Big Data and Online Texas Hold'em A Literature Review

Xianli LI
Department of Computer Science
University of Waterloo
Email: x464li@uwaterloo.ca

Chenyu Yao
Department of Computer Science
University of Waterloo
Email: c5yao@uwaterloo.ca

***Abstract*—As porker is as much about skill as it is luck, online porker has been investigated by numerous researchers. Texas Hold'em, accomplished by maximizing winnings in each individual game within the session, is among the top of the list. Current strategies for playing online Texas Hold'em is more focused on prediction using probabilities given by opponent models, since other strategies, such as Hand Evaluation Analysis and Pot Analysis, are more restricted. When using opponent modelling for prediction, players with a LARGE database of other players' hands often has a huge advantage playing against the others. Data of the stored hands has the attributes of volume, variety, and velocity which could be categorized as Big Data. The goal of this report is to give a general review about the analysis techniques used on the Big Data. We would also give a comparison on the techniques and their decision making strategies. Besides, we would like to find out the tools those techniques used and try to come up with an accuracy-optimized algorithm by combining multiple algorithms using Weighted Majority Algorithm.**

I. Introduction

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. The main difference between board games and porker is that porker involve imperfect information since the others' cards are unknown; tree searching, which is typical a high performance method for the board games, is insufficient to play porker well. However, this attribute, as well as other properties like, multi-agents (more than two players), risk management (betting strategies and their consequences), agent modelling (identifying patterns in the opponent's strategy and exploiting them), deception (bluffing and varying your style of play), and dealing with unreliable information (taking into account opponent's deceptive plays), does make porker an interesting and challenging domain for research.

Previously, there are two main approaches to porker research. One approach is to use simplified variants that are easier to analyze [1]. The other approach is to pick a real variant, and investigate it using mathematical analysis, simulation, and/or ad-hoc expert experience [2]. After that, Koller and Pfeffer [3] have been investigating poker from a theoretical point of view and intended to find an optimal game-theoretic strategy. But they found that the full-scale game is so huge and unlikely to solve it. Researchers of University of Alberta began to be aware of the importance of opponent modelling in success of porker [7]. Since then, progresses in this aspect have continuously been made.

Generally speaking, big data is defined as limitation of analytics and storage capabilities of standard data processing tools like database management systems. Formally, the big data stays that there is a triple "v": volume, velocity and variety. Volume states the fact of data processing limitations that coming from data's huge size. Velocity argues that data input speed is also crucial, since data is generated and inserted into data storage on high speed. Variety stays that data has different formats including structured and unstructured [8].

Data (game logs) used in constructing models of the opponents is usually defined distinctively by different methods and their attributes differs. However, before formulating by criteria, data collected from online Texas Hold'em apparently exceed the capability of a standard data. In online Texas Hold'em, the volume of the data is huge even for a few set of consecutive games. Also, the length of each entry varies in a big range due to player's different betting behaviour in each game. In addition, the speed of input is crucial as the game data would be stored into the data warehouse through online stream. The data, which has the property of high volume, unstructured and high velocity, can be clearly categorized as big data.

This literature review mainly focuses on the techniques used to analyze the data (game logs) and their corresponding strategies based on the analysis. We would first introduce the game of Texas Hold'em in section II. Then, we would mention methods used to capture data online from a real game followed by data storage in section III. In section IV, we would discuss the opponent modelling techniques in detail and make comparisons. In section IV, we would propose a combining model based on Weighted Majority Algorithm as well.

II. Texas Hold'em

Texas Hold'em is considered to be one of the most strategically complex poker variants. The script for Texas Hold'em contains four rounds: *Pre-flop*, *Flop*, *Turn* and *River*. After the betting on the river, the best 5-card hand formed from the two hole cards and five community cards wins the pot.

A. Game Play

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 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.

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. 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.

The betting portion of poker is a multi-round sequence of player actions until some termination condition is satisfied. Without it, each player's 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).

The players are in a fixed seating order around the table (even in a virtual environment, like the online Texas Hold'em, the set of players is referred to as the table). 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.

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. 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. [13]–[17]

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. Here are all the 5-card hands ranked from strongest to weakest: 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.

B. Decision Making

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 to use the game theoretic optimal betting strategy, however, despite the fact that it is very complex to calculate, human opponents do not play optimally 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 (like bluffing). For each decision to be made, one of several variables is compared to some threshold All of these refinements are intended to find quick ways to select the play which hopefully has the largest expected win. We would give a brief introduction for the analysis techniques used except for opponent modelling here.

1) Hand Evaluation: Hand evaluation uses the opponent models and game state to calculate estimates of the winning chances of the cards in the player's 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), pre-flop evaluation is mainly done by expert system or game simulation.

a) Pre-flop Evaluation: For the general case of pre-flop evaluation, there are $\binom{52}{2} = 1326$ possible combinations. The simple brute force method needs $O(50^7)$ comparing times. Thus, using probabilities learned by the expert system and making simulations (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)) are better solutions.

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

$$Threshold = Percent_{win} \times (1 + position/c) \tag{2}$$

The implemented pre-flop betting strategy is preliminary and makes use of expert information and it mainly based on 2 things: pre-flop hand strength and player's position. A threshold equation can be constructed, which could decide the strategies based on positions and percentage of wins calculated by the pre-flop simulation or got in the table. Equation (2) 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, a judgement is added to show that if the player's position is over 2, which means that he/she is not involved in the blind betting, he/she would have higher $FOLD$ threshold as he/she does not put any money in the pot and thus, do not need to bet for the not quite good starting hand.

b) Hand Strength: Hand strength assesses how strong the card in your hand and how strong it is in relation to what other players may hold. 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. On the flop there are $\binom{47}{2} = 1081$ 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.

c) Hand Potential: Hand potential computes the probability that a hand will improve to win, or that a leading hand will lose, as additional community cards appear. For example, a hand that contains four cards in the same suit may have a low hand strength, but has good potential to win with a flush as more community cards are dealt. Conversely, a hand with a high pair could decrease in strength and lose to a flush as many cards of a common suit appear on the board. At a minimum, hand potential is a function of the cards in the hand and the current community cards. However, a better calculation could use all of the additional factors described in the hand strength computation.

The majority of betting decisions are made based on a variable which represents the strength of hand in the current situation. Basic hand strength 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 potentials, 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.

d) Pot Odds: Pot odds is also an 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 4 times out of 5, at a cost of \$20 each time. However, we win 1 time out of 5 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.

III. Data Collection and Storage for Online Texas Hold'em

For offline games, we could easily get access to the game data and store them in the desired data structures. However, for online porker applications, they usually have a large number of commonalities in their interfaces and their interfaces are also the entry point to extract all hand data. Methods for collecting hand data involved in three different approaches:

- Capturing packets between the poker client and server.
- Utilizing provided hand history facilities.
- Scraping the screen to obtain game information.

