CASPER: DESIGN AND DEVELOPMENT OF A CASE-BASED POKER PLAYER

by

Jonathan Rubin

A thesis submitted in partial fulfilment of the requirements
for the degree of **Master of Science** in **Computer Science**,
The University of Auckland, 2007.

Abstract

Poker provides a challenging domain for Artificial Intelligence research due to the game’s properties such as hidden information (the other player’s cards) and non-determinism (random shuffling of the deck). Recent approaches to Poker research have required intensive knowledge engineering efforts. This thesis discusses the design and development of a CASe-based Poker playER (CASPER) that uses the Case-Based Reasoning methodology to make betting decisions at the poker table. The results suggest it is possible to record instances of games played between strong poker players and then reuse these to obtain a similar performance therefore bypassing the need for the initial, intensive knowledge engineering process. An investigation into deriving optimal feature weights using evolutionary algorithms has also been conducted. Casper has been extensively evaluated by challenging various sets of opponents, including both computerised opponents and real opponents.

Acknowledgements

Thank you to my supervisor, Ian Watson, for the opportunities you provided for me and the time and effort you devoted to me. I also need to thank the University of Alberta Computer Poker Research Group. Without their past research and tools this thesis could never have been completed. Lastly, thank you to all my friends and family who encouraged and supported me along the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 AI and Games

Games offer a well suited domain for Artificial Intelligence (AI) investigation and experimentation due to the fact that a game is usually composed of several well-defined rules which players must adhere to. Most games have precise goals and objectives which players must meet to succeed. For a large majority of games the rules imposed are quite simple, yet the game play itself involves a large number of very complex strategies. Furthermore, a performance metric is naturally embedded into the game itself. Success can therefore easily be measured by factors such as the amount of games won or the ability to beat certain opponents.

Games are often classified by the amount of information available to the players. If a player has access to all the information they require about the game during play then the game can be classified as having *perfect information*. However, if some of that information is hidden from the player the game is known as having *imperfect information*. Take for example the game of chess. Chess is a game of *perfect information* because each player can look down upon the board and obtain all the information necessary to make their playing decisions. On the other hand, the game of poker is a game of *imperfect information*. In poker players are given cards which only they can see, therefore players now have to make decisions based on hidden information because they cannot see their opponents' cards.

Games can be further classified as either *deterministic* or *stochastic*. If a game contains chance elements, such as the roll of a dice, this introduces randomness into the game. These types of games are known as *stochastic* games and examples include bridge, backgammon and poker. The absence of these chance elements ensures the game is *deterministic*. Games such as chess, checkers and go are examples of deterministic games.

Until recently the main focus of AI related research has been on *deterministic* games with *perfect information* such as chess (Campbell, et al. 2002) and checkers (Schaeffer, et al. 1996). Success for these types of games has mainly come about through the use of brute-force search techniques and increases in hardware processing speeds (Schaeffer, et al. 1992). However, these approaches have been criticized for a lack of applicability to real world problems. It is hoped that by studying *stochastic* games with *imperfect information* results obtained may be more applicable to real world domains. *Stochastic*, *imperfect information* games make it necessary to handle uncertain knowledge and issues such as dealing with chance and deception (Davidson 2002), issues that are closer to the reality that we live in.

1.2 AI and Poker

Poker is a *stochastic* game with *imperfect information*. It is *stochastic* because the shuffling of cards introduces randomness into the game. It is a game of *imperfect information* because players cannot see their opponent's cards, therefore players need to make decisions based on hidden information. Given the relatively simple rules of the game there are an enormous amount of subtle and sophisticated scenarios that can occur during a hand of play (this is particularly true of the Texas Hold'em variation). Poker ensures that issues such as probabilistic reasoning and opponent modelling needs to be considered. Poker is an inherently psychological game. It is crucial to have an understanding of your opponent and how they think to be able to play well. All these factors make poker a challenging domain for AI related research where advances are likely to be beneficial outside the realm of poker itself.

The University of Alberta Poker Research Group[1] has been extensively researching computer poker for several years. The result of their efforts have been the production of systems such as *Poki* (Davidson 2002) and *PsOpti* (Billings, et al. 2003). *PsOpti* was designed to challenge only one opponent at the poker table, whereas *Poki* is more suited to play at a full table, i.e. consisting of 10 players. *Poki* has been extensively tested against real opponents using "play-money" and the results indicate that *Poki* consistently makes profit against its competition. *Poki* has been rated as having intermediate playing strength at a full poker table (Davidson 2002).

[1] http://www.cs.ualberta.ca/~games/poker/

1.2.1 CBR and Poker

While there has been much focus on AI and poker related research in the recent past, especially from the University of Alberta Poker Research Group, there has been little effort in applying the tools and techniques of Case-Based Reasoning (CBR) to the area of computer poker. CBR is an AI methodology (Mántaras, et al. 2005) that adapts and uses solutions to past problems to resolve current problems. It is often true that poker players act in a way that has proven to be successful in similar, past situations. An introduction to CBR is given in section 1.4.

1.3 The game of Poker

There are numerous variations of the game of poker available. The games differ by various aspects such as the number of *hole cards* dealt (cards which only one player can see and use to make their best hand), the number of community cards dealt (cards which all players can see and use to make their best hand), the order in which players bet and the limits imposed on a player's bet.

1.3.1 Betting

There are two variations which control the amount that a player may bet: *limit* and *no limit.* In a *limit* game player's bets are restricted to a certain amount. Conversely, in *no limit* there is no restriction on the amount that a player can bet. A player's betting decision can be to *fold*, *check*, *call*, *bet* or *raise*. These are described below:

Fold: A player can *fold* their cards if they are facing a bet by another player, but they don't wish to match the bet. Once a player *folds* they are no longer involved in the current hand, but can still participate in any future hands.

Check/Call: When it comes time for a player to make his/her decision they can *check* if there have been no bets made by other players. *Checking* means the player does not need to invest any of their money into the pot to stay in the current hand. If, however, an opponent has made a bet then a player can *call* the bet by adding to the pot the exact value of the current bet. By contributing their own money to the pot they are able to stay in the current hand.

Bet/Raise: A player can add their own money to the pot over and above what is needed to stay in the current round. If the player is able to *check,* but they decide to add money to the pot this is called a *bet*. If a player is facing a bet from an opponent, but instead of deciding to just *call* the bet they decide to add more money to the pot, this is called a *raise*.

1.3.2 5-Card Draw

In the past the most popular poker variation was 5-card draw. In 5-card draw money enters the pot by way of each player's *ante*, i.e. a forced bet by each player before any cards are dealt. This ensures that there is something in the pot to play for. Players are each then dealt five *hole cards*. No community cards are used. A round of betting occurs where each player decides how they wish to play using the above betting decisions (*fold*, *call*, *bet*, *etc*…). After the first round of betting, players can exchange any number of their five cards for new cards from the un-dealt portion of the deck. After a final round of betting, if there are still at least two players in the hand, a *showdown* occurs where all players that are left reveal which cards they were actually holding. The player with the best hand wins all of the money in the pot.

The use of no community cards in 5-card draw poker ensures that all players' cards are hidden. The only information available to a player to help inform their decision is their opponents' betting strategy and the number of cards they choose to discard. 5-card draw no longer remains the most popular poker variation. Texas hold'em is now by far the most popular and most played variation of the game[2]. It is also the variation used to determine the annual World Series of Poker Champion.

1.3.3 Tournament Play

The World Series Champion is determined via a no limit Texas hold'em tournament structure. In tournament play, all players begin with the exact same amount

[2] http://en.wikipedia.org/wiki/Texas_hold_'em

of chips. Forced bets, known as *the blinds*, are imposed on two players during each hand. The *big blind* acts as the minimum bet amount a player must make to stay in a hand. Initial betting usually occurs as multiples of the b*ig blind.* During each round of play one player at the table is assigned the status of *dealer*. This determines the betting order. The player to the immediate left of the dealer is known as the *small blind*. This means that player must make a forced bet of half of the current *big blind*. The player to the left of the *small blind* is known as the *big blind*. This player must make a forced bet of one full *big blind*. These forced bets occur before any cards have been dealt and ensure that there is something in the pot to play for. As the tournament proceeds the *small blind/big blind* values increase. The time taken for the *blinds* to increase normally varies between about twenty minutes to two hours. For example, the *blinds* may increase as follows: 10/20, 15/30, 20/40, 30/60, 40/80, 50/100. This means that the *big blind* is initially $20 worth of chips and the small blind is $10 worth of chips. After a certain time period the *blinds* are then raised to $30 for the *big blind* and $15 for the *small blind*. The raising of the *blinds* continues until all players are knocked out of the tournament, except for one player who holds all of the chips. This player is the winner of the tournament.

1.3.4 Ring Games

Poker can also be played as a *ring game* (or a *cash game*). *Ring games* differ from tournaments in a few areas. Firstly, in *ring games* players gamble with real money in the form of chips. Players can play with any amount of money up to a specified limit. Another difference is that the *blinds* do not increase. These are fixed, normally at a value much lower than a players *chip stack* (the amount the player has to play with). *Ring games* can be played as *limit* or *no limit* games. A ring game is normally composed of 8 – 10 players. Players can leave the game at any time with their winnings (or losses) and they can continue to play as long as they can pay the *blinds*. All results obtained for this thesis are for *limit*, *ring games.*

1.3.5 Texas Hold’em

In the game of Texas hold’em players are dealt two *hole cards* and five community cards are used in total. This strikes the right balance in terms of information availability (Harrington and Robertie 2004) and offers opportunities for better strategic play than other poker variations allow for. Texas hold’em also offers a better skill-to-luck ratio than is offered by other forms of poker. An expert hold ‘em player has more of an advantage because the best hand holds up more often than in any other poker variation (Sklansky and Malmuth 1994). Play in hold ‘em proceeds in the following four stages: *preflop*, *flop*, *turn* and the *river*. These are described below:

Preflop: The game of Texas hold’em begins with each player being dealt two *hole cards* which only they can see. The player to the immediate left of the *big blind* is the first player to act. Once a player has made their decision play continues in a clockwise fashion round the table. If a player, who has not made a forced bet, wishes to play then they must pay at least the big blind value into the pot. The *small blind* and *big blind* only have to match the current bet value to stay in the game. As long as there are at least two players left then play continues to the next stage. During any stage of the game if all players, except one, fold their hands then the player who did not fold his/her hand wins the pot and the hand is over.

