Bot for Texas Hold'em
A REPORT

Xianli Li[1], Chi Zhang[2], and Xiang Gao[3]

[1] University of Waterloo 20545508 x464li@uwaterloo.ca
[2] University of Waterloo 20551692 c335zhan@uwaterloo.ca
[3] University of Waterloo 20457882 x39gao@uwaterloo.ca

1 Introduction

1.1 Texas Hold'em

Poker is a set of multi-player card games that is typically played as a session consisting of a sequential series of multiple games and each player begins the session with a certain amount of money. Poker is a multi-player non-deterministic zero-sum game with imperfect information, where the long-term goal is to net a positive amount of money. This is accomplished by maximizing winnings in each individual game within the session.

1.1.1 Notation of Poker Game

There are numerous variants of poker, and in this report we use the specific variant of Texas Hold'em. We now define the symbols that we use in this report:

- We use a standard deck of 52 cards (4 suits and 13 ranks per suit).
- We use the symbols 2(Deuce), 3(Trey), 4(Four), 5(Five), 6(Six), 7(Seven), 8(Eight), 9(Nine), T(Ten), J(Jack), Q(Queen), K(King) and A(Ace) for card ranks.
- We use the symbols ♢(Diamonds), ♣(Clubs), ♡(Hearts), and ♠(Spades) for card suits.
- A single card is represented by a pair of symbols, e.g. 2♢(Deuce of Diamond) and T♣ (Ten of Clubs)
- A set of cards is represented by a list separated by dashes, e.g. 4♣-5♣-6♣ (Four, Five and Six of Clubs).

1.1.2 Playing Texas Hold'em

Each poker game is consist of several rounds, which in turn involve dealing a number of cards followed by betting. This continues until there is either only one active player left in the game or when all the betting rounds have been completed. In the latter case, the game then enters the showdown to determine the winner(s). Betting involves each player contributing an equal amount of money to the pot. This amount grows as the game proceeds and a player may fold at any time, which means they lose the money they have already contributed and are no longer eligible to win the pot. When all players fold but one, the remaining player wins the pot. Otherwise, if the game proceeds to a showdown, the highest ranked set of cards held by

a player (the highest hand) wins the pot (ties are possible, in which case the pot is split). Note that poker is full of ambiguous terminology; for example, the word hand refers both to one game of poker and to a player's cards.

The specific variant under consideration in this report is Texas Hold'em, the most popular variant played in casinos. It is used in the main event of the annual World Series of Poker championship to determine the World Champion. It is considered to be one of the most strategically complex poker variants. The script for Texas Hold'em is as follows (each of the four rounds is followed by betting):

- **Pre-flop**: each player is dealt two face-down cards (hole cards).
- **Flop**: 3 cards dealt face-up to the board(community cards).
- **Turn**: 1 card dealt face-up to the board.
- **River**: 1 card dealt face-up to the board.

After the betting on the river, the best 5-card hand formed from the two hole cards and five board cards wins the pot.

1.1.3 Ante

Before the initial deal, participating players are required to blindly contribute a fixed amount of money (ante) to the pot. In some variants an alternative system is used where some of the players immediately following the rotating dealer (called the button) are forced to put in fixed size bets (called the blinds). Without these forced bets, risk can be minimized by only playing the best hand, and the game becomes uninteresting.

1.1.4 The Deal

Each round begins by randomly dealing 2 cards (the non-deterministic element of poker). In Texas Hold'em, there are community cards which are shared by all players (see figure 1). Each player receives the same number of cards - each of which is either face-down (known only to this player) or face-up (known to all players). The face-down cards are the imperfect information of poker. Each player knows their own cards but not those of their opponents.

1.1.5 Betting

The betting portion of poker is a multi-round sequence of player actions until some termination condition is satisfied. Without it your probability of winning the game depends solely on nature (the deal of the cards). Betting increases the pot and indirectly gives information about players and their hands. When playing against good opponents who pay attention to the actions of the other players, betting can also be used to give misinformation (the element of deception in poker).

Betting Order

The players are in a fixed seating order around the table (even in a virtual environment the set of players is referred to as the table, see figure 2). The dealer button rotates around in a clockwise fashion, as do betting actions. Betting always begins with the first to act, which in most games is the first active player following the button. Betting proceeds around the table, involving all active players, but does not end at the last active player.

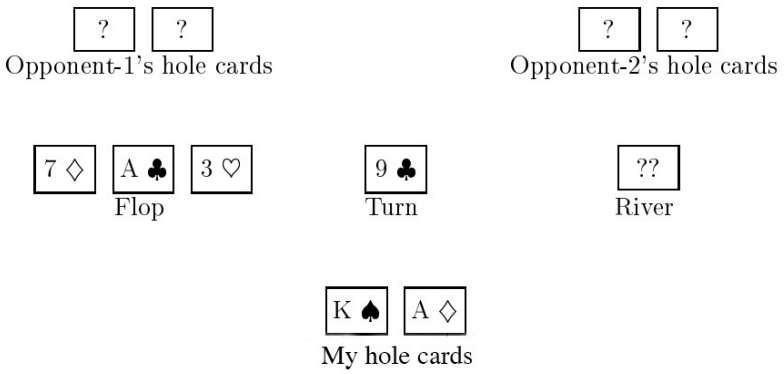

Fig. 1. A Texas Hold'em hand

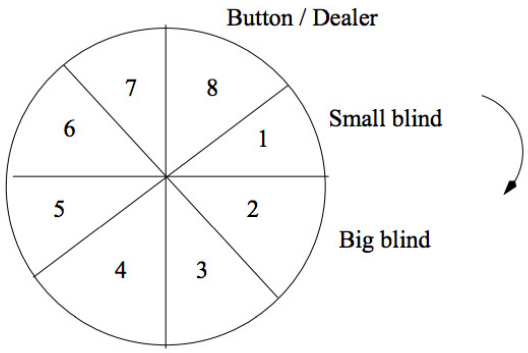

Fig. 2. Seat position and betting order

Termination Condition

Betting continues sequentially around the table until all active players have contributed an equal amount to the pot (or until there is only one active player remaining). The game then proceeds to the next round. In the final round the remaining players enter the showdown. Note that all players must be given at least one opportunity to act before betting is terminated (this allows for the case where all players have equally contributed $0). Being forced to put a blind bet in the pot does not count as having had an opportunity to act. Also, there often is a limit on the number of raises (increments to the contribution amount) which artificially forces an end to the betting.

Betting Actions

Each player has 3 different actions to choose from. Each action directly affects the number of active players, the size of the pot, and the required contribution to remain active in the game. Here are the 3 action categories and 5 different actions that fit into those categories:

– **Fold**: Quit from the game. A player who folds is no longer eligible to win the pot. The player is now out of the current game and loses the money that has been contributed to the pot.
– **Call**: Match the current per-player contribution (e.g. if player **A** has contributed $20 (the most) and player **B** $10, then **B** must place an additional $10, the amount to call, in the pot). A **check** is a special case of **calling** when the current amount to call is $0 (which is usually the case with the first active player). It means you forego opening the betting for the round.
– **Raise**: Increase the current per-player contribution (e.g. if player **A** has contributed $20 and player **B** $10, then **B** can put $20 into the pot (a raise of $10) to make the required contribution $30). A **bet** is a special case of **raising** when the current amount to call is $0. It means you open the betting for the round.