However, the first two methods have some drawbacks introduced by [26]. According to [26], the task of reverse engineering the packet format would be tedious and time consuming. Also, the traffic between the client and server is securely encrypted. While in theory, it should be possible to determine the encryption key through studious examination of local binaries. in practice, this would be extremely difficult, and maybe illegal. Moreover, the packet format and encryption schemes would vary between poker programs, which severely restricting the general applicability of certain extracting prohram. Those three factors make the first method of collecting data infeasible.

For using application native hand history facilities. As noted previously, support for hand histories is uneven. Some programs do not provide them, and others make use of differing formats to maintain log files. The means for accessing these histories are similarly varied. The GUI access points differ, as do the formats in which these histories are transmitted; some are sent by email, others are saved as local text files [26].

In [29], the author mentioned the challenge about collecting player's data using hand histories as well. Typically, the game state and player's strategy are classified by online game log. However, game log does not usually represent the state information. Therefore, replay a game with the log is necessary in order to retrieve desired information. Moreover, the organization of log files are different among different poker platform, author created some parser to parse various game information. In [29]'s approach, a new parser must be added every time a new online platform's information is needed. Handling unstructured data more efficiently would be a speed-up enhancement and a motivation to correlate it with big data collection.

Thus, the most efficient way is to screen capture all relevant game information as third approach. [26] provides a detailed methods of capturing the screen shot using the built-in Java library, Robot, adding other techniques for identifying the data from the captured images. Those techniques include Image Matching, Text Recognition, Game Element Parsing, Game Parsing and Query.

Image matching has two ways of implementations: comparing pixel by pixel and trie based method. Text recognition adopts the trie based method of image matching using a character trie for font detection. Game element parsing has several categorizes: text fields can be achieved by predetermining the locations, and then scanning these locations when requests come; graphic fields could be recognized through the use of simple image matching, while card fields can be detected by trie based image matching.

Game parsing is the main part of game data collection which contains pre-betting action, betting action, and pot resolution [27]. The goal of pre-betting action is to determine which players are in the pot, their stacks, and their names. Betting action covers the betting on all betting rounds, as well as the community cards that are dealt. Pot resolution involves determining who the winner of the pot (or pots, if there are side pots formed by players moving all-in). The final part is to build a query environment to examine the data. These examined data stream could be stored into the relational databases for further analysis.

As the technology improvement of computer vision, parsing a real game recorded by match video becomes possible in the future. In [28], authors shows some progress parsing objects in videos. Although there is still a long way to go, there is no surprise that real game could be served as another source of data in opponent model in next few years.

IV. Techniques for Opponent Modelling

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 it is 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 as well. Actually, there are many studies of opponent modelling in Texas Hold'em [30]–[34]. The most of the contribution would be the studies done by University of Alberta. We would first look at the Re-Weighting System proposed by them and followed by other methods.

A. Re-Weighting System

In [35], researchers from University of Alberta use the betting history of the opponents to determine 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 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 an important performance improvement using only a simple analysis of context.

In their bot, *Loki*, a weight array of the opponent's hand for each opponent p is constructed. The array 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).

The authors then adjust the weight array by using action frequencies. Besides generic opponent modelling by treating each player the same, the re-weighting system use specific opponent modelling. They maintain a model for each opponent and divide the observed actions into 12 different categories (situations). When needing to know the frequency that a player has taken a certain action in a certain situation, they simply compute the ratio of that specific action taken in that situation versus the total number of actions recorded for that context. In particular, before the model accumulating adequate samples, they use the default frequencies based on a typical IRC player.

The learning process is that each time an opponent makes a betting action, the weights for that opponent are modified to account for the action for two distinct levels. The actual transformation function used for the re-weighting is independent among different player. A different weight array is still maintained for each opponent, but a raise observed in a certain category is treated the same for all players.

Every time when they want to make a decision, they simply calculated the summation of the normalized $w'_p[h]$ and compared it with EHS in hands (Equation (3)). They use a reweighing function to retrieve information from history and the weight is based on interpolation in the range $\mu \pm \delta$.

$$Estimated\_Oppo\_EHS = \sum_h w'_p[h] \times HS_h \qquad (3)$$

Later, in [36], *Loki*'s improved betting strategy consists of playing out many likely scenarios to determine how much money each decision will win or lose using simulation (discussed later in section IV, subsection C, part 4). They use selective sampling to select the most possible hands for each opponent which has specific information that can be used to bias the selection of cards. For each opponent, *Loki* maintains a Weight Table like before, and the random generation of each opponent's hole cards is based on the entire set of possible hands. This selective simulation is clearly better than the static one proposed earlier, since it uses a selective search to refine the static evaluation function.

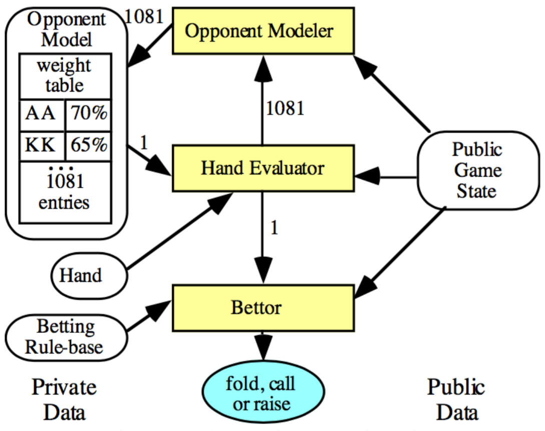


Fig. 1. The architecture of Loki-1 [37].

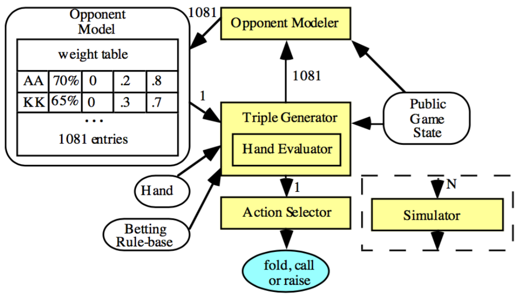


Fig. 2. The architecture of Loki-2 [37].

To make *Loki* perform better, they further modified the structure of *Loki*'s system. As shown in Figure 1, the opponent modeller does not distinguish between the different actions that an opponent might take. A call/check versus a bet/raise gives valuable information about the opponent's cards. In *Loke-2*, they modified the structure (shown in Figure 2) by adding a technique called probability triples which is an ordered triple of values representing the probability distribution that the next betting action in a given context is a fold, call, or raise respectively.

In Figure 2, the Opponent Modeller uses an array of probability triples to update the opponent weight tables by calling Triple Generator for each possible two-card hand. It then multiplies each weight in the Weight Table by the entry in the probability triple that corresponds to the opponent's action. Through using a probability distribution, they account for the uncertainty in their beliefs, which was not handled by their previous architecture. This process of updating the weight table is repeated for each entry.

The enhancement on the Opponent Modeller has improved the performance of the bot (which could be seen from Figure 3). In the figure, three enhancement has been examined including:

- R: changing the re-weighting system to use probability triples.
- B: changing from a rule-based Bettor to an Action Selector that uses probability triples and incorporates a randomized action.
- S: incorporating a Simulator to compute an Expected Value estimate, which is used to determine an action.

