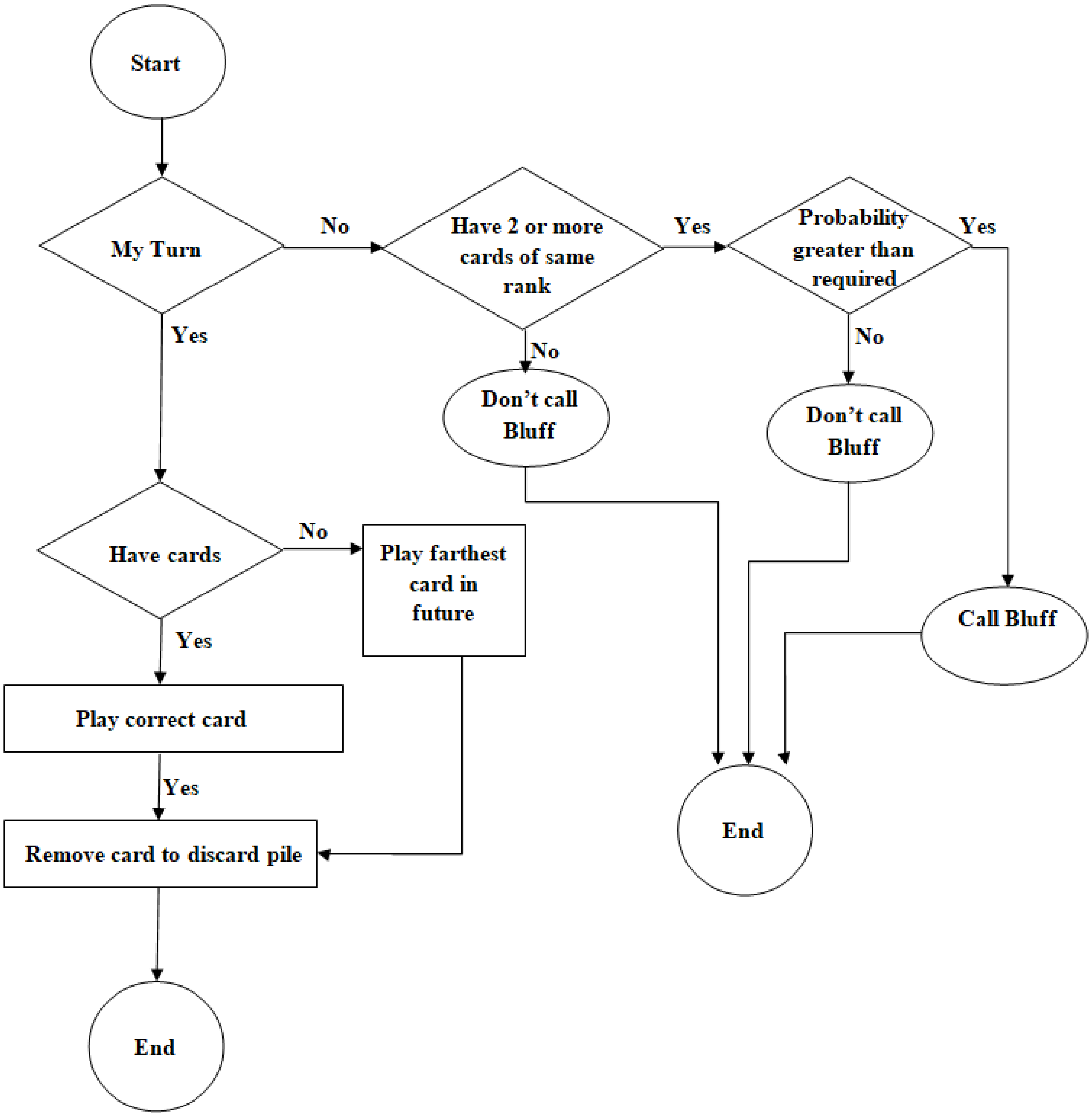

Start
My Turn
No
Have 2 or more cards of same rank
Yes
Probability greater than required
Yes
Yes
No
Don't call Bluff
No
Don't call Bluff
Have cards
No
Play farthest card in future
Call Bluff
Yes
Play correct card
Yes
Remove card to discard pile
End
End

Fig. 4. Game flow of Smart AI

7. REINFORCEMENT LEARNING AI

The Reinforcement Learning AI (RLAI) is an offensive player and uses a more complex strategy as shown in Fig. 5. It can be split into two stages: Training and Testing. The training stage is the learning stage for the AI. In the training stage the Reinforcement Learning AI plays correct card, which is the rank to be played in that turn, if he has it. If he does not have the card, he plays the farthest card in the future. The result of each turn in the learning stage is updated to two 13x13 matrices, namely, the State-Action matrix and the Reward matrix. The State-Action matrix would have the action that was taken during the current state. This means that rows make up the card to be played and the columns make up the card actually played in this turn. For example, in the current state, if the card to be played was Ace, and the player played an Ace, then the value for row and column [0, 0] would be updated to one. If the player did not have an Ace and played some other card, say Ten, then the value for row and column [0, 9] will be updated to one. If some player calls Bluff, the result is updated in the Reward Matrix. For example, in the previous example where the player played the correct card- Aces, if the challenger calls Bluff and loses, then the value at row and column [0, 0] is incremented by one.

After the learning stage, comes testing. In this stage, the player is not explicitly told which card is to be played in the current turn. Instead he is offered a look-up table which has all the possible actions that was taken previously and the reward obtained while playing this rank. The player would then choose the action which would bring the highest reward. He verifies if his hand has the card with the highest reward, if not, he chooses the card with second highest reward and so on. We have observed that the Reward matrix has high values diagonally. This is because, the agent is rewarded the most when he plays cards honestly. This prompts the agent to play the correct cards in his testing phase and thus learn the strategy effectively.