Flop: Once the *preflop* betting has completed three community cards are dealt. Players use their hole cards along with the community cards to make their best hand. Another round of betting occurs. During this round and all future rounds the *small blind* player is the first to act (if the small blind player is no longer in the hand then the first active player to the left of the *small blind* becomes the first to act). The player classified as *dealer* is always the last to act (once again, if the *dealer* is no longer in the hand the first active player to the right of the *dealer* becomes the last player to act). As long as there are at least two players left then play continues to the next stage.

Turn: The *turn* involves the drawing of one more *community card*. Once again players use any combination of their *hole cards* and the community cards to make their best hand. Another round of betting occurs and as long as there are at least two players left then play continues to the next stage.

River: During the *river* the final community card is dealt proceeded by a final round of betting. If at least two players are still active in the hand a *showdown* occurs in which both players reveal their hole cards and the player with the highest ranking hand wins the entire pot (if both players hold hands of the same value then the pot is split between both players).

1.4 Case-Based Reasoning

1.4.1 The CBR Cycle

Case-based reasoning is an AI methodology which stores past problems and solutions and uses these to handle novel situations (Riesbeck and Schank 1989). Past cases are stored in a case-base and when a new problem is encountered the most similar cases are retrieved and evaluated. Case-based reasoning is a cyclical process and is typically composed of the six-*REs* (Watson 2003).

1. REtrieve the most similar case(s).
2. REuse the case(s) to attempt to solve the problem.
3. REvise the proposed solution if necessary.
4. REview the proposed solution to determine whether it is worth retaining.
5. REtain the new solution (if need be) as part of a new case.
6. REfine the case-base over time.

This process is illustrated pictorially in Figure 1.1.

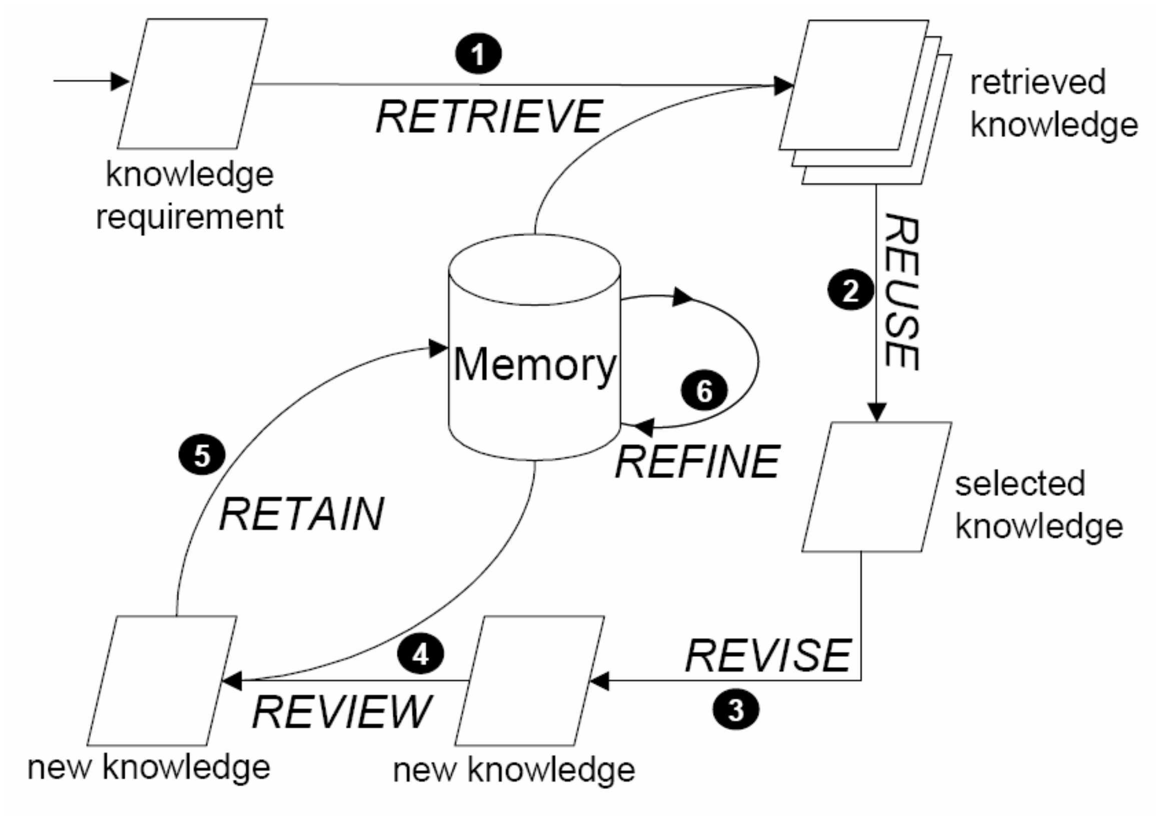


Figure 1.1: The CBR-cycle. Image sourced from (Watson, 2003).

1.4.2 Illustrative Example

Case-based reasoning is probably best explained using an example. The scene for this example is an online poker server where players can play poker for real money against opponents from all around the world. Imagine Jimmy is an average online poker player who specialises in heads-up Texas hold'em tournaments. Heads-up tournaments involve two players who begin with even amounts of chips and play until one player holds all the chips and the other player holds none. Before the tournament begins both players pay a fee to enter the tournament. Jimmy normally pays $10 to enter into a tournament. The player with all the chips at the end of the match is the winner and they are rewarded $20 (their original $10 to enter the tournament plus their opponent's $10).

On the online poker server records are kept for each player specifying how long they have been playing for and how many heads-up games they have won. Assume Jimmy wishes to use this information to establish whether playing a particular opponent will be profitable or not. To do so Jimmy decides to record this information before playing a match along with the outcome of the match. Figure 1.2 shows Jimmy's records after a few months of play. The black dots are games that Jimmy eventually won and the yellow dots are games Jimmy lost.

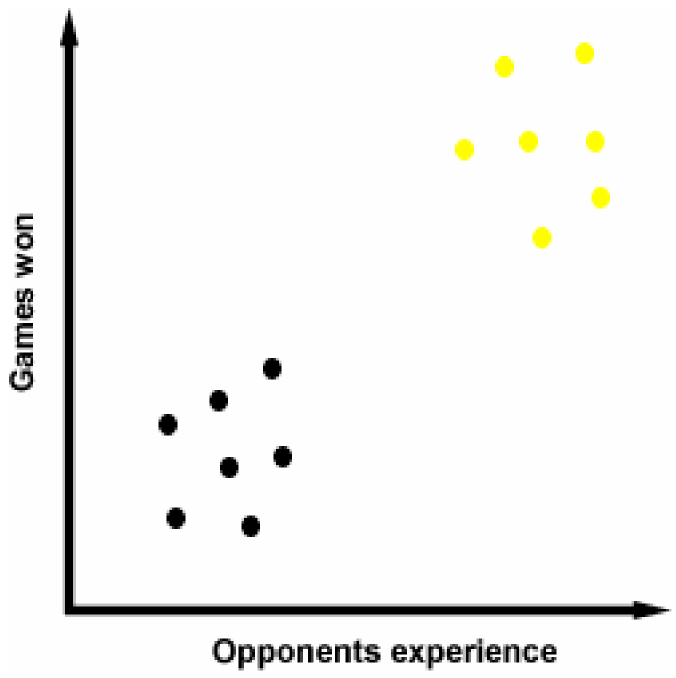


Figure 1.2: Jimmy's opponent data. The black dots indicate games Jimmy won, whereas the yellow dots are games that he lost.

Now, before Jimmy decides to play a particular opponent in a game of heads-up poker he first finds out how long that opponent has been playing for and how many games the opponent has won and he plots this information on his graph. In Figure 1.3 this opponent is represented as the blue square. Once Jimmy has plotted the information on his graph he needs to decide whether or not to challenge this opponent. To do so Jimmy compares how close this opponent's attributes are to previous opponents Jimmy has played against. If these values are similar to opponents that Jimmy played in the past and won against then Jimmy decides to challenge the opponent, however if they are more similar to opponents that Jimmy lost against in the past then Jimmy decides to keep his $10 instead. In this example Jimmy's opponent is closer in proximity to players that Jimmy has won against in the past, so Jimmy decides to challenge the opponent.

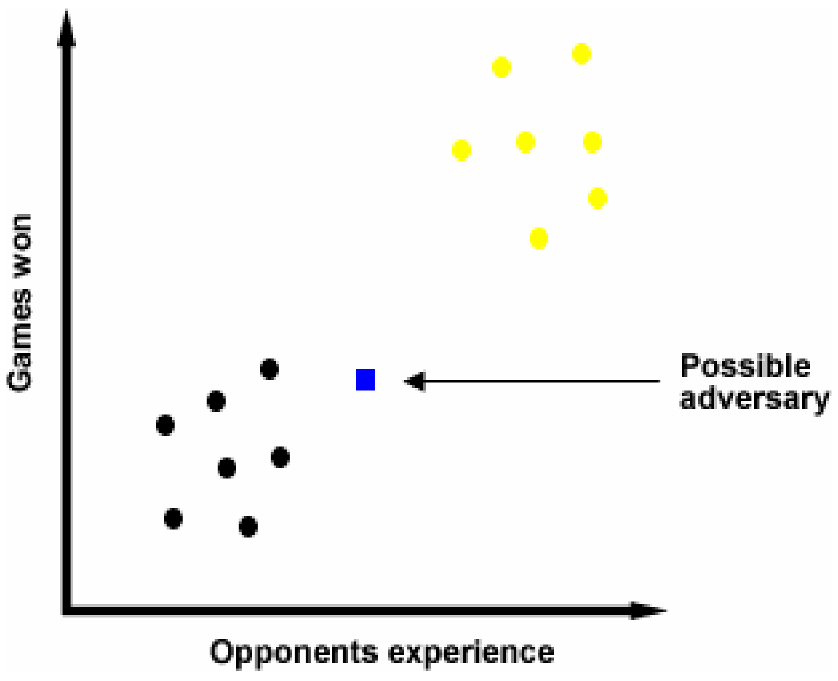


Figure 1.3: Jimmy's opponent data. The blue square represents an opponent Jimmy has not played before.