Figure 3 shows the result of playing Loki against itself with the B and R enhancements individually and combined (B+R). By comparing the win rate (measured by small bets per hands, or sb/h), against the Loki-1 standard, B won $+0.025 \pm 0.007$ sb/hand, R won $+0.023 \pm 0.0125$ sb/hand and the combined B+R won $+0.044 \pm 0.024$ sb/hand, showing that these two improvements are nearly independent of each other. Figure 3 also shows enhancement S by itself and S combined with B and R (B+R+S). Note that each feature is a win by itself and in combination with others.

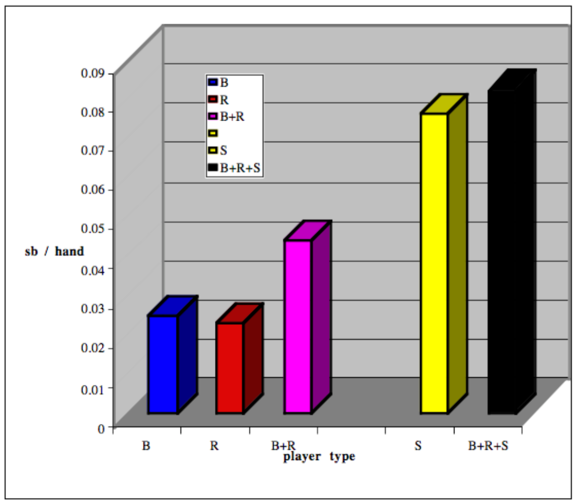


Fig. 3. Experimental results of the basic Loki player versus the Loki player with enhancements [37].

B. *Artificial Neural Network*

Two paper [38], [46] discussing Neural Network from University of Alberta would be reviewed together here first. These two paper are written by Aaron Davidson and are closely related with each others.

In [42], [46], the author discussed the previous method of using re-weighting system for opponent modelling. In re-weighting system, the table is updated after each player action accordingly. Until this paper is published, there were three possible updating technique. The first method is no opponent modelling at all. All values in weighed table are fixed and all opponent actions are ignored. The next technique is to treat all players in the same way according to some pre-selected typical play. This is called generic opponent model. This is a good improve comparing to the previous one. However, there could exist very inaccurate prediction for some particular players. The third method is to treat each opponent differently. All previous collection of hand data is used.

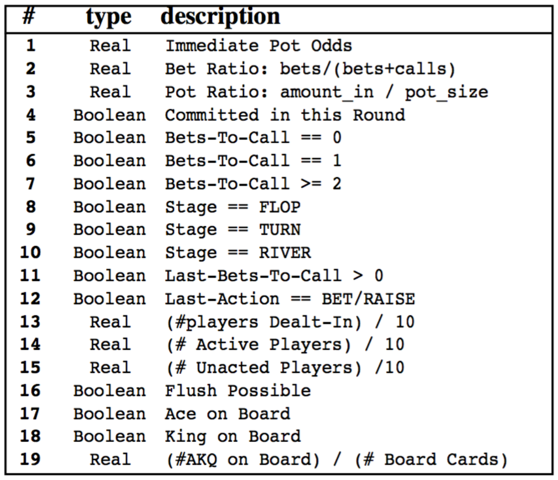
The author mentioned the opponent model could be improved in various ways which utilize more recorded data of each player. The winning rate could be enhanced in such way. The basic data structure of opponent model is a betting frequencies table during various stages.

Besides the number of times each opponents folded,called or raised in each context is recorded, there are still many contextual factors that could affect a player's action. Testing all of these would be hard and unrealistic. In addition, competing with best players would require dynamic learning in playing process. To achieve this goal, more flexible structure is needed. Two particular features are filtered out with the use of artificial neural network (ANN) [41].

[46] discussed ANN in more details. Since the final goal of opponent model is to predict the player's hand based on their action and what a player will do in particular game situation, ANN is well known to learn and identify patterns in noisy data.

The author tested the model through an online poker server called Internet Relay Chat (IRC) against real players since the inherent biases exist in self-training. The training data was based on log files of specific players. Game contexts and the associated observed action was collected for different players and translated into an array of float with range 0 to 1 as shown in Figure 4. The data is then fed into a standard feed ANN. The sigmoid activation function was used for all nodes in the network. Each of the three output nodes represents the network's prediction that an opponent will fold, call, or raise respectively. By normalizing the output nodes, a Probability Triple data structure is constructed by normalize the output nodes. After using back-propagation which is a local hill-climbing method, the network begins to successfully discover the importance of each input feature.

Author used three techniques to deal with some common problems of hill climbing algorithm. To prevent wild oscillations caused by overshooting an optimum configuration, the learning rate is set after each training cycle to the minimum of the default rate (0.4) and the average error. To avoid trapped in local peaks, momentum in the weights allows the system to surpass small obstacles. And a technique called selective simulated annealing is used for more difficult obstacles. Finally, pre-flop actions were ignored from training set since pre-flop is much different and has less information available compared with post-flop play.



#	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)

Fig. 4. Context Information used to train the networks. Boolean values are input as a 0 or 1, Real values as a real number from 0 to 1 [38].

By comparing the actual result and predicted output, accuracy is calculated and a confusion matrix is constructed. The columns shows the predicted frequencies while the rows indicates the actual frequencies. Probability triple and the confusion matrix can be combined to make a new triple which represents a more conservative prediction.

	Prediction			
Accuracy	fold	call	raise	%
fold	13.0	0.3	0.3	13.6
call	0.0	58.4	3.3	61.8
raise	0.0	10.5	14.1	24.7
%	13.0	69.3	17.7	85.6

TABLE I
NEURAL NET PREDICTION ACCURACY CONFUSION MATRIX [38].

Author also showed a graphical representation of ANN. Figures 5 and 6 shows a neural net diagram before and after training. And Figure 7 shows how the network's error and accuracy change after each training cycle. As shown in figure, the accuracy is improved within three stage. Firstly it learns to choose the most frequent action overall. Then, it eventually figures out how to distinct two common actions. Finally, a third distinction occurs and the network can recognize common situations for all three actions.

The experiment makes use of six different players from weak to strong. After 25 training iteration, the network could reach 85% accuracy towards new seventh player. Figure 8 shows the final network. It is easy to find what game features are highly correlated with an opponent's action. Previous action and previous amount to call are two strong features

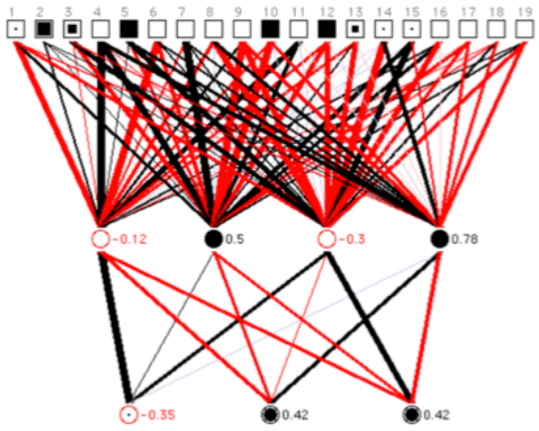


Fig. 5. A Neural Net before training [38].

