PUSHPOP: A CARD GAME

BY

VISHAL SUBRAMANI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

MAY 2012

Copyright © by Vishal Subramani 2012

All Rights Reserved

ACKNOWLEDGEMENTS

I owe my deepest gratitude to my supervising professor, Dr. Hao Che who has guided me throughout my Masters as my mentor. His knowledge, approach to problem solving, simple and clear thinking are few of the qualities which I have tried to imbibe in myself. His unconditional support is the main reason why I choose to pursue my thesis.

I would like to thank my committee members, Dr. Mohan Kumar and Mr. David Levine who have influenced and inspired me in various stages of my Masters program. Thank you for taking interest my research and taking out time for my thesis defense.

My father Mr. R Subramani and my mother Mrs. Vasanthi Subramani have strived hard throughout their life so that I get a chance to pursue my dreams without any constraints. Their blessings are a driving force and reason for my success. I thank them from the bottom of my heart for all that they have done for me.

Far away from home I have my small family of close friends comprised of Irfan, Ebrahim, Amey, Shekhar, Akshay, Diya, Jaineel, Samaya and many more who have helped/tolerated me every way possible for this thesis to reach where it has reached today.

I would like to thank my boss Mr. Jeff Borden, owner of BrandProtectionAgency (BPA), and the entire staff at BPA for their unconditional support throughout my thesis.

April 16, 2012

ABSTRACT

PUSHPOP: A CARD GAME

VISHAL SUBRAMANI, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: HAO CHE

In this thesis we have designed and implemented an Android card game called PushPop. This game is meant to stimulate the interest of kids in math learning. The Android game has cards with mathematical signs instead of the traditional suits. To reach the goal kids have to perform a lot of mathematical operations which helps them learn maths in a fun and effective way.

A study was made to classify PushPop into the various games under the research umbrella of Game Theory. After the analysis we conclude that PushPop is a non-co-operative, symmetric, non-zero sum, simultaneously and imperfect information game which makes the approaches we have proposed very challenging.

All these properties make this game very interesting for research. We have designed and implemented 4 levels of the game with respect to the computer as a player. The simulation results include the win loss ratio of each level against the other.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS v

ABSTRACT vi

LIST OF ILLUSTRATIONS viii

LIST OF TABLES xi

Chapter Page

1. INTRODUCTION 1

1.1 Card Game for Kids 1

1.2 Relater Work 1

1.3 Game Theory 4

1.1.1 History of Game Theory 4

1.1.2 Basic Terminologies in Game Theory 4

1.1.3 Representation of Games in Game Theory 5

1.1.4 Types of Games in Game Theory 7

2. PROPOSED GAME MODEL 11

2.1 Deck of Cards 11

2.2 Distribution of Cards 11

2.3 Objective of the Game 12

2.4 Decision Making Process 12

2.4.1 Decision Process for Computer To Play Card 13

2.4.2 Decision Process for Computer To Select Card on Winning 13

2.5.1 Level 1 - Random Throw Random Select 15

2.4.3 Level 2 – Random Throw Intelligent Select 16

2.4.4 Level 3 – Ordered Throw Intelligent Select 17

2.4.5 Level 4 – Hand Strength Based Throw Intelligent Select 19

3. EXPERIMENTS RESULTS AND ANALYSIS 25

3.1 Experiments and Results 25

3.1.1 Level versus Level 25

3.1.2 Human Player versus Various Levels 30

3.2 Game Theory Analysis 31

3.2.1 PUSHPOP a Non- Co-Operative Game 32

3.2.2 PUSHPOP a Symmetric Game 33

3.2.3 PUSHPOP a Non- Zero-Sum Game 33

3.2.4 PUSHPOP a Simultaneous Game 34

3.2.5 PUSHPOP an Imperfect Information Game 34

3.3 Game Theory Representation of PUSHPOP 35

4. CONCLUSION AND FUTURE WORK 36

4.1 Conclusion 36

4.2 Future Work 37
4.2.1 Possible Future Levels 37
4.2.2 Possible Game Changes 37
4.2.3 Experimentation Activity Kids 37

REFERENCES 38

BIOGRAPHICAL INFORMATION 39

LIST OF ILLUSTRATIONS

Figure Page

1. Normal Form Representation 7

2. A Zero Sum Game with Equal Payoff 8

3. Distribution of Cards. 11

4. Which Card to Throw. 13

5. Selected Card on Winning 14

6. Level 1 Performance against all other Levels 25

7. Level1 versus Level 2. 26

8. Level1 versus Level 3. 27

9. Level1 versus Level 4. 27

10. Average number of rounds 27

11. Win Loss ratio for Level 2 versus Level 3 and Level 4 28

12. Win Loss Ratio for Level 3 versus Level 4 29

13. GameTrace Level 4 vs Level 3 29

14. Human Player vs Levels win loss anaysis 30

LIST OF TABLES

Table Page

1. Normal Form Representation of PUSHPOP 34

2. Normal Form Representation of A in table 1 34

3. Normal Form Representation of B in Table 1 35

CHAPTER 1

INTRODUCTION

1.1 Card Game for kids

PushPop is a game envisioned to stimulate the math learning abilities of kids. In a traditional card game, the suits of the cards are Diamonds, Hearts, Clubs and Spades. We replace this by mathematical suits such as Addition, Subtraction, Multiplication and Division. This goal oriented game encourages kids to solve simple and tricky math operations to win the game. This game not only instills mathematical operations but also instills the impact of numbers and operations in Mathematics.

1.2 Related Work

Many attempts have been made to come up with new card games as educational tools for kids. The analysis of many games and the approaches used by the computer to simulate a player are also very interesting.

In the card game called "Hearts" described in [1], the authors use a partially observable Markov decision process to estimate the unobservable state variables and predict actions of opponent agents. In this game all the 52 cards are distributed among 4 players and the playing of cards is not simultaneous. With this information a player can make a definite prediction that the card which he/she does not possess is definitely present with one of the players. This reduces the imperfect information factor. Also since the players do not play simultaneously their decisions are influenced by the card choice of previous players. Thus in this case the player playing last has an undue advantage of the total information of the cards with which he has to compete. The game algorithm is rule driven based on merit calculation which makes it a rigid

approach. The noteworthy contributions of this paper are the state evaluation mode and the action prediction mode. In the state evaluation mode the environment variables are accessed. In the action prediction mode a merit is evaluated for each possible action which leads to the final decision chosen. It's an attempt to make an autonomous learning agent that plays this multi-player card game called "Hearts".

"Hearts" [1] made further contributions by introducing a one step utility prediction method which is also called a one step pessimistic prediction method. They trained the agent by using their Reinforcement learning method which proved to be better than the rule based agent after several thousand training games. The authors categorized their game according to Game theory concepts into two players, zero sum game where it is optimal for the agent to choose negative/ pessimistic approach in their prediction.

In one of the Fantasy Card game called "Magic" [3], the authors came up with a card game where the traditional suit of 52 cards was replaced by a special suit which comprised of cards with creatures which different skill levels and magical powers inscribed in them. The game consists of multiple varieties of cards and multiple types of resource, consequently the possible interactions between the available cards can become extremely complex. In the Monte Carlo search approach [4] [3], from the set of cards that the computer has and the remaining cards which the player has, all the possible combinations are formed. A play is simulated for both the computer and player's random card throw till someone wins. The winner of the game is recorded and a different simulation of the possible computer cards and player cards is initiated again. This is done for an arbitrary set threshold value. After that the simulation that recorded the least number of moves before completion is selected where the computer had won. This game is non-spontaneous and hence the player playing second has information about the card

played by the previous player. Once the number of cards increases, simulating a game tree for every possible legal card is very difficult.

In [6], the authors propose an interesting concept of emotional recognition model which tries to gather emotional information about the player such as facial expression, eyebrow positioning, voice confidence, smile detection etc. [6] [7] and try to figure out whether the player has good cards or bad cards in the game of poker. In the game of poker, the player just has 2 cards. Based on these 2 cards the player has a decent estimation of his winning chance. Hence his emotions can be captured in case of a confident bet and a bluff bet. A bluff is where the player knows he does not stand a good chance of winning but bets more so as to confuse the opponent into believing that he possesses good cards so that the opponents calls quits. However with enough understanding of the system and ample training, the player can fake emotions to go around the emotional recognition. Also if the cards distributed are more than 2 then the player himself will be confused about his winning chances and hence would not give clear indication of winning or bluffing. Also different levels of players have different emotional quotient which means some players are very expressive and some are very reserved. Hence it is difficult to keep same recognition model for all players.