At any point in the game a player will have three actions available: $FOLD/CHECK/BET$ or $FOLD/CALL/RAISE$ (we use the later one here). An exception occurs in games with blinds where a player was forced to blind bet and everyone else $CALLS$ or $FOLDS$. Since the player was not originally given a choice, they are given an option to $RAISE$ when the action returns to them (the amount to $CALL$ is $0). The available actions are $FOLD/CHECK/RAISE$; $CHECK$ because it is $0 to $CALL$ and $RAISE$ because there has already been a BET (the blind). Another exception can occur because there is normally a limit of 4 $RAISE$s (including the initial bet or blind). If it is a player's turn and the betting has already been capped (no more raises allowed) the available actions are $FOLD/CALL$.

Betting Amounts

There are many different ways to restrict the betting amounts and the various systems can produce very different games (requiring different strategies).

– **No-limit poker**: this is the format used for the World Series of Poker championship main event. The amount a player is allowed to bet/raise is limited only by the amount of money that they have.
– **Pot-limit poker**: this format normally has a minimum amount and the maximum raise is whatever is currently in the pot (e.g. if the pot is at $40 and the amount to call is $20, a player can at most raise $60 by placing $80 in the pot).
– **Spread limit**: a format commonly used in friendly games. There is both a fixed minimum and maximum in each round (e.g. 1-5 Stud is a game where the raise or bet size can be between $1 and $5 in any round).
– **Fixed limit**: a common format used in casinos. There is a fixed bet size in each round (same as spread limit with the minimum equal to the maximum). Usually the bet size is larger in the later rounds. Games that feature the same bet size in all rounds are called **flat limit**.

Specifically, we examine the game of No-limit Texas Hold'em with a structured betting system blinds of 1 and 2 units.

1.1.6 Showdown

When there is only one active player remaining in the game, that player wins the pot without having to reveal their cards. Otherwise, when the final round terminates normally, the game enters the showdown where all active players reveal their cards and the player with the strongest 5-card hand wins the pot. In the case of a tie, the pot is split evenly.

Note that individual cards are ranked from Deuce - the lowest - to Ace. However, in most games, Ace can optionally be used as a low card (comes before Deuce instead of after King). The suit is not used in ranking (but is sometimes used for other purposes, such as awarding an odd chip when splitting the pot in ties). Here are all the 5-card hands ranked from strongest to weakest:

- **Straight Flush**: (e.g. A♠-K♠-Q♠-J♠-T♠) The strongest hand in regular poker - 5-card that form both a straight and a flush. Straight flushes are ranked by the top card in the straight (note that if Ace is used as low in an Ace to Five straight flush then the Five is the top card). An Ace-high straight flush (the highest possible hand) is called a Royal Flush.
- **Four of a Kind**: (e.g. A♠-A♣-A♢-A♡-K♡) 4-card of identical rank and one unmatched kicker. Compare four of a kinds by the rank of the 4 matched cards. The kicker is used to break ties (note that in Texas Hold'em, it is possible for multiple players to hold the same four of a kind).
- **Full House**: (e.g. A♠-A♣-A♢-K♢-K♡) 3-card of one rank and 2-card paired but of another rank. Compare first by the triple and then the pair in the event of a tie.
- **Flush**: (e.g. A♠-K♠-J♠-9♠-6♠) 5-card of identical suit. Rank multiple flushes first by comparing the top card, and then each subsequent card (e.g. A♠-K♠-9♠-5♠-2♠ is better than A♠-K♠-7♠-3♠-2♠).
- **Straight**: (e.g. A♠-K♠-Q♠-J♠-7T♢) 5-card in sequence. Straights are ranked by the highest card.
- **Three of a Kind**: (e.g. A♠-A♣-A♢-K♠-Q♠) 3-card of one rank with 2 kickers of unmatched rank. First compare the rank of the triple, and then examine each kicker (the higher one first).
- **Two Pair**: (e.g. A♠-A♣-K♠-K♣-Q♠) 2-card of one rank, 2-card of another, and one kicker of a third rank. Always compare by the highest pair first, then the second pair, and finally use the kicker.
- **One Pair**: (e.g. A♠-A♣-K♣-Q♣-J♣) 2-card of one rank with 3 kickers of unmatched rank (compare by the rank of the pair and then examine each kicker in order from the highest to the lowest).
- **High Card**: (e.g. A♠-K♠-J♠-T♢-9♠-) 5 unmatched cards. Compare by the highest to lowest cards, like a flush.

We use a special convention for representing Texas Hold'em hands. The designation 8♠-J♠/4♣-5♣-6♣ represents hole cards of 8♠-J♠ with a board of 4♣-5♣-6♣.

2 Related Work

2.1 Findler's work

One of the first studies of computer poker was done by Niolas Findler [1] [2]. The variation of poker used in his research was a simplified version of five-card draw poker. During the years Findler's project was carried out, various poker-playing programs were created, each different in its structure and approach to decision-making. Most of the computer players developed in Findler's research were based on simulating human cognitive processes involved in decision-making under uncertainty and risk. His approaches were based on psychological precepts of human thought rather than mathematically-oriented analysis. He did not produce a strong poker player.

2.2 Bayesian poker

Kevin B. Korb, Ann E. Nicholson and Nathalie Jitnah [3] at Monash University are working in the Bayesian Poker Program (BPP). BPP plays two-player five-card stud poker using a Bayesian network structure to represent the relationships between current hand type, final hand type (after the five-cards have been dealt) and the behaviour of the opponent. Given evidence for BPP's current hand type and the observed cards and cations of the opponent, BPP obtains its posterior probability of winning the game. BPP uses this estimated probability of winning the game to randomly select its action based on probabilistic curves for each betting action. BPP performs opponent modelling. It uses the relative frequencies of the opponent's betting actions to update the conditional probabilities per round of passing or calling versus betting or raising given the opponent's current hand type. BPP is work in progress as pointed out by Korb et al. The authors state that poker appears to be an ideal domain for investigating the application of Bayesian networks, and report positive results of BPP playing against a simple probabilistic program, a rule-based program and non-expert amateur human players.

2.3 University of Alberta

The work done by University of Alberta is the most impressive one [4] [5]and it describes work done on improving the knowledge representation, betting strategy, and opponent modelling of their poker-playing program. First, a randomized betting strategy that returns a probability triple is introduced. A probability triple is a probabilistic representation of betting decisions that indicates the likelihood of each betting action occurring in a given situation. Second, real-time simulations are used to compute the expected values of betting decisions. These simulations use selective sampling to maximize the information obtained with each simulation trial. Experimental results show that each of these enhancements represents a major advance in the strength of their former version and we based mainly on their ideas to construct our Bots.