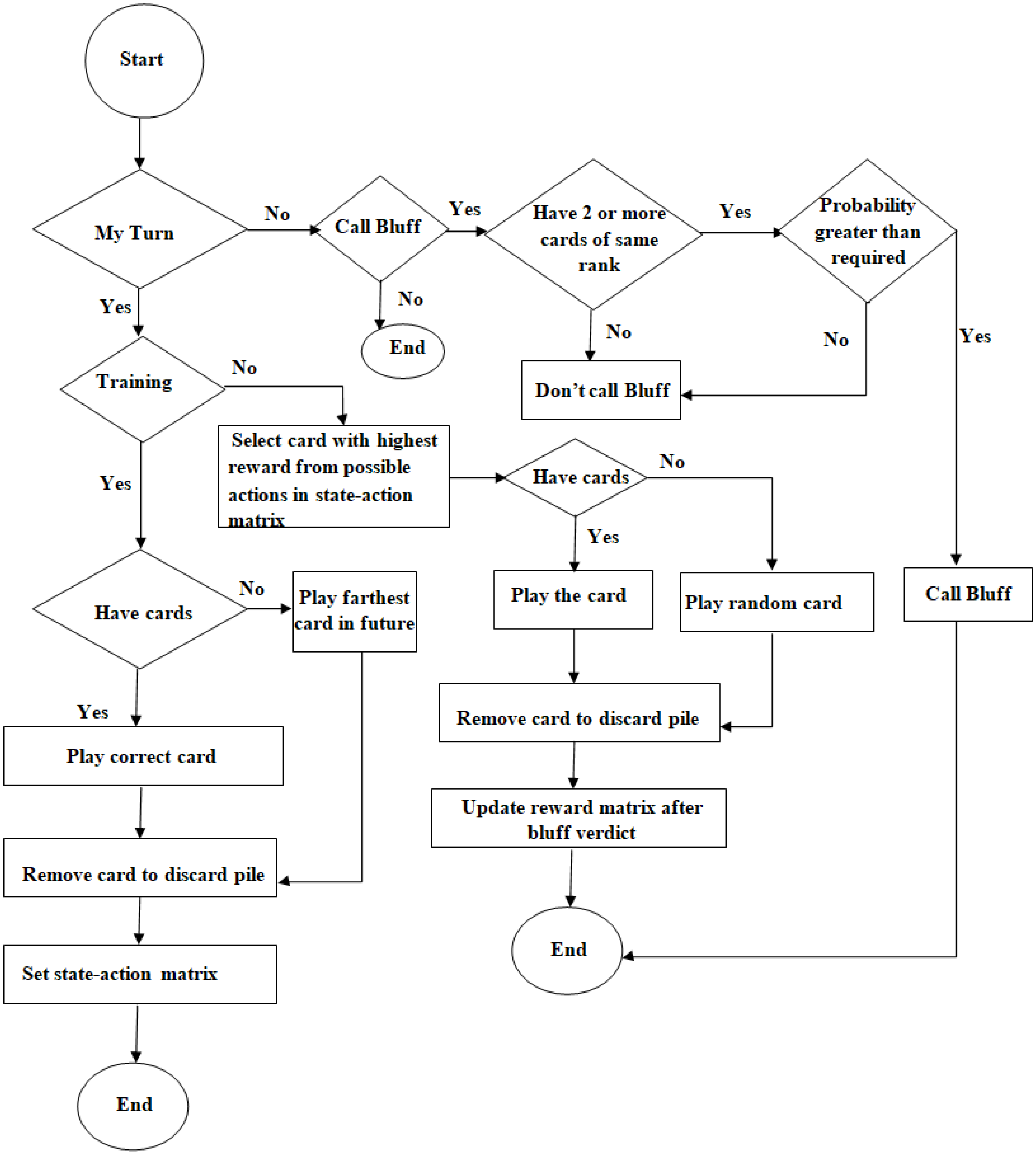
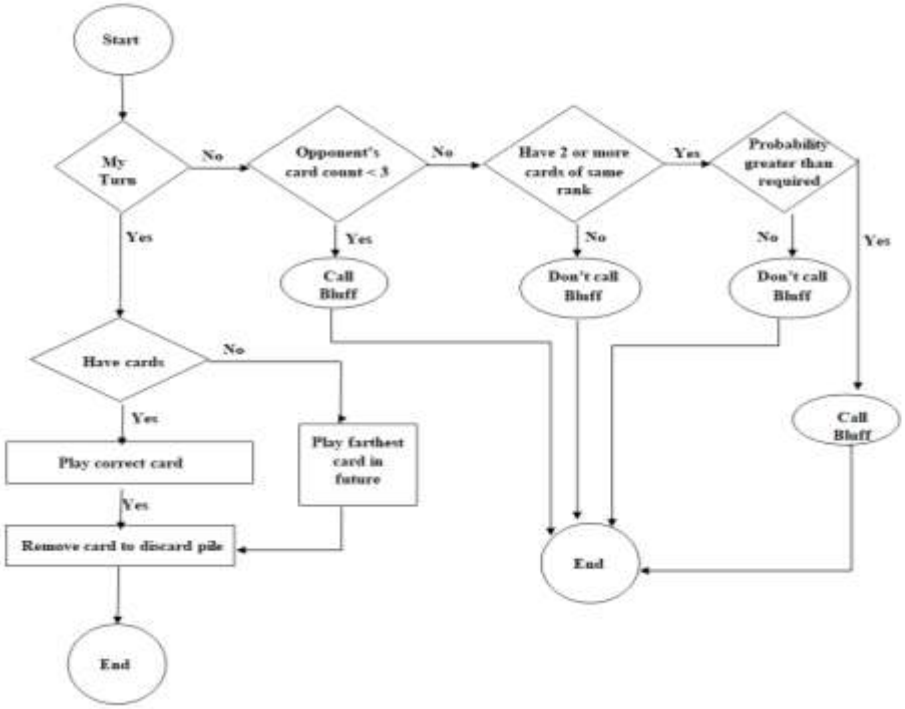

Start
My Turn
No
Call Bluff
Yes
Have 2 or more cards of same rank
Yes
Probability greater than required
No
End
No
No
Yes
Yes
Don't call Bluff
Training
No
Select card with highest reward from possible actions in state-action matrix
Have cards
No
Yes
Yes
Have cards
No
Play farthest card in future
Play the card
Play random card
Call Bluff
Yes
Remove card to discard pile
Play correct card
Update reward matrix after bluff verdict
Remove card to discard pile
End
Set state-action matrix
End

Fig. 5. Game flow of Reinforcement Learning AI

8. ANXIOUS AI

The Anxious AI (AAI) is a defensive player and plays the right card if he has it, but the main technique he uses to win the game is to call Bluff on other players to delay their win. During his turn to call bluff, the Anxious AI becomes anxious if an opponent has less than three cards (and is about to win) and so calls Bluff on any player with less than three cards in their hand. This is because, towards the end, it is very rare that the players have the actual card to play in that turn. This forces them to cheat if they have to win. The AAI takes advantage of this and forces the leading player to get caught and thus delay his win. Now this strategy might prove to be fatal to the AAI as well as the player. If the discard pile has a large number of cards and the player gets it all, then he has a huge disadvantage, or it could be that the discard pile was light and did not harm the player much. It could also happen that the player played the actual card and that the AAI was wrong. This will be analyzed in the experiments.



Start
My Turn
No
Opponent's card count < 3
No
Have 2 or more cards of same rank
Yes
Probability greater than required
Yes
Yes
No
No
Yes
Call Bluff
Don't call Bluff
Don't call Bluff
Have cards
No
Yes
Call Bluff
Play correct card
Play farthest card in future
Yes
Remove card to discard pile
End
End

Fig. 6. Game flow of Anxious AI

9. SAMPLING PLAN

Bluff is a game of win or lose and so it is categorical. For each game, we have an outcome and they are independent of each other. Inferential statistics would help to make an inference about our results from a sample space. But it could have some degree of error or uncertainty that would be captured by the confidence interval. Confidence interval denotes the number of samples required to compute a result with a certain confidence level such as 95% or 99%. Attribute Sampling can be used to determine the sample size for categorical problems, such as classifying an object as good or bad and in our case identifying a win or lose [5]. To determine the least sample size (run size) for our experiment to result in 99% confidence and 99% reliability level, we use the following formula [6]:

$$\text{Run size (n)} = \frac{\text{Ln}(1-\text{ confidence interval})}{\text{Ln}(\text{Reliablity})} = \frac{\ln(1-0.99)}{\ln(0.99)} = 299 \text{ Runs}$$

Therefore from this equation we determine that we should run our experiments greater than or equal to 299 trials and to round off, we run all our experiments for 300 trials.

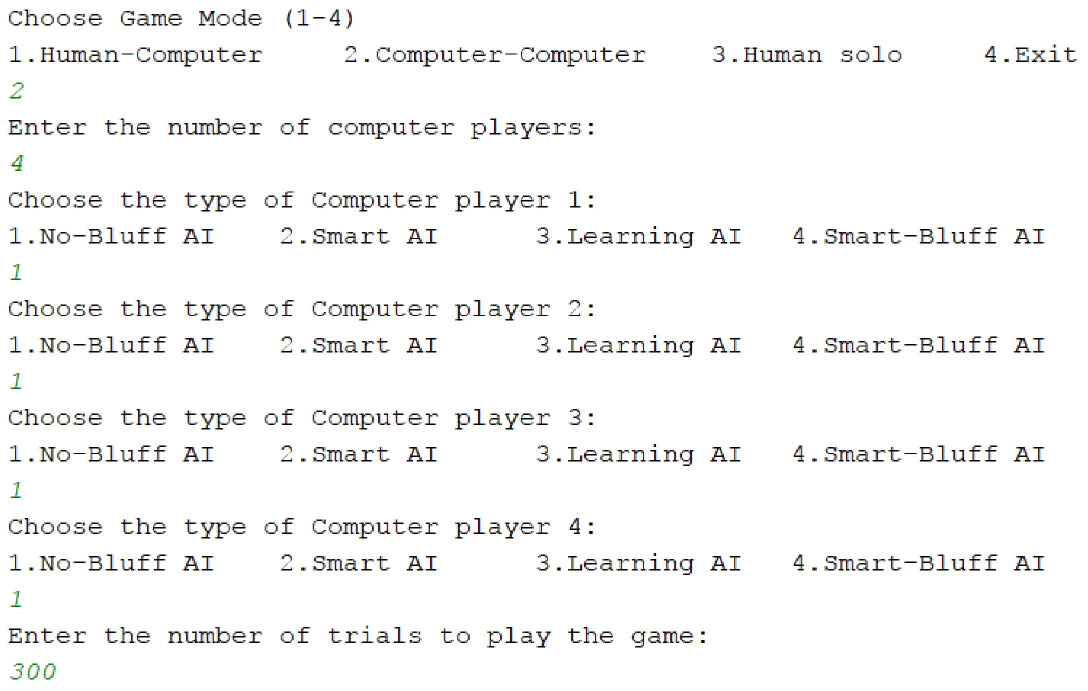
10. EXPERIMENTS

10.1 Experiment 1: Self Play

The first type of experiment we conduct is called self-play, in which the agents compete against themselves. This test was conducted with four players for 300 games on different decks in each trial. There are four possible self-plays:

1. No-Bluff AI vs. itself
2. Smart AI vs. itself
3. Reinforcement Learning AI vs. itself
4. Anxious AI vs. itself

The purpose of these experiments was to find out if any player in a particular position has advantage over the other, even with the same logic. We ran the first experiment with four No-Bluff AIs competing amongst themselves with the following settings as shown in Fig. 7.



```
Choose Game Mode (1-4)
1.Human-Computer     2.Computer-Computer    3.Human solo     4.Exit
2
Enter the number of computer players:
4
Choose the type of Computer player 1:
1.No-Bluff AI    2.Smart AI      3.Learning AI   4.Smart-Bluff AI
1
Choose the type of Computer player 2:
1.No-Bluff AI    2.Smart AI      3.Learning AI   4.Smart-Bluff AI
1
Choose the type of Computer player 3:
1.No-Bluff AI    2.Smart AI      3.Learning AI   4.Smart-Bluff AI
1
Choose the type of Computer player 4:
1.No-Bluff AI    2.Smart AI      3.Learning AI   4.Smart-Bluff AI
1
Enter the number of trials to play the game:
300
```

Fig. 7. Parameters for a sample Self-play - No-Bluff AI against itself

Hypothesis: No position would have advantage over other positions during self play.

Result: In each of the runs, the results were fairly consistent with a confidence interval of 99% and a reliability of 99%. NBAI was tested first against itself in four different positions. NBAI player in position 1 won around 44% of the time, NBAI player in position 2 won 26% of the time, NBAI player in position 3 won around 16% of the time and NBAI player in position 4 won around 12% of the time as shown in Table 2.

Table 2 Win rate of Experiment 1

	NBAI	**SAI**	**RLAI**	**AAI**
Player in Position 1	**44**	**36**	**47**	**34**
Player in Position 2	**26**	**23**	**17**	**14**
Player in Position 3	**16**	**19**	**16**	**24**
Player in Position 4	**14**	**22**	**20**	**28**

Next we tested four SAIs against for 300 games each, with different decks. Here too, the results were fairly consistent with a confidence interval of 99% and a reliability of 99%. SAI player in position 1 won around 36% of the time, SAI player in position 2 won 23% of the time, SAI player in position 3 won around 19% of the time and SAI player in position 4 won around 22% of the time.

When the four RLAIs played against themselves for 300 games RLAI player in position 1 won around 47% of the time, RLAI player in position 2 won 17% of the time, RLAI player in position 3 won around 15% of the time and RLAI player in position 4 won around 20% of the time with a confidence interval of 99% and a reliability of 99%.

The AAIs also played against themselves for 300 games and AAI player in position 1 won around 47% of the time, AAI player in position 2 won 17% of the time, AAI player in position 3 won around 15% of the time and AAI player in position 4 won 20% of the time as in Fig. 8.

There are a few reasons why almost half the time, Player in position 1 won the games even though the deck was shuffled and cards were assigned randomly without any bias to the players.

- Bias towards the player in position 1, since he leads the round
- Distribution of card after shuffling

Just as in the real game between humans, Player in position 1 has the advantage of leading the turn (52 % 4 = 0). Consider the case where each player is left with one card. Player 1 gets to play first in the round and discard the last card in his hand before other players. So he has higher probability of winning. But this scenario is the same in the actual game too. Probability of winning also depends on the distribution of cards after shuffling, since the players with more than one card of the same rank can empty their hand faster.

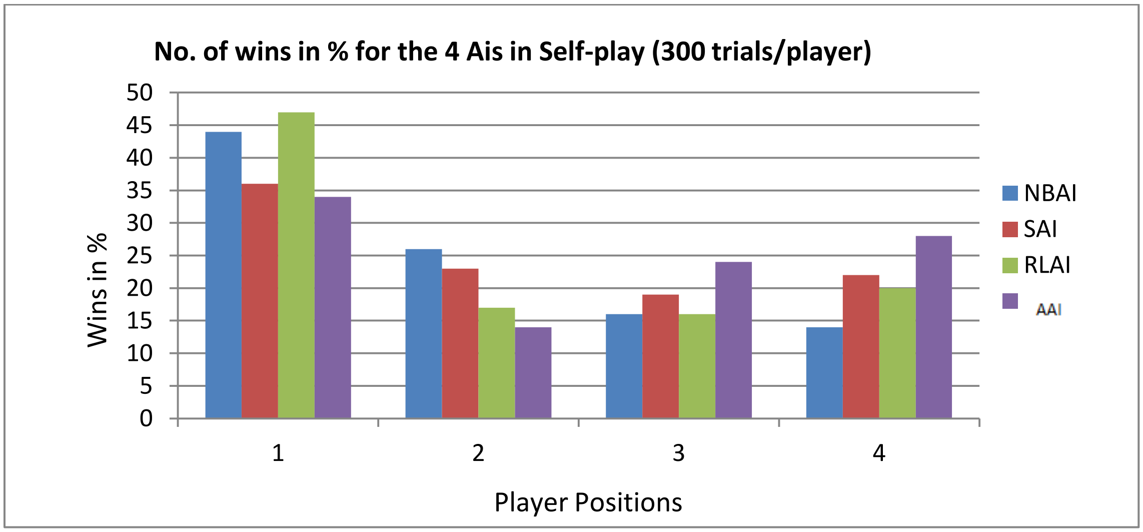


Fig. 8. Experiment Results for Self-play

Conclusion: For all the AIs, we note that player in position one has an advantage over others and so our hypothesis is wrong.

10.2 Experiment 2: No-Bluff AI vs. Smart AI

In this experiment we play No-Bluff AI against Smart AI for 300 games. The expectation was that the Smart AI would beat the No-Bluff AI. But the interesting factor to look for was whether being the first player would give any additional advantage to the No-Bluff AI.

Smart AI calls bluff on the No-Bluff AI whereas No-bluff is trusting and never doubts other players. So Smart AI had an unfair advantage of never being caught even if it cheated.

Hypothesis: Smart AI would beat No-Bluff AI.

Result: In a four player game with players 1 and 3 as the No-Bluff AI and players 2 and 4 as the Smart AI, we see an unexpected result. Contrary to our expectation, Player1, the No-Bluff AI had the most number of wins as shown in Fig. 9 below. Player 1 won 32% of the games, Player 2 won 26% of the games, Player 3 won 20% of the games and Player 4 won 22% of the games. When the same No-Bluff AI was the player 3, Smart AI could beat it.

Conclusion: This experiment shows that when No-Bluff AI is in position 1 he has an advantage over Smart AI, and won the game. But when No-Bluff AI is not in first position, Smart AI could beat him.

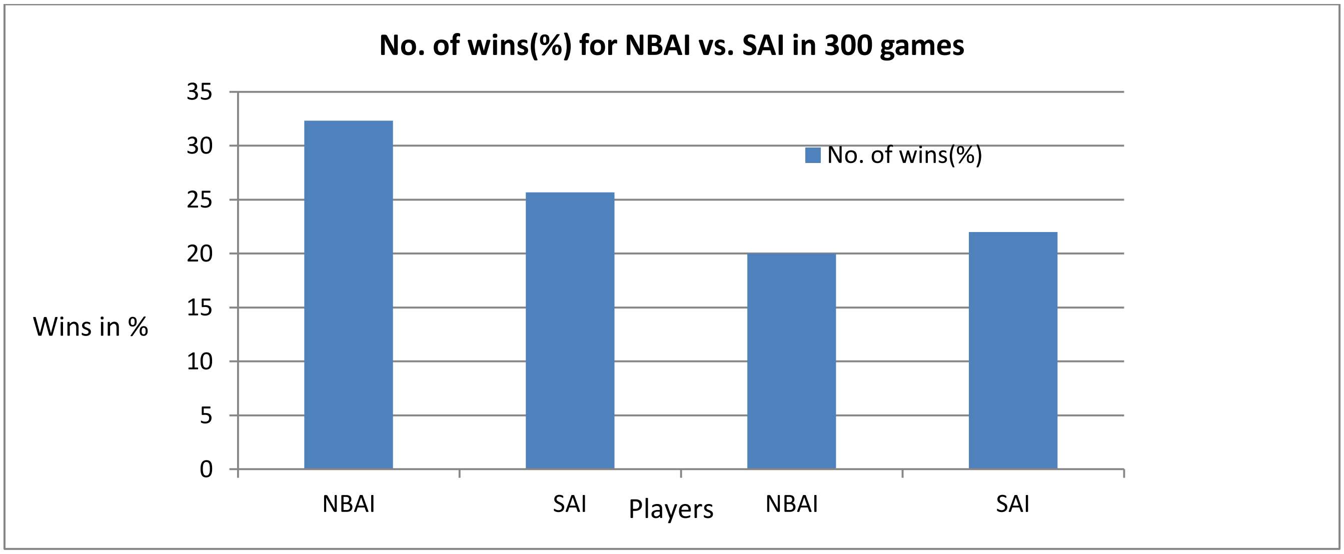


Fig. 9. Result of Expt. 2: No-Bluff AI vs. Smart AI

10.3 Experiment 3: Anxious AI vs. Reinforcement Learning AI

In this experiment we play the AAI against the RLAI for 300 games. The expectation was that the RLAI would beat the AAI. Here too we want to find out how the first player advantage would affect the outcome, if any and whether the logic wins over the first player advantage.

