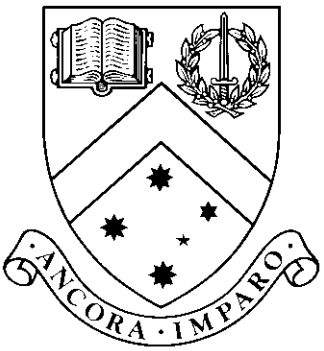
Opponent Modeling in Bayesian Poker

by

Brendon Taylor, BSE



Thesis

Submitted by Brendon Taylor

in partial fulfillment of the Requirements for the Degree of

Bachelor of Software Engineering with Honours (2770)

Supervisors: Ann Nicholson

Co-Supervisor: Kevin Korb

Clayton School of Information Technology
Monash University

October, 2007

Contents

List of Tables

List of Figures

Opponent Modeling in Bayesian Poker

Brendon Taylor, BSE
batay2@student.monash.edu
Monash University, 2007

Supervisor: Ann Nicholson
Ann.Nicholson@infotech.monash.edu.au
Co-Supervisor: Kevin Korb
korb@csse.monash.edu.au

Abstract

Poker is a good testing platform for making decisions under uncertainty. This is due to the fact it is a game of incomplete information and some degree of chance. The factors that lead to incomplete information are the randomly shuffled and dealt cards, the hidden opponent cards, the opponent's strategy and their playing behavior. Complete information games such as chess and checkers have been well studied and mostly solved. The main goal of poker is profitability, therefore economics and mathematics can be used to improve accuracy when making decisions. This research is a valuable contribution to Artificial Intelligence (AI) to help both machines and humans make decisions under uncertainty.

The following thesis work discusses the previous research undertaken. This research includes the development of Bayesian Poker Player (BPP) and the related research undertaken in this field. The Bayesian Poker Player is a computer program developed at Monash University that utilities Bayesian Networks to make decisions under uncertainty. My area of research endeavors to improve the way the Bayesian Network models an opponent. This includes building an accurate static (initial) opponent model which represents any opponent. The second model will be a dynamic (interactive) opponent model which learns during the course of the game. The dynamic opponent model categorises the opponent type based on its behavior. Learning this extra piece of information helps better identify the type of opponent, therefore improving the ability to predict opponent's future moves. Exploiting problems in your opponent's strategy is vital in the game of poker.

Opponent Modeling in Bayesian Poker

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Brendon Taylor
October 27, 2007

Acknowledgments

Firstly I would like to thank my supervisors (**Dr. Ann Nicholson** and **Dr. Kevin Korb**) for their advice and guidance.

Secondly I would like to thank **Steven Mascaro** for his support in the understanding of the code and network.

Finally I would like to thank my parents and friends for their continued support and encouragement throughout the year.

Brendon Taylor

Monash University
October 2007

Chapter 1

Introduction

Poker is an excellent testing platform to perform automated reasoning under uncertainty. The challenge of poker is not all the information is present, for example the player's hidden cards. In most situations this results in players making decisions (assumptions) when they are unsure whether they hold the best hand. Players have to make a decision based on limited information. For example, the player can use factors such as their betting position, their expected hand strength and their potential hand strength to make a decision. Additionally the player has to attempt to model their opponent in order to predict what move their opponent will make next. This will assist in making a better decision due to the fact they can respond to an opponent's move before they make it. For example, if the player is losing and they know the opponent is likely to bet then they may chose to fold. Unfortunately good players do not play consistency. Occasionally players will over represent (bluffing) or under represent (sand bagging) their hand in order to trick their opponent into making an incorrect move. This makes opponent modeling a challenging aspect of the game.

Game theory attempts to use mathematical and economical decisions from the available information to make the best decisions and ultimately produce the best return. Game theory can be used to make intelligent decisions. When we refer to machines (computers) making intelligent decisions it is known as artificial intelligence.

The challenge of the game is not the ability to follow and understand the rules of the game, but rather to implement a complex strategy that will produce the greatest return. Humans can quickly adapt and change their strategy over time, where as computers are limited by their learning ability. This leads to making abstractions and only learning the most important information during the game. My aim is to learn the most important information specifically about the opponent. This field of research can be expanded to domains such as psychology, marketing and economics.

The thesis is broken down into the following areas:

- Section two contains the basics of playing poker.
- Section three describes an overview of the previous research.
- Section four provides contributions to the opponent model.
- Section five presents the conclusions and future work.

Chapter 2

Poker

Poker is a popular non-deterministic card game which consists of between two to ten players. A long term economic goal of a player would be to leave the table with more money, where as a short term goal is to win a hand. A standard playing deck is used consisting of 52 cards. Each playing card contains a suit and a rank. There are four different suits: ♣ Clubs, ♢ Diamonds, ♡ Hearts and ♠ Spades. Additionally there are 13 different ranks (in increasing order): Deuce (2), Three (3), Four (4), Five (5), Six (6), Seven (7), Eight (8), Nine (9), Ten (T), Jack (J), Queen (Q), King (K) and Ace (A). A list of poker hands and their corresponding rank is shown in table 2.1.

Rank	Type	Example	Probability
1.	Straight Flush	3 ♡ 4 ♡ 5 ♡ 6 ♡ 7 ♡	0.0000134
2.	Four of a Kind	6 ♣ K ♢ K ♠ K ♡ K ♣	0.0002476
3.	Full House	9 ♣ 9 ♡ 5 ♠ 5 ♣ 5 ♢	0.0014405
4.	Flush (same suit)	J ♠ 4 ♠ A ♠ Q ♠ 9 ♠	0.0019693
5.	Straight (sequence)	2 ♢ 3 ♣ 4 ♡ 5 ♣ 6 ♢	0.0035492
6.	Three of a Kind (triple)	A ♡ J ♣ J ♢ J ♠ 5 ♣	0.0211037
7.	Two Pair	T ♣ T ♢ Q ♠ 8 ♣ 8 ♡	0.0475431
8.	Pair	9 ♣ 2 ♡ 2 ♢ T ♠ 3 ♡	0.4225703
9.	Busted (high card)	4 ♠ 9 ♡ 3 ♠ 2 ♣ A ♣	0.5015629

Table 2.1: Poker Hands ranked from strongest to weakest.

Prior to the commencement of a game an opening bet (ante) is required by each player. All bets are accumulated in the centre of the table, which is known as the pot. The pot is distributed amongst the winning player(s) at the end of the game. A poker game consists of multiple rounds. During each round there is a number of cards randomly dealt, followed by a series of betting actions. Each active player has a opportunity to make a betting action at least once for each round. There are three main betting actions a player can take when it is there turn to act.

1. **Fold**: The player can chose to forfeit their hand and contributions to the pot. This player can play no further part in the current game.
2. **Call**: The player matches the highest bet.
3. **Bet**: The player increases the amount of money required to call for other active players.

A betting round is concluded when all bets have been called or only one active player remains in the game. In the second scenario the player automatically wins the pot without

being forced to show their hidden (hole) cards. At the conclusion of the final round, if there is more than 1 active player, the player with the highest hand rank wins the game. This is known as a showdown.

There are many variations of poker. However the three main categories are stud games (no shared cards), flop games (shared community cards) and draw games (all cards are hidden). Stud and flop are interesting to research due to the public knowledge of visible cards during the game. Additional knowledge can be learnt when games are subject to a showdown.

2.1 5 card stud

In this version of poker after the initial ante each player is dealt 5 cards. The first card is hidden (face down) and the remaining 4 cards are visible to all players. There is a betting round after 2, 3, 4, and 5 cards have been dealt. Some variations of the game include changing the amount of face up and face down cards for each player. In the stud category of poker there are no community cards (shared cards).

The initial research used the 5 card stud version of poker.

2.2 Texas Hold'em

This modern version of poker has become very popular and used in most professional poker tournaments. This example of poker is a flop game. The current version of the BPP supports this type of poker.

Each player is dealt 2 cards face down and a betting round commences (PreFlop). The dealer then places 3 shared community cards in the middle of the table (PostFlop). Another round of betting commences. An additional card is placed in the center (Turn) followed by a round of betting. The final card is placed in the middle and the last round of betting takes place (River). Each player combines their hidden (hole) cards with the 5 community cards to form their best poker hand. An example game of Texas Hold'em is shown in Figure 2.1. In this example, player 2 wins with four queens.

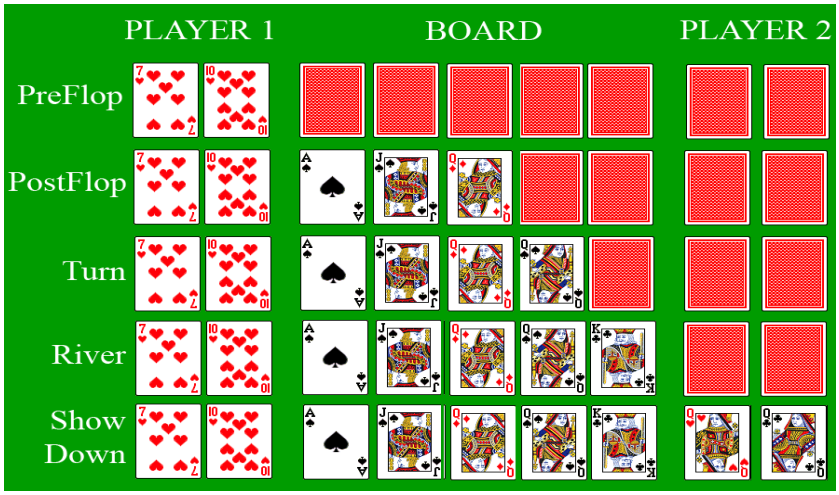


Figure 2.1: An example game of Texas Hold'em

Chapter 3

Previous Research

3.1 Findler 1977

AI techniques for making decisions under uncertainty were first applied to poker by Findler (Findler, 1977). Findler identified two different ways of playing poker. Tactical considerations would focus on short term goals (how will I win this round). Where as strategic considerations would focus on long term goals (how much money will I win/lose for the evening). He identified two different types of players. A static player would ignore both tactical and strategic considerations. Where as a learning (player/component) would consider both considerations. Findler identified four aspects of decision making; alternate courses of action, manipulating information, limited financial resources and probability/dynamic assessment of players. Findler found it important to use the past knowledge of a player's behavior and to counteract a players betting strategy.