3 AI Construction

3.1 Architecture

Figure 3 is a complete poker-playing program architecture (able to play a full game of Texas Hold'em unaided). There are three main co-dependent components which control the play of the program. These components are discussed in the following sections. They are hand evaluation (using the opponent models and game state, it generates values which roughly correspond to the probability of holding the strongest hand), betting strategy (it uses the values generated by hand evaluation, the opponent models, and the game state to determine the best action), and opponent modelling (it translates the betting history of the opponent into information about betting behaviour and possible hands held).

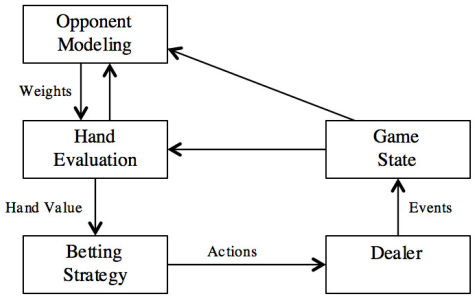

Fig. 3. Architecture

3.2 Hand Evaluation

Hand evaluation uses the opponent models and game state to calculate estimates of the winning chances of the cards in your hand. However, since there are more cards to come on the flop and turn round, the present strength of a hand is not a sufficient one. For this reason, post-flop hand evaluation is broken into two parts: strength and potential. Strength is the probability of a hand currently being the strongest and potential is the probability of the hand becoming the strongest after future cards have been dealt. Due to the computational complexity of potential for the pre-flop (with only the the first two hole cards), we use a simulation or pre-flop table look up evaluation in the preflop-round of the game.

3.2.1 Pre-flop Evaluation

For the general case of pre-flop evaluation, there are $\binom{52}{2} = 1326$ possible combinations, if we just want to use simple brute force to model all the 1326 possible combinations, it needs

approximately $\binom{52}{2} \times \binom{50}{3} \times \binom{47}{2} = O(50^7)$ comparing times. However, the 1326 possible combinations has only 169 distinct hand types (13 paired hands, $\binom{13}{2} = 78$ suited hands and $13 \times \binom{4}{2} = 78$ unsuited hands). Because this property, we can greatly reduce the comparing times needed to get the result to $\binom{48}{3} \times 169 \times 169 = O(50^3)$ still a great many comparing times, and we actually tried to run this and got a 1.6 GB file in the end and end up using more than 7 days to get that result. The most awful thing is that every time we want to do a pre-flop evaluation, we need to load the 1.6 GB database and using a hash or binary tree search to find the exact entry which is cumbersome and unnecessary.

Thus, we decide to take a method that is more quicker to do this. Basically, we have two methods to use to have a successful pre-flop decision. The first one is to use table learned by the expert system to give a probability table for each of the entries and just retrieve the data from the table and use this data to make a decision. We have constructed tables for 2-player, 3-player, 4-player and 5-player games and as the number of players increases, we find that the possibility of each combination is decreasing. This is reasonable as the more players involved in the game, the less likely one player would win from others with the same combination (not the absolute strong ones or the weak ones). Also, for the suited ones, we find they are more likely to win than the unsuited ones (In the table 1, 2 and 3, the lower triangle of the table represents the suited ones while the upper one represents the unsuited one).

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	0.504	0.000	0.024	0.052	0.047	0.061	0.123	0.191	0.263	0.339	0.424	0.507	0.569
3	0.076	0.552	0.078	0.109	0.105	0.119	0.142	0.213	0.289	0.367	0.450	0.519	0.582
4	0.125	0.179	0.600	0.163	0.158	0.170	0.202	0.236	0.314	0.393	0.475	0.532	0.595
5	0.152	0.206	0.259	0.647	0.213	0.230	0.259	0.289	0.337	0.421	0.501	0.548	0.608
6	0.148	0.203	0.256	0.303	0.687	0.282	0.309	0.346	0.388	0.438	0.513	0.560	0.609
7	0.163	0.215	0.269	0.321	0.370	0.730	0.358	0.397	0.438	0.491	0.526	0.573	0.625
8	0.226	0.240	0.296	0.346	0.393	0.438	0.772	0.444	0.494	0.520	0.551	0.585	0.639
9	0.283	0.309	0.326	0.380	0.428	0.475	0.511	0.812	0.519	0.546	0.575	0.610	0.652
T	0.351	0.377	0.401	0.420	0.472	0.508	0.533	0.557	0.854	0.572	0.603	0.638	0.680
J	0.425	0.450	0.472	0.500	0.507	0.532	0.556	0.580	0.605	0.889	0.614	0.650	0.692
Q	0.501	0.514	0.524	0.539	0.551	0.561	0.585	0.608	0.633	0.644	0.924	0.661	0.705
K	0.544	0.556	0.569	0.582	0.593	0.607	0.619	0.642	0.666	0.676	0.690	0.959	0.716
A	0.603	0.616	0.627	0.640	0.640	0.656	0.670	0.682	0.706	0.716	0.729	0.740	1

Table 1. Pre-flop expert table: 2 players

$$P = \frac{Number\ of\ Winning\ Times}{Total\ Simulation\ Times} \tag{1}$$

Except from the expert system method, we can, for each one of the 169 possible hand types, do a simulation of 10000 games against 2, 3 and 4 opponent, where each player is simple and always call to the end of the game and calculated the possibility of that particular combination wins using equation (1). And we can also get the same table as follow. By comparing, the tables that generated by the two methods are of the same trend, the difference is that the

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	0.362	0.000	0.035	0.065	0.033	0.009	0.052	0.094	0.157	0.205	0.279	0.368	0.505
3	0.192	0.452	0.107	0.145	0.113	0.093	0.078	0.127	0.185	0.237	0.312	0.401	0.517
4	0.221	0.292	0.516	0.220	0.194	0.173	0.159	0.150	0.213	0.268	0.346	0.437	0.527
5	0.251	0.328	0.396	0.548	0.272	0.260	0.242	0.231	0.242	0.307	0.381	0.472	0.537
6	0.219	0.298	0.371	0.440	0.582	0.335	0.327	0.320	0.327	0.335	0.415	0.502	0.531
7	0.203	0.271	0.353	0.426	0.500	0.620	0.413	0.410	0.422	0.425	0.450	0.515	0.547
8	0.243	0.268	0.340	0.417	0.497	0.521	0.661	0.500	0.504	0.507	0.514	0.527	0.562
9	0.282	0.311	0.332	0.410	0.494	0.521	0.544	0.708	0.533	0.536	0.543	0.537	0.577
T	0.339	0.368	0.389	0.421	0.495	0.524	0.548	0.575	0.759	0.572	0.582	0.594	0.613
J	0.398	0.423	0.453	0.486	0.502	0.526	0.552	0.579	0.612	0.810	0.596	0.611	0.631
Q	0.467	0.496	0.507	0.516	0.525	0.534	0.559	0.588	0.621	0.636	0.867	0.629	0.652
K	0.515	0.524	0.532	0.541	0.551	0.564	0.574	0.602	0.636	0.652	0.670	0.926	0.674
A	0.556	0.567	0.576	0.585	0.580	0.594	0.608	0.622	0.657	0.673	0.691	0.710	1

Table 2. Pre-flop expert table: 4 players

values in the simulation table is lower. The reason is that what we use in the expert system is a proportion of each combination's income rate to combination that has the highest income rate. Both of the two method is helpful in pre-flop decision, and in order to better use them we should give appropriate thresholds in betting decision respectively.