Hypothesis: RLAI would beat the AAI

Result: As shown in Fig. 10, in a four player game with Players 1 and 3 as the AAI and Players 2 and 4 as the RLAI, we see that AAI in position 1 gets 49% of wins while in position 3, it gets only 5% of the wins. Player 2, the RLAI got 31% of the wins in position 2 and 15% of the wins in position 4. The RLAI beat the AAI when it was not in position 1.

Conclusion: This experiment also proves that the Player 1 has an advantage over other player, which can be proved by the RLAI winning over AAI when it was not in position 1.

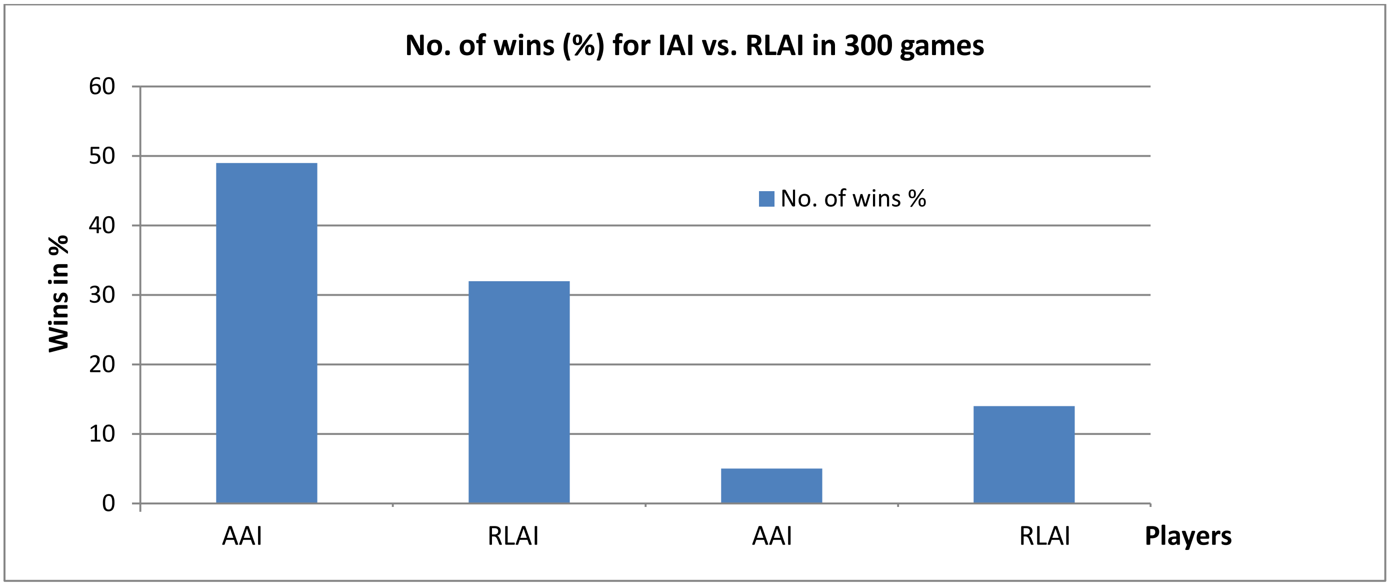


Fig. 10. Result of Expt. 3: AAI vs. RLAI

10.4 Experiment 4: NBAI vs. SAI vs. RLAI vs. AAI

In this experiment we play the four AIs with each other in all possible combinations as shown in Table 3, and noted the number of wins for each player for 300 trials/position. For ease, we denote each player by number, to represent the run order. No-Bluff AI is denoted as 1, Smart AI is 2, Reinforcement Learning AI is 3 and Anxious AI is 4. Run order simply means the position in which each agent played for a set of 300 games. For example run order 1234 means that NBAI was Player 1, Smart AI was Player 2, Reinforcement Learning AI was Player 3 and Anxious AI was Player 4 for 300 trials. We ran a total of 7200 games for each player.

Null Hypothesis (H_o): Reinforcement Learning AI would have the highest number of wins since Reinforcement Learning AI has the knowledge of previous outcomes, which other players lack.

Alternate Hypothesis (H_1): Reinforcement Learning AI would have equal or lower win rates when compared to other players.

Experimental setup: All possible combinations of the four AI players were tested for 300 trials, totaling of 7200 games. The results of experiment are shown below in Table 3.

Result: This experiment was crucial to benchmark the performance of all the agents. We have conducted the experiment with agents in all possible positions to eliminate the possibility of unfair advantage by occupying position 1.We have some key findings from this experiment.

- The NBAI was the best performer followed closely by SAI
- The SAI has very good performance rate and is closely followed by the RLAI
- The RLAI could not beat other players as we expected it to
- The AAI was the lowest performer

Table 3 Win rate of Experiment 3

	Total no. of wins in 300 trials			
Run order	**NBAI**	**SAI**	**RLAI**	**AAI**
1234	107	79	110	4
1243	154	80	66	0
1324	119	102	78	1
1342	145	74	80	1
1423	167	51	82	0
1432	149	83	68	0
2134	103	104	91	2
2143	78	158	64	0
2314	97	94	109	0
2341	90	129	80	1
2413	87	154	57	2
2431	93	127	79	1
3124	113	95	91	1
3142	83	80	137	0
3214	85	121	93	1
3241	86	90	121	3
3412	98	59	142	1
3421	96	63	139	2
4123	99	84	113	4
4132	100	110	86	4
4213	114	101	85	0
4231	116	85	98	1
4312	110	96	91	3
4321	102	111	87	0
Total wins for each player	**2591**	**2330**	**2247**	**32**
Win%	**36%**	**32%**	**31%**	**1 %**

The RLAI could not beat the other agents like we expected it to, unless it was given the first player advantage. The RLAI has a win rate of 31% which is very identical to the SAI. This could be because, during the training phase, the Reinforcement Learning AI follows the strategy of the Smart AI. We could say that we trained our Learning agent to be as smart as the model it followed. But since it did not follow the No-Bluff the winning strategy, which we found out from this experiment, it could not become the winner like we expected.

The Anxious AI was expected to have a high win rate with its defensive strategy of calling Bluff on other players with less than 3 cards in hand. This strategy gave the agent only 32 wins in total which makes it only 1% of wins. Upon analyzing this problem, we found that the Anxious AI was penalized a lot for calling random Bluffs on all players with fewer cards. Since the agents tend to play the correct card when possible, a lot of times the Anxious AI got the discard pile.

The No-Bluff AI had a very strong win rate of 36%. This is because, there are very few ways for the No-Bluff AI to acquire cards from discard pile compared to all the other agents. The normal ways for agents to get more cards from discard pile are:

i. When they play the wrong cards and get caught

ii. When the agent is the challenger and the player had played the right cards.

Since the No-Bluff AI does not call Bluff on other players, there is no possibility of acquiring more cards unless it played a Bluff and was caught, which was rare. The No-Bluff AI tries to play the correct cards and so, the chance to get more cards is very few. Even if the No-Bluff AI was in positions other than one, it showed a steady number of wins with less variance as shown in Fig. 11.