In [4] the authors have described a class activity involving kids and have noted that using an electric card game increases the interest and knowledge of children in an effective way. They used a popular card game called ‘concentration’ and substituted the traditional RFID cards whit a computer system prototype. Through this demonstration they concluded that kids could effectively transform a binary number to its decimal equivalent.

The authors of [14] developed an educational card game named ‘Program and Programmers’ which teaches students software engineering concepts and life cycle. It was

concluded that when this card game was used in conjunction with lectures and projects it allowed students to gain deep understanding of real world lessons that were not understood earlier.

In the next section we shall look at a research domain which is closely linked with every game and other real life decision making scenarios to understand the decision making process and in what ways a game can be represented mathematically for analysis and also try to classify games based on their properties.

1.3 Game Theory

Game Theory is a mathematical tool which is used for strategic decision making. More formally it is 'the study of mathematic models of conflict and co-operation between intelligent rational decision makers where the participating decision makers must make choices that potentially affect the interest of other decision makers' [1]. Game theory model and concepts apply to a wide variety of fields where actions and decisions of several agents are interdependent. It provides a technical language to design, formulate, structure, analyze and understand strategic scenarios in the fields of Social Science, Political Science, Business Law and Economics, Strategic Auctions, Game designing etc.

1.3.1 History of Game Theory

The earliest known example of a game theoretic analysis was made by Mathematician Antoine Cournot in 1838 in the study of duopoly. Mathematician John von Neumann and Economist Oskar Morgenstern released the groundbreaking text in context to Application of Game Theory in the field Economics in their book '*Theory of Games and Economic Behavior*', published in 1944 by Princeton University Press. The basic terminologies and proofs

established in this book are used even today. The book is described as "the classic work upon which modern-day game theory is based."

1.3.2 Basic Terminologies in Game Theory

Game Theory is a preferred analytic tool since is represents a scenario in a formal representation using well defined and unambiguous mathematical objects. There are many terminologies associated with Game Theory which will be discussed over the span of this document. For the moment here are few of the basic terminologies required for representing a game:

- Game

  A game is a formal description of a strategic situation which requires analysis.

- Game theory:

  Game theory is the formal study of decision-making where the decision makers must make choices that potentially affect the interests of the other participants.

- Player

  A player is said to be a rational decision maker in the game who he seeks to play in a manner which maximizes his own payoff. It is often assumed that the rationality of all players is based on common knowledge.

- Payoff

  Payoff is a number or a utility function which is obtained as an intermediate result of interaction between the players and the game environment which are stepping stones in deciding the outcome of the final game at hand.

- Nash equilibrium

Nash equilibrium, also called strategic equilibrium, is a list of strategies which has the property that no player can unilaterally change his strategy and get a better payoff.

1.3.3 Representation of Games in Game Theory

The games of strategic scenarios which are studied under Game Theory have formal representation using well defined mathematical objects. Here are some of the classic representations of a game:

- Extensive Form

The Extensive form is used to formalize games with a time sequencing of moves by the different players. The data structure used to represent this for is a tree. Here each node represents a point of choice for a player. The player is specified by a number listed by the vertex. The lines out of the vertex represent a possible action for that player. The payoffs are specified at the bottom of the tree. The extensive form can be viewed as a multi-player generalization of a decision tree.

- Normal Form

The normal (or strategic form) game is usually represented by a matrix as shown in Fig 2 which shows the players, strategies, and pay-offs. It is generally represented by any function that associates a payoff for each player with every possible combination of actions. In the accompanying example there are two players; one chooses the row and the other chooses the column. Each player has two strategies, which are specified by the number of rows and the number of columns. The payoffs are provided in the interior. The first number is the payoff received by the row player (Player 1 in Fig 2); the second is the payoff for the column player (Player 2 in Fig 2). Suppose that Player 1 plays *Up* and that Player 2 plays *Left*. Then Player 1 gets a payoff of 4, and Player 2 gets 3. When a game is presented in normal form, it is

presumed that each player acts simultaneously or, at least, without knowing the actions of the other. If players have some information about the choices of other players, the game is usually presented in extensive form.

	Player 2 chooses *Left*	Player 2 chooses *Right*
Player 1 chooses *Up*	**4**, **3**	**–1**, **–1**
Player 1 chooses *Down*	**0**, **0**	**3**, **4**

Fig 1 Normal Form Representation

1.3.4 Types of Games in Game Theory

Game Theory is a general research domain which has different types of games under it. These games differ on the basis of the environment of the game, the sequencing of the moves between players and the goal set for the players.

- Co-operative versus Non-Co-operative

A game is *cooperative* if the players are able to form binding commitments. For instance the legal system requires them to adhere to their promises. In non-cooperative games it is not possible for the player to form binding commitments. In co-operative games, solving the game is the main concern and hence a common goal/meeting point is set up for all players so that they co-operate with one another to solve the game at large.

- Symmetric versus Asymmetric

A symmetric game is a game where the payoffs for playing a particular strategy depend only on the other strategies employed, not on who is playing them. If the identities of the players can be changed without changing the payoff to the strategies, then a game is symmetric. Many of the commonly studied 2×2 games are symmetric. The standard representations of chicken, the prisoner's dilemma, and the stag hunt are all symmetric games. Some scholars would consider certain asymmetric games as examples of these games as well. However, the most common payoffs for each of these games are symmetric.

- Zero-Sum and Non-Zero Sum

Zero-sum games are a special case of constant-sum games, in which choices by players can neither increase nor decrease the available resources. In zero-sum games the total benefit to all players in the game, for every combination of strategies, always adds to zero (more informally, a player benefits only at the equal expense of others). Poker exemplifies a zero-sum game because one wins exactly the amount one's opponents lose. Other zero-sum games include matching pennies and most classical board games like chess.

Many games studied by game theorists (including the infamous prisoner's dilemma) are non-zero-sum games, because some outcomes have net results greater or less than zero. Informally, in non-zero-sum games, a gain by one player does not necessarily correspond with a loss by another.

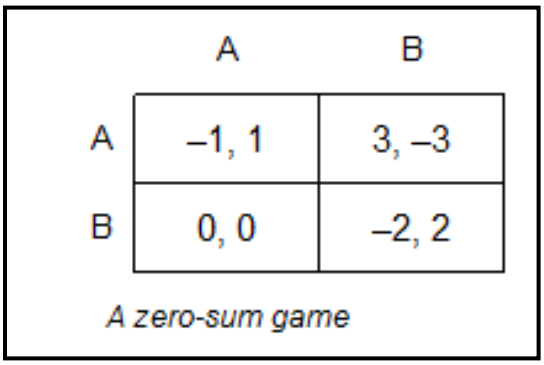

Fig 2 A zero Sum game with equal payoff

- Simultaneous versus Sequential

Simultaneous games are games where both players move simultaneously, or if they do not move simultaneously, the later players are unaware of the earlier players' actions (making them *effectively* simultaneous). Sequential games (or dynamic games) are games where later players have some knowledge about earlier actions. This need not be perfect information about every action of earlier players; it might be very little knowledge. For instance, a player may know that an earlier player did not perform one particular action, while he does not know which of the other available actions the first player actually performed.

The difference between simultaneous and sequential games is captured in the different representations discussed above. Often, normal form is used to represent simultaneous games, and extensive form is used to represent sequential ones. The transformation of extensive to normal form is one way, meaning that multiple extensive form games correspond to the same normal form. Consequently, notions of equilibrium for simultaneous games are insufficient for reasoning about sequential games.

- Perfect Information versus Imperfect Information

An important subset of sequential games consists of games of perfect information. A game is one of perfect information if all players know the moves previously made by all other players. Thus, only sequential games can be games of perfect information, since in simultaneous games not every player knows the actions of the others. Most games studied in game theory are imperfect-information games, although there are some interesting examples of perfect-information games, including the ultimatum game and centipede game. Recreational games of perfect information games Example: chess. Many card games are games of imperfect information, for instance poker or contract bridge.

- Combinatorial Games

Games in which the difficulty of finding an optimal strategy stems from the multiplicity of possible moves are called combinatorial games. Examples include chess and go. Games that involve imperfect or incomplete information may also have a strong combinatorial character, for instance backgammon. There is no unified theory addressing combinatorial elements in games. There are, however, mathematical tools that can solve particular problems and answer some general questions.

operations and chapter paste and rename processes until you have created a template that has as many chapters as your work has.