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	0.338	0.144	0.172	0.183	0.168	0.168	0.189	0.203	0.191	0.223	0.228	0.237	0.274
3	0.258	0.384	0.192	0.189	0.197	0.191	0.196	0.205	0.199	0.191	0.232	0.257	0.269
4	0.233	0.294	0.382	0.192	0.223	0.241	0.186	0.198	0.201	0.238	0.241	0.261	0.304
5	0.217	0.255	0.251	0.386	0.209	0.233	0.218	0.214	0.216	0.239	0.224	0.273	0.296
6	0.219	0.237	0.260	0.293	0.386	0.239	0.217	0.245	0.249	0.224	0.254	0.290	0.293
7	0.212	0.285	0.260	0.307	0.338	0.417	0.236	0.275	0.241	0.263	0.269	0.276	0.322
8	0.217	0.251	0.263	0.314	0.313	0.369	0.465	0.294	0.283	0.263	0.292	0.324	0.337
9	0.234	0.250	0.291	0.279	0.335	0.352	0.364	0.471	0.303	0.311	0.315	0.337	0.333
T	0.223	0.251	0.271	0.294	0.313	0.370	0.348	0.400	0.483	0.323	0.325	0.375	0.372
J	0.244	0.249	0.271	0.313	0.324	0.343	0.375	0.428	0.448	0.519	0.334	0.383	0.351
Q	0.215	0.267	0.283	0.317	0.350	0.341	0.408	0.409	0.426	0.477	0.550	0.371	0.373
K	0.215	0.258	0.270	0.298	0.306	0.333	0.348	0.408	0.461	0.445	0.502	0.598	0.392
A	0.247	0.245	0.274	0.276	0.303	0.312	0.383	0.411	0.410	0.475	0.503	0.540	0.636

Table 3. Pre-flop simulation table: 4 players

3.2.2 Hand Strength

Hand strength assesses how strong the card in your hand and it is in relation to what other players may hold. Critical to the program's performance, it is computed on the flop, turn

and river by a weighted enumeration which provides an accurate estimate of the probability of currently holding the strongest hand. Unlike the pre-flop one, this time, the calculation is feasible in real-time: on the flop there are 47 remaining unknown cards so $\binom{47}{2} = 1081$ possible hands may have in an opponent hand and on the turn and river $\binom{46}{2} = 1035$ and $\binom{44}{2} = 990$ possible hands respectively.

Algorithm 1 Hand Strength Calculation

```
win, tied, loss ← 0
ourrank ← Rank(ourcards, boardcards)
for eachcase(oppocards) do
    opporank ← Rank(oppocards, boardcards)
    if ourrank > opprank then
        win++
    else if ourrank == opporank then
        tied++
    else
        loss++
    end if
end for
handstrength ← (win + tied/2)/(win + tied + loss)
```

Algorithm 1 gives hand strength calculation. Suppose our starting hand is $A\clubsuit$-$Q\spadesuit$ and the flop is $3\heartsuit$-$4\spadesuit$-$J\diamondsuit$ (1,081 possible opponent hands). To estimate hand strength we need to count the number of possible hands that are better than ours (any pair, two pair, A-K, or three of a kind: 444 hands), hands that are equal to ours (9 possible remaining A-Q combinations), and hands that are worse than ours (628 hands). Counting ties as half, this corresponds to a hand rank (HR) of 0.585, that is, there is a 58.5% chance that our hand is better than a random hand. This measure is with respect to one opponent, as the opponent number grows larger, the hand rank is the power of the number of active opponents (HR_n is the hand rank against n opponents, $HR \equiv HR_1$).

$$HR_n \equiv (HR)^n \tag{2}$$

As the growing of players the HR_n would become very small and it is not a good idea to use it to make betting decisions, thus we should make compensation for HR_n to make it appropriate to be HS. We adopt a scheme to make HS affected less from the number change of players by equation (3). This equation gives a ratio to enlarger HR_n to make HS enough for betting decisions and c is a constant number that controls the enlarger ratio. We can control HS through changing c to make it more realistic.

$$HS = HR_n \times (1 + number\_of\_players\_left/c) \tag{3}$$

3.2.3 Hand Potential

In practice, only using hand strength is insufficient to assess the quality of a hand. Hand potential assesses the probability of the hand improving as additional community cards appear.

Consider the hand 8♣-7♣ with a flop of 9♣-6♢-2♣. The probability of having the strongest hand is very low, even against one random opponent (11.5%). On the other hand, there is a tremendous potential for improvement. With two cards yet to come, any ♣, 10, or 5 will give us a **Flush** or a **Straight**. Hence there is a high probability that this hand will improve substantially in strength, so the hand has a lot of value. We need to be aware of how the potential affects hand strength.

5 cards	7 cards			
	Ahead	Tied	Behind	Sum
Ahead	449005	3211	169504	$621720 = 628 \times 990$
Tied	0	8370	540	$8910 = 9 \times 990$
Behind	91981	1036	346543	$439560 = 444 \times 990$
Sum	540986	12617	516587	$1070190 = 1081 \times 990$

Table 4. Potential of $A\diamondsuit$-$Q\clubsuit$/3♡-4♣-$J\heartsuit$

We use definition introduced by [5] $PPOT$: the probability of pulling ahead when we are behind. These can be computed by enumeration in real-time. We have $\binom{47}{2} = 1081$ possible subcases on the flop and $\binom{45}{2} = 990$ on the turn. For example, the potential for $A\diamondsuit$-$Q\clubsuit$/3♡-4♣-$J\heartsuit$ is shown in table 4. The table shows what the result would be after seven cards, for cases where we are ahead, tied or behind after five cards. For example, if we did not have the best hand after five cards, then there are 91981 combinations of cards (pre-flop and two cards to come) for the opponents that will give us the best hand. Of the remaining hands, 1036 will leave us tied with the best hand, and 346543 will leave us behind. In other words, if we are behind we have roughly a $PPOT = 21\%$ chance of winning against one opponent in a showdown using equation (4).

$$PPOT = \frac{T_{B,A} + \frac{T_{B,T}}{2} + \frac{T_{T,A}}{2}}{T_{B,S} + \frac{T_{T,S}}{2}} \tag{4}$$

Based on the equation (4), we can show how to compute hand potential by algorithm 2.