3.2 Pearl 1988

Bayesian Networks were first introduced by Pearl (Pearl, 1988). They have been identified as a valuable statistical tool. Graph theory is combined with probability theory to provide a useful data structure to reason under uncertainty. Bayesian Networks have became a very powerful tool in both Artificial Intelligence (AI) and machine learning.

Bayesian Networks are directed acyclic graphs (DAGs) that represent a real life problem. They contain a number of nodes that represent random variables. The arcs between each node show the dependence between variables. Each node contains a conditional probability table (CPT) that contain a probability distribution over all the combinations of its parent states. Repeated use of Bayes' rule is used to compute the conditional probability for each combination. Bayesian Networks are the main tool used in our program to reason under uncertainty.

3.3 Billings 1997 - 2004

The University of Alberta have been researching ways to build agents that will perform well in perfect information games such as Poker. Their initial research began using rule based methods. These methods had limited success, therefore there was a shift to using game-theoretical methods. Additionally imperfect information game trees were used to model a game of poker. Once they had successfully implemented the initial game trees, they enhanced it to incorporate opponent modeling. This ensures the game tree adapts

to opponent actions during the course of a game. More recently they have been focussing on evaluating the performance of each approach.

3.3.1 Knowledge-based Methods and Simulation (1997 - 2001)

The University of Alberta initially defined five characteristics that define a world class poker player. These include hand strength, hand potential, bluffing, opponent modeling and unpredictability.

Hand Strength

Hand Strength (HS) is a percentile ranking of a player's current hand. It is not used in the first round of betting (preflop). It is important to note this will only be a partial hand until the final round of betting. All the possible hand combinations a particular opponent may have are enumerated. For example on the flop this would be 47 C 2 = 1,081 combinations. This is because 5 cards are already known (the 3 flop cards and our 2 hole cards). Suppose our starting hand was A♢ - Q♣ and the flop was 3♡ - 4♣ - J♡. The player's two possible hole cards and combined with the board cards to give all the possible partial hands for the opponent. Each possible hand combination is ranked and compared to our current partial hand. Partial flush and straight draws are not considered, only the basic hand categories are considered (one pair, two pair, etc). The amount of hands that are worse than ours (have a lower ranking), equal to ours (have the same ranking) and are better than ours (have a better ranking) are counted. This will give us a sum for each of these three categories. In this example, our partial hand is ahead of 628 partial hands, equal to 9 partial hands and behind 444 partial hands. To calculate the hand strength we take the ratio of hands our partial hand is ahead of or equal to over all the possible partial hand combinations (where n represents the number of active opponents).

$$\begin{aligned} HS &= \frac{(ahead + equal/2)^n}{(ahead + equal + behind)} \\ &= \frac{(628 + 9/2)^n}{(47C2)} \\ &= \frac{632.5^n}{1081} \\ &= 0.585^n \end{aligned}$$

Therefore our current partial hand has a 58.5% chance of being better than a random partial hand an opponent may hold. A higher amount of active opponents decreases our hand strength. For example, 3 active opponents would reduce the hand strength to 0.200 giving only a 20% chance of being better than a random partial hand.

Hand Potential

Hand strength is not sufficient to evaluate the strength of a player's hand. Remember in the hand strength calculations draws are not considered. Therefore a player could have a high flush or straight draw but have a low hand strength. Hand potential is used to resolve this problem. The hand potential is the chance of improving the current partial hand with the remaining board cards. For example on the flop there are two more cards to be dealt and on the turn there is one more card to be dealt.

After 7 cards have been dealt there are 45 C 2 (990) possible partial hand combinations. Suppose for each of the 1081 partial hands on the flop we combine them with the possible partial hands at the end of the game. For example, there were 628 partial hands we were ahead of on the flop. Therefore for each of these hands there are 990 possible hand combinations for the final hand. This gives us 621,720 combinations. Similar to the hand strength calculation we work out of these combinations how many we are ahead of, how many we are equal to and how many are behind. The following table gives an overview of these calculations.

Five cards	Seven cards			
	Ahead	Tied	Behind	Total
Ahead	449,005	3,211	169,504	$621,720 = 628 * 990$
Tied	0	8,370	540	$8,910 = 9 * 990$
Behind	91,981	1,036	346,543	$439,560 = 444 * 990$
Total	540,986	12,617	516,587	$1,070,190 = 1081 * 990$

Table 3.1: An example of hand potential. Adapted from (Billings et al., 2002).

From this table we can calculate the chance of improving our position, remaining the same or worsening our position. For example if we are behind, there's two ways of improving our position, one way of equaling our position and zero ways of worsening our position. Therefore we can work out the chance of improving our position and the change of worsening our position. We define these two categories as positive potential and negative potential.

$$\begin{aligned} PP(PositivePotential) &= \frac{(T[B,A] + T[B,T]/2 + T[T,A]/2)}{(T[B,S] + T[T,S]/2)} \\ &= \frac{(91,981 + 1,036/2 + 0/2)}{(439,560 + 8,910/2)} \\ &= \frac{92,499}{444,015} \\ &= 0.208 \end{aligned}$$

$$\begin{aligned} NP(NegativePotential) &= \frac{(T[A,B] + T[A,T]/2 + T[T,B]/2)}{(T[A,S] + T[T,S]/2)} \\ &= \frac{(169,504 + 3,211/2 + 540/2)}{(621,720 + 8,910/2)} \\ &= \frac{171,379.5}{626,175} \\ &= 0.274 \end{aligned}$$

Calculating the hand potential from the flop is computationally expensive. Therefore there are two alternatives to the above approach. Using only a one card look-ahead or using an algorithm to approximate the hand potential. Using the algorithm, it was shown to have less than 5% error.

Effective Hand Strength

The effective hand strength (EHS) is a combination of the hand strength and the hand potential to give an overall rank of the player's current hand compared to an active opponent.

$$\begin{aligned} Pr(Win) &= Pr(ahead) * Pr(OpponentWorsens) + Pr(behind) * (PlayerImproves) \\ &= HS * (1 - NP) + (1 - HS) * PP \end{aligned}$$

In betting situations we want to bet when we have a strong hand or potentially have a strong hand. Therefore negative potential is not a strong aspect of the formula, so NP is set to 0. As above, n represents the number of active opponents.

$$\begin{aligned} EHS &= HS^n * (1 - 0) + (1 - HS^n) * PP \\ &= HS^n + (1 - HS^n) * PP \end{aligned}$$

Using the above example with 1 active opponent.

$$\begin{aligned} EHS &= HS^n + (1 - HS^n) * PP \\ &= 0.5851 + (1 - 0.5851) * 0.208 \\ &= 0.585 + 0.086 \\ &= 0.671 \end{aligned}$$

Therefore the effective hand strength gives us a 67.1% chance of having the best hand against 1 active opponent. A threshold is used to determine the betting decision. For example a threshold of 0.5 could be used. The two betting categories could be check or bet. A rule based system would be used to decide what action to take. It important to add some randomisation to the result to avoid becoming predictable, therefore the threshold should never remain the same.

3.3.2 Game-Theoretic Methods (2002-2003)

The University of Alberta decided to change from a formula-based system to one that uses game theory. Imperfect information game trees were implemented to represent the game of poker. The initial research attempted to make abstractions of the game without altering the structure of the game. Reducing the size of the game trees can reduce the size of the problem.

Abstractions

Initially there were four types of abstractions they applied.

1. Suit equivalence or rank near equivalence.
2. Desk reduction (less than 52 playing cards).
3. Hole card reduction (less than 2 hole cards).
4. Flop reduction (less than 3 flop cards).

Betting Round Reduction

In a normal game of Texas Hold'em there is a raise limit of 4. This means there can be no more than 4 raises in a single betting round. In a single round this produces 19 possible betting sequences (or paths) through the game tree. There are 2 sequences that are impossible, therefore they are removed. The complexity of the game tree can be further simplified by reducing the amount of allowed bets in a single betting round.

Composition of models

Instead of using one model on all betting rounds, a combination of the above models can be used during different betting rounds. For example we might limit the maximum number of bets to 2 on the preflop where as we may limit the maximum number of bets to 3 on the postflop.

Bucketing

The purpose of bucketing is to group similar hands together. There are too many distinct hand types (7462) in the game of poker, therefore each hand type is categorised into buckets. For example you may want to chose 19 buckets. A classification method has to be defined to determine which hand types are categorised to which bucket. For example we might want to use our formula based methods (hand strength and hand potential) to determine the similarity of hands. Transitive probabilities are used to determine the chance of going from a pair of buckets on the flop to a pair of buckets on the postflop.

3.3.3 Adaptive Imperfect Information Game-Tree Search (2004)

The program was modified so that it can search an imperfect information game tree. An opponent model was used at all opponent decisions nodes and leaf nodes. There were two main concerns.