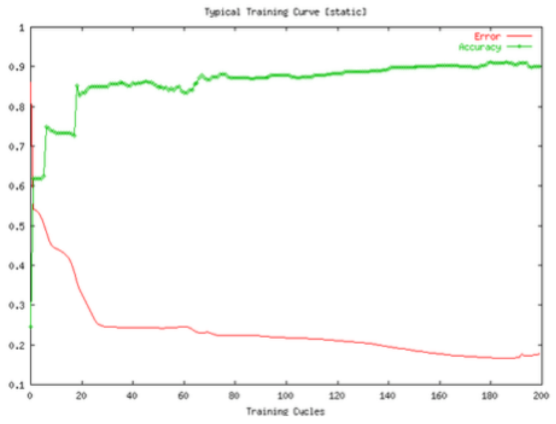


Fig. 7. A Typical Training Curve for a Static Opponent Modelling Problem [38].

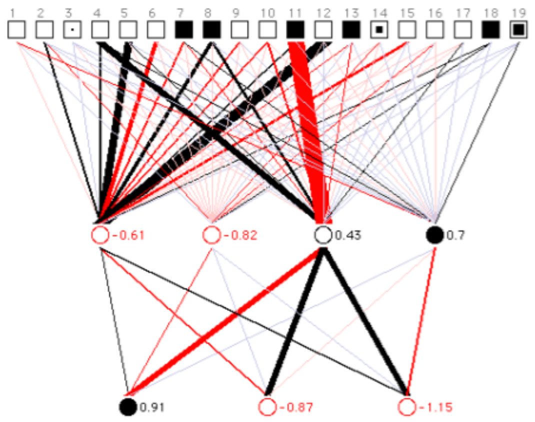


Fig. 6. A Network trained by composite data (Shown successfully predicting a fold) [38].

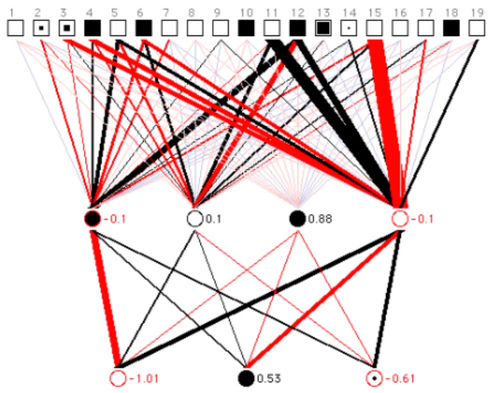


Fig. 8. A Network after being trained on an opponent (Shown correctly predicting a call) [38].

and are noticed by author which enhance the opponent model in [38].

Table was given to show the result of comparison of previous opponent model, enhanced opponent model and ANN among seven distinct players with their ability from weak to strong. As a result, ANN could make a much more reliable prediction compared to previous opponent model.

The result also shows an big improvement for enhanced opponent model by considering those two features. Although a complete re-design could result in better outcome, a small enhancement of opponent model could achieve good improvement.

By comparing the win rate between old and enhanced opponent model, an interesting phenomenon has been brought to table. Old model won at a rate of $+0.09$ sb/h against both beginner and advanced level of players. While new model won at a rate of $+0.22$ sb/h against beginner level players but a $+0.08$ sb/h against advanced level players. Since the action of the player gives more weighted in enhanced model, enhanced model is more successful against predictable players. However, strong players are able to detect and exploit this characteristic which makes enhanced model relatively weaker while against to them.

ANN could not only serve as a model to predict the opponent's action, but could also discover features having strong correlations to the predict outcomes. Opponent model could be strongly enhanced by adding discovered features by ANN.

These two papers are published in around 15 years ago. There are still many data which is potentially useful but not been taken into consideration due to the limitation of computing resource. By applying big data technique, more features could be discovered from ANN. In addition, ANN introduced here is an offline algorithm which surely has limitation against strong players. A real time analysis against player and self adaptable abilities are needed. Applying big data analytics using neural network has become the trend and could certainly be beneficial. Also, note the result of enhanced opponent model shows a predictable vulnerability against advanced level players. Re-design the current opponent model and adding some self adjustment power to it would be a realistic approach.

Another paper [50] introduced a fundamentally different

Train	Test	Prev	Enhc	ANN	sb/h
218	361	63.4	69.5	90.0	-0.017
250	217	52.1	64.1	75.6	0.131
1323	615	58.2	72.2	80.0	-0.076
237	116	56.0	72.4	75.6	-0.078
325	109	55.1	73.4	82.6	0.127
90	322	51.2	70.2	82.6	0.166
86	138	65.2	80.4	81.2	-0.138
361	268	57.3	71.7	81.1	0.016

Fig. 9. Comparison of three prediction techniques [38].

approach of opponent model using neural network. Rather than constructing some probability distribution table and predict opponent directly, this approach would attempt to classify the opponent by type. In this paper, a mixture approach which all opponents are represented as probability distribution of some cardinal opponents is discussed by author.

In this approach, the neural networks are trained using NeuroEvolution of Augmenting Topologies (NEAT) [59]. Game state and mixture identifier are two main factors. Game state is represented by the following input parameters:

- The estimated win probability, p,
- The ratio of expected winnings if CALL is selected,
- The ratio of expected winnings if RAISE is selected,
- The current round,
- The size of the pot,
- The size of their own bankroll,
- The number of raises made by the opponent this round,
- The required cost of a bet.

The reason of using win probability to visualize the card hold rather than more complete states is to keep the number of parameters small to ensure the feasibility of training.

For mixture identifier, the mixture parameters are obtained by partitioning the game state which was broken into nine independent region based on win probability and expected winning as shown in Figure 9. In each turn, a player has three options which represented by Call probability and Raise Probability. Therefore there are 10 output parameters in total.

In the mixture identifier experiment, a mixture identifier network was trained to guess the mixture governing a generated opponent on average. Then, two separated networks was trained to validate this module. Mixture-based player took both game state and mixture identifier's output as input while Control player only take the game state as input. The result was shown in Figure 10 which shows the mixture-based player has more stable and higher winning,

The main contradiction in the approach is efficiency against effective. The neural network require smallest possible input parameters in order to speed up training process in real-time to prevent sluggish. However, the game state and the output of mixture identifier require more parameters to fully describe the game context and player's strategy. Thus, making use of big data analysis strategy might be enhancement to both speed and

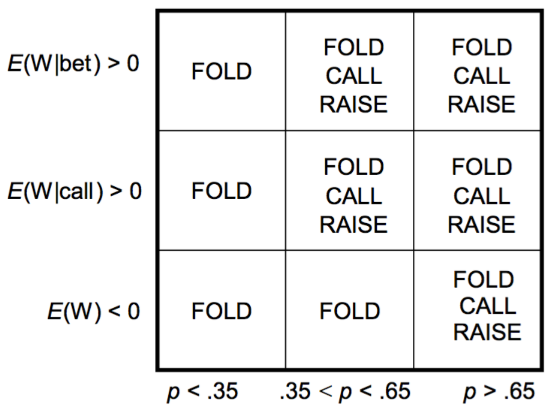


Fig. 10. A 10-dimensional opponent space for poker. Vertical axis is expected winnings. Horizontal axis is the probability of winning the hand based on the visible cards. Only five of the nine regions have more than one viable action, and a probability distribution over each of these has two degrees of freedom, for a total of 10 parameters in the model [50].