The above example, while simplistic, illustrates the idea of using past experiences and their solutions to make decisions about novel situations. Each previous experience is stored as a case in the case-base and each case consists of a number of attributes with associated values and the final solution. In the above example, the attributes that made up each case was:

1) *the opponent's experience (days, weeks, months… playing on the server) and,*
2) *the amount of games they had won.*

These attributes are known as *indexed* attributes, i.e. their values are used to find similar cases in the case-base. However, a case can also be composed of other *non-indexed* attributes which simply record useful information about the case. In this example a *non-indexed* attribute may be the opponent's name. The solution for each case was whether Jimmy had won or lost the match. Case-based reasoning assumes that similar problems have similar solutions (Leake 1996). This is represented pictorially in Figure 1.4.

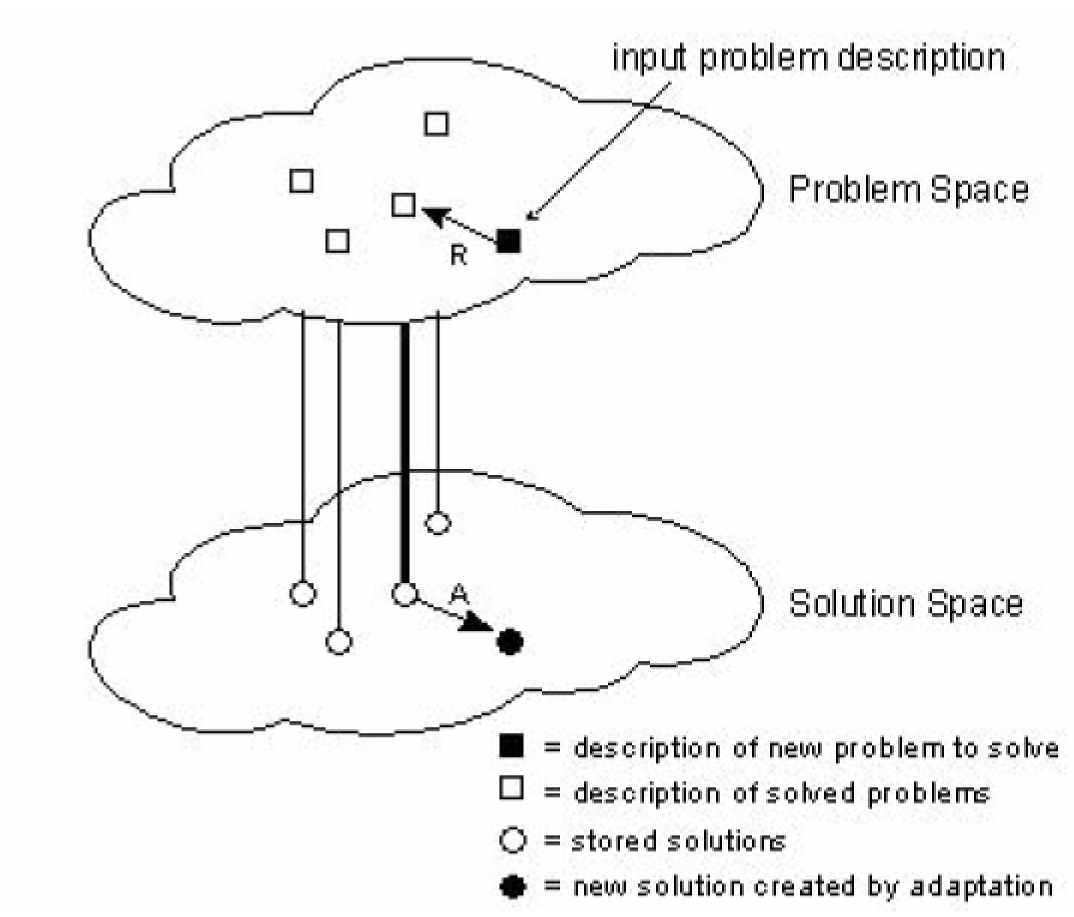

input problem description
Problem Space
R
Solution Space
A
■ = description of new problem to solve
□ = description of solved problems
○ = stored solutions
● = new solution created by adaptation

Figure 1.4: Similar problems have similar solutions. Image sourced from (Leake 1996).

1.4.3 Case Retrieval

There are two common methods for the retrieval of similar cases from the case-base. One involves using the k-nearest neighbour algorithm and the other inductive retrieval.

1.4.3.1 K-Nearest Neighbour

The k-nearest neighbour algorithm involves positioning a target case (*T*) in an n-dimensional search space of source cases (*S*). Each dimension in the space records the value for one of the indexed attributes which makes up the case. Similarity between the target and source cases individual attributes is calculated using a distance metric. For example, the absolute difference between two numeric attributes $|T_i - S_i|$ where *i* refers to the specific attribute in the case.

The target case must be compared to every case in the case-base and similarity computed for each attribute in the case. Global similarity between two cases is computed as follows:

$$\text{Similarity}(T, S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i \tag{1.1}$$

Here, *f* refers to a similarity function, *i* refers to each individual attribute in the case, *n* refers to the number of cases in the case-base and w_i refers to a weighting for attribute *i* to indicate its importance in the similarity measure.

The above equation simply states that global similarity is the sum of the local similarities between attributes. Similarity values are often normalised to fall in the range of 0 to 1, where 0 refers to least similarity and 1 refers to an exact match.

Of the *n* cases in the case-base, *k* cases with the highest similarity are retrieved.

1.4.3.2 Inductive Retrieval

Inductive retrieval is another method which has been used for case retrieval. Instead of summing similarity between separate attributes inductive retrieval works by examining cases in the case-base and building a decision tree. The ID3 induction algorithm is generally used to build the tree. The attributes that make up the case are examined and the *information gain* heuristic is used to order the attributes position in the decision tree. Attributes which do a good job of partitioning cases in the case-base are favored and are selected earlier, resulting in placing their nodes higher in the tree. Inductive retrieval has not been used in this thesis and is merely mentioned for completeness. For a more in-depth discussion of inductive retrieval and ID3 see (Watson 1997) or (Mitchell 1997).

1.5 Research Goals / Thesis Contributions

The work completed in this thesis has focused solely on the area of limit Texas Hold'em. Particular interest has been given for making betting decisions at a full poker table, i.e. one consisting of approximately 8 – 10 players. Successful strategies differ markedly at a full table compared to games with fewer players e.g. *heads-up* - where there are only two players in total.

A major goal of the research was to investigate the application of case-based reasoning tools and techniques to make betting decisions for the game of Texas hold'em and the quality of performance that was possible using this approach. This required the design and development of appropriate case-representations for encoding poker knowledge, investigating appropriate case comparison methods and extensive performance testing. By investigating the above problems it was intended to add to and improve upon the modest CBR related approaches to the game of poker that were found in the literature.

A case-based poker player, nicknamed Casper (CASe-based Poker playER), was successfully developed and tested. Casper was shown to be able to record games from strong players and then reuse these to obtain a similar performance. This bypassed the need for any initial, intensive knowledge engineering effort required of other poker-bots.

The rest of this thesis proceeds as follows:

- Chapter 2 discusses past research related to AI and games. Historic and recent approaches to the game of poker are extensively discussed and examined.
- Chapter 3 details the design and development decisions made during the construction of the Casper system.
- Chapter 4 describes various attempts at improving the performance of the system through the use of evolutionary algorithms.
- Chapter 5 summarises all results obtained for the Casper system.
- Chapter 6 discuses conclusions and possible future work.

Chapter 2

Related Work

Games provide a well suited domain for AI research. This is due to the fact that a game is usually composed of several well defined rules which players must adhere to. For a large majority of games the rules imposed are quite simple, yet the game play itself involves a large number of very complex strategies. This is especially true of games such as chess and checkers which offer opportunities to make very sophisticated and intricate plays. This statement is also true of the game of Texas hold'em and is nicely summed up by a popular quote coined by Mike Sexton which states "Poker takes a minute to learn and a lifetime to master"[3]. Another reason why games offer a beneficial environment for AI research is the fact that goals and objectives of the game are clearly defined. This is advantageous to research as a performance metric is implicitly embedded in the game. Success can easily be measured by factors such as the amount of games won, the ability to beat certain opponents or, as in the game of poker, the amount of money won.

2.1 Games and AI

2.1.1 Chess

Up until recently AI research has mainly focused on games such as chess and checkers. Successes like *Deep Thought*, *Deep Blue* and *Chinook* are usually the first to come to mind when contemplating AI and games. The chess automaton, Deep Thought, evolved through work started by Hsu in 1985. It was the first machine to achieve Grandmaster level performance over 25 consecutive rated games in 1988. For this achievement it won the second Fredkin Intermediate Prize (Hsu, et al. 1995). The next

[3] http://www.pokerlistings.com/poker-beginner-guide

year in October 1989 saw the first exhibition match between Deep Thought and Gary Kasparov, the then World Chess Champion. Kasparov won the match-up. Deep Thought's successor, Deep Thought 2, was completed in 1991 and had reached close to Super Grandmaster strength by 1995 (Hsu, et al. 1995). Deep Thought 2 was effectively a prototype for IBM's Deep Blue computer chess system. Deep Blue 1 played 6 games against Gary Kasparov in February of 1996. Once again Kasparov was the victor. The final score being 4-2. After this loss to Kasparov development of Deep Blue 2 commenced. Various improvements were made to Deep Blue 1. The evaluation function, i.e. the weighted sum of features which indicates the strength of a particular board, was significantly improved. Deep Blue 2 now used an evaluation function which consisted of 8000 individual features. The search speed was improved to on average 250 million positions per second (Campbell, et al. 2002) and the selective search aspect of the system, which meant that resources were dedicated to search interesting lines of play and dead ends in play were quickly abandoned, also underwent various modifications and improvements. The result of these efforts saw Deep Blue 2 defeat Garry Kasparov in 1997 by a score of 3.5 – 2.5. For the victory Deep Blue 2 was awarded the Fredkin prize.