Algorithm 2 Hand Potential Calculation

```
1: procedure HandPotential                                   ▷ index : ahead, tied, behind
2:     HP[3][3] ← 0
3:     ourrank5 ← Rank(ourcards, boardcards)
4:     for eachcase(oppcards) do
5:         opporank ← Rank(oppocards, boardcards)
6:         if ourrank5 > opporank then
7:             index ← ahead
8:         else if ourrank == opporank then
9:             index ← tied
10:        else
11:            index ← behind
12:        end if
13:        for eachcase(additionalboard) do
14:            board ← boardcards + additionalboardcards
15:            ourrank7 ← Rank(ourcards, boardcards)
16:            opporank ← Rank(oppocards, boardcards)
17:            if ourrank7 > opporank then
18:                HP[index][ahead] + +
19:            else if ourrank == opporank then
20:                HP[index][tied] + +
21:            else
22:                HP[index][behind] + +
23:            end if
24:        end for
25:    end for
26:    Ppositive ← HP[behind][ahead] + HP[behind][tied]/2 + HP[tied][ahead]/2
27:    Ptotal ← HP[behind][SUM] + HP[tied][SUM]/2
28:    PPOT ← Ppositive/Ptotal
29: end procedure
```

3.3 Betting Strategy

Betting strategy of poker games determines whether to $FOLD$, $CALL$, or $RAISE$ using the game state, opponent models and hand evaluation. But exactly what information is useful and how should it be used are not obvious. One approach would be using the game theoretic optimal betting strategy, despite the fact that it is very complex to calculate, and human opponents do not play optimally which may not be the best decision in practice.

The simplest betting strategy could be based only on hand strength (i.e. ignore hand potential and simply $CALL$ based on the immediate strength of our hand). Refinements to the betting strategy would involve the addition of high-level strategy concepts (slow-playing or bluffing). For each decision to be made, one of several variables ($PPOT$) is compared to some threshold (which may be based on another variable, Pot Odds). All of these refinements are intended to find quick ways to select the play which hopefully has the largest expected value (EV), since computing the exact EV is not feasible. There is a real-time constraint, in a single game of poker only takes a few minutes to play, and as we described before, the game tree can be very large.

Our betting strategy consists of the pre-flop betting strategy, a post-flop (flop, turn and river) betting strategy (using hand strength and hand potential), bluffing and calling with pot odds in post-flop rounds, and we will talk about betting strategies that based on opponent modelling in the later section.

3.3.1 Pre-flop Betting Strategy

The implemented pre-flop betting strategy is preliminary and makes use of expert information and mainly based on two things: pre-flop hand strength and player's position. We can construct a threshold equation on deciding the strategies based on positions and percentage of wins calculated by the pre-flop simulation or got in the table.

$$threshold(strategy) = percentageOfWins \times (1 + position/c) \tag{5}$$

The equation (5) mainly shows that the later you involved in this game, the more chance you can win, as you have more chance to observe the others' decisions and could make decisions with more information. Also, we add a judgement that if the player's position is over 2, which means that he is not involved in the blind betting, he would have a higher $FOLD$ threshold as he does not put any money in the pot and thus, easier to give up the not quite good starting hand. Following these rules, we provide a pre-flop betting strategy algorithm 3 that helping the AI to make the first judgement.

Algorithm 3 Pre-flop Betting Stategy

$threshold(strategy) \leftarrow percentageOfWins \times (1 + position/(2 \times numberOfplayers))$
if $position > 2$ **and** $percentageOfWins < 0.3$ **then**
    Return $FOLD$
end if
if $threshold > 0.5$ **then**
    $X = (threshold - 0.5)/0.6$
    $AMOUNT = e^{X-1} \times MONEY\_IN\_HAND$
    Return $RAISE$ **and** $AMOUNT$
else if $threshold < 0.5$ **and** $threshold > 0.25$ **then**
    Return $CALL$
else
    Return $FOLD$
end if

3.3.2 Post-Flop Betting Strategy

Effective Hand Strength

The majority of betting decisions are made based on a variable which represents the strength of hand in the current situation. Basic hand strength (HS_n) is the probability that it presently has the strongest hand. This alone is not fully adequate when there are more cards to come which can easily change the situation. For this reason, we compute the $PPOT$, which tell us

the probability of winning given that it is currently behind/ahead. Using these values we can compute an estimate of our bots' chances of winning at the showdown.

$$EHS = HS_n + (1 - HS_n) \times PPOT \tag{6}$$

We then define effective hand strength as equation (6). This is an estimate which means the probability that we are currently leading, or that we will improve to the best hand by the showdown. Algorithm 4 shows the normal post-flop decision process based on EHS.

Algorithm 4 Post Flop Betting Decision Based on EHS

```
if EHS > 0.8 then
    Return RAISE and AMOUNT
else if EHS > 0.4 then
    Return CALL
else
    Return FOLD
end if
```

Deception: bluffing

The decision to bluff is also a good strategy that is ranked only after EHS. Bluffing is used when the calculated EHS or percentage of wins calculated is small and usually, in which circumstance, we would just $FOLD$ to quit this game. By using bluffing, we attempt to deceive the opponents that we have a strong hand and win the game by make the opponent quit the game. The victory achieved by bluffing is usually not proceeding to the showdown part. However, as using bluff is a way of deception and it can not stop the opponent with strong hand continuing playing the game, we need to choose the time carefully when to use bluffing. Also, in order to counter detection of deception, we also need to set a bluffing rate deciding how frequently the bot would use bluffing.

In our settings, the action returned by bluffing is divided by the round we are currently in. As in $TURN$ and $RIVER$ round, players have more information and it is difficult to cheat them, if our EHS is still small in those two rounds the bluffing risk is much higher than the $FLOP$ round as there is no less new community cards to come. Thus, in those two round we bluff only when there are not many players left in the game and they do not $RAISE$ too much (This reflects that their hand are not so strong). Besides, we categorize the player to different level players according to their bluffing rate: Aggressive (80% chance of bluffing),Normal (50% chance of bluffing,bluffing in this manner has a positive side effect by contributing to deception, as it is like a random bluffing.), Conservative (20% chance of bluffing). Algorithm 5 show the procedure of a normal bluffing.

Besides bluffing, deception includes strategies such as slow-playing and check-raising. It causes the opponent to make wrong assumptions about the current state of the game. Deception can also be used to make a play that does not necessarily lead to the highest expected value for the current game but rather intended as to mislead the opponent which increases profits in future. To be complete, all possible actions that one can witness from the bot should

Algorithm 5 Bluffing Normal

```
1: n ← rand(0, 1)
2: if EHS < 0.2 and n > 0.5 then
3:     if BettingRound == FLOP then
4:         Return RAISE and AMOUNT
5:     else
6:         if Players left == FEW and Pot == FEW  then
7:             Return RAISE and AMOUNT
8:         else
9:             Return FOLD
10:        end if
11:    end if
12: else
13:    Return Decision using Other Strategies
14: end if
```

have a dual interpretation so no conclusions can be made with certainty. For example, in the present system, if the bot checks, a knowledgeable observer can infer that this bot has a hand with $EHS < 0.5$.