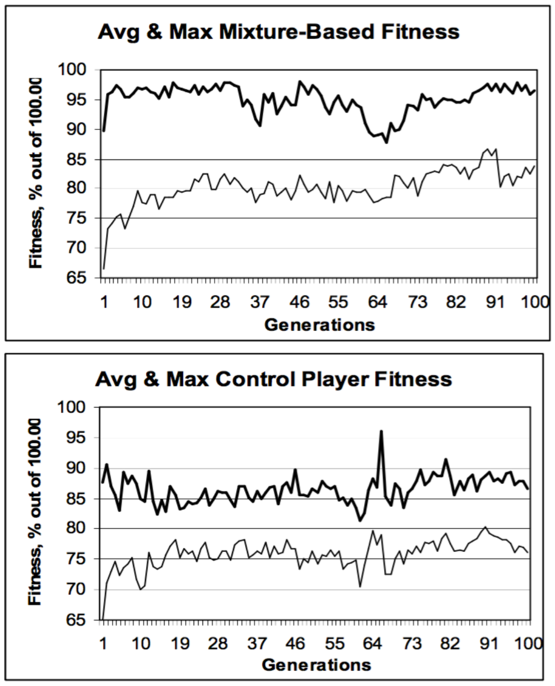


Fig. 11. Fitness graphs for the mixture-based and control players on a sample trial run. Fitness is the average percentage of money won by the player. Average and maximum fitness for the mixture-based player is higher and more stable in general [50] .

accuracy of the classify model since big data usually requires real-time processing.

C. Other Methods

1) Supervised Learning: [29] briefly discussed an agent using static strategy. The strategy was obtained by classifying game state instances, using supervised learning algorithms. Author used WEKA to facilitate the learning process and used the Meerkat API to implement agent. In detail, agent determine which actions are doable, and change the tactic based on the game state. Then, it tried to predict the action with current tactic classifier. If the prediction failed, the agent checks, or otherwise fold. The agent in this approach is still weak since it uses static tactic which can be easily learned by opponent and easily defeated. In order to overcome this weakness, author made agent change tactic once it start losing money.

There still many improvement potentially exist in this approach. First, the build of agent is an off-line algorithm. That is to say, the agent is not able to learn the opponent during the actual game. Although the enhanced version of agent is able to change the tactic if sitting on an unfavourable situation, the agent is not able to decide and choose the best possible tactic. It is possible that agent choose a worse tactic and make itself far beyond redemption. Also, the strength of agent strongly depends on the size of learning dataset. As shown in paper, if agent is not able to predict the correct action, it simply checks every time until bankrupt. This decision is curtness and could be easily exploit and cause serious loss. To avoid such tragedy, the best solution without re-design agent is to enlarge its train set to reduce the probability of failed prediction. Due to the difficulty of parsing log files from various online poker engine, it is needed to handle unstructured log data with big data collection technique.

2) Bayesian Decision Networks: Another approach using Bayesian Decision Network [56] is introduced by [60]–[62]. Figure 12 shows a scratch version of decision network for poker. This decision network bets aggressively with a strong hand and passively with a weak hand which could be quickly learned by adversaries. To minimize such disadvantage, some randomization are added for the purpose of bluffing. In addition, the ability of predict straight or flush from a partial hand is added by extending the decision network as shown in Figure 13. As a result, the modified version shows an increase of 0.128 ± 0.123 sb/h compared with previous one.

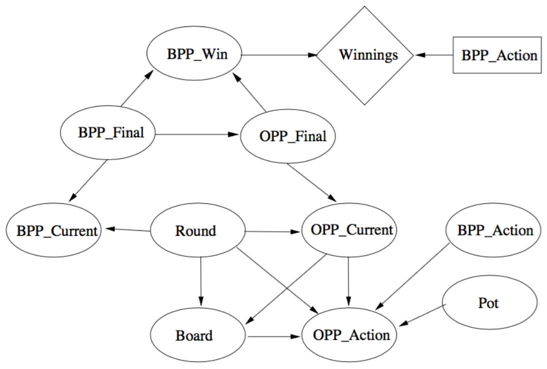


Fig. 12. A decision network for poker [60] .

The model described above is still a static opponent model which is not very valuable in real world use. However, it still plays an important role at the initial stage of game and in situation where useful modelling techniques are not available. The main challenge of implementing a dynamic model lies in accurately predicting opponents. CPT for the OPP_{Action}

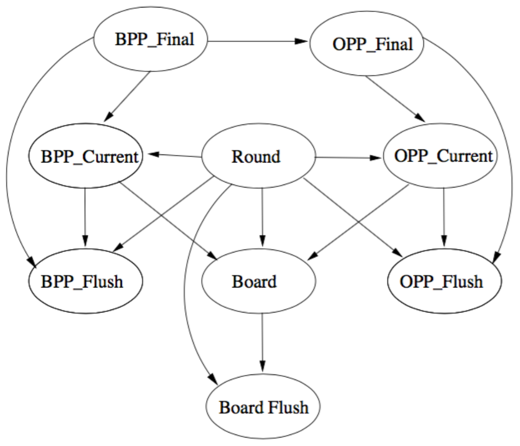


Fig. 13. Modified network fragment showing the ability to represent partial flushes in the board, opponent's current hand and the player's current hand [60] .

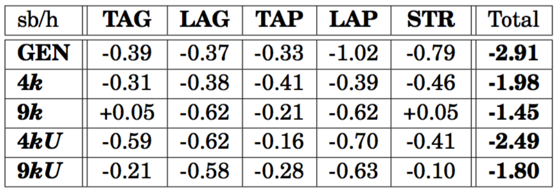
contains 200,000 entries which is too great to ensure short time learning. After considering some methods to reduce the size of CPT, author chose to use a decision tree to represent CPT. They applied a decision tree learner D-Graf trading off the learning rate with accuracy. The data served as training set is obtained mainly from self-play trials.

The approach introduced in this paper has some potentials but still a long walk towards the final goals. The model is aimed on heads-up poker game which is far less complicated compared to multiple opponent games which might result in exponentially growth of decision tree.