1. That each branch will be taken at an opponent decision node.

2. Expected value of a leaf node.

Expecti-Max Search

The purpose is to calculate the expected values over the possible betting actions. Expectimax combines the minimisation and maximization nodes of minimax search with the addition of chance nodes, where a random event happens (for example, randomly dealt cards). It is important to note that the nodes in a poker game are not dependent of each other. There is hidden information contained in the opponent's hand, therefore it must be appropriated using a probability distribution over all hands. Algorithms are used to calculate the expected value at decision nodes, by treating them as chance nodes (probability distribution) based on information known.

A minimax search uses a mixed node for the opponent's decision and a max node for our decision. However choosing a maximum strategy for our action is predictable, therefore a miximix approach is used.

There are four main types of nodes present in the game tree. You can calculate the Expected Value (EV) at each node. Pwin is equal to the probability of winning, Lpot is equal to the size of the pot and Lcost is the cost of getting there.

1. Chance nodes (based on the dealt cards)
   EV = average EV over all branches.

2. Opponent decision nodes (approximation of opponent's hand)
   EV = weighted sum of possible actions

3. Player (program) decision nodes
   EV = max(EV(action))

4. Leaf nodes
   EV = (Pwin x Lpot) - Lcost

Opponent Modeling

There must be a dynamic way of learning about the opponent. The sensible way would be to keep a history of the opponent's betting actions. It is important to note professional players will have a dynamic playing strategy, therefore it is important to use a decay function to give more importance to recent information.

3.4 Korb and Nicholson 1999

Monash (Korb, Nicholson and Jitnah, 1999) made modifications to the BPP which modeled its opponent based on the opponent's action. The BPP was based on the (Jitnah, 1993) model. This represented a base case for opponent modeling where only a small amount of information was used (opponent's last action) to model the opponent. (Korb et al., 1999) refined the hand types from 9 to 17 to include more busted and pair sub hand types. This enabled the BPP to perform better when it had a busted hand or a pair. The 17 hand types are shown in Table 3.2.

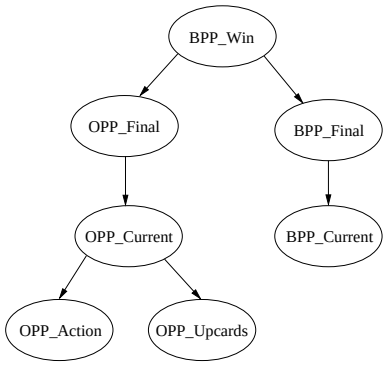


Figure 3.1: Initial implementation of the Bayesian Network

The BPP knows exactly what hand it is holding, therefore we know exactly the current hand type BPP is holding. Using that information we can calculate the probability (depending on BPP's current hand) of what hand type BPP Final will hold at the end of the final round.

However the opponent's cards are hidden to other players. The opponent's hand type has to be inferred from the opponent's action and the opponent's upcards. Using this knowledge we can calculate the probability of the opponent's final hand type depending on the opponent's current hand. Once we have calculated the probability distribution of

what hand type the opponent has and the hand type BPP holds, we can calculate the chances of winning. This is represented in the BPP Win node, which is a probability distribution over the states win and lose.

3.5 Carlton 2000

Carlton (Carlton, 2000) discovered that the model was losing information unnecessarily. He proved that because the cards were dealt from the same deck the two players hand types were dependent on each other. Therefore an arc was added between the two players final hand types. Interestingly this did not see significant improvement until the hand types were refined. Carlton refined the hand types so that individual Low Pair and Medium Pair were used. The 25 hand types are shown in Table 3.2. BPP was bluffing in situations when the visible cards provided enough information to determine the best hand, therefore a new node BPP Upcards node was added to the network to prevent BPP betting in inappropriate situations. The refined Bayesian Network is shown in Figure 3.2.

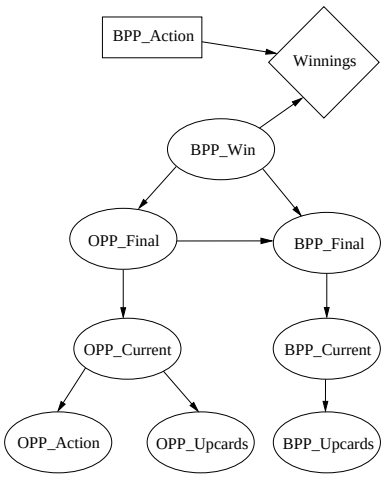


Figure 3.2: Carlton's implementation of the Bayesian Network

Carlton's main change to the architecture was the conversion to a decisions network. The betting strategy previously was determined using betting curves, without the use of the network. Carlton refined the model to include a winnings utility node and a BPP decision node. The utility determines the expected return given our chances of winning and our next action. We can use these values to determine what action to take. We endeavor to make actions that will give us a high return while remaining unpredictable.

3.6 Boulton 2003

Boulton (Boulton, 2003) decided to the change the network to incorporate Texas Hold'em in order to play other bots. He converted the old network in Figure 3.2 to comply with Texas Hold'em. Rather than having a network for each round, he added a new round node which represented the current round. The hand types were slightly changed to comply with Texas Hold'em as shown in Table 3.2. The network was changed from a dynamic network to a normal Bayesian Network. The other major change to the network was to include the importance of flush and straight draws. He added three separate nodes to

represent flush predictions. This includes the player flush node, the opponent flush node and the board flush node. With all this information you can predict whether there will be a flush on the board. The action and winning nodes are not included in the diagram. The refined Bayesian Network is shown in Figure 3.3.

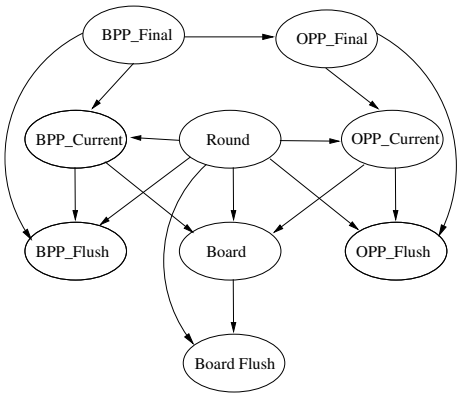


Figure 3.3: Boultons's implementation of the Bayesian Network

3.7 Mascaro 2007

Mascaro (Mascaro, 2007) has made some changes to the Bayesian Network and program over the last couple of years. He switched the partial hand types into three different categories including sequence, suit and tuple. Boulton's flush and straight modification affected the learning rate of the network. Therefore, Mascaro added the flush and straight draws into the partial hand categories. These hand types apply to the BPP current hand, opponent current hand and board card nodes. The 16 final hand types are shown in Table 3.2. Additionally Mascaro has been working on opponent modeling. Opponent modeling is now based on whether the opponent wins, loses or draws, therefore an extra arc has connected the oppAction and win nodes. Some minor changes include removing two busted sub-states and two pair sub-states to the hand resolution. Finally he added a new draw state to the win node. The refined Bayesian Network is shown in Figure 3.4.

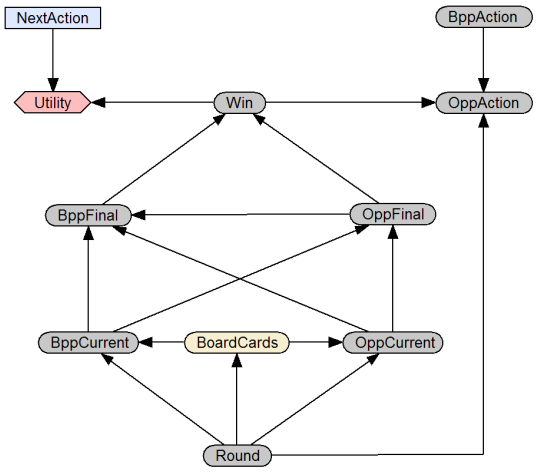


Figure 3.4: Mascaro's implementation of the Bayesian Network

		Probabilities					
No.	Hand Type	1993	1999	2000	2003	2007	Sub types
1	Busted Low		0.0024	0.0024	0.0024		18
2	Busted Medium		0.0306	0.0306	0.0306		228
3	Busted Queen		0.0280	0.0280	0.0280		209
4	Busted King		0.0441	0.0441	0.0441		329
5	Busted Ace	0.1711	0.0661	0.0661	0.0661	0.1711	493
6	Pair of Twos			0.0295	0.0295		220
7	Pair of Threes			0.0295	0.0295		220
8	Pair of Fours			0.0295	0.0295		220
9	Pair of Fives			0.0295	0.0295		220
10	Pair of Sixes			0.0295	0.0295		220
11	Pair of Sevens			0.0295	0.0295		220
12	Pair of Eights		0.2064	0.0295	0.0295	0.2064	220
13	Pair of Nines			0.0295	0.0295		220
14	Pair of Tens			0.0295	0.0295		220
15	Pair of Jacks		0.0884	0.0295	0.0295		220
16	Pair of Queens		0.0295	0.0295	0.0295	0.1179	220
17	Pair of Kings		0.0295	0.0295	0.0295	0.0295	220
18	Pair of Aces	0.3833	0.0295	0.0295	0.0295	0.0295	220
19	Two Pair Low					0.0310	231
20	Two Pair Medium					0.0501	374
21	Two Pair Kings					0.0162	121
22	Two Pair Aces	0.1150	0.1150	0.1150	0.1150	0.0177	132
23	Three of a Kind	0.1150	0.1150	0.1150	0.1150	0.1150	858
24	Straight (sequence)	0.0013	0.0013	0.0013	0.0013	0.0013	10
25	Flush (same suit)	0.1711	0.1711	0.1711	0.1711	0.1711	1277
26	Full House	0.0209	0.0209	0.0209	0.0209	0.0209	156
27	Four of a Kind	0.0209	0.0209	0.0209	0.0209	0.0209	156
28	Straight Flush	0.0013	0.0013	0.0013	0.0013	0.0013	10
Count		9	17	25	25	15	
Total		1.0000	1.0000	1.0000	1.0000	1.0000	7462

Table 3.2: The probability table above shows the probabilities for each hand type. The hand type probability is the sum of the probability of its subtypes; for example: the probabilities for Busted Ace in 1993 is the sum of the probabilities for 1999 of hands types 1 to 5. A low hand sub type normally represents sub types from 2 to 8 inclusive, where as a medium sub type normally represents hand types from 9 to Jack inclusive.