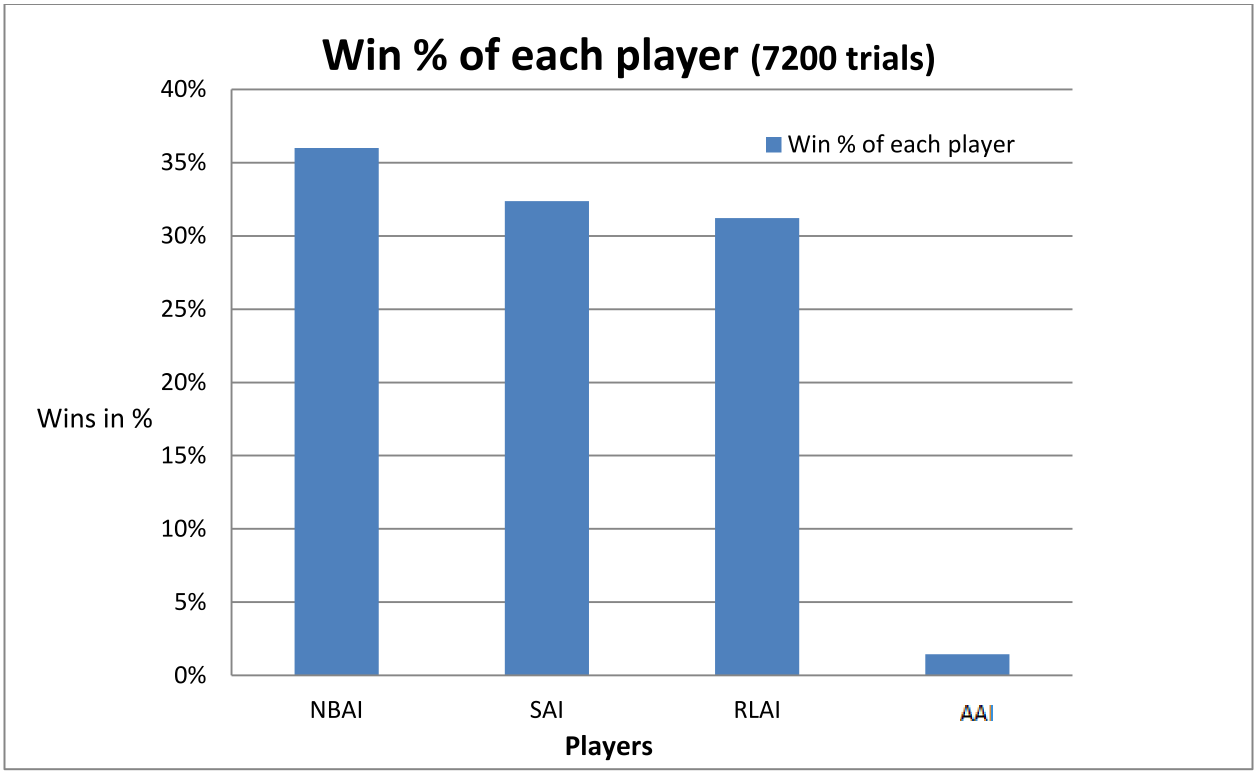


Fig. 11. Game result of all AIs for 7200 trials

We could see from Table 3 that each player when occupying the first position bagged the most number of wins compared to the second, third or fourth position. In the Box plot in Fig. 12, we can see the win rate of No-Bluff AI for each of the positions for 300 trials/position. We find that the mean of second, third and fourth position is around 100, but when playing in position 1, the mean is around 150. This means that the opening player has roughly around 50% advantage than the rest of the players.

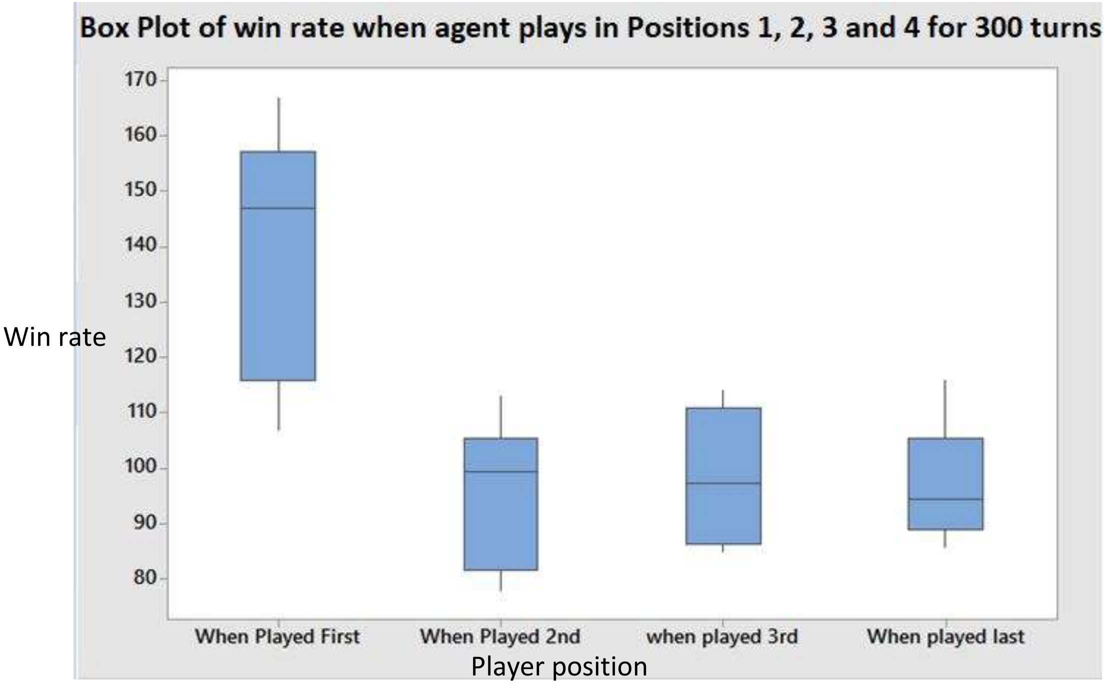


Fig. 12. Win rate for player 1 in each position

Conclusion: From the results it is evident that the Alternate Hypothesis (H_1) is true and Null hypothesis can be rejected with No-Bluff AI having the most wins of all players.

10.5 Experiment 5: True Bluff calls vs. False Bluff calls

In this experiment we aim to find which players made the most number of Bluff calls, their percentage of correct Bluff calls and false Bluff calls in 1200 games.

Null Hypothesis: Anxious AI would have the most number of False Bluff calls as Anxious AI tends to call Bluff every time if its sees an opponent with less than 3 cards in hand.

Alternate Hypothesis: Anxious AI would have the highest success rate in catching Bluff, since most players would not have the correct card to play towards the end.

Experimental Setup: 1200 games were played among all four AIs in all possible combinations and both true and false bluff call results were observed.

Result: As shown in Table 4, No-Bluff AI did not make any Bluff calls as demanded by logic, Smart AI made around 2114 correct Bluff calls and 1251 false Bluff calls. Reinforcement Learning AI is better at catching Bluff than Smart AI and made around 2988 correct Bluff calls and 1310 false Bluff calls.

Table 4 True Bluff vs. False Bluff

	Number of Correct Bluff calls in 1200 Games			
	NBAI	SAI	RLAI	AAI
Total	0	2114	2988	9508
True Bluff %	0.0%	62.8%	69.5%	75.0%
	Number of False Bluff calls in 1200 Games			
	NBAI	SAI	RLAI	AAI
Total	0	1251	1310	3163
False Bluff %	0.0%	37.2%	30.5%	25.0%

We can see that the Anxious AI has the winning strategy and made around 9508 correct Bluff calls and 3163 false Bluff calls. Calling Bluff on other players whenever they have less than 3 cards has increased the number of Bluff calls for Anxious AI tremendously. Around 62% of the total Bluff calls were made by Anxious AI, followed by Reinforcement Learning AI with 21% and Smart AI with 16% of Bluff calls. Though Smart AI and Reinforcement Learning AI share a close percentage of success in catching Bluffs (SAI – 62.8% & RLAI – 69.5%), it was clear that Reinforcement Learning AI had better success in catching Bluff.

Conclusion: The Null Hypothesis was rejected and the Alternate Hypothesis was accepted as Anxious AI had the best success rate at calling Bluff.

10.6 Experiment 6: Modeling Bluff Using Evolutionary Game Theory

This experiment is based off of Evolutionary game theory, which had helped to model competition and evolution. Each player analyzes the opponent's strategy and makes his own choice of moves with an objective to maximize payoff. Strategy success is determined by how well one strategy is, in presence of a competing strategy. The players aim to replicate themselves by culling the weakest player and thus defeating the competing strategy.

Replicator dynamics model is defined as a strategy which does better than its opponents and replicates at the expense of strategies that do worse than the average. This model is used to conduct our experiment.