A case-based reasoning approach to chess was conducted by (Sinclair 1998). Sinclair used a collection of 16,728 chess games played by grandmasters with varying styles of play. The database of chess games was then analysed using a multivariate technique known as *principal component analysis* to build a case-base for each of the board positions and the corresponding move made was recorded. Future board positions encountered are matched against the case-base and the most similar cases retrieved. Sinclair reported that when the similarity measure is high the quality of the solutions returned is very good, but recall is low, whereas if the degree of similarity was reduced the quality of returned solutions is lowered, but recall improves (Sinclair 1998).

2.1.2 Checkers

The main contributor to the success of chess programs such as Deep Blue was the use of brute-force search techniques along with improvements to hardware processing speeds. A lot of the same techniques that were used for the game of chess have also been successful when applied to checkers. Checkers is less strategically complex than chess. In checkers there are only two types of pieces to play with, while in chess there are 6. There is also a reduction in the amount of legal squares on the board. In checkers there are 32 legal squares as opposed to twice that amount in chess. These simplifications can actually be more beneficial when it comes to AI research (Schaeffer, et al. 1992). Many of the same research questions are still being addressed as in chess, but without the undue complexity. In August 1990, a checkers program called *Chinook* competed in the U.S. National Open. Work on *Chinook* had begun in June 1989 at the University of Alberta (Schaeffer, et al. 1991). *Chinook* ended up coming second in the National Open, after the then World Champion, Dr. Marion Tinsley. By coming second to Dr. Tinsley *Chinook* had earned the right to challenge him in a 40 game match for the World Championship title. Unfortunately, the American Checker Federation (A.C.F) and English Draughts Association (E.D.A) refused to sanction the match on the grounds that they did not want a computer vying for a human title. Instead a new "Man-Machine" World Championship was created. *Chinook* commenced playing Tinsley in August 1992 - the final outcome saw Tinsley as the victor winning four of the matches, losing two and drawing 33 matches (Schaeffer, et al. 1996). Two years later *Chinook* was set to challenge Dr. Tinsley again. *Chinook* had undergone significant improvements in the two years, including additions to the endgame database and improvements to the evaluation function. The first 6 matches all resulted in draws. The seventh match was never to be played. Unfortunately, due to health reasons, Dr. Tinsley resigned the match and *Chinook* became the Man-Machine World Champion by forfeit.

As with Deep Blue in chess, *Chinook*'s success was related to its deep search capabilities as well as a strong evaluation function. *Chinook* also employed use of an end-game database which was able to supply perfect information for all board positions for 6 pieces or less remaining. At the time of writing this thesis *Chinook* had extended the endgame databases to provide perfect information for all checker positions involving 8 or fewer pieces on the board.

A different approach to the game of checkers which is noteworthy is the work of Fogel (2000). Fogel's approach consisted of evolving neural networks to play the game of checkers. Initially a set of random neural networks competed against each other for survival. The only human knowledge that was provided to the networks was the piece differential, i.e. the difference between the number of one player's pieces versus the other player's pieces. After a certain number of games were completed the networks were given a number associated with how well they were playing. The best networks were kept and offspring created from them. After 250 such generations the best evolved neural network was used as an evaluation function and combined with standard minimax search to play games against actual human opponents. Although Fogel's program is no match for *Chinook* his work is impressive due to the lack of human expertise involved in creating an expert checkers player.

CBR has also been applied to the game of checkers (Powell, et al. 2004). The result was a system called CHEBR (CHEckers case-Based Reasoner). In traditional case-based reasoning systems case-bases are often constructed manually, for example via interaction with a domain expert who can supply prototypical scenarios. CHEBR differs from conventional case-based reasoning in that it actually acquires knowledge in real-time by playing checkers, i.e. CHEBR begins with an empty case-base and adds cases as it plays the game. This is known as *automatic case elicitation*. Initially, CHEBR has no knowledge of the game whatsoever; this includes the difference between legal and illegal moves. It begins by randomly selecting moves it has not tried in the past until a legal move is found. Once a legal move is found CHEBR records how successful the move was. As the system encounters similar situations it retrieves the most similar successful cases from its case-base and takes the appropriate action, if the situation encountered does not match previously stored cases or only unsuccessful cases are present then the system generates a new action either randomly or by combining actions from other successful cases. Results indicate that extra experience (gained through playing many games of checkers) can compensate for a lack of predefined knowledge.

2.1.3 Other Games

Apart from chess and checkers there have also been attempts to create programs to play games such as Backgammon, Go and Bridge. Gerald Tesauro's *TD-Gammon* is a neural network that trains itself to be an evaluation function for the game of backgammon by playing itself and learning from the outcome (Tesauro 1995; Tesauro 2002). *GIB*, developed by Matthew L. Ginsberg, has achieved success in the game of Bridge (Ginsberg 1999; Ginsberg 2001). And in Othello, Michael Buro's *Logistello* challenged and defeated the World-Champion Takeshi Murakami (Buro 1997).

2.2 Poker and AI

Games such as chess, checkers and backgammon are classified *as two-person, zero-sum* games with *perfect information*. This means that there is one winner and one loser (zero-sum) and the entire state of the game is accessible by both players at any point in the game (perfect information), i.e. both players can look down upon the board and see all the information they need to make their playing decisions. These types of games have achieved their success through the use of fast hardware processing speeds, selective search, effective evaluation functions and better opening books and endgame databases. While these achievements are impressive, their scope is rather limited. They offer little insight into other areas where AI techniques may be useful.

Games such as poker on the other hand are classified as stochastic, imperfect information games. The game involves elements of chance, the actual cards which are dealt, and hidden information in the form of other player's *hole cards* (cards which only they can see). This ensures that players now need to make decisions with uncertain information present. This is still an open research question in the AI community and research efforts are likely to be beneficial outside the realm of poker itself. For AI to be useful for most real world problems, challenges that imperfect information and a stochastic environment offers need to be addressed.

There have been a small number of early machine learning attempts made in the domain of five-card draw poker. More recent approaches to poker research can be classified into three broad categories: the investigation of game-theoretic optimal solutions, heuristic rule-based systems and simulation/enumeration-based systems.

2.2.1 Early Poker Research

Nicholas Findler is credited with the earliest attempts to apply machine learning principles to the game of 5-card draw poker (Billings 1995). In the 1970s Findler created programs to play 5-card draw poker which used various playing strategies (Findler 1977). Findler's machine players were classified as either static players, which did not take their opponents' behaviour into account, or learning players, that adapted their style of play according to the game conditions. Findler's analysis of these machine players quality of play and rate of improvement has been questioned (Billings 1995) due to its subjective nature and lack of scientific rigor. Although the machine players developed by Findler produced weak to mediocre poker players this was not the main concern of the research. Rather, the game of poker was used as an environment in which to study theories of decision making, human behaviour and cognition in a risk-taking environment.

Another early researcher who used 5-card draw as a research test-bed was Waterman (1970). Waterman investigated the machine learning of heuristics (represented as production rules) to make a betting decision given information such as the amount of money currently in the pot, the belief measure that an opponent could be susceptible to a bluff, the number of cards the opponent replaced (used in deducing which possible hand the opponent may hold) and a measure of how conservative the opponent is believed to be. Waterman reported that the use of a small set of production rules "*produces play at roughly the same level of skill as an experienced human player*" (Waterman 1970).

2.2.2 Heuristic-Based Systems

A heuristic/rule-based system approach to computer poker uses various pieces of information to inform a betting strategy. For example, information such as a player's *hole cards*, the current community cards, the player's current position at the table and the previous betting history of the hand may form part of some heuristic which dictates whether the player should fold, check/call or bet/raise when it is their turn to act.

As mentioned previously, an earlier attempt at an heuristic based system was made by Waterman (1970) who attempted the machine learning of heuristics using the poker variation of five-card draw. Sklansky and Malmuth have detailed various heuristics for different stages of play (preflop, flop, turn and river) in the game of Texas hold'em (Sklansky 1994; Sklansky and Malmuth 1994). In particular detailed guidelines for *preflop* play, given a player's relative betting position at the table, are provided. As well as the grouping of various *hole cards* into eight separate equivalence classes ordered on the strength of the hand (Sklansky and Malmuth 1994). The purpose of these rules, however, has been to guide human players who are looking to improve their game rather than the construction of a computerised expert system. Nevertheless, a poker playing program that used to play on Internet Relay Chat (IRC) called *r00lbot* developed by hobbyist Greg Wohletz used Sklansky and Malmuth's recommendations to determine its preflop play (Papp 1998).

The University of Alberta Poker Research Group's formula based version of *Poki* uses *ad hoc* rules and formulas defined by a domain expert to generate a fold, check/call and bet/raise probability distribution which specifies a betting decision (Billings, et al. 2002). The distribution is represented as a *probability triple* whose components sum to 1.0. For example, given the probability triple *(0.0, 0.2, 0.8)* a player should fold 0% of the time, check/call 20% of the time and bet/raise the remaining 80% of the time.

Information such as a player's *effective hand strength* (the probability that the player currently has the best hand or can improve to make the best hand), relative betting position and the model of an opponent's play all contribute to the generation of the probability triple. The exact details of the expert system have not been specified (Billings, et al. 2002).

Poki's performance was tested by playing both real and machine opponents. The formula-based version of *Poki* played in both low limit and higher limit games on the IRC poker server. *Poki* was a consistent winner in the lower limit games as well as in the higher limit games where it faced tougher opposition (Billings, et al. 2002).

While expert defined rule-based systems can produce poker programs of reasonable quality (Billings, et al. 2002), various limitations are also present. As with any knowledge-based system a domain expert is required to provide the rules for the system. In a strategically complex game such as Texas hold'em it becomes impractical to write rules for all the scenarios which can occur. Moreover, given the dynamic, nondeterministic structure of the game any rigid rule-based system is unable to exploit weak opposition and is likely to be exploited by any opposition with a reasonable degree of strength. Finally, any additions to a rule-based system of moderate size become difficult to implement and test (Billings, et al. 1999).