3.7.1 The Current Approach of Bluffing

BPP will bluff randomly depending on a bluffing vector. A number between 0 and 1 is defined for each round. If a random number for a given round is less than the one defined in the bluffing vector than BPP will decide to bluff. For example if the number was 0.20 for a particular round, then BPP will bluff anytime the number generated is less than 0.20. This means on average BPP will bluff 20% of the time during that round. Once BPP has committed to bluffing, it will bluff for the rest of the game.

When bluffing BPP will over-represent its current hand strength. For example, if BPP currently believes its final hand will be a full house, it may over-represent it to be four of a kind. The BppFinal node is shown in Figure 3.5 before bluffing. Once the BppFinal node

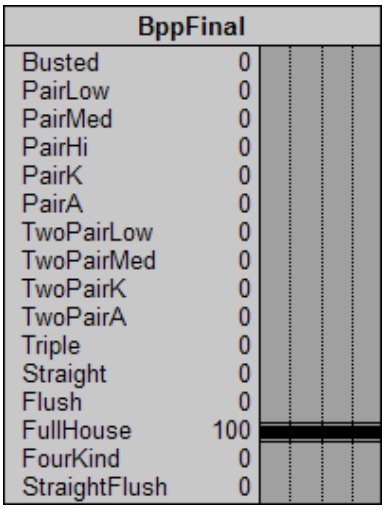


Figure 3.5: The BppFinal node before bluffing

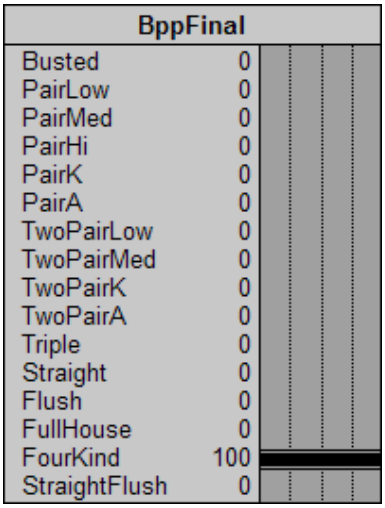


Figure 3.6: The BppFinal mode after bluffing

has been updated, this will propagate throughout the network. The chances of winning will be increased in the win node. Therefore the expected utility will be increased and this will cause BPP to be more aggressive in betting. The BppFinal node after bluffing is shown in Figure 3.6.

Chapter 4

Opponent Modeling

Opponent modeling is an important aspect in the decision making process. There are two main types of opponent models. The first type is a static opponent model. A static opponent model does not change or adapt during the interactions with the opponent. Alternatively a dynamic opponent model, would initially start with a static opponent model and adapt to the interactions with an opponent during the course of a match. A dynamic playing strategy helps avoid becoming predictable. A professional opponent will avoid being predictable, therefore the challenge is to adapt to the opponent's strategy, whether that be fixed or dynamic. It is important in some domains to note an opponent may attempt to trick their opponent, by implying something that is not true. An example would be in poker when an opponent attempts to over represent their hand strength. The Monash Research group and the University of Alberta Research Group both use a dynamic opponent modeling strategy.

4.1 Initial Model

At the start of the game we require an initial opponent model. We want to start with a generic model that represents a good poker player. The initial state of our OppAction CPT (Conditional Probability Table) represents our initial opponent model. We require data to learn more about opponents' behavior. Therefore, it was decided to use the logs from a machine vs machine poker competition to provide us with an initial starting point.

4.1.1 AAAI Competition

The University of Alberta (`http://www.cs.ualberta.ca/~pokert/2006/results.html`) held a poker competition in Vancouver, BC which allowed computer programs (also known as bots) to be entered and compete against each other. There were five entries last year including Hyperborean, BluffBot, GS2, Monash BPP and Teddy. Even though there were 5 different entries, only four competed in each competition. The poker bots competed against each other in two styles of play; the bankroll competition and the series competition. The object of the bankroll competition was to earn the most amount of money (bankroll) over a certain period. Where as the series competition focussed on which bot won the most games. A series of logs were kept from each of the games. I processed these logs to provide data for our initial opponent model. In our OppAction CPT we need to know the round, whether BPP won and BPP's action. For each match we know which round it is, we know who won and we know BPP's action by looking at the previous betting action. Therefore I can keep a frequency count of each opponent action given the round, the result and the last betting action. There are two main ways to process the logs;

as eight separate bots or combine them into one opponent. The second approach gives a good variation of opponent behavior.

Although this data was helpful, it seems the bots were not making the best decisions, therefore they could not be totally relied upon to produce a good initial opponent model. I used the combinatory bot as a baseline. During the course of the game, when calculating the win probabilities it plays out a number of hands and counts the number of wins, loses and draws. These frequency counts are converted to percentages and updated in the win node. Interested readers can refer to Boultons's (Boulton, 2003) paper to learn more about the combinatory model. The results can be seen in Table 4.1.

	Combinatory Bot			
	SBU	Hands Won	Showdowns Won	Distance
Bot 1	+0.20 ± 0.26	50.74%	53.33%	0.05554
Bot 2	+0.30 ± 0.29	51.05%	51.11%	0.05201
Bot 3	+0.16 ± 0.29	52.79%	47.96%	0.03011
Bot 4	-0.16 ± 0.25	48.90%	61.65%	0.14473
Bot 5	-0.01 ± 0.23	51.28%	52.73%	0.04185
Bot 6	+0.02 ± 0.28	51.70%	51.79%	0.04573
Bot 7	+0.14 ± 0.28	52.33%	48.22%	0.03004
Bot 8	+0.23 ± 0.31	50.77%	50.27%	0.04003
Combined Bot	-0.05 ± 0.29	50.88%	55.13%	0.02380

Table 4.1: The above results show the combinatory bot verses each bot generated from the AAAI 2006 logs. The distance shows the Bhattacharyya distance (Bhattacharyya, 1943) between the new generated OppAction CPT and the original OppAction CPT. A distance of more than 0.1 indicates a large variation. The large standard deviation indicates these results are not statistically significant.

4.1.2 Sensible OppAction CPT

We decided to generate the new OppAction CPT based on sensible values. We looked at four different scenarios. When the opponent was weak-early, weak-late, strong-early and strong-late. Based on these scenarios we generated likely probabilities distributions over the possible actions. This was only a starting point in generating the OppAction CPT. This only accounted for the first round of betting (preflop - early) and the last round of betting (river - late). It was decided to use a linear distribution to generate the values in between the first and last round. Additionally the draw states were generated by using an average over the win and loss states. Image the round was the PreFlop and the last action was a raise. To generate the draw probability distribution, I would find the average between the two probabilities distributions whose parents states are Win, PreFlop, Raise and Lose, PreFlop, Raise. It is important to note if BPP does not bet (or the opponent is first to act) the opponent should never fold. You can see the base case OppAction CPT in Table 4.2.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	73	0	0	27
Win	PreFlop	Raise	45	0	45	0	10
Win	PreFlop	None	0	0	50	0	50
Win	River	Check	0	85	0	15	0
Win	River	Bet	20	0	70	0	10
Win	River	Raise	20	0	70	0	10
Win	River	None	0	85	0	15	0
Lose	PreFlop	Call	0	35	0	0	65
Lose	PreFlop	Raise	10	0	30	0	60
Lose	PreFlop	None	0	0	35	0	65
Lose	River	Check	0	45	0	55	0
Lose	River	Bet	5	0	45	0	50
Lose	River	Raise	5	0	45	0	50
Lose	River	None	0	45	0	55	0

Table 4.2: The 'Win' state represents when the opponent is weak, where as a 'Lose' state represents a strong opponent. When BPP's action is 'None', it represents the opponent acting first.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	55	0	0	45
Win	PreFlop	Raise	5	0	74	0	21
Win	PreFlop	None	0	0	74	0	26
Win	River	Check	0	52	0	48	0
Win	River	Bet	23	0	44	0	33
Win	River	Raise	1	0	46	0	53
Win	River	None	0	42	0	58	0
Lose	PreFlop	Call	0	47	0	0	53
Lose	PreFlop	Raise	0	0	74	0	26
Lose	PreFlop	None	0	0	58	0	42
Lose	River	Check	0	16	0	84	0
Lose	River	Bet	0	0	35	0	65
Lose	River	Raise	0	0	34	0	66
Lose	River	None	0	17	0	83	0

Table 4.3: An opponent model generated by recording the frequencies from a base opponent.