Replicator Equation is defined as: $\dot{x}_i = x_i[f_i(x) - \phi(x)]$

Where, $\phi(x) = \sum_{j=1}^{n} x_j \; f_j\,(x)$

x_i *– Proportion of type i in the population*

$f_i(x) - is\ the\ fitness\ of\ type\ i$

$\phi(x) - is\ the\ average\ population\ fitness$

From the above Replicator equation it can be understood that the growth rate is the difference in average payoff of a particular player strategy against the average payoffs of the entire player population. The player that evolves and dominates the entire population is considered to be in Evolutionarily Stable State.

Evolutionarily Stable Strategy (ESS): A given strategy is called an evolutionarily stable strategy if a population adopting this strategy cannot be defeated by a small group of invaders using a different strategy which was initially weak [7].

10.6.1 Experiment 6a: Finding the dominant strategy in the population

Aim: To find the Evolutionarily Stable Strategy among the four agents.

Experiment Setup: In this experiment, we ran four agents for one Evolution (set of 300 games) and observed the fitness of a player against other players' fitness. Fitness was evaluated as a measure of number of wins against other opponents. We repeated this experiment over several evolutions and results were observed.

Result: The results of the experiments are shown in Table 5.

For each evolution, we calculated the fitness of each player using the replicator equation and eliminated the player with the weakest strategy (least fit) and replicated the agent with the strongest value to take its position. The calculations for the first Evolution is shown below.

- In the very first Evolutionary run, AAI had the weakest strategy of all players and was eliminated with an offspring of RLAI.
- In the second evolutionary run, an offspring of RLAI was culled by a SAI offspring.
- In the third evolutionary run, Reinforcement Learning AI was eliminated and replaced with SAI offspring.
- In the fourth evolutionary run, No Bluff AI was eliminated with SAI offspring dominating the entire game population.
- In the fifth evolutionary run, the whole population is using the SAI strategy and has reached the stable state as shown in Fig. 14.

Table 5 Win % for each evolution (300 trials/evolution)

Evolution Time Period	Player Type	Percentage of wins %				Remarks
Evolution 1	NBAI, SAI, RLAI, AAI	27%	26%	32%	15%	Eliminated Player AAI and replicated RLAI
Evolution 2	NBAI, SAI, RLAI, RLAI	30%	34%	24%	12%	Eliminated Player 4 (RLAI) and replicated SAI
Evolution 3	NBAI, SAI, RLAI, SAI	24%	28%	20%	28%	Eliminated Player 3 (RLAI) and replicated SAI
Evolution 4	NBAI, SAI, SAI, SAI	15%	21%	20%	44%	Eliminated Player 3 (NBAI) and replicated SAI
Evolution 5	SAI, SAI, SAI, SAI					Evolutionarily Stable State

Calculations:

$f_j(x) -$ is the fitness of type j and is calculated as the number of wins.

x_j – Proportion of type j in the population

$\phi(x) -$ is the average population fitness

In the first Evolution, the total wins of each players are as shown in Table 6.

$\phi(x) = \sum_{j=1}^{n} x_j \; f_j \; (x)$

= Sum (Proportion of j * Fitness of j)

= (0.25* 82) + (0.25*79) + (0.25*95) + (0.25*44)

= 18.75

No-Bluff AI has a total of 82 wins. The Replicator Equation for No-Bluff AI is calculated as follows:

$f_i(x) - \phi(x)$ = Total wins – Average population fitness

= 82 – 18.75 = 63.25

$x_i[f_i(x) - \phi(x)]$ = 0.25 * 63.25

= 15.8125

RLAI has a value of 19.0625 and is the best strategy (strongest agent).RLAI is followed by NBAI and SAI. SAI has a value of 15.0625 and AAI has a value of 6.3125. We eliminate the agent with the least fitness. So after the first evolution, AAI has been eliminated and replaced with a replica of RLAI which was the strongest strategy in this round. The values for all the agents are shown in Table 6.

Table 6 Totals wins of four players in first evolution (300 trials)

Players	NBAI	SAI	RLAI	AAI
Total wins in 300 games	82	79	95	44
$f_i(x) - \phi(x)$	63.25	60.25	76.25	25.25
$x_i[f_i(x) - \phi(x)]$	15.8125	15.0625	19.0625	6.3125

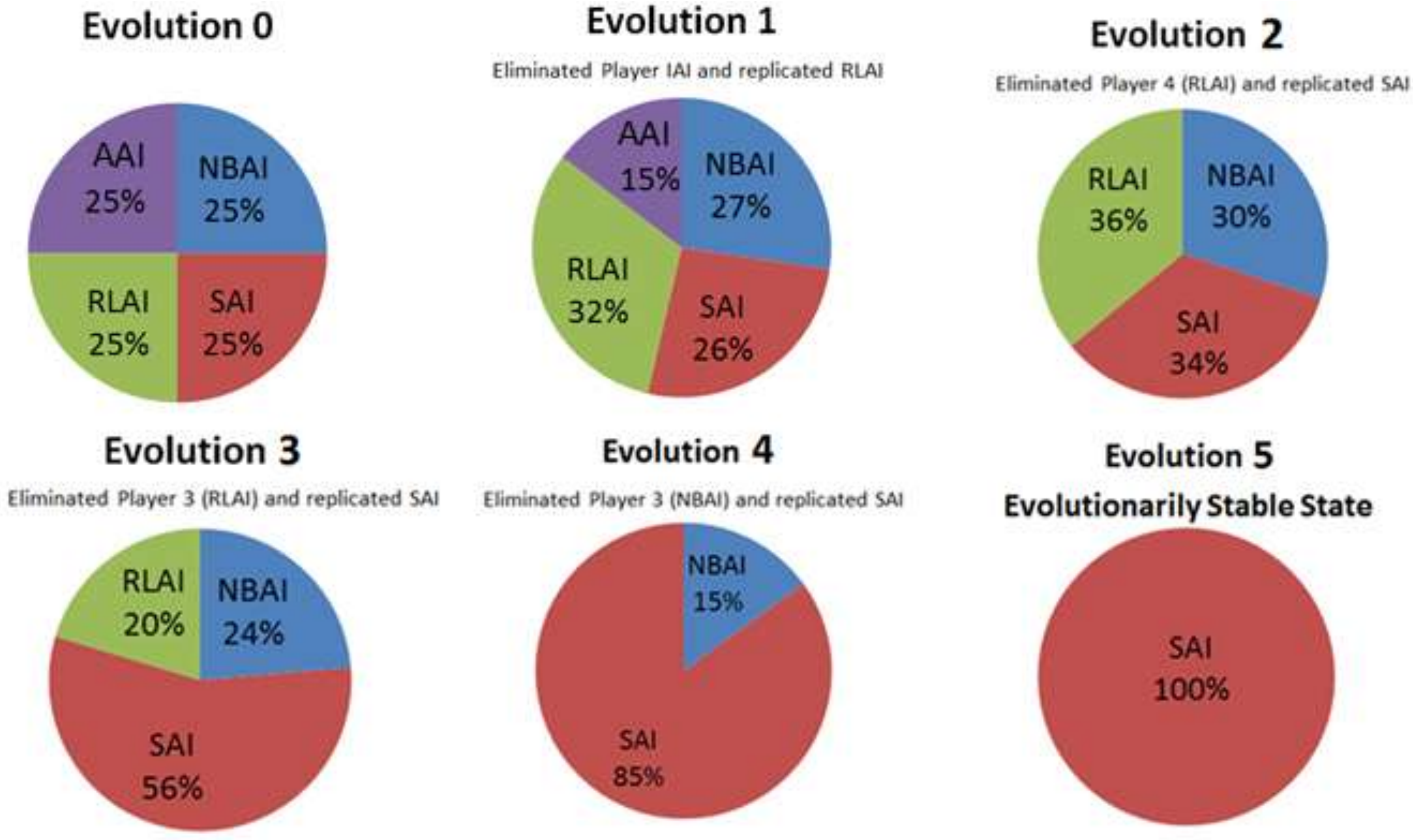


Fig. 13. Population growth of Players using Evolutionary Game Theory

Conclusion: SAI has overcome all other competing strategies and successfully multiplied its own strategy into the entire population.SAI may possibly be the ESS given that it has successfully established its population.

To verify ESS a subsequent experiment (Experiment 6b) has to be conducted with a small group of invaders.

10.6.2 Experiment 6b: Test for finding the Evolutionarily Stable Strategy

Aim: To test the stability of Evolutionarily Stable Strategy with invaders.