CHAPTER 2

PROPOSED GAME MODEL

2.1 Deck of Cards

The Game Push-Pop is a card game. The 1st version of this game currently consists of 20 cards in the entire deck. The numbers on the cards are {1, 2, 3, 4, 5} and the Suits are {Add(+), Subtract(-), Multiply(*), Divide(/)}. Hence in total the deck contains 20 cards {1 of Add, 1 of Subtract……5 of Divide}

2.2 Distribution of Cards

This is a two player game consisting of human player and computer player. The Game consists of multiple rounds. At the start of each round, 7 cards are randomly distributed to each player and remaining 6 cards are kept aside. In this game total uncertainty is maintained whereby each player only has access to his/her/its 7 cards and does not know about the 7 cards of the other player and the 6 cards which are kept out of the game. The figure below displays the game interface showing the distribution of cards.

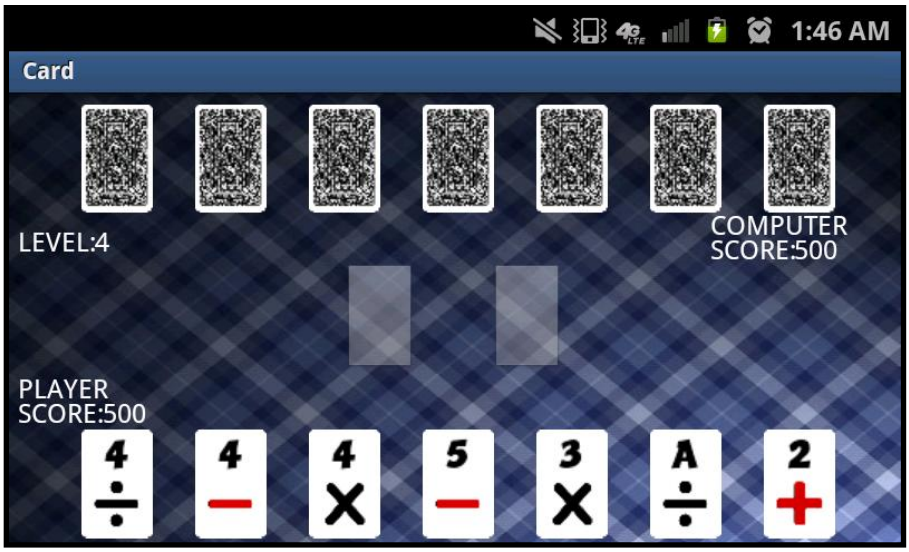

Fig 3 Distribution of Cards

2.3 Objective of the Game

At the start of every game during the 1st round, the player and computer have an initial score of 500 points each. The cards are distributed as mentioned in Section 2.2. Now both the player and the computer select a card simultaneously. So in other words both reveal their selected card simultaneously. This implies that neither of the two has prior information about the card the opponent is going to select. The 2 cards are revealed on the table. A comparison is made between the two cards based on the following rule:

- *Rule for Determining Winner*

If the numeric value of Card A is greater than numeric value of Card B then Card A wins else Card B wins. In case of a tie, the card with higher operator precedence wins. The precedence of the operators in higher order of precedence is {Divide, Multiply, Subtract, Add}. For example if player throws 4 of multiply and computer throws 3 of divide then player wins. If player throws 4 of multiply and computer throws 4 of divide then computer wins.

On winning a hand, the winner chooses either Card A or Card B and applies it to either his own score or the opponents score. For example if player throws 4 of multiply and computer throws 3

of divide then player wins. On winning Player chooses either 4 of multiply on his own score or 3 of divide on computer score.

- *Ultimate Objective of Game*

The Ultimate objective of the game is that the winner should either reach at least 35000 points or he/she/it should reduce the opponent to at most zero.

2.4 Decision Making Process

In this game the agents/players/participants have to make key decisions during the course of the game. The first decision they have to make is which card they should play for a particular game round. This decision depends on a lot of factors such as Current Score, gut feeling, certain strategy to predict the opponent's card which is going to be chosen etc.

On wining a round where the card put by participant A is greater than card put by participant B, the second decision is to be made which involves which card should participant A select after winning and on whose score should the card mathematically be applied.

2.4.1 Decision Process for Computer to play card

In this game, the computer can only see the cards dealt to it by a Random function generator which randomly distributes the cards as described in Section 2.2. The computer has to choose a strategy to pick one card during each throw of cards. The computer has no knowledge of the cards the player possesses. Neither does the computer know which cards are kept aside which are neither dealt to the computer nor to the player. The computer in this game chooses one of the 4 strategies based on the game level decided.

Fig 4 Which Card to Throw

2.4.2 Decision Process for Computer to select card on Winning

Once the Computer plays a card and wins, it has to make a decision whether to select its card or the player card and to whose score should it apply the selected card. This decision mainly depends on the pay-off the computer gets on choosing a particular card on a particular score. There are a lot of factors which have to be taken into account before making this decision which will be explained in the various approaches. The game currently has 4 levels and based on the level selected, the computer chooses a strategy for making the decision about which card needs to be mathematically applied to whose score.

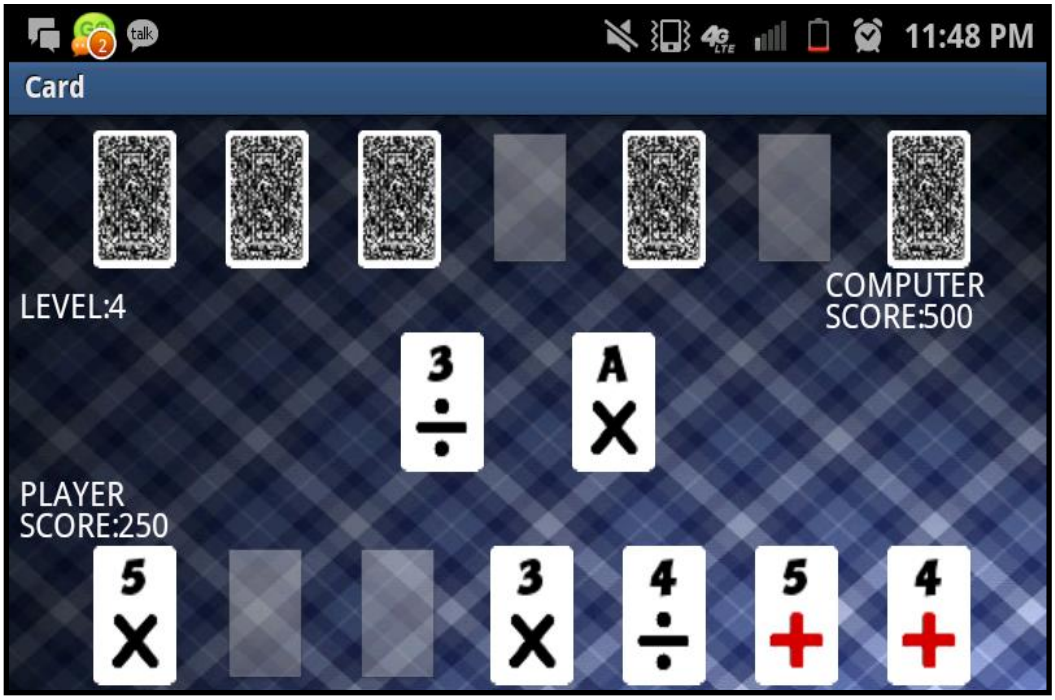

Fig 5 Selecting Card on winning

2.5 Levels of the Game

The game Push-Pop considers 4 different levels. The main goal of this exercise is to list the possible strategies in increasing order of optimality. Hence we would have Level 1 being the least optimal and level 4 being the most optimal. To prove the decisions as more optimal, in the later sections we would analyze all the levels using game theory principles and practically prove them by simulating games between different levels of the game and see how they fare against each other. Another simulation metric we use is that the game be played by different human groups and rate the different levels and see the win loss ratio among the 4 different levels. It must be realized that this analysis might vary from person to person so the results must be quantified for better results and understanding.

2.5.1 Level 1- Random Throw Random Select

In the level 1 of the game, at the beginning of the round, distribution of cards takes place as explained in Section 2.2. The computer in level one is acting as a dumb terminal which leaves everything to chance. Hence all the decisions pertaining to the computer are taken in a random order.