4.2 Interactive Model

The purpose of the interactive model is that it adapts to a specific opponent during the course of the game. It is likely at the start of the game our image of the opponent is wrong, due to the fact it is based on a generic opponent. However the longer the game progresses the more accurate our model of the opponent will be. For example, we can enter the betting history of the opponent into our CPT during the course of the game to better predict future actions. It is important we start with a reasonable initial opponent model so that it does not take long to development an accurate model of the opponent. This is challenging because we do not know what types of opponents we will face, that is why the initial opponent model was based on a skilled opponent.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	-18	0	0	18
Win	PreFlop	Raise	-40	0	29	0	11
Win	PreFlop	None	0	0	24	0	-24
Win	River	Check	0	-33	0	33	0
Win	River	Bet	3	0	-26	0	23
Win	River	Raise	-19	0	-24	0	43
Win	River	None	0	-43	0	43	0
Lose	PreFlop	Call	0	12	0	0	-12
Lose	PreFlop	Raise	-10	0	44	0	-34
Lose	PreFlop	None	0	0	23	0	-23
Lose	River	Check	0	-29	0	29	0
Lose	River	Bet	-5	0	-10	0	15
Lose	River	Raise	-5	0	-11	0	16
Lose	River	None	0	-28	0	28	0

Table 4.4: A comparison between the original base CPT and the generated base CPT.

4.2.1 Opponent Types

In poker there are four main types of players, conservative-loose, conservative-tight, aggressive-loose and aggressive-tight. A conservative player will tend to have more conservative betting actions. They will fold, check or call rather than betting or raising. An aggressive player will have the opposite behavior. These types of players are focussed on short term goals. A tight player will tend to fold more compared to a loose player. These types of players can be determined by looking at long term behavior. It is important to note these four characteristics are mutually exclusive and can be summarised in in Table 4.5. Unfortunately I only had time to look at the two types of behaviors (conservative and aggressive).

	Tight	Loose
Conservative	Conservative-Tight	Conservative-Loose
Aggressive	Aggressive-Tight	Aggressive-Loose

Table 4.5: A summary of the different types of opponent behaviors.

Conservative Opponent

A conservative player will fold, check or call more, rather than betting or raising. The challenge was to generate a CPT that would represent a conservative player. A conservative CPT was generated by shifting the opponents actions up or down from the base CPT. For example, fold/check/call actions were shifted up where as bet/raise actions were shifted down. You can view the conservative OppAction CPT in Table 4.6.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	93	0	0	7
Win	PreFlop	Raise	65	0	35	0	0
Win	PreFlop	None	0	0	70	0	30
Win	River	Check	0	95	0	5	0
Win	River	Bet	30	0	65	0	5
Win	River	Raise	30	0	65	0	5
Win	River	None	0	95	0	5	0
Lose	PreFlop	Call	0	55	0	0	45
Lose	PreFlop	Raise	20	0	25	0	55
Lose	PreFlop	None	0	0	45	0	55
Lose	River	Check	0	65	0	35	0
Lose	River	Bet	10	0	50	0	40
Lose	River	Raise	10	0	50	0	40
Lose	River	None	0	65	0	35	0

Table 4.6: An OppAction CPT that represents a conservative player.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	73	0	0	27
Win	PreFlop	Raise	26	0	65	0	9
Win	PreFlop	None	0	0	92	0	8
Win	River	Check	0	55	0	45	0
Win	River	Bet	30	0	42	0	28
Win	River	Raise	1	0	46	0	53
Win	River	None	0	50	0	50	0
Lose	PreFlop	Call	0	62	0	0	38
Lose	PreFlop	Raise	0	0	84	0	16
Lose	PreFlop	None	0	0	81	0	19
Lose	River	Check	0	23	0	77	0
Lose	River	Bet	0	0	40	0	60
Lose	River	Raise	0	0	36	0	64
Lose	River	None	0	20	0	81	0

Table 4.7: An opponent model generated by recording the frequencies from a conservative opponent.

Aggressive Opponent

The second type of player we wanted to represent was an aggressive player. An aggressive player will bet or raise more, rather than folding, checking or calling. An aggressive player shows the opposite behavior of a conservative player. Therefore the bet/raise actions were shifted up and the fold/check/call actions were shifted down. You can view the aggressive OppAction CPT in Table 4.9.

Linear Combination

In order to prove the correctness of the new opponent type CPTs I had to show the Base CPT was a linear combination of the aggressive CPT and conservative CPT. I had to find

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	-20	0	0	20
Win	PreFlop	Raise	-39	0	30	0	9
Win	PreFlop	None	0	0	22	0	-22
Win	River	Check	0	-40	0	40	0
Win	River	Bet	1	0	-24	0	23
Win	River	Raise	-29	0	-19	0	48
Win	River	None	0	-45	0	45	0
Lose	PreFlop	Call	0	7	0	0	-7
Lose	PreFlop	Raise	-20	0	59	0	-39
Lose	PreFlop	None	0	0	36	0	-36
Lose	River	Check	0	-42	0	42	0
Lose	River	Bet	-10	0	-10	0	20
Lose	River	Raise	-10	0	-14	0	24
Lose	River	None	0	-45	0	45	0

Table 4.8: A comparison between the original conservative CPT and the generated conservative CPT.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	53	0	0	47
Win	PreFlop	Raise	25	0	55	0	20
Win	PreFlop	None	0	0	30	0	70
Win	River	Check	0	75	0	25	0
Win	River	Bet	10	0	75	0	15
Win	River	Raise	10	0	75	0	15
Win	River	None	0	75	0	25	0
Lose	PreFlop	Call	0	15	0	0	85
Lose	PreFlop	Raise	5	0	35	0	65
Lose	PreFlop	None	0	0	25	0	75
Lose	River	Check	0	25	0	75	0
Lose	River	Bet	0	0	40	0	60
Lose	River	Raise	0	0	40	0	60
Lose	River	None	0	25	0	75	0

Table 4.9: An OppAction CPT that represents an aggressive player.

the correct alpha value that would satisfy the following formula.

$$BaseCPT = \alpha * AggressiveCPT + (1 - \alpha) * ConservativeCPT$$

For each alpha value I used the Bhattacharyya distance (Bhattacharyya, 1943) to show the difference between the original Base CPT and the Base CPT generated by the above formula. The goal was the minimise the distance. The alpha value was 0.492479 which minimised the Bhattacharyya distance. You can not exactly produce the original CPT because each combination was not equally shifted.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	32	0	0	68
Win	PreFlop	Raise	1	0	64	0	36
Win	PreFlop	None	0	0	51	0	49
Win	River	Check	0	42	0	58	0
Win	River	Bet	13	0	44	0	43
Win	River	Raise	1	0	43	0	56
Win	River	None	0	34	0	66	0
Lose	PreFlop	Call	0	29	0	0	71
Lose	PreFlop	Raise	0	0	61	0	39
Lose	PreFlop	None	0	0	41	0	59
Lose	River	Check	0	10	0	90	0
Lose	River	Bet	0	0	28	0	72
Lose	River	Raise	0	0	33	0	67
Lose	River	None	0	14	0	86	0

Table 4.10: An opponent model generated by recording the frequencies from an aggressive opponent.

Win	Round	BppAction	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	0	-21	0	0	21
Win	PreFlop	Raise	-24	0	8	0	16
Win	PreFlop	None	0	0	21	0	-21
Win	River	Check	0	-33	0	33	0
Win	River	Bet	3	0	-31	0	28
Win	River	Raise	-9	0	-32	0	41
Win	River	None	0	-41	0	41	0
Lose	PreFlop	Call	0	14	0	0	-14
Lose	PreFlop	Raise	0	0	26	0	-26
Lose	PreFlop	None	0	0	16	0	-16
Lose	River	Check	0	-15	0	15	0
Lose	River	Bet	0	0	-12	0	12
Lose	River	Raise	0	0	-7	0	7
Lose	River	None	0	-11	0	11	0

Table 4.11: A comparison between the original aggressive CPT and the generated aggressive CPT.

Aggressive and Conservative Opponents

To test the new opponent models I had to produce different opponent types. This included an aggressive and conservative version of BPP. BPP uses betting curves to decide which action to take (fold, check/call or bet/raise). Interested readers are encouraged to refer to Carlton's (Carlton, 2000) paper to learn more about betting curves. The easiest way to modify BPP's betting behavior is to shift the betting curves. A shift to the left represents more aggressive behavior where as a shift to the right represents more conservative behavior. The main challenge was finding the correct shift amount to represent each player type. Additionally the shifting amount might not necessarily be equal due to the fact BPP might be slightly conservative or slightly aggressive. The betting curves were original introduced to provide randomisation to BPP's behavior.

Generated Opponent CPTs

During the course of a game I kept a frequency count of each of the opponents actions. I used the Base CPT as the first opponent and the three different versions (aggressive hybrid, hybrid, conservative hybrid) of BPP as my second opponent. There were two main aims in viewing the opponents actions. The first aim was to see if the betting curve adjustments were producing aggressive/conservative players compared to the base version. The second aim was to compare the generated opponent CPTs with our opponent type CPTs. This had the additional benefit of finding any incorrect BPP behavior. For example, I discovered in BPP's behavior, it were folding unnecessarily. Therefore, I set those fold actions to 0 in the CPTs. Table 4.10, table 4.3 and table 4.7 show the different generated opponent CPTs.

BPP already had an over-write rule to never fold if it could check. However on the preflop check is an unallowed action. Therefore this over-write was not being affective when BPP could simple call (without having to contribute more money). This was clearly noticeable in the generated opponent CPTs where BPP was folding when it did not need to. I extended the over-write rule to fix this problem.

Comparison

As mentioned above I compared the generated opponent CPTs with the opponent type CPTs. It is important to note depending on how BPP plays a large variation may not be a problem. It depends on a lot of factors such as the hands played, bluffing amount and opponent type. I looked at the numbers which represented a high difference to try and improve the opponent type CPTs. As mentioned above, this helped find the difference between expected behavior and actual behavior. Table 4.11, table 4.4 and table 4.8 show the comparisons between the original opponent CPT and the generated opponent CPT. Further research could be undertaken to fully utilise these findings.