3) Clustering: [66] shows that clustering player logs and giving new opponents to a predefined opponent model is beneficial to the performance of simulation based systems. Instead of training only one model to give predictions on all possible playing strategies, the clustered models cover subsets of these strategies and thus can learn and predict the actions of opponents within these subsets to a higher degree of accuracy. Also, as the amount of data increases, using clustering is more likely to cover more possible situations.

In Texas Hold'em, there are basically four groups of Poker players; tight-passive, tight-aggressive, loose-passive, and loose-aggressive. As each group would have a specific playing style, the authors would first to discover styles through the processed data from hand logs using k-means clustering. Then, they explored the possibilities for different numbers of groups to see if more than just the four basic player profiles existed. After using this technique getting classified groups, neural networks are used for prediction.

Figure 14 shows a clear increase in winnings as the number of clusters increase. The generic agent, GEN, plays with the lowest winning rate of -2.91 sb/h. The 4k and 9k agents both improve upon this by 0.93 sb/h and 1.46 sb/h, respectively. The clustered agents that use unique opponents models, 4kU and 9kU, also perform better than the generic agent, however their winning rates are lower, only improving by 0.5 sb/h and



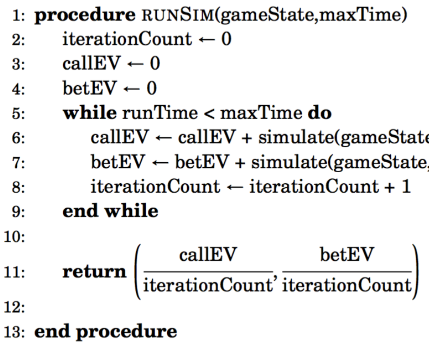
sb/h	TAG	LAG	TAP	LAP	STR	Total
GEN	-0.39	-0.37	-0.33	-1.02	-0.79	**-2.91**
4k	-0.31	-0.38	-0.41	-0.39	-0.46	**-1.98**
9k	+0.05	-0.62	-0.21	-0.62	+0.05	**-1.45**
4kU	-0.59	-0.62	-0.16	-0.70	-0.41	**-2.49**
9kU	-0.21	-0.58	-0.28	-0.63	-0.10	**-1.80**

Fig. 14. Direct comparison of agent configurations [66].

1.11 sb/h, respectively.

4) Simulation: Prediction given by neural networks is usually a set of possibilities of the opponent's actions and we need to use a scheme to define the optimal actions for us to take. Monte Carlo (MC) methods/simulations are a set of algorithms that use repeated random sampling to find approximate solutions to numeric problems that would otherwise be difficult, or computationally infeasible to solve [67]. Thus MC is a way of calculating expected values fetching the optimal actions for the players.

When the agent is faced with a decision, it would run a selective sampling simulation for each possible action to determine which choice is the best to take. For each of the possible actions, the game is simulated until showdown and the amount won by the agent is determined using an uniform probability distribution given by the opponent modelling. Figure 15 gives the general idea of the simulation process based on two possible actions, a call and a bet.



```
1:  procedure RUNSIM(gameState,maxTime)
2:      iterationCount ← 0
3:      callEV ← 0
4:      betEV ← 0
5:      while runTime < maxTime do
6:          callEV ← callEV + simulate(gameState,callAction)
7:          betEV ← betEV + simulate(gameState,betAction)
8:          iterationCount ← iterationCount + 1
9:      end while
10:
11:     return $\left(\frac{\text{callEV}}{\text{iterationCount}}, \frac{\text{betEV}}{\text{iterationCount}}\right)$
12:
13: end procedure
```

Fig. 15. Algorithm: Calculate Expected Value of possible actions [66].

Other works done on MC includes [67]–[72] focusing on improving simulation based systems with the introduction of Monte-Carlo Tree Search methods. The MCTS builds a search tree according to the results of the simulated games. Using the search tree, the algorithm can focus on exploring promising paths through the tree. This in turn shows that the algorithm will simulate the positive actions more thoroughly and extensively, therefore returning the expected value of the actions with higher confidence and accuracy.

D. Weighted Majority Algorithm Model

After reviewing a number of research about opponent modelling, we found that there are various approaches about how to predict the opponent's action. Certain approaches may perform well in certain situation against certain opponents. For example, in paper Improved Opponent Modelling in Poker, the enhanced opponent model perform better against beginner level of opponent while worse against advanced level opponent compared with old version opponent model. Combining these model and choose the suitable model accordingly would be a considerable approach.

Weighted Majority Algorithm is used to design a master algorithm to give prediction while given a pool of prediction algorithm. It does not need to have prior knowledge about which algorithm in pool makes fewest mistakes [74].

The detail application of weighted majority algorithm in opponent modelling is as follows: At first, all model are given a weight 1. At each round, all models make predictions about the opponent's behaviour ($RAISE$, $CALL$, $FOLD$). Then the predicted action with most weight is considered as the opponent's action and the agent takes action responding to opponents' action. After opponent actually takes action, reduce the weight of models who make wrong prediction by half as shown in table. Denote the best model makes OPT mistakes after n round. A mathematical proof shows the master algorithm makes no more than $2.41 \times (OPT + log_2\ n)$. Therefore, if there is a suitable model that makes few mistakes, the agent starts performing well after $O(log\ n)$ steps.

	M1	M2	M3	M4	M5	Prediction	Actual
weight	1	1	1	1	1		
predict	C	C	C	R	F	C	C
weight	1	1	1	0.5	0.5		
predict	F	R	R	F	C	R	C
weight	0.5	0.5	0.5	0.25	0.5		
...							

TABLE II
WEIGHTED MAJORITY ALGORITHM.

In addition, if less weights are taken from wrong prediction experts, the agent would perform better but require a longer training time. Suppose the weight reduced from w to $w(1-\epsilon)$ after a mistake, the result becomes the form of Equation (4).

$$ALG \leq OPT \times log_{\frac{2-\epsilon}{2}}\ (1-\epsilon) + log_{\frac{2}{2-\epsilon}}\ n \tag{4}$$

It is also possible to enhance the combining model by randomization [74]. The only modification to the weighted majority algorithm is the following: at each time, suppose the weights of the experts are $w_1, w_2, \ldots, w_n$ and $W = \sum w_i$ be the total weight. Then the algorithm follows expert's opinion with probability w_i/W. The expected number of mistakes made by the randomized weighted majority algorithm is upper bounded by Equation (5).

$$(1+\epsilon) \times OPT + \frac{1}{\epsilon} \times log\ n \tag{5}$$

In theory, the combining model should do a good work if there is some opponent working well. However, before combining model could make any prediction, all models in pool must have finished their prediction which might require much time and the actual players would feel very sluggish.

Since models in pool are independent with each other, parallel computing could be used in this approach. Also, handling the unstructured log files produced by online game server in real-time requires big data analysis technology.