Pseudo-Code for Level 1:

Level 1 (Hand A)

1. X= Random(0 to [A.size-1]) //Select Random number between 0 and Hand size minus one
2. CardToBeThrown=A[X] // Select Card at position X
3. Play(CardToBeThrown)
4. Remove(A,CardToBeThrown) //Removes the card which is played from the hand A. And reduces size of Hand A by 1.
5. If (Win)

a. X1=Random(0,1)
b. If (X1==0)
    i. X11=Player Card
c. Else
    i. X11= Computer Card
d. X2=Random(0,1)
e. If (X2==0)
    i. Player Score= Player Score (X11.Operator) X11.Number //Update Player Score
f. Else
    i. Computer Score= Computer Score (X11.Operator) X11.Number // Update Computer Score

2.5.2 Level 2-RandomThrow Intelligent Select

In Level 2 some intelligence is induced which makes the computer competent to use a particular function which calculates which card selection on whose card will result in maximum payoff. The calculation of this function is based on a priority queue which has a list of all the cards in order of priority of pay off. In the algorithm a function call is made to CardSelectOnWinning(). The 2 parameters to this function are the cards thrown by player and computer. The return type of this function is of type Card. The 2 parameters passed to the function are compared to each other and the card which offers the maximum payoff is returned by the function. Once the card to be mathematically applied is selected, on whom should this card be applied is to be decided. For this we have a function OnWhom(). This function takes the card which was returned by CardSelectOnWinning() as a parameter and decided on whom should the card be applied to make the payoff beneficial for the computer. This function returns 0 for the card to be applied

mathematically to Player Score and 1 for the card to be applied mathematically to Computer Score.

Pseudo-Code for Level 2:

Level 2 (Hand A)

1. X= Random(0 to [A.size-1]) //Select Random number between 0 and Hand size minus one
2. CardToBeThrown=A[X] // Select Card at position X
3. Play(CardToBeThrown)
4. Remove(A,CardToBeThrown) //Removes the card which is played from the hand A. And reduces size of Hand A by 1.
5. CardSelect= CardSelectOnWinning(PlayerCardThrown, ComputerCardThrown)
6. X1= OnWhom(CardSelect)
7. If(X1==0)
    a. Player Score= Player Score (CardSelect.Operator) CardSelect.Number //Update Player Score
8. Else
    a. Computer Score= Computer Score (CardSelect.Operator) CardSelect.Number //Update Player Score

2.5.3 Level 3-Ordered Throw Intelligent Select

In Level 3 the computer adopts a strategy to throw a card in a particular order. The computer tries to gain advantage by adopting an Attack and Defend strategy to throw its cards. In this strategy, the computer sorts the hand in increasing order of the numeric value of each card. After sorting the cards, the computer divides the Hand Array in 2 equal sections. The left section consists of cards having lower numeric value than the cards on the right section. The computer toggles between the two sections and selects a card in random within the section to play the

card. If in the 1st throw of cards the left section is selected then a random card from the left section is selected and thrown as the computer card. If the Computer wins then the computer adopts the same strategy as in Level 2 to select either the player card or the computer card which is applied to either player score or computer score. Here a global Boolean variable 'Left' is used which is initially set to true. Whenever this variable is true, a random number is chosen between 0 to Hand Size/2 i.e the card to be thrown is selected from the left section and the Boolean variable is turned off (reset to false). When the variable 'Left' is false, a random number is chosen between Hand-Size/2+1 and Hand-Size i.e. the card to be thrown is selected from the right section and the Boolean variable is turned back on.

Pseudo-Code for Level 3:

Left = true;

Level 3 (Hand A)

1. Sort(A)
2. If(Left)
    a. X=Random(0 to [A.size-1]/2)
    b. CardToBeThrown=A[X]
3. Else
    a. X=Random([A.size-1]/2 to A.size)
    b. CardToBeThrown=A[X]

4. Play(CardToBeThrown)
5. Remove(A,CardToBeThrown) //Removes the card which is played from the hand A. And reduces size of Hand A by 1.
6. CardSelect= CardSelectOnWinning(PlayerCardThrown, ComputerCardThrown)
7. X1= OnWhom(CardSelect)
8. If(X1==0)

a. Player Score= Player Score (CardSelect.Operator) CardSelect.Number //Update Player Score

9. Else
    a. Computer Score= Computer Score (CardSelect.Operator) CardSelect.Number //Update Player Score

2.5.4 Level 4-Hand Strength Based Throw Intelligent Goal oriented Select

In Level 4 we use a small concept of level 3 for the partitioning of the hand. The left array of the Hand is the set of the weaker cards whereas the right array of the Hand is the set of stronger cards. According to Nash Equilibrium we play a card which is our best bet during the current scheme of things, even if it means we have to incur a negative payoff just to minimize the co-lateral damage. This level algorithm starts by sorting the hand in increasing order of the numeric value of each card. After that we calculate the hand strength of the computer cards and predict the hand strength of the Player Hand.

We do this using functions called *HandStrengthCalculatorForComputer()* and *HandStrengthPredictorforPlayer().* The first function *HandStrengthCalculatorForComputer()* takes in as parameter the current Hand array of the computer. It also takes in the current computer score and player score. In this function, all the cards which have an addition sign or a multiplication sign are applied to the current computer score. All the cards which have a subtraction sign or a division sign are applied to the current player score. After these operations, the ratio of the computer score to the player score is returned by the function *HandStrengthCalculatorForComputer()* which is stored in a variable ‘HSRC’. Before introducing the second function, let us define environment memory state array which is a global array and maintains the set of cards which are not present in the computer hand and the cards which are already thrown by the computer and the player. This global array keeps track of the cards which

the player might have and the cards which are kept out of the game. Initially this array has 13 cards. The computer can easily guess these 13 cards. Out of the total deck the cards which are not a part of the computer hand form the 13 cards. When the computer and the player play a card, the computer removes the card thrown by the player from this array. The function *HandStrengthPredictorforPlayer()* takes as parameter the current player score and the computer score and the size of the Computer hand. This function also has access to the global array which is maintaining the state of the remaining cards. The cards in the Global array are sorted in the ascending numeric value of each card. If the number of cards in the computer hand are even and suppose S, then from the beginning of the global array S/2 cards are chosen and from the ending of the global array S/2 cards are chosen to form an array of S cards which represent the average case card representation of the player. If the number of cards in the computer hand are odd, then from the beginning of the global array S/2-1 cards are chosen and from the ending of the global array S/2 cards are chosen and middle element of Global array is chosen to form an array of S cards which represent the average case card representation of the player. In this function, all the cards from this newly formed array which have an addition sign or a multiplication sign are applied to the current player score. All the cards which have a subtraction sign or a division sign are applied to the current computer score. After these operations, the ratio of the player score to the computer score is returned by the function. This ratio is stored in a variable say 'HSRP'.

Using the 2 functions *HandStrengthCalculatorForComputer()* and *HandStrengthPredictorforPlayer()* we have 2 ratios HSRC and HSRP which represent the approximate hand strength ratios of computer and player respectively. If the hand strength ratio of the computer is greater than the hand strength ratio of the player then the computer can predict that it has a better hand currently than the player so it would try to attack by choosing a random card from the right section of Computer hand. If the hand strength ratio of the computer is less than the hand strength ratio of the player then the computer can predict that it has a

weaker hand currently than the player so it would try to defend by choosing a random card from the left section of Computer hand. This is how the card to be thrown is decided.

In Level 4 intelligence is coupled with a goal oriented approach in which not only makes the computer competent to use a particular function which calculates which card selection on whose card will result in maximum payoff but also observes the goal to be achieved and checks if it can reach its goal by accepting a smaller payoff. The calculation of this function is based on a priority queue which has a list of all the cards in order of priority of pay off. In the algorithm a function call is made to CardSelectOnWinning(). The 2 parameters to this function are the cards thrown by player and computer. The return type of this function is of type Card. The 2 parameters passed to the function and are initially checked whether choosing a single card out of the two can help the computer to reach 35000 directly or reduce the player to zero directly. If so then the card which fulfills the goal is returned. If not then the two cards are compared to each other and the card which offers the maximum payoff is returned by the function.

Once the card to be mathematically applied is selected, on whom should this card be applied is to be decided. For this we have a function OnWhom(). This function takes the card which was returned by CardSelectOnWinning() as a parameter and decided on whom should the card be applied to make the payoff beneficial for the computer. This function returns 0 for the card to be applied mathematically to Player Score and 1 for the card to be applied mathematically to Computer Score.