Experiment Setup: In this experiment, we ran six agents (four SAI and two mutated AAI) for one Evolution (set of 300 games) and observed the fitness of a player against other players' fitness. We repeated this experiment over several evolutions and results were observed.

Result: The results of the experiments are shown in Table 7.

- Anxious AI was modified to call bluff on opponents with less than 2 cards and then introduced as the fifth and sixth players (mutants) to invade the SAI ESS state.
- Over three generations, the Mutant-Anxious AI population was eliminated by the SAI strategy. Therefore the SAI strategy is the Evolutionarily Stable Strategy and this state is called Evolutionarily Stable State as shown in Fig. 15.

Table 7 Win % for each evolution after introducing mutants (300 trials/evolution)

Evolution Time Period	Player Type	Win Percentage %						Remarks
Evolution 6	AAI, AAI, SAI, SAI, SAI, SAI	1%	21%	9%	22%	24%	22%	Eliminated AAI and multiplied SAI
Evolution 7	SAI, AAI, SAI, SAI, SAI, SAI	50%	3%	4%	13%	16%	15%	Eliminated AAI and multiplied SAI
Evolution 8	SAI, SAI, SAI, SAI, SAI, SAI	18%	17%	19%	15%	15%	15%	Evolutionarily Stable State

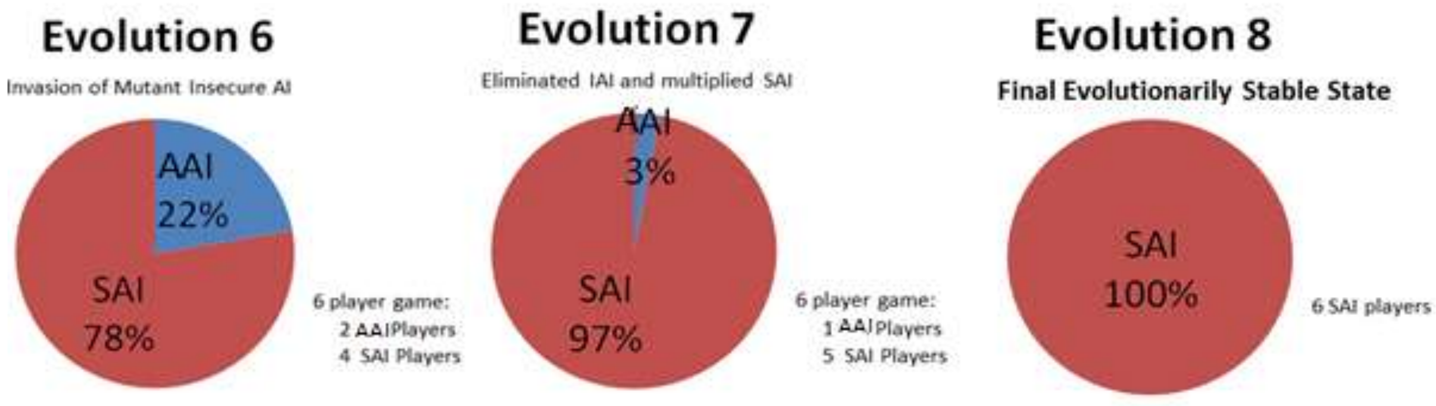


Fig. 14. Test for ESS stability in 6 player game with mutants

Conclusion: SAI strategy is the Evolutionarily Stable Strategy and this state is called Evolutionarily Stable State.

A small group of invading population using a strategy T would have lesser fitness than the evolutionarily stable strategy S and would be overcome by majority population, provided the disturbance by the invading strategy T is not too large [8].

More formally, we will phrase the basic definitions as follows:

- The fitness of a player is based on the expected payoffs from the interactions with other players.
- Strategy T invades a strategy S at level x, where x is a small positive number and denotes the population that uses T and (1 – x) denotes the population using S
- Finally, strategy S is said to be evolutionarily stable if a strategy T invades S at any level $x < y$, where y is a positive number, and the fitness of strategy S is strictly greater than the fitness of a strategy T.

11. SOLVING BLUFF WITH A TIT FOR TAT APPROACH

Nash equilibrium is a set of strategies, where each player's strategy is optimal and no player has incentive to change his or her strategy given what other players are doing.

According to Nash's Theorem, the game of Bluff is bounded by finite number of players with finite strategy space and therefore there exists at least one Nash Equilibrium. When the players play honestly without challenging, Nash Equilibrium is achieved and can be best explained by, what you are doing is optimal based on what I am doing with no regrets for both players.

Table 8 is a simple payoff matrix for Player X and Y at a turn M to illustrate the possible reward and penalty.

- (2, 2) – The state is Nash equilibrium because no player has incentive to change his or her strategy given what the other players are doing.
- (-3,3) – If player X bluffs and gets caught the penalty is maximum. Player Y has most payoffs if player X is caught bluffing.
- (2,-2) & (2, 2) – Player X has identical payoff for being honest. On the other hand Player Y has one strategy with Penalty of 2 and another with reward of 2.

Table 8 Payoff matrix of two player scenario

		Player Y	
		Challenge	**No Contest**
Player X	**Bluff**	(-3,3)	(1,-1)
	No Buff	(2,-2)	(2,2)

The Tit for Tat strategy, cooperates on the first move, and then replicates the action that its opponent has taken in the previous move. On the equilibrium path when matched with all-co-operate strategy Tit for Tat player always cooperate. On the off-equilibrium path Tit for Tat always defects after the first round, when matched against all-defect strategy. This gives Tit for Tat player with both the advantage of getting the full benefit of cooperation and of defecting when matched with players of different strategy.

If, On-Equilibrium payoff $\geq$ Off-Equilibrium payoff, then there is no incentive to choose to deviate from on-equilibrium path.

But if inequality doesn’t hold i.e., On-Equilibrium payoff $\leq$ Off-Equilibrium payoff, then it is profitable to deviate from the on-equilibrium path and adopt defecting strategy.

11.1 Combat of Tit for Tat player against different types of Bluff AI Players:

1. **Tit for Tat vs. No Bluff AI:**

   When matched against No Bluff AI, Tit for Tat player will always cooperate with No Bluff AI and exhibit similar behavior of No Bluff AI.

2. **Tit for Tat vs. Smart AI:**

   When matched against Smart AI, Tit for Tat player will cooperate most of the time, until Smart AI defects when it estimates a bluff. However Smart AI has higher chance of winning against the Tit for Tat player because it defects only when it calculates and estimates a bluff by the opponent. But when Tit for Tat defects it has only 50% chance of catching a bluff, therefore Smart AI strategy would dominate against Tit for Tat player.

3. **Tit for Tat vs. Reinforcement Learning AI:**

   Reinforcement Learning AI has similar strategy as of Smart AI. Therefore similar outcome is expected as of Tit for Tat player against Smart AI.

4. **Tit for Tat vs. Anxious AI:**

   When matched against Anxious AI, Tit for Tat player will cooperate in the beginning until Anxious AI defects when it suspects a bluff by the opponent, then Tit for Tat strategy will defect back in the next round. However when Anxious AI detects less than 3 cards with Tit for Tat player it defects all the time, which might create a chain of bluff calls between Tit for Tat and Anxious AI.

5. **Tit for Tat vs. Tit for Tat**

   When matched against itself, the tit for tat strategy always cooperates and takes On-equilibrium path.

12. CONCLUSION AND FUTURE WORK

In this project, we created four different AIs with different tactics. The No-Bluff AI started as the naïve agent and was not expected to produce many wins, but in fact it proved to be the most efficient strategy. The Smart AI was a good strategy and could beat all other AIs except the No-Bluff AI. While our Anxious AI indeed caught many true Bluffs, it got caught many times for false Bluffs and so did not produce a winning strategy to top the other players. The Reinforcement Learning AI indeed produced good learning results, but it could not show great results against a simple strategy which was to not lie as much as possible and not get caught, followed by the No-Bluff AI. We tested our agents and found that SAI strategy is the Evolutionarily Stable Strategy and this state is called Evolutionarily Stable State.

Currently our Reinforcement Learning AI learns the strategy of only one player. In future, it would be interesting to note if an AI could learn the strategies of multiple players and thus achieve more wins against them by using different strategies in different levels of the game.