V. Conclusion

We have presented a review of techniques used in online Texas Hold'em and introduced a number of approaches for opponent models. We found big data analysis technique has many potentials for opponent models. The game state and opponent's state is recorded either as game logs or screen shots. Both of them requires handling big volume of unstructured data. In addition, as the technique improvement of big data and computer vision, parsing a real game through record video could be realized in the future. Clustering data technique could be used as preprocessing. Re-weighting model and artificial neural network are two main focus for opponents model in current study. Besides these two approach, classifying opponent through supervised learning and modelling based on bayesian network are two novel techniques. Monte Carlo Tree method is the most popular decision making technique for players to choose their strategies based on opponent models. Among these different approaches, re-weighting system is the fastest method and ANN combining with clustering and MC simulation technology is the most accurate method so far. All of these approaches could benefit from big data analysis to speed up real-time learning process. The performance of certain approaches could also be determined by certain situation. Selecting the best model based on different state and environment could be benefit. We have proposed a combining model based on Weighted Majority Algorithm to find the optimal model from a pool of models. The combining model requires parallel computation and real-time learning to prevent sluggish. Unlike most of board games, uncertainty and imperfect information make Texas Hold'em harder to predict and solve. Combining big data technique and opponent model seems a interesting topic and contains a number of potential areas worth studying on.

Acknowledgment

The authors wish to thank all those whose work was mentioned in this review paper. Great efforts have been taken to present third party research as accurately as possible and any errors in interpretation are solely the responsibility of the authors.

References

[1] Findler, N. *Studies in Machine Cognition Using the Game of Poker*, CACM 20(4):230-245, 1997.

[2] Sklansky, D. and Malmuth, M. *Hold'em Poker for Advanced Players*, Two Plus Two Publishing, 1994.

[3] Koller, D. and Pfeffer, A. *Representations and Solutions for Game-Theoretic Problems*, Artificial Intelligence 94(1-2), 167-215, 1997.

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Mateo, Calif., 1988.

[5] J. Rubin, I. Watson, *Investigating the effectiveness of applying case-based reasoning to the game of Texas hold'em*, in: Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, 2007, pp. 417-422.

[6] J. Rubin, I. Watson, *A memory-based approach to two-player Texas hold'em*, in: AI 2009: Advances in Artificial Intelligence, 22nd Australasian Joint Conference, 2009, pp. 465-474.

[7] Denis Papp, *Dealing with Imperfect Information in Poker*, 1998.

[8] Laney D. *3-D Data Management: Controlling Data Volume, Velocity and Variety*, META Group Research Note, February 2001.

[9] J. Noble, R.A. Watson, Pareto coevolution: *Using performance against coevolved opponents in a game as dimensions for Pareto selection*, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Morgan Kaufmann, 2001, pp. 493-500. [65] M.J. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, Cambridge, MA, 1994.

[10] N.A. Risk, D. Szafron, *Using counterfactual regret minimization to create competitive multiplayer poker agents*, in: 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), 2010, pp. 159-166.

[11] J. Rubin, I. Watson, *Similarity-based retrieval and solution re-use policies in the game of Texas hold'em*, in: Case-Based Reasoning. Research and Development, 18th International Conference on Case-Based Reasoning, in: Lecture Notes in Computer Science, vol. 6176, Springer-Verlag, 2010, pp. 465-479.

[12] K. Waugh, D. Schnizlein, M.H. Bowling, D. Szafron, *Abstraction pathologies in extensive games*, in: 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), 2009, pp. 781-788.

[13] K. Waugh, M. Zinkevich, M. Johanson, M. Kan, D. Schnizlein, M.H. Bowling, *A practical use of imperfect recall*, in: Eighth Symposium on Abstraction, Reformulation, and Approximation, SARA 2009, 2009.

[14] M. Zinkevich, M. Johanson, M.H. Bowling, C. Piccione, *Regret minimization in games with incomplete information*, in: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, 2007.

[15] B. Sheppard, *World-championship-caliber scrabble*, Artif. Intell. 134 (2002) 241-275.

[16] M. Zinkevich, M.H. Bowling, N. Burch, *A new algorithm for generating equilibria in massive zero-sum games*, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007, pp. 788-793.

[17] I. Watson, J. Rubin, Casper: *A case-based poker-bot*, in: AI 2008: Advances in Artificial Intelligence, 21st Australasian Joint Conference on Artificial Intelligence, 2008, pp. 594-600.

[18] P. Morris, *Introduction to Game Theory*, Springer-Verlag, New York, 1994.

[19] Y. Nesterov, *Excessive gap technique in nonsmooth convex minimization*, SIAM J. Optim. 16 (2005) 235-249.

[20] G. Nicolai, R. Hilderman, *No-limit Texas hold'em poker agents created with evolutionary neural networks*, in: CIG-2009, IEEE Symposium on Computational Intelligence and Games, 2009, pp. 125-131.

[21] J. Noble, *Finding robust Texas hold'em poker strategies using Pareto coevolution and deterministic crowding*, in: Proceedings of the 2002 International Conference on Machine Learning and Applications - ICMLA 2002, 2002, pp. 233-239.

[22] M. Ponsen, M. Lanctot, S. de Jong, MCRNR: *Fast computing of restricted Nash responses by means of sampling*, in: Proceedings of Interactive Decision Theory and Game Theory Workshop (AAAI 2010), 2010.

[23] H. Quek, C. Woo, K.C. Tan, A. Tay, *Evolving Nash-optimal poker strategies using evolutionary computation*, Frontiers Comput. Sci. China 3 (2009) 73-91.

[24] J. Rubin, Casper: *Design and development of a case-based poker player*, Master's thesis, University of Auckland, 2007.

[25] RoboCup, *Robocup world championship*, 2010, http://www.robocup.org/.

[26] Haruyoshi Sakai, *Internet Poker: Data Collection and Analysis*, Brown University, 2005.

[27] P. McCurley, *An artificial intelligence agent for Texas hold'em poker*, Undergraduate dissertation, University of Newcastle Upon Tyne, 2009.

[28] Borislav Antić and Björn Ommer, *Video parsing for Abnormality Detection*, In ICCV, 2011.

[29] Luis Filipe Teofilo and Luis Paulo Reis, *Building a No Limit Texas Hold'em Poker Agent Based on Game Logs using Supervised Learning*, Lecture Notes in Computer Science, 6752 LNAI, pp. 73-82, 2011.

[30] T. Salonen, The bluffbot website, 2009, http://www.bluffbot.com/.

[31] A. Selby, *Optimal heads-up preflop poker*, 1999, www.archduke.demon.co.uk/simplex.

[32] J. Schaeffer, *The games computers play*, Adv. Comput. 52 (2000) 190-268.

[33] A. Sandven, B. Tessem, *A case-based learner for poker*, in: Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006), 2006.

[34] D. Schnizlein, M.H. Bowling, D. Szafron, *Probabilistic state translation in extensive games with large action sets*, in: IJCAI 2009, Proceedings of the 21st International Joint Conference on, Artificial Intelligence, 2009, pp. 278-284.

[35] Denis Richard Papp, *Dealing with Imperfect Information in Poker*, ACM, 1998.

[36] Jonathan Schaeffer, Darse Billings, Lourdes Pea, Duane Szafron, *Learning to Play Strong Poker*, ACM 2001.