New Network Structure

A new node (conservative) was added to the network to represent these two types of players. The first state false, represents an aggressive player where as the true state represents a conservative player. Initially this node has an equal distribution over true and false, because we are unsure about our opponent's behavior at the start of the game. Therefore the opponent CPT at the start of the game would roughly represent the Base CPT. The changes to the network are shown in Figure 4.1.

This change to the network means that each combination in the OppAction table will now be split into two probability distributions. Therefore for each stage in the game we can determine what actions a conservative player will be likely to make and what actions an aggressive player will make. Normally we would not categorise a player as aggressive or conservative but rather suggest a player favors a type of play. For example, if a player bets a lot we may suggest they are more aggressive (perhaps 70% aggressive, 30% conservative). We can combine the conservative and aggressive CPTs to generate our final Combined CPT as shown in Table 4.12. Therefore at any stage of the game we can determine the opponents action based on the probabilities in the opponent type node. For example when the player type node is set to 100% conservative, it will use the true state in the OppAction table. Alternatively when the player type node is set to 0% conservative it will use the false state in the OppAction table. Any percentage in between well use a linear combination of the two probability distributions.

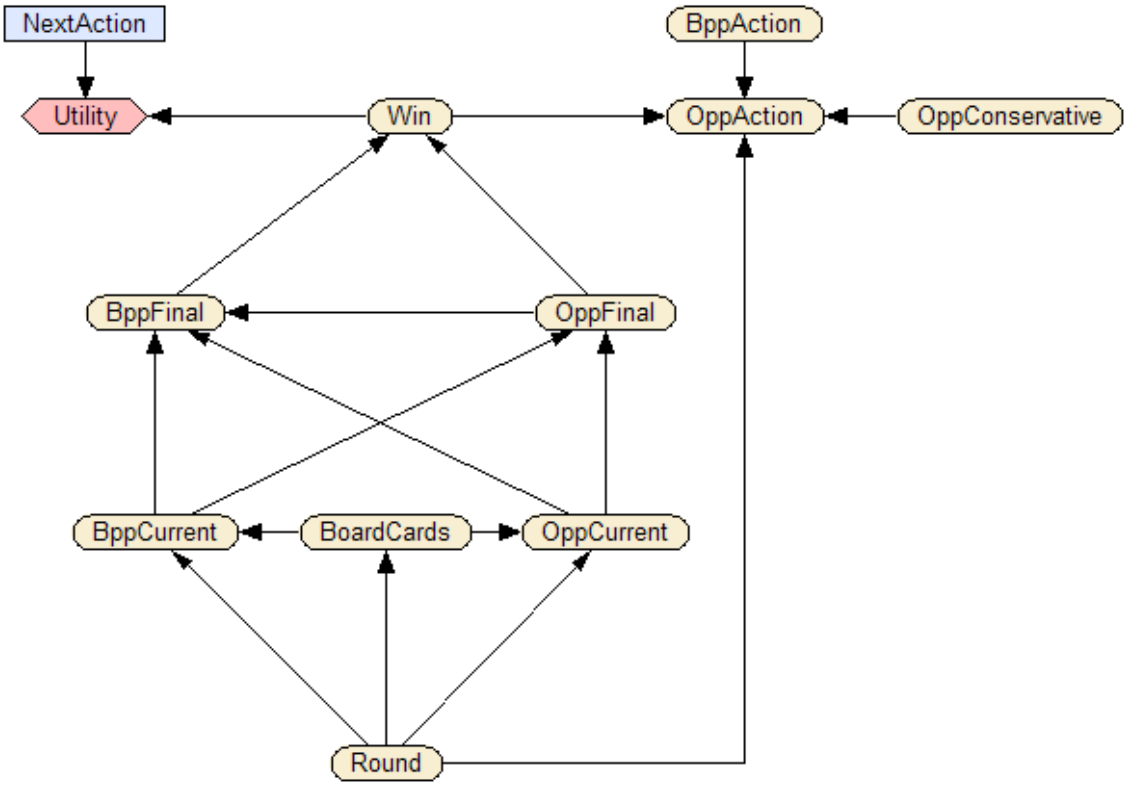


Figure 4.1: The new network structure.

4.2.2 Results

To determine the accuracy of the new models I performed run tests between the different versions of the bots. The first opponent represents the various opponent models, the second opponent represents different behaviors of the combinatory and hybrid models. When updating the win node during the course of the game, the combinatory model plays out a number of hands to work out the chances of winning. Where as the hybrid model uses a percentage of the combinatory and a percentage of the network to calculate the chances of winning. The hybrid bot in my examples use 70% combinatory, 30% network. I used the following agents as the second opponent. Please note the first table uses the Old CPT for the opponent agent where as the second table uses the Base CPT for the opponent agent.

The following abbreviations were used in the results tables.

- A. Hybrid: Aggressive hybrid (betting curves shifted left).
- Hybrid: Base hybrid.
- C. Hybrid: Conservative hybrid (betting curves shifted right).
- A. Comb.: Aggressive combinatory (betting curves shifted left).
- Comb.: Base combinatory.

Win	Round	BppAction	OppConservative	Fold	Check	Call	Bet	Raise
Win	PreFlop	Call	false	0	53	0	0	47
Win	PreFlop	Call	true	0	93	0	0	7
Win	PreFlop	Raise	false	25	0	55	0	20
Win	PreFlop	Raise	true	65	0	35	0	0
Win	PreFlop	None	false	0	0	30	0	70
Win	PreFlop	None	true	0	0	70	0	30
Win	River	Check	false	0	75	0	25	0
Win	River	Check	true	0	95	0	5	0
Win	River	Bet	false	10	0	75	0	15
Win	River	Bet	true	30	0	65	0	5
Win	River	Raise	false	10	0	75	0	15
Win	River	Raise	true	30	0	65	0	5
Win	River	None	false	0	75	0	25	0
Win	River	None	true	0	95	0	5	0
Lose	PreFlop	Call	false	0	15	0	0	85
Lose	PreFlop	Call	true	0	55	0	0	45
Lose	PreFlop	Raise	false	5	0	35	0	65
Lose	PreFlop	Raise	true	20	0	25	0	55
Lose	PreFlop	None	false	0	0	25	0	75
Lose	PreFlop	None	true	0	0	45	0	55
Lose	River	Check	false	0	25	0	75	0
Lose	River	Check	true	0	65	0	35	0
Lose	River	Bet	false	0	0	40	0	60
Lose	River	Bet	true	10	0	50	0	40
Lose	River	Raise	false	0	0	40	0	60
Lose	River	Raise	true	10	0	50	0	40
Lose	River	None	false	0	25	0	75	0
Lose	River	None	true	0	65	0	35	0

Table 4.12: The Final OppAction CPT. This is a combination of the aggressive CPT and the conservative CPT.

- C. Comb.: Conservative combinatory (betting curves shifted right).

	Opponent Agent (Old CPT)					
Opponent Model	A. Hybrid)	Hybrid	C. Hybrid	A. Comb.	Comb.	C. Comb.
Base CPT	0.55 ± 0.28	0.37 ± 0.57	0.14 ± 0.24	0.02 ± 0.25	0.20 ± 0.29	0.20 ± 0.15
Conservative CPT	0.47 ± 0.24	0.22 ± 0.24	0.22 ± 0.23	0.02 ± 0.21	0.30 ± 0.17	0.17 ± 0.13
Aggressive CPT	0.63 ± 0.35	0.27 ± 0.31	0.19 ± 0.30	0.03 ± 0.25	0.30 ± 0.31	0.20 ± 0.14
Combined CPT	0.45 ± 0.25	0.37 ± 0.34	0.22 ± 0.28	0.10 ± 0.32	0.28 ± 0.33	0.20 ± 0.13
Last 10,000	0.46 ± 0.26	0.33 ± 0.30	0.23 ± 0.20	0.10 ± 0.33	0.47 ± 0.30	0.13 ± 0.10

Table 4.13: Results for the Opponent Type implementation after 20,000 hands. A positive number represents the Opponent Model winning, where as a negative number represents the Opponent Agent winning.

You will notice the Opponent Models were able to correctly identify the Opponent Agent. For example, the conservative CPT performed best against the conservative implementations of BPP. Likewise the aggressive CPT performed best against the aggressive implementations of BPP. This is because we started with the correct (or close to) the correct initial opponent model. The other models had to adapt to the opponents actions before it could generate the correct opponent model. Interestingly the Hybrid was best modeled by the Base/Combined CPTs where as the combinatory model had no clear winner. It is important to note a lot of these results have a high standard deviation, therefore they are not fully statistically significant. To be statistically significant the SBU would need to be at least double the standard deviation. Some of the results above are close to this requirement however most of them fall short. Therefore, further experiments would need to be under taken against different bots they are not variations of BPP. Unfortunately I was unable to compete against other bots due to the fact I would need to write a program to comply with the University of Alberta's online protocol. Other alternatives such as their training program required the bot to be ported to Java.

Statistically significant results may have been achieved by setting up the experiment as follows. Generate an Opponent CPT by recording the actions of BPP. This would allow us to exactly identify our opponent when we use BPP as the opponent. The OppAction CPT could be shifted up or down accordingly, depending on the shift amount applied to the betting curves. A comparison could be used to compare the new opponent type CPT with the generated opponent type CPT. You would expect there to be minimal difference; in fact you would try to minimise this difference. Not only would you expect these experiments to produce a higher SBU but the standard deviation should be lower, which in turn would produce statistically significant results.