Pot Odds

Pot odds is also a useful element that contributes to the betting strategies. Pot odds is the comparison of player's winning chances to the expected return from the pot. For example, if there is only a 20% chance that the player has the best hand on $RIVER$, then how to make betting decisions? Assume the pot contains \$100 after the only opponent bets \$20. $CALL$ in this situation will lose four times out of five, at a cost of \$20 each time. However, we win once out of five for a profit of \$100. Therefore, under these assumptions, a $CALL$ is better than a $FOLD$, resulting in an average profit of \$4 per hand. However, if the pot only contained \$60, we obviously should $FOLD$, since $CALL$ would yield an average loss of \$4 per hand.

$$Pot\_Odds = \frac{to\_call}{pot\_size + to\_call} \tag{7}$$

On the $FLOP$ and $TURN$ round, the $CALL$ decision is based on $PPOT$. If $PPOT = 0.3$ which says that there is a 30% chance that the next card will give the player a very strong hand. As pot odds is the ratio of player's winning chances to the expected return from the pot, thus we can compare it to the potential winning to decide whether to $CALL$ or $FOLD$ using equation (7) which functioned in algorithm 6.

Algorithm 6 Post Flop Betting Decision Using Pot Odds

```
1: if BettingRound == RIVER and EHS > pot_odds then
2:     Return CALL
3: else if BettingRound != RIVER and PPOT > pot_odds then
4:     Return CALL
5: else
6:     Return FOLD
7: end if
```

3.4 Opponent Modeling

In Artificial Intelligence, especially in porker games, opponent modelling is always an significant part, and it can be the primary distinguishing feature between players at different skill levels. If a set of players all have a comparable knowledge of poker fundamentals, the ability to alter decisions based on an accurate model of the opponent may have a greater impact on success than any other strategic principle. The actual method of gathering information and using it for betting decisions is a complex and interesting problem. Not only is it difficult to make appropriate inferences from certain observations and then apply them in practice, it is not even clear how statistics should be collected or categorized. Actually, there are many studies of opponent modelling in Texas Hold'em.

3.4.1 Different Methods

Re-weighting

In [5], they use the betting history of the opponents, it determines a likely probability distribution for their hidden cards which is used by the hand evaluation system. A minimal system might use a single fixed model for all opponents in a given hand, based on the cards played to the flop by typical players. The system lets their bot generate a model for each opponent and maintains information between games. It is a simplistic first approximation that takes specific observed information and transforms it to a more useful form. However, this is sufficient to demonstrate an important performance improvement.

They construct weight-array of the opponent's hand for each opponent p, and this is an array of weights $w_p[h]$ where h represents each possible two card hand. There is a weight associated with each h (reset to 1 each new hand) which approximately represents the conditional probability that player p would have played in the observed manner (given that they held hand h). They call these values weights because they act as multipliers in the enumeration computations. If we want a probability distribution of possible hands held by p, we normalize the array by dividing each entry by the total sum of all the entries (the number of valid entries depends on which hands are still possible). The normalized entries $w'_p[h]$ represent the conditional probability that player p holds hand h (given the betting history).

$$Estimated\_Oppo\_EHS = \sum_h w'_p[h] \times HS_h \tag{8}$$

Every time we want to make a decision, we simply calculated the summation of the normalized $w'_p[h]$ and compared it with EHS in our hand (equation (8)). They use a reweighing

Algorithm 7 Re-weighting Function [5]

for *eachcase*(*hand*) **do**
  $reweights \leftarrow (handvalue - \mu + \delta)/(2 \times \delta)$
  if $reweights < lowest\_weight$ **then**
    $reweights \leftarrow lowest\_weight$
  end if
  if $reweights > highest\_weight$ **then**
    $reweights \leftarrow highest\_weight$
  end if
  $weight[hand] \leftarrow weight[hand] \times reweight$
  if $weight[hand] < lowest\_weight$ **then**
    $reweights \leftarrow lowest\_weight$
  end if
end for

function to retrieve information from history and the weight is based on interpolation in the range $\mu \pm \delta$.

Decision Trees

Decision trees have short training time and thus is very beneficial to be served as the baseline for opponent models. These decision trees could be trained in a matter of seconds on different sets of features, providing insight into the value of each. Some bots online use decision trees as their tools to construct their model base like Fell Oman.

Artifical Neural Networks

Most of the opponent modelling strategy remember a few action frequencies in a very limited context (i.e. bets-to-call and game stage). However, context which a player may use to assist in making decisions is far richer than just those two factors. For instance, the number of players in the game, the size of the pot, draw potentials, and their positions in the betting are all known to strongly affect the way a player will behave. By ignoring all of this context, we are discarding volumes of valuable opponent information. Different players will have different sensitivities to particular contexts. The problem we are faced with is sorting out what factors within all of this data actually affect a particular player's decisions.

[6] use a lot of context data into the network. This data is then fed into a standard feed forward ANN (also known as a multilayer perceptron) with a four-node hidden layer and three output nodes. The sigmoid activation function was used for all nodes in the network. Each of the three output nodes represents the networks prediction that an opponent will $FOLD$, $CALL$, or $RAISE$ respectively. An output node can give a real value from 0 to 1, so by normalizing the output nodes, we are given a ready to use probability distribution (known as 'Probability Triple' data structure [5]). By training the network on all of our data, and using back-propagation to perform a gradient descent on the connection weights in the network, the network begins to successfully discover the importance of each input feature with regards to the opponent's decision process.

#	type	description
1	Real	Immediate Pot Odds
2	Real	Bet Ratio: bets/(bets+calls)
3	Real	Pot Ratio: amount_in / pot_size
4	Boolean	Committed in this Round
5	Boolean	Bets-To-Call == 0
6	Boolean	Bets-To-Call == 1
7	Boolean	Bets-To-Call >= 2
8	Boolean	Stage == FLOP
9	Boolean	Stage == TURN
10	Boolean	Stage == RIVER
11	Boolean	Last-Bets-To-Call > 0
12	Boolean	Last-Action == BET/RAISE
13	Real	(#players Dealt-In) / 10
14	Real	(# Active Players) / 10
15	Real	(# Unacted Players) /10
16	Boolean	Flush Possible
17	Boolean	Ace on Board
18	Boolean	King on Board
19	Real	(#AKQ on Board) / (# Board Cards)

Table 5. Context information used to train the networks[6]. Boolean values are input as a 0 or 1, Real values as a real number from 0 to 1.

3.4.2 Our Method

We adopt a basic opponent modelling method using the betting history of the opponents, it determines a likely probability distribution for their hidden cards which is used by the hand evaluation system. A minimal system might use a single fixed model for all opponents in a given hand, based on the cards played to the flop by typical players. Our bot generates a model for each opponent and maintains information between games. It is a simplistic first approximation that takes specific observed information and transforms it to a more useful form. However, this is sufficient to demonstrate a significant performance improvement using only a simple analysis of context.