Pseudo-Code for Level 4:

Hand GlobalEnvironment [];

Level 4 (Hand A, CurrentCompScore, CurrentPlayerScore)

1. Sort(A)
2. HSRC= *HandStrengthCalculatorForComputer(A, CurrentCompScore, CurrentPlayerScore)*

3. HSRP= *HandStrengthCalculatorForPlayer(CurrentCompScore, CurrentPlayerScore, A, A.size)*
4. If(HSRC<HSRP)
    a. X=Random(0 to [A.size-1]/2)
    b. CardToBeThrown=A[X]
5. Else
    a. X=Random([A.size-1]/2 to A.size)
    b. CardToBeThrown=A[X]

6. Play(CardToBeThrown)
7. Remove(A,CardToBeThrown) //Removes the card which is played from the hand A. And reduces size of Hand A by 1.
8. CardSelect= CardSelectOnWinning(PlayerCardThrown, ComputerCardThrown)
9. X1= OnWhom(CardSelect)
10. If(X1==0)
    a. Player Score= Player Score (CardSelect.Operator) CardSelect.Number //Update Player Score
11. Else
    a. Computer Score= Computer Score (CardSelect.Operator) CardSelect.Number //Update Player Score

Pseudo-Code for HandStrengthCalculatorForComputer:

HandStrengthCalculatorForComputer(A, CompScore, PlayerScore)

1. For X=0 to A.size-1
    a. If(A[X].ContainsAddOrMultiply)
        i. CompScore= CompScore (A[X].operator) A[X].value

b. Else

i. PlayerScore= PlayerScore(A[X].operator) A[X].value

2. Return CompScore/PlayerScore

Pseudo-Code for HandStrengthCalculatorForPlayer:

HandStrengthCalculatorForPlayer(CompScore, PlayerScore, A, size)

1. X=0; Y=0
2. If(size%2==0)
   a. For X=0 to size/2-1
      i. AveragePlayerHand[Y]= GlobalEnvironment[X]
      ii. X= X+1
      iii. Y= Y+1
   b. For X=GlobalEnvironment.Size to GlobalEnvironment.Size-1-size
      i. AveragePlayerHand[Y] = GlobalEnvironment[X]
      ii. X= X-1
      iii. Y=Y+1
3. Else
   a. For X=0 to size/2-1
      i. AveragePlayerHand[Y]= GlobalEnvironment[X]
      ii. X= X+1
      iii. Y= Y+1
   b. For X=GlobalEnvironment.Size to GlobalEnvironment.Size-size
      i. AveragePlayerHand[Y] = GlobalEnvironment[X]
      ii. X= X-1
      iii. Y=Y+1

c. AveragePlayerHand[size/2]= GlobalEnviornment[GlobalEnvironment.Size/2]

3. For X=0 to AveragePlayerHand.Size-1

c. If(AveragePlayerHand [X].ContainsAddOrMultiply)

ii. PlayerScore= PlayerScore (A[X].operator) A[X].value

d. Else

ii. CompScore= CompScore(A[X].operator) A[X].value

4. Return PlayerScore/CompScore

CHAPTER 3

EXPERIMENTS RESULTS AND ANALYSIS

3.1 Experiments and Results

Pushpop is designed as a two player game. The Experiments are divided into two modules. In the first module we conduct a set of experiments where each level plays against every other level. In the next module we conduct a set of experiments where humans play against every level. In all the experiments we record the win loss ratio, no of rounds for which the game ran before finishing and the scoring patterns based on graphs.

3.1.1 Level versus Level

We use Java programming language to code the simulation test bed in Netbeans IDE 6.9.1 environment. We have a console application where every level plays against different levels. In the first case, Level 1 lost against all the other 3 levels. The win loss ratio was heavily stacked against Level 1. In the Figure 6 we see the performance of level 1 against other levels. In this experiment, Level 1 played against all the other levels for 30 games.

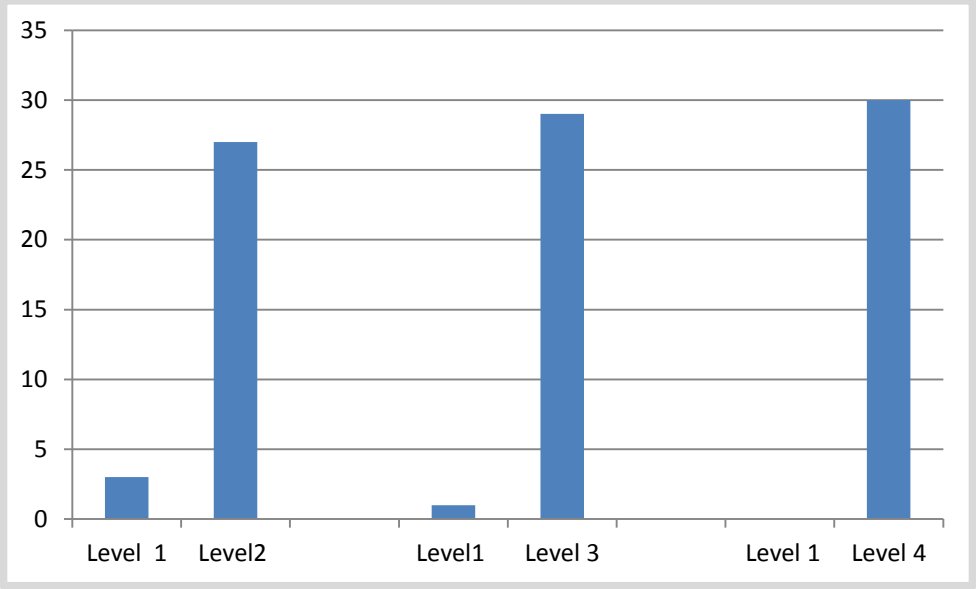

Fig 6 Level 1 Performance against other Levels

In Fig 7, we show one of the 30 game's play by play trace between Level 1 and Level 2 which went on for 8 rounds.

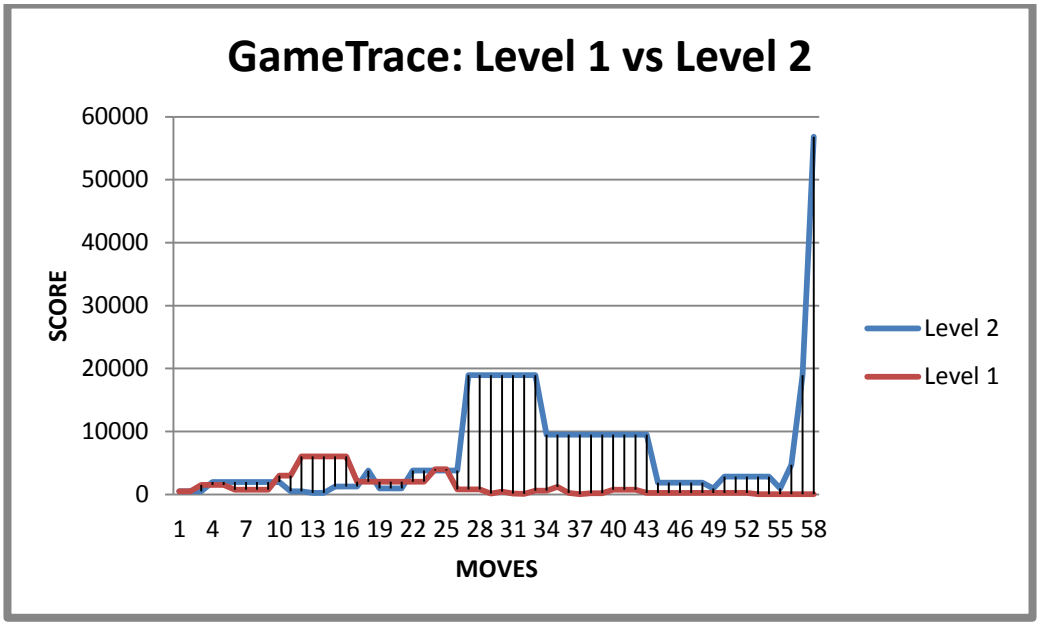

Fig 7 Game trace for play by play level1 versus level 2.

In Fig 8, we show one of the 30 game's play by play trace between Level 1 and Level 3 which went on for 5 rounds.