2.2.3 Simulation-Based Approaches

A simulation-based betting strategy is analogous to selective search in perfect information games such as chess and checkers. Rather than expanding all nodes in the game-tree with equal probability, biases towards expanding certain nodes are introduced in the hope of obtaining better information in less time by initially examining important nodes.

In poker a simulation-based betting strategy consists of playing out many scenarios from a certain point in the hand and obtaining the *expected value* (EV) of different decisions. The simulations occur when it is time for the program to make a betting decision. The amount of money won or lost for each betting decision is determined and the average of this becomes the EV for that decision. The EV of a fold decision is always 0 because no money can be won if the hand is folded. The EV of a check/call or a bet/raise decision is obtained by playing out the hand to the end using a certain number of trials. Each trial begins by assigning each opponent possible hole cards and then simulating how the opponent might play the hand.

During the simulation it becomes necessary to make future betting decisions as well as predict an opponent's betting decisions along the way. The University of Alberta Poker Research Group's implementation of *Loki* (Billings, et al. 1999) and simulation-based version of *Poki* (Billings, et al. 2002) use probability triples as the main data structure to handle these decisions. A probability triple *(f, c, r)* is generated which specifies how often the program and the opponent would fold, call or raise at a particular point in the game. The use of probability triples allows elements such as game-specific information, expert defined rules and knowledge of human behaviour to effectively be treated as a 'black box'. These 'messy' elements are constrained to the construction of the probability triple (Billings, et al. 1999).

Simulation based approaches have been combined with opponent modeling methods in both *Loki* and *Poki* (Billings, et al. 1999; Billings, et al. 2002). At the beginning of a simulation trial opponents need to be assigned two hole cards to inform their possible future actions. Rather than assigning hole cards to different opponents with uniform probability a weight table is maintained for each opponent which lists all possible two card holdings and the likelihood that those cards would have been played to the current stage in the game. After observing a betting action from an opponent the weights are updated using a probability triple generated by the current opponent model. The use of the weight table allows the assignment of hole cards to an opponent to be biased towards certain cards, rather than assuming equal probability. For example, if an opponent has consistently been raising in a hand it is more likely that the opponent has a good hand rather than a random one.

Results for the simulation based versions of *Poki* and *Loki* were somewhat mixed (Billings, et al. 2002). While they performed better than their formula-based counterparts in the lower level games on the IRC poker server, they were only able to break even on the more advanced level games whereas the formula-based versions would routinely win.

2.2.4 Game Theory and Poker

The game of Texas hold'em consists of multiple players, with conflicting goals, making decisions given the information they have access to. *Game theory* provides the tools to model and analyse situations such as this and allows "rational" strategies to be developed for different players (Rasmusen, 2001).

There have been various manual applications of game theory to simplified versions of poker (Kuhn 1950; Nash and Shapley 1950). While game-theoretic approaches are manageable for games with perfect information, the introduction of imperfect information greatly increases the computational costs and the complexity of the problem. A consequence of this was that game-theoretic analysis could only be performed manually on over-simplified versions of poker and as such any results obtained from studying these simple versions of the game do not transfer well to full-scale poker (Koller and Pfeffer 1997).

Koller and Pfeffer have investigated the application of game theory to large imperfect information games, such as poker, in their *Gala system* (Koller and Pfeffer 1997). The Gala system is made up of two distinct sections. The first section involves the ability to describe a game using a special language to specify the rules that make up the game. Once the game has been specified a tree is constructed which is very similar to the "standard" AI game tree in which states are represented as nodes in the tree and an agents possible decisions are represented as arcs. This tree is known to game theorists as the *extensive form* representation of the game. The extensive form representation extends the standard game tree by adding information about a player's information state at particular nodes (Koller and Pfeffer 1997).

The second component of the Gala system performs analysis on the game tree and finds *randomised optimal strategies* for the game. Randomised strategies involve some proportion of random decisions being made, for example how often to bluff, with random cards, in a game of poker. Randomised strategies are employed by the Gala system for games with imperfect information as any deterministic strategy is liable to be exposed and exploited. An optimal strategy is one in which a player cannot do any better by changing his or her strategy provided that their opponents are also using an optimal strategy, moreover a player can reveal their optimal strategy yet not be vulnerable to exploitation by their opponent (Koller and Pfeffer 1997). These concepts have been illustrated using the game of *rock-paper-scissors* (Billings, et al. 2002; Davidson 2002; Billings, et al. 2003).

The game of rock-paper-scissors (or RoShamBo) is played by two people. Each person simultaneously chooses rock, paper or scissors. Players make their choices known to each other at the same time, usually via hand signals. The winner of the game is determined as follows: rock beats scissors, scissors beats paper and paper beats rock. So as an example if player 1 chooses rock and player 2 chooses scissors then player 1 is the winner of that round. If both players choose the same item then the outcome is a draw. The skill in the game of rock-paper-scissors comes from a player knowing their opponent so well that they are able to predict which option their opponent will choose i.e. they know their opponents' strategy. Now, in the game of rock-paper-scissors there exists an *optimal strategy* which says to randomly pick rock, paper or scissors with equal probability. If a weak player deviates from this strategy a strong player is liable to *outplay* the weak player by finding out and exploiting their *sub-optimal strategy*. However, if the weak player just uses the optimal strategy and chooses rock, paper or scissors, each with a probability of $\frac{1}{3}$, then the strong player no longer can exploit the weak player by predicting what they will choose. By choosing the optimal strategy the weak player ensures a breakeven result rather than a losing result. In fact, the weak player can even tell the strong player their strategy and still not do any worse.

The Gala system was applied to a simplified version of two-player poker. An eight card deck was used in which the lowest card was a 6 and the highest card was a King. Each player is dealt one card and has to make a forced bet of one dollar. Each player also has one extra dollar with which to bet. Players then have the option to *check* and not wager their remaining dollar or to *bet* their remaining dollar or to *fold* their hand. If either player folds the other player automatically wins the forced bets. The game consists of up to three rounds. In the first round player 1 decides whether they wish to check or bet. In round two player 2 now makes their decision. If player 1 checked in the first round then player 2 now has the option of either checking or betting.

If player 1 bet in the first round player 2 can now either fold or call. The third round only takes place if player 1 checked in the first round and player 2 bet in the second, now player 1 can decide to fold or to also bet. Once betting is complete the cards are revealed and the player with the highest card wins the pot. The rules for this simplified game of poker were input into the Gala system and the game tree was generated and an optimal strategy derived. The results of the analysis show that bluffing is game-theoretically optimal in poker. While this simplified 8 card poker variation was able to be solved using the Gala system the authors note that they are nowhere near being able to solve full-scale poker due to the size of the game trees generated (Koller and Pfeffer 1997).

The University of Alberta Computer Poker Research Group have attempted to apply game-theoretic analysis to full-scale 2-player poker (Billings, et al. 2003). The poker variation that they investigated was limit Texas hold'em. The group attempts to overcome the computational complexities associated with full-scale poker by using various abstraction techniques to reduce the search space while still retaining the key properties and structure of Texas hold'em. By using abstractions such as limiting the number of bets a player is allowed per round, eliminating some betting rounds (for instance the *river*) and the grouping of hands into equivalence classes the group are able to determine "pseudo-optimal" strategies. The result is a class of programs known as *PsOpti* (*PsOpti1*, *PsOpti2 …*) which are "*able to defeat strong human players and be competitive against world-class opponents*" (Billings, et al. 2003).

The outcome of applying game-theoretic solutions to games produces *optimal* strategies, rather than *maximal* strategies. An optimal strategy assumes an opponent will play optimally. It gives no consideration to exploiting any weaknesses of an opponent and is only concerned with not losing rather than winning. A maximal strategy, on the other hand, will try to win by exploiting sub-optimal play to maximise gains. This implies that while game-theoretic approaches may not lose against very strong opponents, they also may not win against weak opponents. Consider once again the example of rock-paper-scissors. Imagine a very weak player whose strategy is to only ever play *rock*. A maximal strategy will eventually detect this weakness and play *paper* ensuring a win every time. Whereas, an optimal strategy will not consider the opponents play and will continue to play *rock*, *paper* or *scissors* with equal probability, ensuring a breakeven result. To overcome these constraints opponent modeling needs to be addressed (Billings, et al. 2003).

2.2.5 Case-Based Reasoning and Poker

Relatively few attempts to apply the principles and techniques of CBR to the game of poker have been undertaken. A case-based learner for Texas Hold'em, called Casey, was constructed by (Sandven and Tessem 2006). Casey recorded information from poker hands and used these to make future betting decisions. Each case was made up of several attributes including hand strength, relative position, number of opponents and number of bets to call. The solution offered by a case is a *strategy* which consists of an action to be taken and a follow up response if applicable. For example a *strategy* may consist of one action such as fold or multiple actions such as checking to an opponent and raising if the opponent bets (this is known as a check-raise in poker and is considered to be a sign of strong hand strength).

Initially, Casey began with an empty case-base and therefore had to begin by employing random strategies (i.e. making random decisions) to build up the case-base. The decisions made by Casey were then evaluated by the outcome of the hand and this was recorded in the case-base. As play proceeded more similar scenarios where encountered and the need for random strategies decreased.

Sandven and Tessem tested Casey by playing in poker games of 4, 6 and 8 players through the University of Alberta Poker Research Group's commercial product known as Poker Academy[4]. Casey's opponents were instances of another poker-bot known as *RuleBot*. *RuleBot*, as its name suggests, is a rule-based system which was provided with the Poker Academy software. As Casey begins by never folding and playing randomly to generate a sufficient case-base the initial results are obviously quite poor. As more cases are added to the case-base slight improvement is shown in the results. Sandven and Tessem report that Casey plays on a par with *RuleBot* in 4-handed play.

The Casper system, which is the focus of this thesis, improves upon Casey's results (Rubin and Watson 2007)[5]. No other systems that used case-based reasoning to make betting decisions at the poker table could be found in the literature.

An attempt to apply CBR specifically to the area of opponent modeling was made by (Salim and Rohwer 2005). Opponent modeling attempts to predict the hand strength of an opponent given how that opponent has been observed playing in the past.