	Opponent Agent (New CPT)		
Opponent Model	A. Hybrid	Hybrid	C. Hybrid
Base CPT	0.02 ± 0.21	0.04 ± 0.16	0.16 ± 0.17
Conservative CPT	0.07 ± 0.26	-0.07 ± 0.13	0.20 ± 0.16
Aggressive CPT	0.13 ± 0.16	0.09 ± 0.22	0.10 ± 0.18
Combined CPT	0.20 ± 0.25	0.03 ± 0.24	0.15 ± 0.19
Last 10,000	0.28 ± 0.23	0.06 ± 0.20	0.18 ± 0.17

Table 4.14: Results for the Opponent Type implementation after 20,000 hands. A positive number represents the Opponent Model winning, where as a negative number represents the Opponent Agent winning.

I ran similar experiments using the new base CPT for the Opponent Agent rather than using the old CPT. This will remove the bias from using an improved initial opponent model. Therefore these rules show purely the improvements made using the new adaptive opponent model. Like the previous results the opponent types that match perform well. For example, the Conservative CPT performs well against the Conservative Hybrid. It is important to note these results are not fully statistically significant due to the fact they have a high standard deviation.

An aggressive player will normally win more matches but not necessarily more money, where as a conservative player will win less matches. In the above results this can be shown by the fact the aggressive Opponent Agents won more matches where as the conservative Opponent Agents won less matches compared to the opponent model. As you

	Opponent Agent					
Opponent Model	A. Hybrid	Hybrid	C. Hybrid	A. Comb.	Comb.	Comb. C.
Base CPT	40.02%	46.80%	53.70%	41.89%	46.28%	66.03%
Conservative CPT	35.53%	42.86%	50.81%	37.19%	43.20%	62.82%
Aggressive CPT	41.45%	48.85%	55.10%	45.14%	48.51%	68.33%
Combined CPT	41.71%	49.17%	55.31%	45.64%	48.68%	68.39%
Last 10,000	42.46%	48.94%	55.46%	44.26%	49.79%	66.57%

Table 4.15: Results for the Opponent Type implementation after 20,000 hands. These results are represented using win/lose.

would expect the hybrid and the conservative opponent agents are in between the aggressive and conservative implementations. Interestingly both the hybrid and combinatory implementations won more matches, suggesting they are aggressive. This result can be confirmed by the fact the opponent actions for the hybrid bot were aggressive (please refer to generated opponent CPT section). It is important to note a higher win percentage is not the main goal. The main goal is to receive the highest return. Therefore a higher SBU is favorable over a high win rate.

Opponent Type Learning

The final objective was to ensure the conservative node was correctly identifying the type of player. At the start of the game when we do not know anything about the type of player the priors at set to 50% true, 50% false. A conservative opponent should cause the prior to head towards 100% true and 0% false, where as an aggressive player should head towards 0% true and 100% false. The following graph in figure 4.2 shows the conservative node over the course of a game, with a conservative opponent and an aggressive opponent.

When trying to find the correct shift amount to produce a conservative and aggressive player, it became obvious BPP was aggressive in its betting behavior. Therefore there was not a need to shift the curves to produce aggressive play. However, to produce conservative behavior became a bigger challenge. A substantial shift was required to produce sensible results. Even with a large shift, it seemed BPP was still producing some aggressive behavior. There are different experiments that could be undertaken to compare to these results. You could use other bots that have a certain type of behavior and see if BPP detects that behavior. Perhaps you could even manually play against BPP and see if BPP detects your behavior based on your betting actions. Either way, by just shifting the curves did not provide completely sound results.

Normally during the course of a match, the opponent will alter their behavior. For example, at the start of the game they may bet conservatively to gage their opponents and towards the end of the game they may bet more aggressively. In fact, probably during the course of a game the type of opponents behavior may vary considerably. So, rather than testing one type of behavior I decided to used a mixed strategy. At the start of the match I altered the betting curves to produce conservative behavior and towards the end of the match, I readjusted the curves to produce aggressive behavior. This type of match is more realistic compared to an opponent that exibits one type of game play. The graph can be seen in Figure 4.3.

It is important to note that BPP is detecting the opponent very quickly. In fact if we assume there are approximately four actions per game on average, then after about

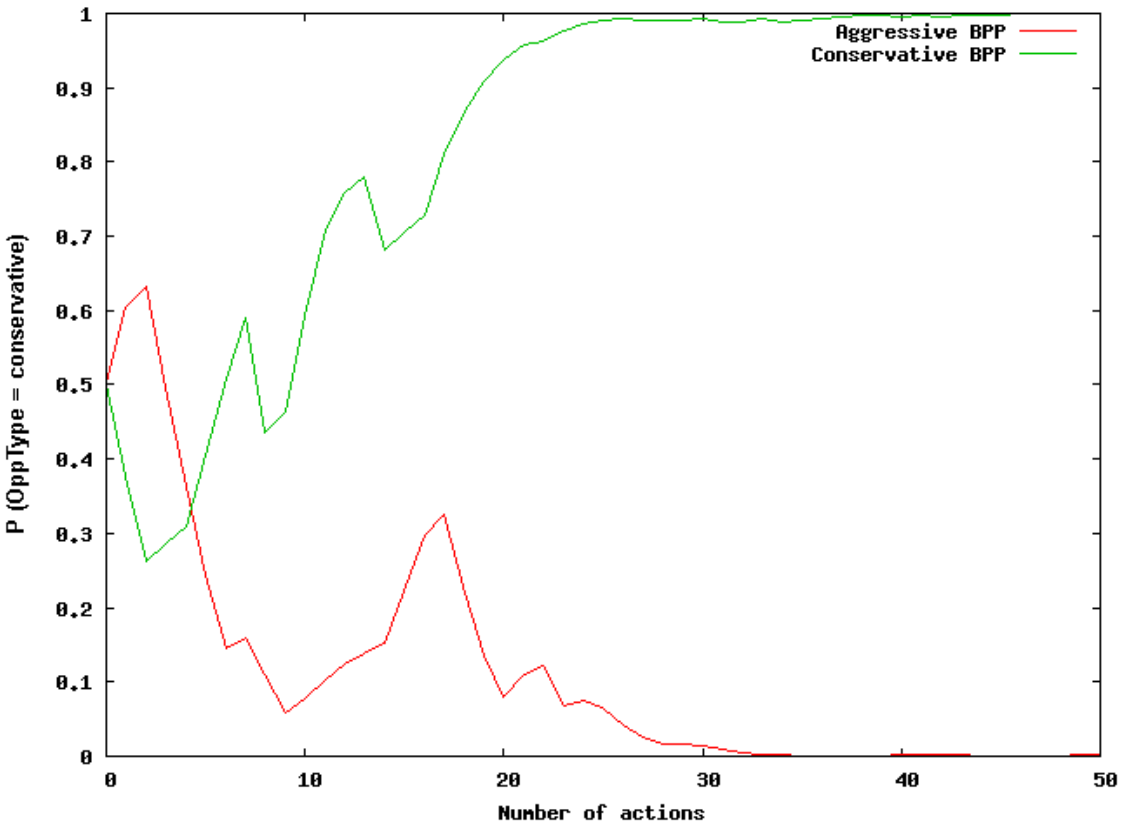


Figure 4.2: A graph of the prior probabilities in the opponent type node during the course of a match.

ten games BPP has detected its opponent. These results seem strange, because you would not expect BPP to learn that fast. I decided to run further experiments to see if this was consistent behavior over many trials. I ran BPP against the different opponent types over 10 trials and then took the average. All the trails show similar results; BPP is learning very quickly. Interestingly, on average BPP is detecting aggressive actions quicker than conservative actions. Perhaps, this confirms BPP has aggressive behavior. The number of actions, required to detect a certain behavior is dependent on the types of hands. For example, an player may seem to be aggressive just because they have been dealt strong hands (not because they are aggressive). This could explain why there is a lot of variation in the results. These results can be seen in Table 4.16.

	1	2	3	4	5	6	7	8	9	10	Mean
Aggressive	45	51	87	30	37	33	43	100	51	47	52.40
Conservative	65	68	48	41	77	82	74	54	71	90	67.00

Table 4.16: The table shows the number of actions required to detect an aggressive player and the number of actions required to detect a conservative player.