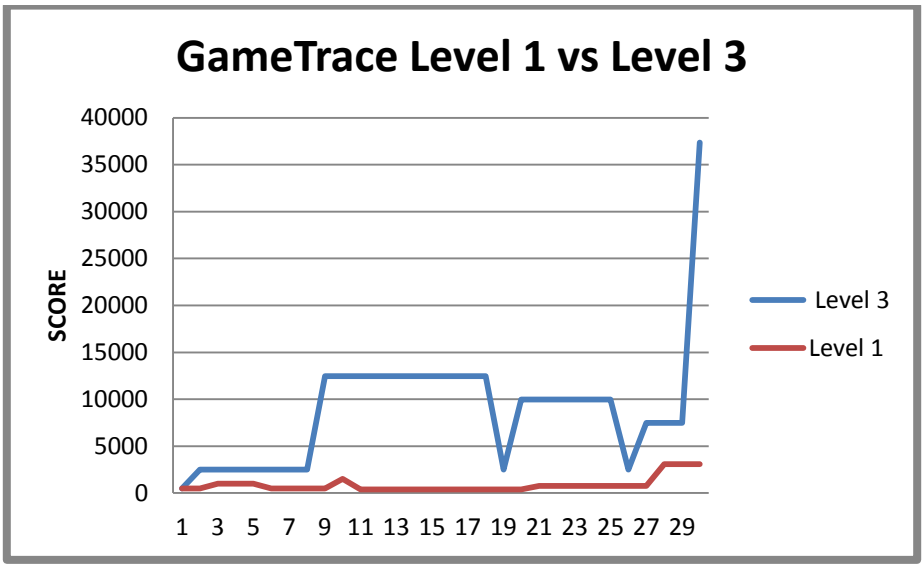

Fig 8 Game trace for play by play level1 versus level 3.

In Fig 9, we show one of the 30 game's play by play trace between Level 1 and Level 4 which went on for three rounds.

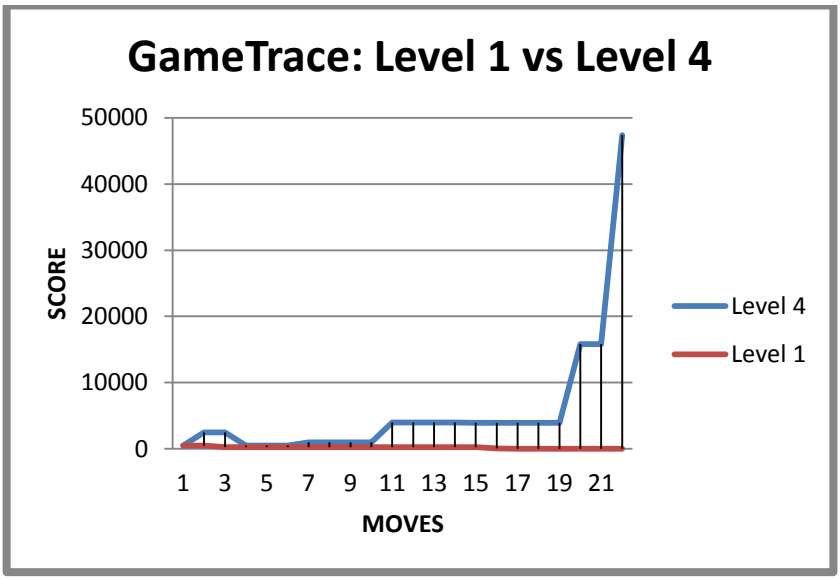

Fig 9 Game trace for play by play level1 versus level 4.

On average, the number of rounds by Level 2, 3 and 4 against Level 1 were ten, seven and four respectively as shown by Fig 10 below:

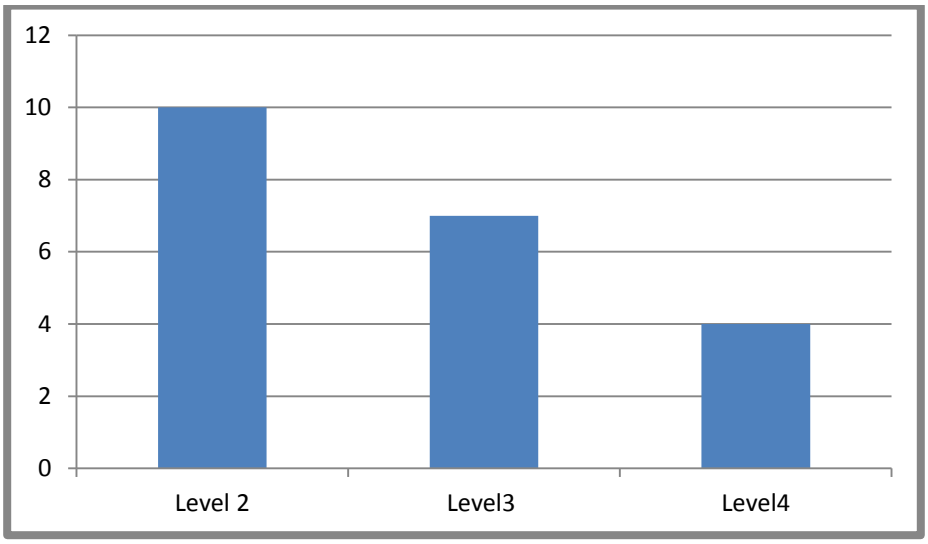

Fig 10 Average No of Rounds taken by other Levels to finish game against Level 1

With these results for Level 1 we can conclude that Level 1 lost against other Levels and one of the main reasons for this was the lack of intelligent decision making after winning the game. Hence in the 2nd Level we have random throw but intelligent decision of selecting appropriate card on winning. This helps it make better decisions. In the 2nd case we pit Level 2 against Level 3 and 4. The win loss ratio for level 2 versus level 3 was 9-21 respectively which indicates that Ordered selection of cards to be thrown lead to better results. The win loss ratio for level 2 versus level 4 was recorded at 6-24 which give a more optimized version of the algorithm. The Fig 11 shows the win loss ratio of Level 2 versus Level 3 and 4

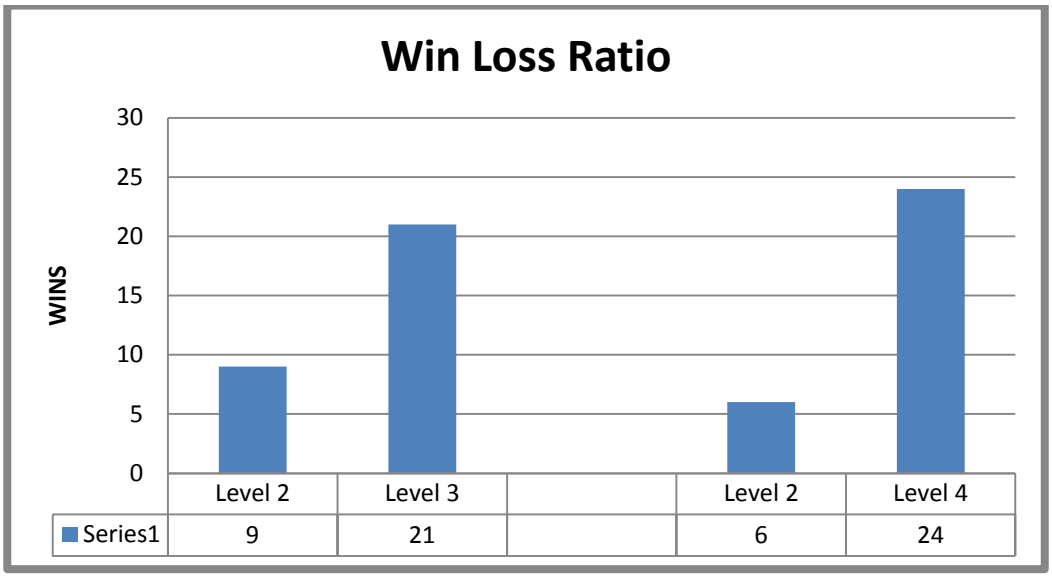

Fig 11 Win Loss ratio for Level 2 versus Level 3 and Level 4

In Fig 12 we have the win loss ratio of Level 3 versus Level 4. Out of the 30 games played between these two levels, Level 4 wins 20 games whereas Level 3 wins 10 games.