[4] http://www.poker-academy.com
[5] See Appendix A for a full re-print of this paper

In poker it is imperative to know how ones opponents play, for instance are they aggressive or conservative players? Does one opponent bluff too much? This information is used to exploit weak opposition and to reduce the chances of being exploited by stronger opposition. This insures that accurate opponent modeling is critical to the success of any computerised poker player (Billings, et al. 1998; Billings, et al. 1999; Davidson 2002). CBR seems to be an obvious candidate to handle this aspect of the game, i.e. by recording cases of how a particular opponent has played in the past and using the most similar cases in the case-base when faced with a new decision. Salim and Rohwer attempted to use CBR to predict an opponent’s future play given how they played their last 100 hands. Each opponent had their own individual case-base which recorded their average hand strength and variance. The results show that the predictions made using CBR did not achieve great success and were inferior to results obtained by simply recording long-term average statistics for opponents (Salim and Rohwer 2005).

Chapter 3

CASPER: Developing a Case-Based Poker Player

For this thesis a poker player was developed that used the CBR methodology. The design and implementation of this CASe-based Poker playER, or CASPER, involved the identification and experimentation of various aspects such as case representation; feature usage, significance and similarity; and case retrieval. The design decisions employed are detailed in this chapter.

3.1 Case-base Construction

Casper uses the CBR methodology to make a betting decision. This means that when it is Casper's turn to act it evaluates the current state of the game and then consults its case-base (i.e. its knowledge of past poker experiences) to try and find similar scenarios which may have been encountered. These past experiences dictate how Casper should play the hand. Initially, Casper's case-base was constructed by analysing approximately 7000 poker hands played between two types of poker bots developed by the Computer Poker Research Group at the University of Alberta[6]. The two bots used were the well known *Pokibot* and the simulation based *Simbot*. Both Pokibot and Simbot were the result of an intensive knowledge engineering process. Both bots have proven to be profitable against human competition in the past (Davidson 2002) so it is believed that the data obtained is of greater quality then it might be from other sources, such as free money games on the internet composed of real players. Every decision made during each hand was recorded as one case in Casper's case-base. Casper then reuses these recorded instances to make decisions at the poker table and therefore bypasses the intensive knowledge engineering effort required of other poker-bots.

[6] http://www.cs.ualberta.ca/~games/poker/

For each stage of the game (*preflop*, *flop*, *turn* and *river*) a separate case-base is used. Table 3.1 records how many cases were collected for each of the separate case-bases. This became known as the first version of Casper, or Casper01. After observing preliminary results it was decided to improve Casper01 by using a larger case-base. Another 13,000 poker hands were played and recorded as cases in Casper02's case-base. Table 3.2 records the total number of cases recorded for each of Casper02's separate case-bases.

Stage	***Total cases***
Preflop	28,224
Flop	9,998
Turn	7,023
River	5,691

Table 3.1: Casebase totals for Casper version 1.

Stage	***Total cases***
Preflop	167,540
Flop	50,948
Turn	34,634
River	27,507

Table 3.2: Casebase totals for Casper version 2.

As can be seen above Casper02 uses substantially more cases then Casper01 to make a betting decision.

3.2 Case Representation

The features that make up each case differ slightly depending on the current stage. The features used for each stage are described below.

3.2.1 Preflop Cases

Feature:	*Type:*	*Range:*	*Explanation:*
Case number	int	1 - 150000	A unique number identifying the case
Game number	int	1 - 20000	The game number from which this case was derived
Player name	String	{a – z}	The name of the player who made the decision
Hole cards[7]	String	"27o" - "AAo"	One of 169 different groups of *hole* cards that a player could be dealt. Made up of each card's rank and whether they are the same suit 's' or offsuit 'o'.
Absolute position	int	1 - 10	What order the player acts. 1 represents the small blind, 2 the big blind and 10 is the button.
Number of players	int	1 - 10	Number of players that were dealt cards.
Relative position	double	0.0 - 1.0	What order the player acts relative to other players at the table. 0.0 means the small blind and 1.0 is the button.
Players in current hand	int	0 - 9	The number of players that are currently in the hand, i.e. players that have called or raised.
Players yet to act	int	0 - 9	The number of players that still need to make a future betting decision.
Small bets committed	double	0.0 - 5.0	A multiple of the small bets the player has committed to the pot.
Small bets to call	double	0.0 - 5.0	A multiple of the small bets the player has to commit to the pot to stay in the hand.
Pot Odds	double	0.0 - 0.5	(Bets to call) / (Bets to Call + Pot total), i.e. is a risk reward measure.
Hand ranking	int	1 - 169	A number indicating the rank of the players *hole* cards. 1 indicates AAo (the best preflop hand) whereas 169 indicates 27o (the worst preflop hand)
Action	char	{f, k, c, r}	A character representing the decision which was made. f = fold, k = check, c = call, r = raise.

Table 3.3: Representation of a *preflop* case, made up of 5 unindexed features, 8 indexed features and 1 outcome.

[7] All 169 *preflop* hand groupings are listed and ranked in Appendix B.

Each preflop case is made up of five unindexed features, eight indexed features and one outcome. Table 3.3 lists each of these features. The first five features in table 3.3 are unindexed, i.e. they are not used in the retrieval process, but merely provide important contextual information. The eight features that follow are indexed features and are believed to be predictive of a case's outcome.

The above indexed features were chosen by the author to represent a preflop case because they are believed to capture important information needed to make a preflop betting decision. Because all of the above indexed features are quantitative this becomes advantageous when computing case similarity and during case retrieval.

3.2.2 Postflop Cases

Once again the first 5 features in the *postflop* cases are unindexed and therefore they are not used in the retrieval process. Cases for the *flop* and the *turn* consist of 12 indexed features and cases for the *river* consist of 10 indexed features. Each case has one outcome. Table 3.4 lists all features and their range of values.

As with the *preflop* case, these postflop features were chosen by the author because they are believed to provide important information that must be assessed before making a betting decision *postflop*. Once again, all indexed features are quantitative and are easily comparable.

During casebase construction values were calculated and assigned to all case features for all *preflop*, *flop*, *turn* and *river* cases. This was achieved by recording instances of games played between both *Pokibot* and *Simbot* and then extracting and calculating the appropriate information from these instances.

Features:	*Type:*	*Range:*	*Explanation:*
Case number	int	1 - 50000	A unique number identifying the case.
Game number	int	1 - 20000	The game number from which this case was derived.
Player name	String	{a – z}	The name of the player who made the decision.
Hole cards	String	"27o" - "AAo"	One of 169 different groups of hole cards that a player could be dealt.
Board cards	String	"{2 - A}, {s,c,d,h}"	The set of community cards which have currently been dealt. Each card is described by its rank {2 – A} and suit {s,c,d,h} where s = spades, c = clubs, d = diamonds and h = hearts.
Number of players	int	1 – 10	Number of active players at the beginning of the round (*flop*, *turn* or *river*).
Relative position	double	0.0 - 1.0	What order the player acts relative to other players at the table. 0.0 means the player is first to act in the round, 1.0 means the player is last to act.
Previous round total bets	int	0 - 5	How many bets or raises occurred during the previous round of betting.
Players in current hand	int	0 - 9	The number of players that are currently in the hand, i.e. players that have checked, bet, called or raised.
Players yet to act	int	0 - 9	The number of players that still need to make a future betting decision.
Bets committed	double	0.0 - 5.0	A multiple of the current bet size the player has committed to the pot. Small bets are used during the *flop* and big bets are used during the *turn* and *river*.
Bets to call	double	0.0 - 5.0	A multiple of the current bet size the player has to commit to the pot to stay in the hand.
Small bets in pot	double	0.0 - 300.0	The total amount in the pot divided by the value of the small bet size.
Pot Odds	double	0.0 - 0.5	(Bets to call) / (Bets to Call + Pot total), i.e. is a risk reward measure.
Immediate hand strength (IHS)	double	0.0 - 1.0	A numerical measure of the strength of a player's postflop hand. 0.0 represents the worst possible hand whereas 1.0 represents an unbeatable hand ("the nuts").
Positive potential (PPOT)[8]	double	0.0 - ~0.40	A numerical measure which represents the chance that a player who does not currently hold the best hand will improve to the best hand after future cards are dealt.
Negative potential (NPOT)[8]	double	0.0 - ~0.30	A numerical measure which represents the chance that a player currently holding the best hand no longer holds the best hand after future community cards are dealt.
Action	char	{f, k, c, b, r}	A character representing the decision which was made. f = fold, k = check, c = call, b = bet, r = raise.

Table 3.4: Representation of the *postflop* cases.

[8] These features are not used for *river* cases because no future community cards are dealt.

3.3 Similarity Metrics

Each of the above indexed case features was assigned a local similarity metric which computes how similar its value is to another instance. Recall that when it is time to make a betting decision a target case is constructed and all features are assigned particular values. The target case is then compared to the cases stored in the case-base, known as the source cases. This comparison is performed by computing the local similarity for each of the indexed features. The local similarity metric tells us how similar the target case's feature value is to each source case's feature value.

In general two main similarity metrics were used to compute local similarity between individual features. These included the standard 1-dimensional Euclidean distance and an exponential decay function. The similarity metric used for all *preflop* and *postflop* case features are explained below.

3.3.1 Number of Players (preflop only)

Similarity for the number of players was measured by grouping different numbers of players into equivalence classes and assigning a similarity measure of 1.0 if they were assigned to the same group. Otherwise a similarity value of 0.0 was assigned for values in separate groups. The groups used are displayed in Table 3.5.