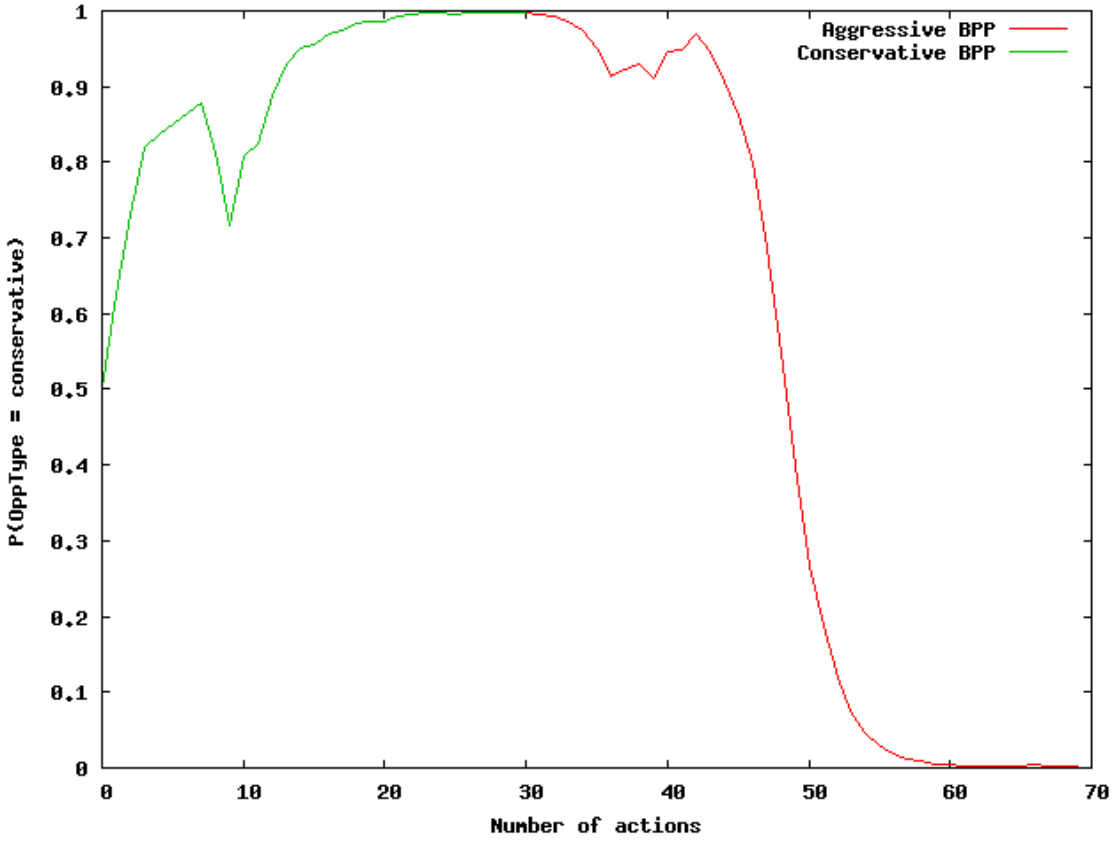


Figure 4.3: A graph of the prior probabilities in the opponent type node during the course of a match with a mixed strategy.

Chapter 5

Conclusion

Perfect information games such as chess and Othello have been extensively researched, where as imperfect information games have had less attention. Even though there have been ideas on how to solve the problem, no solution has provided a complete answer to the problem. This paper has described the contributions to our poker program (BPP). I have shown the changes have caused improvements to BPP.

The improvements to the initial OppAction CPT and the additional opponent type node have provided improvements to the opponent model. BPP is now able to identify the type of opponent. This provides additional information about our opponent which will enable us to better identify the opponent's actions based on their type. Further research could investigate the benefit of using the opponent type directly in BPP's betting decision process. For example, when calculating the utility of a certain betting action you could also take into account the type of opponent.

There are many possible extensions to the current implementation. For example, future enhancements may include adding two more personality types loose and tight. It is harder to model these types of players due to the fact you need observe long-term behavior for a player to determine if they are tight or loose. Another node that possibly could be added to the network would include a bluffing node, to suggest whether the opponent is bluffing or not. Using these strategies of the opponent, combined with BPP's strategy you could improve the betting decision making process for BPP.

To be a good player player, you must adapt to your opponent. Additionally you must not be predictable. The purpose of this paper was to improve the opponent model. A dynamic opponent model is required to determine the betting strategy of a specific opponent. Even though the poker program out performs previous implementations, it is still a long way off beating professional poker players.

Appendix A

Perl Source Code

A.1 Config.pl

A.2 Evaluate.pl

A.3 Frequency.pl

A.4 GenerateCPT.pl

A.5 Parse.pl

Appendix B

Python Source Code

B.1 CalculatePlayerType.py

B.2 CompareLogs.py

B.3 CPTProof.py

B.4 OpponentLogging.py

B.5 UpdateCPT.py

References

Albrecht, D. W., Zukerman, I. and Nicholson, A. E. (1998). Bayesian models for keyhole plan recognition in an adventure game, *User Modeling and User-Adapted Interaction* **8**(1–2): 5–47.

Amit, A. and Markovitch, S. (n.d.). Learning to bid in bridge, *Mach. Learn.* **63**(3): 287–327.

Ankeny, N. C. (1981). *Poker Strategy: Winning with Game Theory*, first edn, Basic Books.

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions, *Bulletin of the Calcutta Mathematics Society* **35**: 99–110.

Billings, D. (1995). *Computer poker*, Master's thesis, Department of Computer Science, University of Alberta.

Billings, D. (2006a). *Algorithms and Assessment in Computer Poker*, PhD thesis, Department of Computer Science, University of Alberta.

Billings, D. (2006b). A tool for the direct assessment of poker decisions, *Technical report*.

Billings, D., Burch, N., Davidson, A., Holte, R. C., Schaeffer, J., Schauenberg, T. and Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker, *in* G. Gottlob and T. Walsh (eds), *IJCIA*, Morgan Kaufmann, pp. 661–668.

Billings, D., Castillo, L. P., Schaeffer, J. and Szafron, D. (1999). Using probabilistic knowledge and simulation to play poker, *AAAI/IAAI*, pp. 697–703.

Billings, D., Davidson, A., Schaeffer, J. and Szafron, D. (2002). The challenge of poker, *Artificial Intelligence* **134**(1–2): 201–40. Special Issue on Games, Computers and Artificial Intelligence.

Billings, D., Papp, D., Schaeffer, J. and Szafron, D. (1998a). Opponent modeling in poker, *Proceedings of the Fifteenth National AAAI Conference*, Sweden, pp. 493–499.

Billings, D., Papp, D., Schaeffer, J. and Szafron, D. (1998b). Poker as a testbed for AI research, *in* R. E. Mercer and E. Neufeld (eds), *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence (AI-98)*, Vol. 1418 of *LNAI*, Springer, Berlin, pp. 228–238.

Billings, D., Pena, L., Schaeffer, J. and Szafron, D. (1999). Using probabilistic knowledge and simulation to play poker, *AAAI/IAAI*, pp. 697–703.

Boulton, D. (2003). Bayesian poker, Honours thesis, School of Computer Science and Software Engineering, Monash University.

Cactus Kev's Poker Hand Evaluator (2006). `http://www.suffecool.net/poker/evaluator.html`.

Carlton, J. (2000). Bayesian poker, Honours thesis, School of Computer Science and Software Engineering, Monash University.

Davidson, A. (2002). *Opponent modelling in poker: Learning and acting in a hostile and uncertain environment*, Master's thesis, Department of Computer Science, University of Alberta.

Davidson, A., Billings, D., Schaeffer, J. and Szafron, D. (2000). Improved opponent modeling in poker, *International Conference on Articial Intelligence*.

Doyle, A. (1995). Graphical user interface for poker playing, Honours thesis, Department of Computer Science, Monash University.

Findler, N. (1977). Studies in machine cognition using the game of poker, *Communications of the ACM* **20**: 230–245.

Jitnah, N. (1993). An expert system that uses bayesian reasoning to play poker, Honours thesis, Department of Computer Science, Monash University.

Kan, M. (2007). *Postgame analysis of poker decisions*, Master's thesis, Department of Computer Science, University of Alberta.

Koller, D. and Pfeffer, A. (1995). Generating and solving imperfect information games, *IJCAI-95*, pp. 1185–1192.

Koller, D. and Pfeffer, A. (1997). Representations and solutions for game-theoretic problems, *Artificial Intelligence* **94**: 167–215.

Korb, K. B. and Nicholson, A. E. (2004). *Bayesian Artificial Intelligence*, Computer Science and Data Analysis, Chapman & Hall / CRC, Boca Raton.

Korb, K. B., Nicholson, A. E. and Jitnah, N. (1999). Bayesian poker, *in* UAI'99 (ed.), *Proceedings of the 15th International Conference on Uncertainty in Artificial Intelligence*, Sweden, pp. 343–350.

Lourdes, M., Szafron, D., Lu, P. and Mcquarrie, S. (1999). *Probabilities and simulations in poker*, Master's thesis, Department of Computer Science, University of Alberta.

Markovitch, S. and Reger, R. (2005). Learning and exploiting relative weaknesses of opponent agents, *Autonomous Agents and Multi-Agent Systems* **10**(2): 103–130.

Mascaro, S. (2007). Personal Communication.

McLeod, J. (2006). Rules of card games: Poker hand ranking, `http://www.pagat.com/vying/pokerrank.html`.

Mud Faq (2006). `http://www.mudconnect.com/mudfaq/`.

Online Bridge Games, lessons, videos (2006). `http://www.bridgedoctor.com/bridge-lessons-intro.php`.

Papp, D. (1998). *Dealing with imperfect information in poker*, Master's thesis, Department of Computer Science, University of Alberta.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Mateo, CA.

Poker Rules for Beginners (2007). `http://www.pokerlistings.com/poker-rules`.

Richards, M. and Amir, E. (2007). Opponent modeling in scrabble, *in* M. M. Veloso (ed.), *IJCAI*, pp. 1482–1487.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, Prentice Hall Series in Artificial Intelligence, second edn, Prentice Hall, Englewood Cliffs, NJ.

Schaeffer, J., Billings, D., Pena, L. and Szafron, D. (1999). Learning to play strong poker, *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 697–703.

Schauenberg, T. (2006). *Opponent modelling and search in poker*, Master's thesis, Department of Computer Science, University of Alberta.

Smith, C. and Zinkevich, M. (2006). Computer poker tournament at AAAI'06, `http://www.cs.ualberta.ca/~pokert/`.

Thomson (1994). Bayesian poker, Honours thesis, Department of Computer Science, Monash University.

von Neumann, J. and Morgenstern, O. (1953). *Theory of games and economic behaviour*, Princeton.

Zadeh, N. (1974). *Winning poker systems*, Prentice Hall, inc.