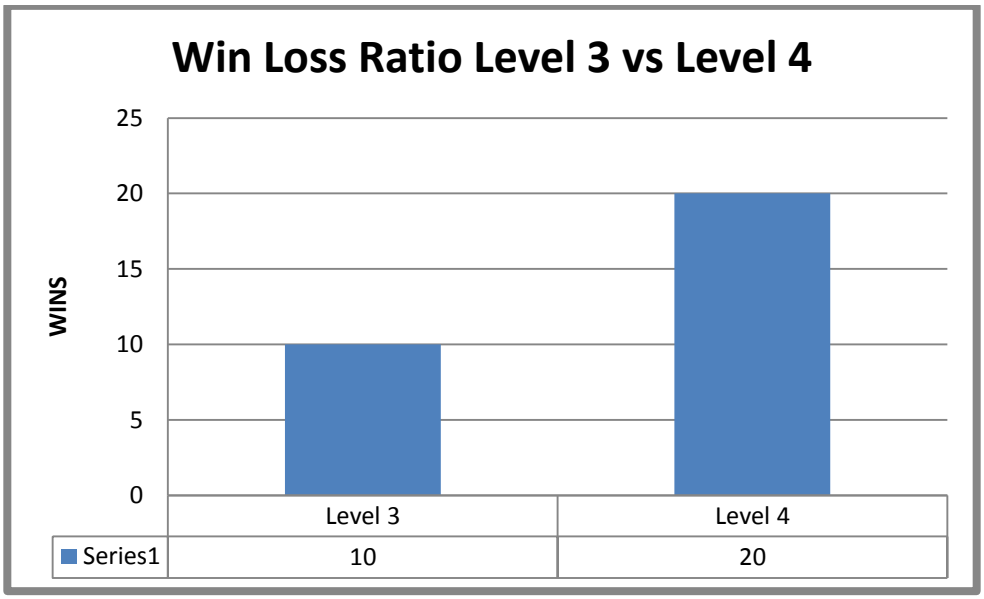

Fig 12 Win Loss Ratio for Level 3 versus Level 4

Fig 13 shows a GameTrace which is a move by move description of Game 10 out of the 30 games played between Level 3 and Level 4. This particular game lasted for 6 rounds and out of 30 games played between Level 3 and Level 4 the average number of rounds required to finish

the game was 7. According to the graphs Level 4 adapts its strategy based on the game scenario.

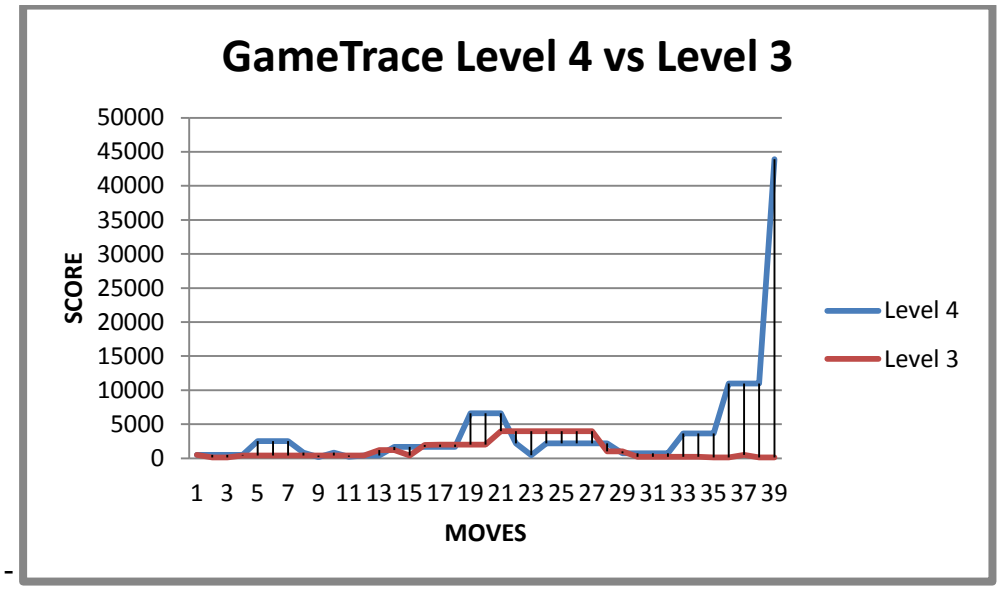

Fig 13 GameTrace Level 4 vs Level 3

3.1.2 Human Player versus Various Levels:

In the 2nd module of simulations, we conducted experiments on human players to see how the various levels fair against human players. We implemented an Android version of the game and installed it on the android phones of these human players. A human player is very adaptive dynamic and constantly changes his/ her strategies based on the game play. Let us understand our human player group's background to understand the results that we have obtained from them.

The average age of the players in the experiment was 24 and all the players were pursuing their graduate studies. So the results below should be analyzed taking into account the intellect and acumen of these players. In the experiment 5 players were chosen which are considered by us as 'experts' in playing card game and general analytical skills. All of them have a GRE average score of 1250 which is considered a very good GRE score. All the players were given adequate knowledge of the game rules before the experiments. They were given a

practice game for each level before actually recording results so that they can familiarize themselves with the game and understand the game scenarios that they will face during the experiments.

Each of the five players played each of the four levels 10 times. During this experiment the results obtained in terms of the wins for the human group and the levels is shown in Fig 14.

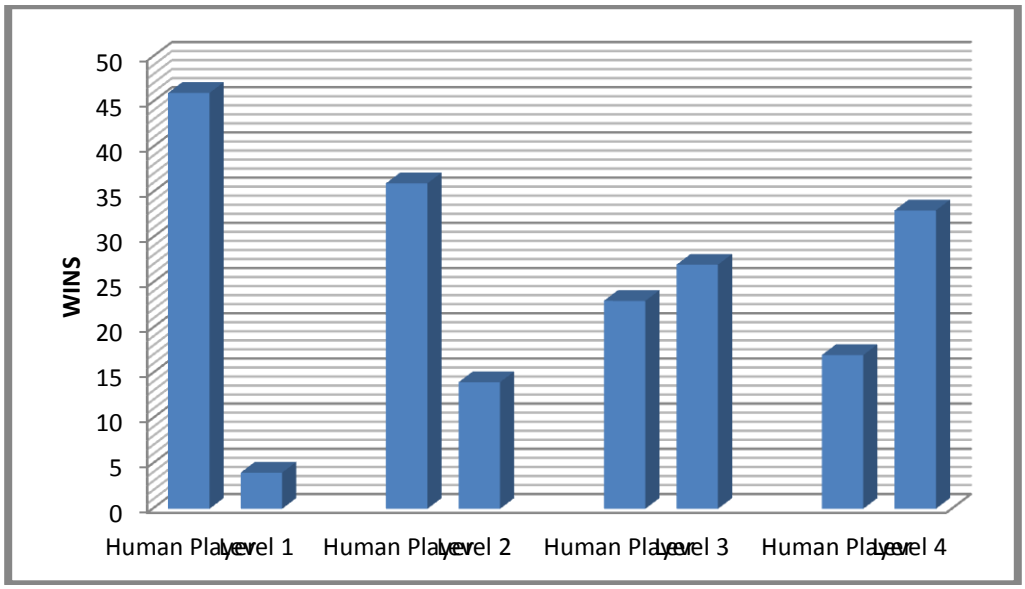

Fig 14 Human Player vs Levels win loss anaysis

3.2 Game Theory Analysis

Pushpop game is a non-co-operative, symmetric, non-zero sum, simultaneously and imperfect information game based on the game theory concepts. Let us analyse why this game has come under these categories.

3.2.1 PushPop a Non-Co-operative game

In a co-operative game, a group of players can form collaborations to solve the game at hand. In our game, the computer and the player compete against each other and hence they do not

form any collaboration to solve the game. Hence pushpop is a non- cooperative game. Traditionally finding a strategy in a co-operative game is easier as compared to a non-co-operative one because the collaborations formed during the game provide a lot of inferred information about the game environment. In pushpop the computer views the game environment the same way as the player views. In a co-operative game, the game's end point is kept common. For example in a game of chess, the game ends when either player can enforce a 'checkmate' (a game scenario where the opponent king has no legal move for survival) on the opponent's 'King'. By keeping a common goal, competing opponents solve the game in a co-operative manner. By killing other entities in the Chess board, the board is cleared for the ultimate Checkmate trap. In pushpop we set 2 opposite goals whereby the player tries to reach 35000 or drag the opponent to zero which makes the play non-cooperative.

3.2.2 PushPop a Symmetric game

In a symmetric game, the payoffs are independent of the player who played thee strategy to win that pay off. In pushpop a pay-off of 5 of multiply is applied to uniformly irrespective of the player who won this round. A symmetric game is very competitive as against an asymmetric game because in an asymmetric game a card would reflect as one pay-off to computer and some other pay-off to player. In the game called 'Ultimatum', an economic experiment is conducted between two players where one player initiates the contract by giving a proposal to player two. Player two has the right to reject the offer or accept the offer. In this game if the proposal is accepted by player two, the money won by the two players during the course of the game will be split according to the proposal made by player 1. So in this case the pay-off on winning is different for different players based on the agreement they had during the negotiation. In this case the player which is given the option of accepting or rejecting is in a powerful position than the player who makes the offer. This gives the second player an advantage which he can use to get maximum pay-off on common win between the two players.