Group	*Number of players*
1	8, 9, 10
2	5, 6, 7
3	3, 4
4	2

Table 3.5: Equivalent groups

3.3.2 Relative position, Players in current hand, Players yet to act, Number of players (postflop only), Small bets in pot and Pot Odds

The similarity measure for the *preflop* and *postflop* features including:

1) relative position,
2) players in current hand,
3) players yet to act, and
4) pot odds

as well as the *postflop* features:

5) number of players, and
6) small bets in pot,

were calculated by computing the absolute difference between the target case's feature value and the source case's feature value and then dividing this difference by the maximum possible difference (given by the range for each feature in Table 3.3 and 3.4). This gives a value between 0.0 and 1.0 where 0.0 indicates an exact match and 1.0 indicates completely different values. The inverse of this value was then taken by subtracting it from 1.0. This ensures exact matches now have a similarity of 1.0 and entirely different values have a similarity of 0.0. The following is summarised mathematically below:

$$s_i = 1 - \left(|x_1 - x_2| \middle/ MAX\_DIFF \right), \tag{3.1}$$

where x_1 refers to the target value and x_2 refers to the source case value and *MAX_DIFF* is the greatest difference in values obtained from Table 3.3 and Table 3.4.

This metric was used because it generates a smoothly varying, continuous function. Figure 3.1 shows an example of how similarity values change based on the difference between a target case and a source case's value for the 'players in current hand' feature. This feature records the number of players who have willingly committed chips to the pot during the *preflop*. The x-axis (Difference in value) represents $|x_1 - x_2|$ in equation 3.1, whereas the y-axis represents the value s_i.

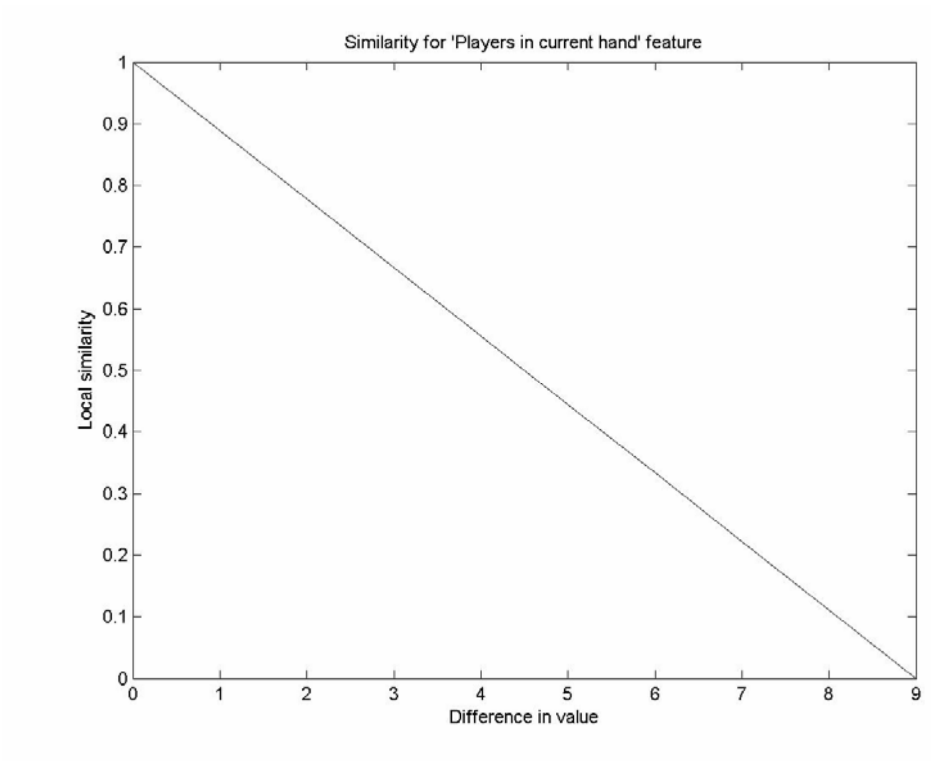


Figure 3.1: Similarity values for the ‘players in current hand’ feature.

So for example, when the target and source case values are exactly the same then local similarity is calculated to be 1.0. When the target and source cases values differ by 1 then similarity is calculated to be 0.89.

3.3.3 Bets committed, Bets to Call and Previous Round Total Bets

Both the preflop case-base and the postflop case-bases record how many bets have been committed during the round (‘bets committed’) and the minimum number of bets that need to be called to stay in (‘bets to call’). All postflop case-bases also record the total number of bets or raises that occurred during the previous betting round (‘previous round total bets’). During the *preflop* and *flop* stages bets are in increments of the small bet and during the *turn* and *river* bets are in increments of the big bet. The similarity metric used for ‘bets committed’, ‘bets to call’ and ‘previous round bets’ differed from the previously mentioned features as differences between their values were believed to have greater significance. For these reasons an exponential decay function was used to compute similarity:

$$s_i = e^{-k\,(|x_1 - x_2|)}, \qquad (3.2)$$

where, x_1 refers to the target value and x_2 refers to the source value and k is a constant that controls the rate of decay. If x_1 has the same value as x_2 similarity is 1.0, indicating an exact match, whereas as the difference between x_1 and x_2 increases the value of s_i rapidly decreases. This is represented pictorially in figure 3.2 below. As can be seen the greater the exponential constant, k, the faster similarity decreases.

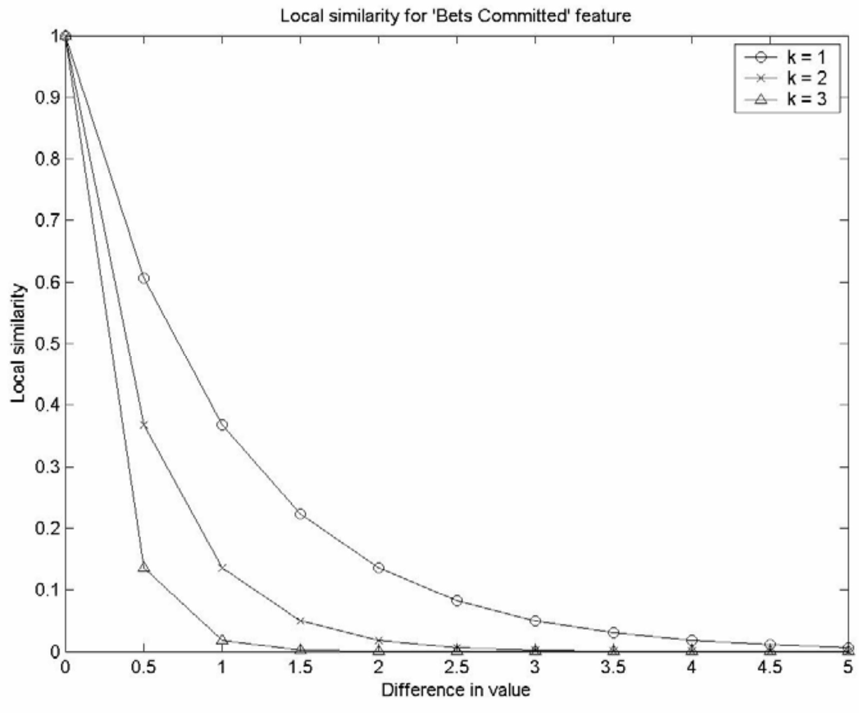


Figure 3.2: Similarity values for the 'bets committed' feature.

Figure 3.2 plots the similarity computed for the 'bets committed' feature using the exponential decay function. Once again the x-axis (Difference in value) represents the absolute difference between the target value and the source value, $|x_1 - x_2|$ in equation 3.2.

Consider an example case where a player has committed no bets to the pot yet. All cases in the case-base which have a value of 0 for the feature 'bets committed' would be given a local similarity measure of 1.0, whereas all cases where a player had already committed one bet would have a similarity of 0.368 (assuming a value of $k = 1$), whereas if the above Euclidean distance measure from section 3.3.2 was used this would result in a local similarity value of 0.8. This dramatic drop in similarity is desirable as scenarios where a player is making their first betting decision are normally quite different from those encountered when a player has already contributed money to the pot.

For the ‘bets committed’ and ‘previous round total bets’ features *k* was assigned the default value of 1.0, whereas ‘bets to call’ used *k = 4.0.*

3.3.4 Hand Ranking

The most important *preflop* feature is hand ranking; a numeric value assigned to the pair of *hole* cards a player has been dealt. During the *preflop* stage there are $\binom{52}{2}$ = 1326 different combinations of cards a player could be dealt, however a lot of these combinations are effectively equal. For example, the exact suit of a card is no longer important as no community cards are yet to be revealed, therefore a hand like **A♥- K♥** can be considered equivalent to **A♣-K♣** and can be classified as AKs, where *s* stands for suited (meaning the same suit). Furthermore a hand such as **T♥-J♦** can be considered equivalent to **T♥-J♠** and can be classified as TJo, where *o* stands for off-suit (meaning separate suits). This means that there are in fact only 169 distinct groups that a preflop hand can fall into. Each of these distinct hands can be ranked by assigning it a number between 1 and 169 where 1 indicates the best possible preflop hand (i.e. AAo, followed by KKo, QQo…) and 169 indicates the worst preflop hand (i.e. 27o). A complete listing of all preflop hand ranks can be found in Appendix A.

These hand rankings were used in an exponentially decaying similarity metric given by the following formula:

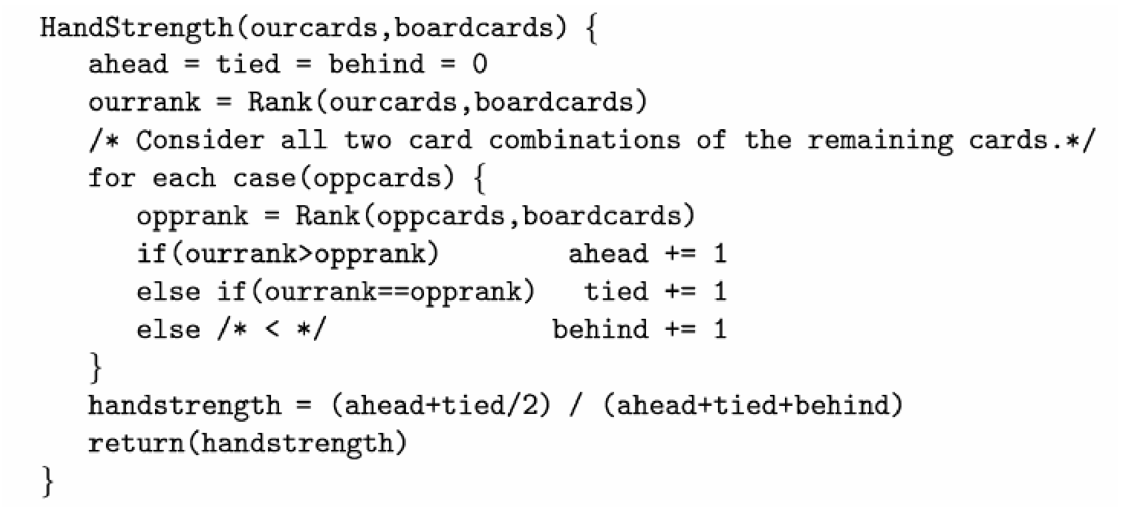
$$s_i = e^{-k\left(|x_1 - x_2| / MAX\_DIFF\right)}, \qquad (3.3)$$

where x_1 refers to the target case’s hand ranking value and x_2 refers to the source case’s hand ranking value. Equation (3.3) only differs from equation (3.2) through the use of *MAX_DIFF* which represents the difference between the highest hand ranking value and the lowest, given by 169 – 1 = 168. An exponential decay constant of $k = 4$ was chosen. These values were chosen so that cards that were close in rank did not produce a similarity measure that was too close.