Specifically, for every opponent we store every hand that reaching to the showdown part with corresponding betting action, betting amount and the action occurred round and their hand strength. The betting action stored is that if this player had RAISE this round, then store RAISE, otherwise store $CALL$ (we do not store $FOLD$ here, as once the player take the FOLD action, his cards would not be shown to us). In oder to get more samples, we categorize the betting amount to: Few, $Moderate$, $Many$. Thus, for every opponent, we can put all the similar actions together (like below) to get our model ("1" stands for PRE_FLOP round, "2" stands for $FLOP$ round, "3" stands for $TURN$ round and "4" stands for $RIVER$ round). We can also calculate the *average* μ and *deviation* δ of the opponent's HS of that type and adding it to the property of that type.

- Player 3:
- $\mu = 0.253$ $\delta = 0.113$
  1: RAISE, Few; 2:CALL, Moderate; 3:CALL, Few; 4: CALL Moderate; Card: 2♣ 9♡

1: RAISE, Few; 2:CALL, Moderate; 3:CALL, Few; 4: CALL Moderate; Card: 5♣ J♣
1: RAISE, Few; 2:CALL, Moderate; 3:CALL, Few; 4: CALL Moderate; Card: 4♡ J♡
1: RAISE, Few; 2:CALL, Moderate; 3:CALL, Few; 4: CALL Moderate; Card: 6♠ 6♡
1: RAISE, Few; 2:CALL, Moderate; 3:CALL, Few; 4: CALL Moderate; Card: 5♣ J♡
. . . . . .

– $\mu = 0.383$ $\delta = 0.193$
1: CALL, Few; 2:RAISE, Moderate; 3:RAISE, Many; 4: RAISE Moderate; Card: 9♣ 9♡
1: CALL, Few; 2:RAISE, Moderate; 3:RAISE, Many; 4: RAISE Moderate; Card: K♣ 9♣
1: CALL, Few; 2:RAISE, Moderate; 3:RAISE, Many; 4: RAISE Moderate; Card: 10♣ 9♡
1: CALL, Few; 2:RAISE, Moderate; 3:RAISE, Many; 4: RAISE Moderate; Card: J♣ 2♡
1: CALL, Few; 2:RAISE, Moderate; 3:RAISE, Many; 4: RAISE Moderate; Card: 4♣ 7♢
. . . . . .

This way, we generate the model of the opponent. And in practice, every time when we want to make an action based on the model we construct, we would compare the action of the current round of a certain player and match it with the stored one to get the relative *average* and *deviation*. Once we get the average and deviation, we would first check that if this category has more than 10(or 5 if the number of players is small) samples. If this category has less than 10 samples of hands, we think that the model does not strong enough to clearly shows that when this player playing in this way that he has a high chance of this particular kind of hand strength indicated by the type constructed in the modeller. However, if there exist more than 10 samples of this type, we can say with a high probability that the indicating hand strength of this type is close to the hand strength in this opponent's hand right now. Once, we get to this, we would use the following algorithm to make our decision not solely based on the effective hand strength we calculated but add some criteria based the estimated hand strength that the opponent use.

Algorithm 8 Opponent Modeling Decision Making

if $EHS < \mu - \delta$ **then**
  Return $FOLD$
else if $EHS < \mu$ **then**
  Return $CALL$
else if $EHS > \mu + \delta$ **then**
  $n \leftarrow rand(0, 1)$
  $AMOUNT = 0.5 \times (1 + n) \times MONEY\_IN\_HAND$
  Return $RAISE$ **and** $AMOUNT$
else
  $X = (EHS - \mu)/\delta$
  $AMOUNT = e^{X-1} \times 0.5 \times MONEY\_IN\_HAND$
  Return $RAISE$ **and** $AMOUNT$
end if

4 Experiment

In the experiment, we construct 3 phases of bots:

- **Phase I**: use only hand strength to bet (pre-flop: use a simple decision function based on current hand; post-flop: use hand strength(with no hand potential information to bet).
- **Phase II**: use results from pre-flop simulation on pre-flop, use EHS(hand strength and hand potential) on post-flop, as well as use bluffing and pot odds. (Differentiate the players by different attitude towards bluffing: Normal, Aggressive and Conservative).
- **Phase III**: use techniques adopted by **Phase II** players and combined them with opponent modeling when making decisions (Differentiate the players by different attitude towards trust of model: Aggressive and Conservative).

4.1 General Comparation

All the techniques used by the bots are described in section 3, and we use a self play simulation letting different levels of players to compete with each other and get figure 4.1 using the majority data of 10 trails. For a curve that reaches 0, it means that the player was beaten and quitting the game in that round.

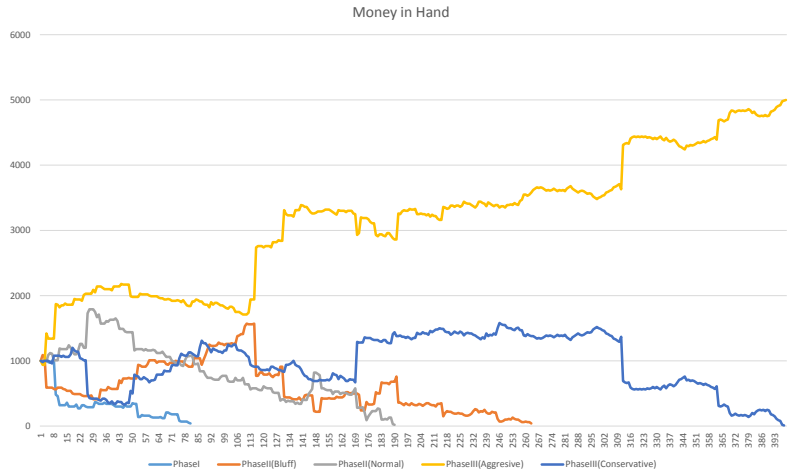

Fig. 4. Change of money in hand of each player

From figure 4.1 we can see clearly that the **Phase I** bot quits the game very early as making decisions based only on his hands is not sufficient and can only give bad performance;

Phase III is the best one, as all the **Phase III** players proceed to the final round while **Phase II** bot drops down earlier, which says that the opponent modeller does worked here.

4.2 Bluff Rate Comparation

As we differentiate the players by different attitude towards bluffing: Normal, Aggressive and Conservative, thus we would like to find out which is better. Thus we analysis this by simulations of those players.