Reinforcement learning lies between supervised learning and unsupervised learning and works on a reward and penalty system [9] as shown in Fig. 15. The agent is not explicitly told what action to take in a turn, but forced to take a decision that would yield the most results in the current turn. The training data is the reward for an action taken in a state and is sparse, delayed and not independent. To solve this problem they used experience replay mechanism, which randomly samples past moves from the set of all past moves, to smooth out any irregularities in the distribution. The action to be taken in this turn is chosen randomly from among all the possible actions for the current state. Then the Q-value (where Q stands for quality) for the next state is calculated based on the function Q(s, a) which represents the maximum discounted reward (or the best score at the end of the game) when we take action a in state s. The Bellman equation denoted $\mathrm{Q(s,a) = r + \ \gamma max_{a'} Q(s^{'}, a')}$ is used to approximate the Q-function. The Q value is calculated for each turn and stored in Q-table. Recent work by same team [10] involving neural networks instead of Q tables has given much better results with minimal history.

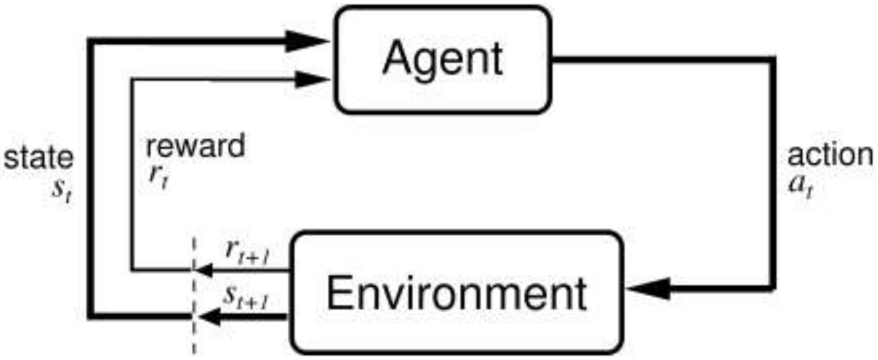
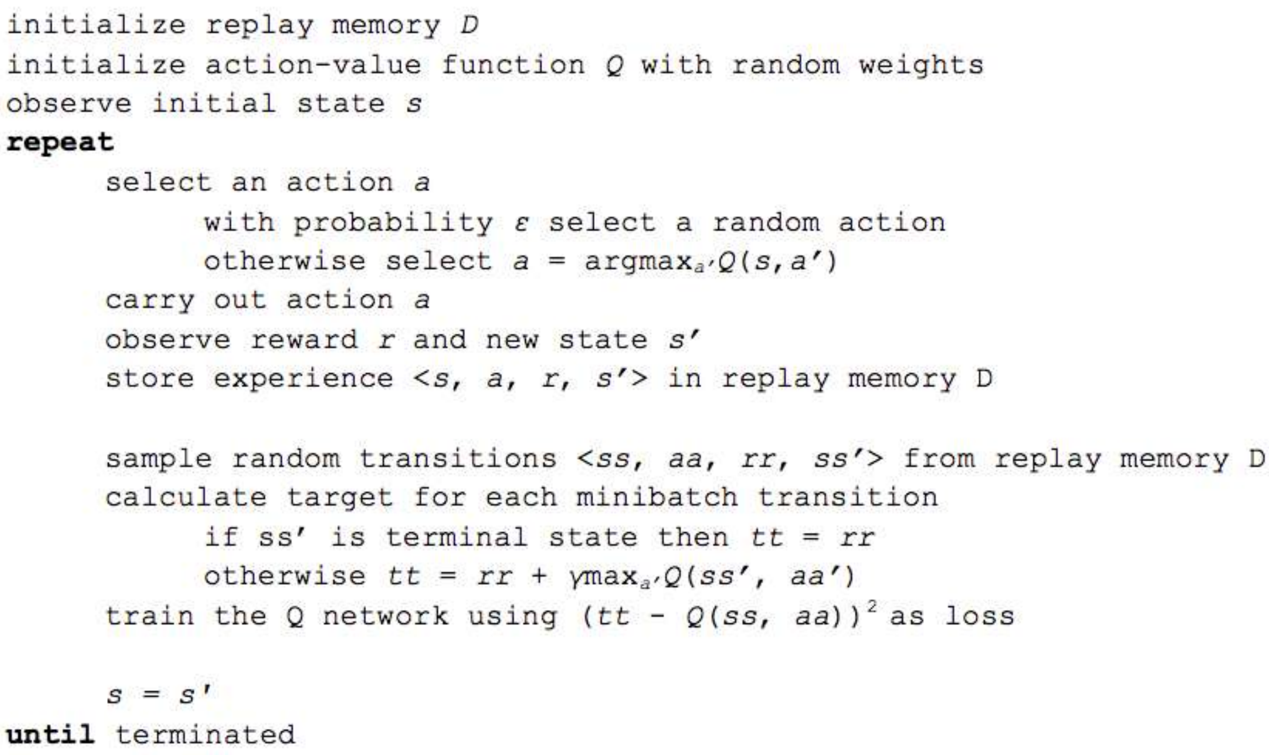


Fig. 15. The reinforcement learning problem

To improve our existing learning agent, the Deep Q-Learning agent with experience replay as shown in Fig. 16 can be used. Even though we may consider only very few parameters to train the agent, we can see that the resulting number of states are quite large. Consider the example where only 2 players are involved and we check the states based on the cards in each player's hand. The number of different states would be:

$\boldsymbol{\int_0^{52} (52!/(52-k)!) * (52-k) \approx 5 * \ 10^{63}}$.



```
initialize replay memory $D$
initialize action-value function $Q$ with random weights
observe initial state $s$
repeat
      select an action $a$
            with probability $\varepsilon$ select a random action
            otherwise select $a = \mathrm{argmax}_{a'} Q(s,a')$
      carry out action $a$
      observe reward $r$ and new state $s'$
      store experience $<s, a, r, s'>$ in replay memory D

      sample random transitions $<ss, aa, rr, ss'>$ from replay memory D
      calculate target for each minibatch transition
            if ss' is terminal state then $tt = rr$
            otherwise $tt = rr + \gamma \mathrm{max}_{a'} Q(ss', aa')$
      train the Q network using $(tt - Q(ss, aa))^2$ as loss

      $s = s'$
until terminated
```

Fig. 16. Deep Q-Learning algorithm with experience replay

Two learning algorithms would have to be implemented since there are two different decisions for the agent to make, namely:

i. Which card to play and
ii. When to call bluff.

It would be best to consider taking an action only based on the number of cards in the players hand before and after each action, since this is the aim of any player in the game. Each state could be considered as a terminal state, rather than waiting till the end of the game to identify the winner.

13. REFERENCES

[1] "Cheat - Card Game - GameSlush," Gameslush. Nov. 18, 2017. [Online]. Available: https://www.gameslush.com/cheat

[2] E. Hurwitz and T. Marwala, "Learning to bluff," 2007 *IEEE International Conference on Systems, Man and Cybernetics*, Montreal, Que., 2007, pp. 1188-1193.   Available: http://ieeexplore.ieee.org/document/4413589/

[3] D. Billings, "Algorithms and assessment in computer Poker," *University of Alberta.* 2007 Available: http://poker.cs.ualberta.ca/publications/billings.phd.pdf

[4] E. Hurwitz and T. Marwala, "A Multi-agent approach to Bluffing," 2009 Salman Ahmed and Mohd Noh Karsiti (Ed.), InTech, DOI: 10.5772/6603.   Available: https://www.intechopen.com/books/multiagent_systems/a_multi-agent_approach_to_bluffing

[5] J.Colton, "How many samples do you need to be confident your product is good," 2017 Available: http://blog.minitab.com/blog/the-statistical-mentor/how-many-samples-do-you-need-to-be-confident-your-product-is-good

[6] J. Frost, "Regression Analysis: How do I interpret R-squared and assess the goodness of a fit," 2013. Available: http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

[7] Easley David and Kleinberg Jon. 2010. "Networks, Crowds, and Markets: Reasoning about a Highly Connected World*,"* *Cambridge University Press*, New York, NY, USA. [Online]. Available: https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch07.pdf

[8] E.V. Belmega, S. Lasaulce, H. Tembine, M. Debbah. "Game Theory and Learning for Wireless Networks: Fundamentals and Applications", *Academic Press*, *Elsevier*, pp.122-124, 2011. [Online]. Available: https://www.elsevier.com/books/game-theory-and-learning-for-wireless-networks/lasaulce/978-0-12-384698-3

[9] V. Mnih , K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available:

[10] http://arxiv.org/abs/1312.5602T. Matiisen, "Demystifying deep reinforcement learning,"*University of Tartu, Estonia*, 2015. [Online]. Available: https://www.intelnervana.com/demystifying-deep-reinforcement-learning/