[37] Darse Billings, Lourdes Pena, Jonathan Schaeffer, and Duane Szafron, *Using Probabilistic Knowledge and Simulation to Play Poker*, AAAI, 1999.

[38] Aaron Davidson, *Using Artificial Neural Networks to Model Opponents in Texas Hold'em*, www.cs.ualberta.ca/ davidson/poker/nnpoker.pdf, 1999

[39] C.K. Riesbeck, R.C. Schank, *Inside Case-Based Reasoning*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1989.

[40] N.A. Risk, *Using counterfactual regret minimization to create a competitive multiplayer poker agent*, Master's thesis, University of Alberta, 2009.

[41] M. Hlynka., *A Framework for an Automated Neural Network Designer using Evolutionary Algorithms*, FGS - Electronic Theses and Dissertations, 1997.

[42] Billings, D., Papp, D., Schaeffer, J., and Szafron, D. *Opponent Modeling in Poker*, Proceedings of 15th National Conference of the American Association on Artificial Intelligence. AAAI Press, Madison, WI, 1998, 493-498.

[43] D.P. Schnizlein, *State translation in no-limit poker*, Master's thesis, University of Alberta, 2009.

[44] T. Schauenberg, *Opponent modelling and search in poker*, Master's thesis, University of Alberta, 2006.

[45] J. Schaeffer, *A gamut of games*, AI Magazine 22 (2001) 29-46.

[46] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. *Improved Opponent Modeling in Poker*, Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI'2000) 1467-1473, 1999.

[47] D. Sklansky, M. Malmuth, *Hold'em Poker For Advanced Players*, Two Plus Two Publishing, Las Vegas, Nevada, 1994.

[48] B. Wilson, Wilson software: *Official site of turbo poker*, 2008, http://www.wilsonsoftware.com/.

[49] G. Tesauro, *Temporal difference learning and td-gammon*, Commun. ACM 38 (1995) 58-68.

[50] Alan J. Lockett and Risto Miikkulainen, *Evolving Opponent Models for Texas Hold'Em*, In IEEE Conference on Computational Intelligence in Games, Perth, Australia 2008.

[51] Barone, L. and While, L. *Adaptive Learning for Poker*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO- 2000). Kaufmann, San Francisco, 2000, 566-573.

[52] DiPietro, A., Barone, L., and While L. *Learning In RoboCup Keepaway Using Evolutionary Algorithms*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002). Kaufmann, San Francisco, 2002, 1065-1072.

[53] Hoehn, B., Southey, F., Holte, R. C., and Bulitko, V. *Effective ShortTerm Opponent Exploitation in Simplified Poker*, Machine Learning. Volume 74, Issue 2, 2009, 159-189.

[54] Riley, P., and Veloso, A. *Planning for Distributed Execution Through Use of Probabilistic Opponent Models*, IJCAI-2001 Workshop PRO- 2: Planning under Uncertainty and Incomplete Information. 2001.

[55] Cover, T. M. and Thomas, J. A. *Elements of Information Theory*, Communications of the ACM, Vol. 20, 230-245. 1991.

[56] Friedman, N. and Goldszmidt, M. *Learning Bayesian networks with local structure*, UAI96 - Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence. 252-262. 1996.

[57] Howard, R. and Matheson, J. Influence Diagrams. *Readings in Decision Analysis* (eds. R. Howard and J. Matheson), pp. 763-771. Strategic Decisions Group, Menlo Park, CA. 1981.

[58] Lockett, A., Chen, C., and Miikkulainen, R. *Evolving Explicit Opponent Models in Game Playing*, Proceedings og the Genetic and Evolutionary Computation Conference (GECCO-07). Kaufmann, San Francisco, 2007, 2106-2113.

[59] Stanley, K. and Miikkulainen, R. *Continual Coevolution Through Complexification*, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002). Kaufmann, San Francisco, 2002, 113-120.

[60] Ann E. Nicholson, Kevin B. Korb and Darren Boulton, *Using Bayesian Decision Network to Play Texas Hold'em Poker*, Technical Report, Faculty of Information Technology, Monash University, 2006.

[61] Korb, K., Nicholson, A., and Jitnah, N. *Bayesian Poker*, Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-99). 1999, 343-350. [5] Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., and Billings, D. Bayes' Bluff: Opponent Modeling in Poker. Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI- 05). 2005, 550-558.

[62] Bard, N. and Bowling, M. Particle *Filtering for Dynamic Agent Modeling in Simplified Poker*, Proceedings of the 22nd Conference on Artificial Intelligence. AAAI Press, Madison, WI, 2007, 515-521. Stanley, K. and Miikkulainen, R. Continual Coevolution Through Complexification, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002). Kaufmann, San Francisco, 2002, 113-120.

[63] R.J. Vanderbei, *Linear Programming: Foundations and Extensions*, Kluwer Academic, Boston, 2001.

[64] K. Waugh, N. Bard, M. Bowling, *Strategy grafting in extensive games*, in: Advances in Neural Information Processing Systems 22 (NIPS), 2009, pp. 2026-2034.

[65] J. Shi, M.L. Littman, *Abstraction methods for game theoretic poker*, in: Computers and Games, Second International Conference, CG, 2000, pp. 333-345.

[66] Jack Pimbert, *Clustering Player Profiles to Improve Opponent Modelling in Simulation Based Poker Agents*, University of Edinburgh, 2014.

[67] A. Davidson, *Opponent modeling in poker: Learning and acting in a hostile and uncertain environment*, Master's thesis, University of Alberta, 2002.

[68] M. Ponsen, G. Gerritsen, and G. Chaslot, *Integrating Opponent Models with Monte-carlo Tree Search in Poker*, In Interactive Decision Theory and Game Theory, AAAI Workshops. AAAI, 2010.

[69] G. V. den Broeck, K. Driessens, and J. Ramon, *Monte-carlo Tree Seach in Poker Using Expected Reward Distributions*, In ACML, volume 5828, pages 367-381. Springer, 2009.

[70] J.B. MacQueen, *Some methods for classification and analysis of multivariate observations*, in: Proceedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability, vol. 1, University of California Press, 1967, pp. 281-297.

[71] N. Metropolis, S. Ulam, *The Monte Carlo method*, J. Amer. Statist. Assoc. 44 (1949) 335-341.

[72] I. Schweizer, K. Panitzek, S.H. Park, J. Frnkranz, *An exploitative Monte-Carlo poker agent*, Technical Report, Technische Universitt Darmstadt, 2009.

[73] A.A.JvanderKleij, *Monte-carlo Tree Search and Opponent Modeling Through Player Clustering in No-limit Texas Hold'em Poker*, Master's thesis, University of Groningen, 2010.

[74] N. Littlestone, M.K. Warmuth, *The Weighted Majority Algorithm*, Information and Computation, Volume 108, Issue 2, 1 February 1994, Pages 212-261, ISSN 0890-5401, http://dx.doi.org/10.1006/inco.1994.1009.