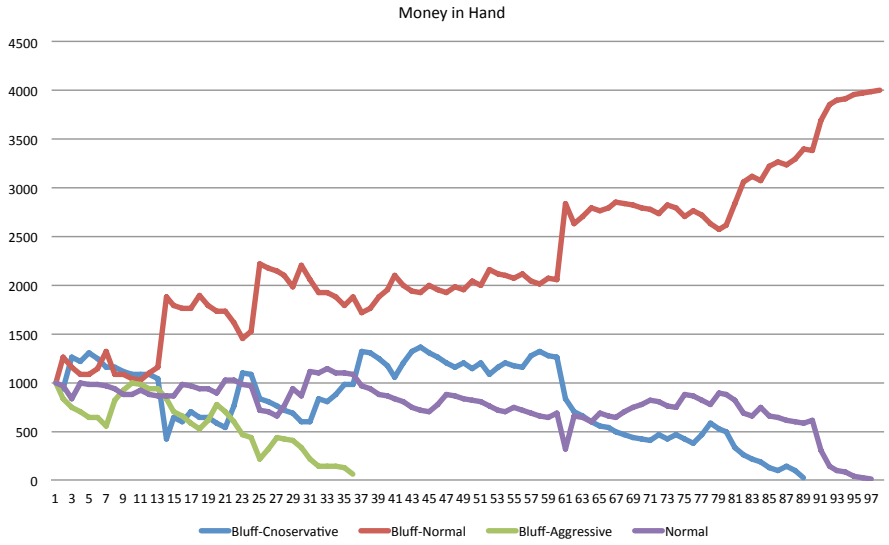

Fig. 5. Change of money in hand of each player

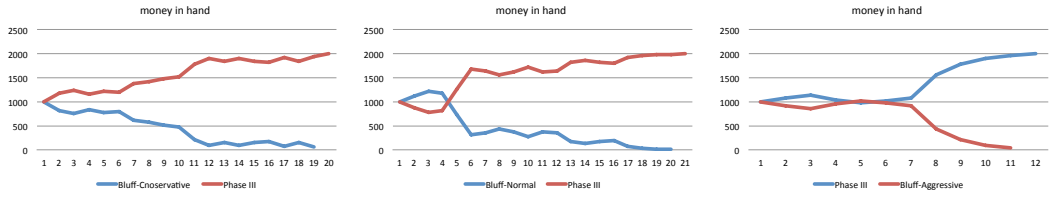

Fig. 6. Change of money in hand of each player

From the figure 6 we can see that using normal bluffing is better (summarize from 10 trails), and a higher and lower bluffing rate is not that good (even normal player without

using bluff could win). It is also the same thing when letting the **Phase II** Bluffing bot to compete with **Phase III** , although **Phase III** players still wins a lot, but it has less advantage over **Phase II** Normal Bluffing (bluffing rate 0.5, random bluffing), as random is more hard to learn than the bias one (we can see this by the following figures).

4.3 Trust of Model Comparation

For **Phase III** players, we categorize them by the attitude towards trust of models: Aggressive and Conservative. The Conservative player just do what we describe in algorithm 8, fully trust the models, while the Aggressive player acts a little bit different. Instead of $FOLD$ in the situation that $EHS < \mu - \delta$, it $RAISE$ by the probability of 0.5. We adopt this action by letting the Aggressive player consider abnormal behaviour of the opponent. Through competition of two **Phase III** players and each of them versus **Phase II** players, we can get from the figure that the Aggressive one is better than the Conservative one. We can conclude that the consideration of abnormal behaviour when using opponent modelling is important.

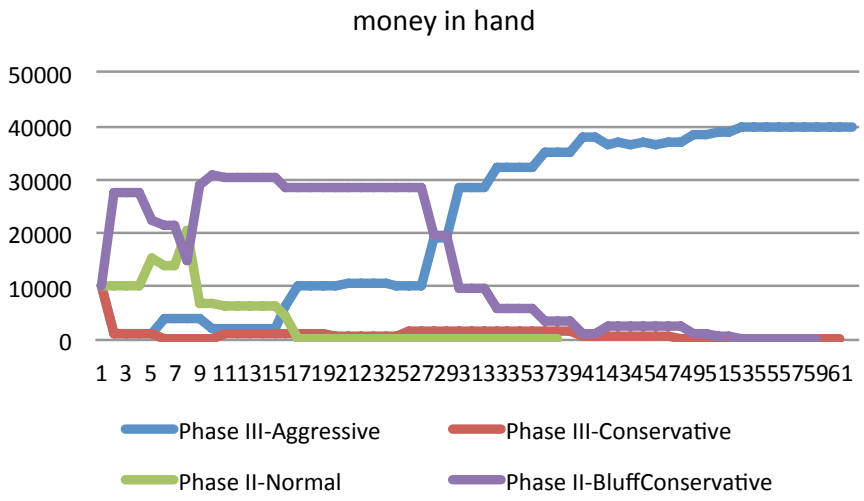

Fig. 7. Change of money in hand of each player

5 Conclusion

In the experiment, our bot successfully demonstrates the beneficial of opponent modelling in a high-performance game-playing program. In self-play simulations it was clear that using

modelling is absolutely better. However, it does not necessarily shows that it will be equally successful in games against strong human players. Humans can be very good at opponent modelling, and less predictable than the players in these experiments. Also, the not low winning rate of using bluffing shows that the techniques other than only hand strength is a good addition to decision making in Texas Hold'em.

Poker is an amazing game and it is also hard to analysis. Implementing bots for poker games needs tremendous loads of works, especially opponent modelling. It is the most difficult part as doing this is quite like thinking as human letting the player to handle all aspects of the game adequately.

6 Future work

We have made an basic bot of Texas Hold'em with opponent modelling and we are pleased with the results. Wherever possible, the final goal should be driven to remove whatever human expert information is used. Betting strategy is clearly a major component that needs to be addressed. However, there are also a lot of thing should be done such as more sophisticated opponent modelling. Eventually, more sophisticated simulations for learning good pre-flop play learned by the bot itself.

For betting strategies, we can adopt other strategies like slow-playing and showdown odds in our bot to decide if those strategies are useful to our bot and judge the power of it to our existing techniques. For pre-flop simulation, we should also adopt good ideas letting the bot learn the threshold itself rather than set by the expert system. And for opponent modelling, there are at least two models we could try next time as described in the opponent modelling part: decision tree and artificial neural networks. We can decide by self-simulation on which model is better. We hope that continuing this way, we can build a world best player one day.

References

[1] H. Iida, I. Kotani, J.W.H.M. Uiterwijk, and H.J. van den Herik. Gains and risks of OM search. In H.J. van den Herik and J.W.H.M. Uiterwijk, editors, Advances in Computer Chess 8, pages 153 -165, Maastricht, The Netherlands, 1997. Department of Computer Science, Universiteit Maastricht.

[2] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. Artificial Intelligence, 94 (1-2):167-215, July 1997.

[3] Kevin B. Korb, Ann E. Nicholson, and Nathalie Jitnah. Bayesian poker. In UAI-99, 1999. In press.

[4] Maria de Lourdes Peña Castillo, Probabilities and Simulations in Poker, Graduate thesis, University of Alberta.

[5] Denis Richard Papp, Dealing with Imperfect Information in Poker, Graduate thesis, University of Alberta.

[6] Aaron Davidson, Using Artifical Neural Networks to Model Opponents in Texas Hold'em, 1999.

[7] Wikipedia - Poker/Texas Hold'em, Wikipedia: The Free Online Encyclopedia.

[8] CPRG, The University of Alberta Computer Poker Research Group.