3.2.3 PushPop a Non-Zero Sum Game

In a zero sum game, the total benefits to all the players, for every combination of strategies, always adds up to zero. In other words the amount of payoff won by a player is the exact payoff deducted from the opponent. In pushpop on winning there is a decision point where the player has to decide which pay off to choose on whose score. So in this case only one player's score is affected by the payoff. Hence pushpop is a Non-Zero Sum game. Non-Zero sum games are very popular research topics for game theorists like the prisoner's dilemma, because some outcomes have net results greater or less than zero. Informally, in non-zero-sum games, a gain by one player does not necessarily correspond to a loss by another. In chess when a player kills a pawn, the player gains a pawn kill count while the opponent losses a pawn kill count. Therefore chess is a Zero sum game.

3.2.4 PushPop a Simultaneous Game

In a simultaneous game, both the players play simultaneously which means while throwing the cards none of the participating players are aware of what the opponent's move is going to be. Due to this we have a wide range of strategies to choose from since we are not narrowed down to a set of cards based on the opponent move. This makes the algorithm which will choose the cards to be played very interesting.

3.2.5 PushPop an Imperfect Information Game

A subset of Simultaneous games is imperfect information games where the players are unaware of the move which is played by the opponent. Due to this the game tree approach is not possible in imperfect information games. In perfect information games like chess, game tree is a tried and tested approach which takes into account the moves played by the opponent in the previous move and plays his move accordingly to gain maximum payoff.

3.3 Game Theory Representation of PuspPop

As mentioned in the previous section, pushpop is an imperfect information game which implies that the player currently playing the card does not have prior information of the card which will be played by the opponent. Hence in this case we cannot use extensive form which is used when the time sequencing of moves is possible in a game tree environment. For pushpop we use the Normal form shown in table 1, 2 and 3 to represent the scoring and choice patterns.

Table 1 Normal Form representation of PUSHPOP

LeftPlayer\TopPlayer	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y

Table 2 Normal Form Representation of A in table 1

A	Addition, Subtraction (+/-)	Addition, Multiplication(Left +)	Addition, Division (Left +)
	Subtraction, Addition(-/+)	Multiplication, Addition(Top +)	Division, Addition(Top +)
	Multiplication, Subtraction (Left -)	Multiplication, Division (*//)	Division, Subtraction(Left -)
	Subtraction, Multiplication (Top -)	Division, Multiplication (//*)	Subtraction, Division (Top -)

Table 3 Normal Form Representation of B in Table 1

B	Addition, Subtraction (Left -)	Addition, Multiplication(Top *)	Addition, Division (Left /)
	Subtraction, Addition(Top +)	Multiplication, Addition(Left *)	Division, Addition(Top +)
	Multiplication, Subtraction (Left -)	Multiplication, Division (Left /)	Division, Subtraction(Left -)
	Subtraction, Multiplication (Top *)	Division, Multiplication (Top*)	Subtraction, Division (Left /)

Similarly we have the choice Normal forms for C, D,….Y. which represent all the possibilities that we encounter during the course of the game.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

In this thesis we have come up with an interesting Android card game as an educational tool for kids. We believe that this card game will help the learning abilities of kids in the field of mathematics. We are of the opinion that such gaming activity which can lead to the brain development of the kids should be strongly encouraged in schools.

During the course of this thesis we have discussed four possible approaches which transform to the four levels in the android game that we have developed. These levels were designed from the computer point of view which was competing to play against a human player. The hypothesis that Level 1 is the least optimal player and level 4 is the most optimal player was vindicated by the experimental results which suggest that these levels are correctly placed in their order of optimality with the least optimal termed as Level 1 to the most optimal which is the Level 4. Also the human simulations prove that some of the algorithms not only compete but also do better than human players.

A study was made to classify pushpop into the various games under the research umbrella of Game Theory. After the analysis we conclude that pushpop is a non-co-operative, symmetric, non-zero sum, simultaneously and imperfect information game which makes the approaches we have proposed very challenging.

4.2. Future Work

We divide the future work into the possible future levels that could improve the computer play better, the game changes that make the game more interesting and actual implementation of PushPop as an educational tool.

4.2.1 Possible Future Levels

One of the possible future levels could be that of ‘Player profiling’. In player profiling, before the start of the game, the name of the player is stored and a neutral component of the environment captures the Player hand and the manner in which the player plays the game. This component does not share any of this information with the computer during the game. However once the game is over, this neutral component stores the snapshot of the player hand and the choices this player made during the game in the knowledge base of the computer, During the consecutive game with the same player, the computer understands the game physic of this player and tries to modify its strategy based on a customized prediction.

Another possible future level could be obtained by deriving the Nash equilibrium from this game theory model. Nash Equilibrium is obtained by analyzing the game model to its entirety and coming up with a strategy which results in least negative pay off.

4.2.2. Possible Game changes

Our current version of the game supports 2 players namely the computer and the player. We could extend this game into a 4 player game by involving, 3 remote human players and a computer competing against them. With this we introduce a co-operative game scenario which could make the game more interesting with collaborations.

We can extend our deck size from 20 to 40 by having numbers from 1 to 10 instead of 1to 5 and also change the final threshold limit accordingly so the game lasts at least 2 rounds.

4.2.3 Experimentation Activity with Kids

We would like to conduct simulations on kids and track their progress report in mathematics to see if they mark an improvement and benefit from this game.

REFERENCES

[1] “A reinforcement learning scheme for a multi-agent card game”, Fujita, H.; Matsuno, Y.; Ishii, Systems, Man and Cybernetics, 2003. IEEE International Conference, Volume: 5, 2003.

[2] “Model-based reinforcement learning for a multi-player card game with partial Observability” Fujita, H.Ishii, S.Intelligent Agent Technology, IEEE/WIC/ACM International Conference, 2005.

[3] “Monte Carlo search applied to card selection in Magic: The Gathering Ward”, C.D.; Cowling, P.I., Computational Intelligence and Games, IEEE Symposium, 2009.

[4] “Determinization and information set Monte Carlo Tree Search for the card game Dou Di Zhu”, Whitehouse, D.Powley, E.J.; Cowling, P.I.Computational Intelligence and Games (CIG), 2011 IEEE Conference, 2011.

[5] “Do Children Understand Binary Numbers by Electric Card Game?” Mitsuhara, Hiroyuki; Ogata, Hiroaki; Kanenishi, Kazuhide; Yano, Yoneo Digital Game and Intelligent Toy Enhanced Learning, 2007. DIGITEL '07. The First IEEE International Workshop, 2007

[6] “Human-agent communication in poker game” Osone, K.; Gong Weiyi; Onisawa, T. Systems Man and Cybernetics (SMC), 2010 IEEE International Conference, 2012.

[7] “HoldemML: A framework to generate No Limit Hold'em Poker agents from human player strategies” Teofilo, L.F.; Reis, L.P. Information Systems and Technologies (CISTI), 2011 6th Iberian Conference, 2011.

[8] Roger B. Myerson (1991). “*Game Theory: Analysis of Conflict”,* Harvard University Press, 1991.

[9] R.J. Aumann ([1987] 2008). "Game theory," Introduction, The New Palgrave Dictionary of Economics, 2nd Edition. Abstract, 1987.

[10] Leyton-Brown, Kevin; Shoham, Yoav (2008), Essentials of Game Theory: A Concise, Multidisciplinary Introduction, San Rafael, CA: Morgan & Claypool Publishers, ISBN 978-1-59829-593-1, 2008.

[11] Osborne, Martin J. (2004), “An introduction to game theory”, Oxford University Press, 2004.

[12] “Combining UCT and Nested Monte Carlo Search for Single-Player General Game Playing”, Méhat, J.; Cazenave, T.Computational Intelligence and AI in Games, IEEE Transactions, 2010.

[13] "An analysis on the applicability of supervised learning algorithms on card games", Popeea, T.Constantinescu, A.; Radulescu, F.Rughinis, R.ICT and Knowledge Engineering (ICT & Knowledge Engineering), 2011 9th International Conference, 2011.

BIOGRAPHICAL INFORMATION

The author, Vishal Subramani, is pursuing his Masters in Computer Science from University of Texas at Arlington. He has maintained a 4.0 C.G.P.A. throughout his Masters. His majors are Networks and Information Security. He has represented UT Arlington at the AT&T foundry October 2011 pitch day event. He has over 2 years of Software Development experience with companies like BPA, Capgemini, Ayoka and Rasinfotek.

He would be graduating in May 2012 and would pursue a software development engineer role in the industry. I am keen to work with companies in the gaming industry to continue my research interest in Game theory.

His current role in Brand Protection Agency is of software developer. The company is into online Price monitoring through web crawlers and reporting tools.