Initially similarity for hand ranking was computed using a Euclidean distance measure, however it was soon discovered that using this measure produced too high a similarity measure when there were differences between *hole* card groups. Once again a function was needed to ensure similarity dramatically decreased as soon as any difference between the target and source value occurred. This effectively represented the importance of giving preference to cases in the case-base that had the exact same hand rank value.

3.3.5 Immediate Hand Strength, Positive Potential and Negative Potential

During the *preflop* it is only necessary to consider 169 different rankings to get an idea of *preflop* hand strength. However, the use of community cards during and after the flop ensures that a lot more calculations are needed to compute *postflop* hand strength. Immediate hand strength (IHS) can be computed by calculating all the hands which are currently better than ours as well as all the hands which are currently worse and those that are equal (Davidson, 2002). Using these values the probability of having the best hand against one opponent with random *hole* cards, i.e. IHS, can be determined. The algorithm for computing immediate hand strength is given in Figure 3.3. By assigning all combinations of possible *hole* cards (the initial 52 cards in the deck minus our two *hole* cards minus three community cards gives $\binom{47}{2}$ = 1081 combinations) to one opponent and determining how many of these combinations we can beat, draw or lose to, immediate hand strength is given by the amount we can beat plus half of the draws divided by the total.



```
HandStrength(ourcards,boardcards) {
   ahead = tied = behind = 0
   ourrank = Rank(ourcards,boardcards)
   /* Consider all two card combinations of the remaining cards.*/
   for each case(oppcards) {
      opprank = Rank(oppcards,boardcards)
      if(ourrank>opprank)        ahead += 1
      else if(ourrank==opprank)   tied += 1
      else /* < */               behind += 1
   }
   handstrength = (ahead+tied/2) / (ahead+tied+behind)
   return(handstrength)
}
```

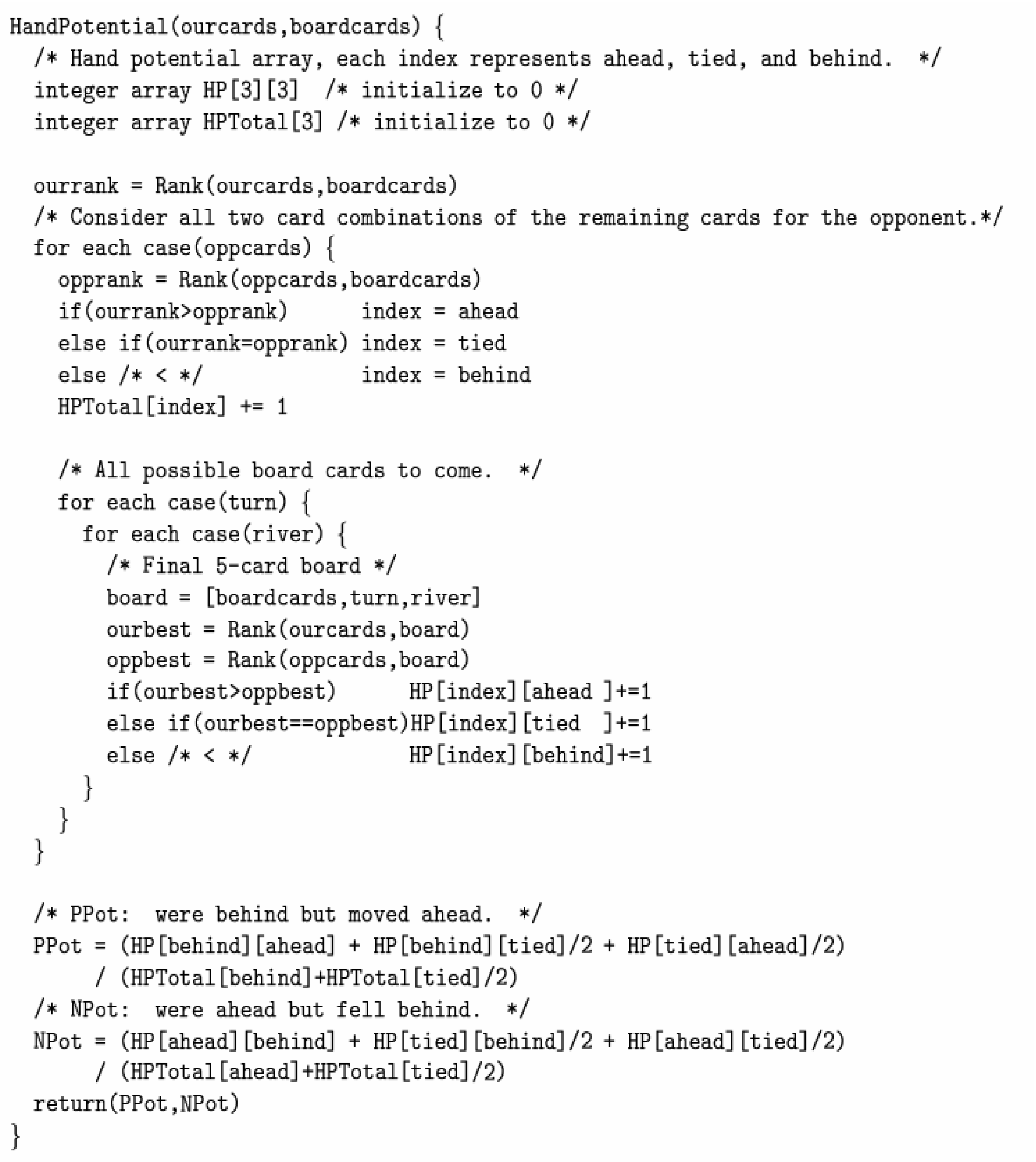
Figure 3.3: Immediate hand strength algorithm. Image sourced from (Davidson 2002).

This gives the hand strength against one opponent, however if more than one opponent is still active in the hand the probability of holding the best hand decreases. Therefore, immediate hand strength when against multiple opponents is computed as follows:

$$\text{IHS} = \text{IHS}^n, \qquad (3.4)$$

where n is the number of active opponents.

Apart from immediate hand strength it is also useful to know the probability of currently having a worse hand which improves to a winning hand after extra community cards have been dealt. This is known as *positive potential* or *ppot* (Davidson 2002). In contrast to positive potential there also exists *negative potential, npot,* which is the probability of currently having the best hand, but after extra community cards are dealt this hand is no longer the best. Figure 3.4 shows the algorithm used for computing ppot and npot.



```
HandPotential(ourcards,boardcards) {
  /* Hand potential array, each index represents ahead, tied, and behind.  */
  integer array HP[3][3]  /* initialize to 0 */
  integer array HPTotal[3] /* initialize to 0 */

  ourrank = Rank(ourcards,boardcards)
  /* Consider all two card combinations of the remaining cards for the opponent.*/
  for each case(oppcards) {
    opprank = Rank(oppcards,boardcards)
    if(ourrank>opprank)      index = ahead
    else if(ourrank=opprank) index = tied
    else /* < */             index = behind
    HPTotal[index] += 1

    /* All possible board cards to come.  */
    for each case(turn) {
      for each case(river) {
        /* Final 5-card board */
        board = [boardcards,turn,river]
        ourbest = Rank(ourcards,board)
        oppbest = Rank(oppcards,board)
        if(ourbest>oppbest)      HP[index][ahead ]+=1
        else if(ourbest==oppbest)HP[index][tied  ]+=1
        else /* < */             HP[index][behind]+=1
      }
    }
  }

  /* PPot:  were behind but moved ahead.  */
  PPot = (HP[behind][ahead] + HP[behind][tied]/2 + HP[tied][ahead]/2)
       / (HPTotal[behind]+HPTotal[tied]/2)
  /* NPot:  were ahead but fell behind.  */
  NPot = (HP[ahead][behind] + HP[tied][behind]/2 + HP[ahead][tied]/2)
       / (HPTotal[ahead]+HPTotal[tied]/2)
  return(PPot,NPot)
}
```

Figure 3.4: Positive and Negative potential. Image sourced from (Davidson 2002).

In the final version of Casper *immediate hand strength* was used in all postflop case-bases, and *positive potential* and *negative potential* were used as features in the *flop* and *turn* case-bases. However initially, experimentation was performed where both the *flop* and *turn* cases used *effective hand strength* and *negative potential* as indexed features instead. Effective hand strength combines immediate hand strength and positive potential into one value. Effective hand strength is calculated as follows:

$$\text{EHS} = \text{IHS}^n + (1 - \text{IHS}^n) \times \text{PPOT} \qquad (3.5)$$

Equation (3.5) states that effective hand strength is given by the probability of having the best hand against *n* opponents and the probability of not having the best hand multiplied by the probability of that hand improving to the best hand. The use of effective hand strength ensures that hands will be bet aggressively even if the chance is high that an opponent will make a better hand with future community cards to come. This is desirable as it forces opponents to pay the highest price for drawing hands (Billings, et al. 2002).

Similarity for both immediate and effective hand strength was measured using the standard 1-dimensional Euclidean distance given by equation (3.1), where MAX_DIFF is equal to 1.0. No noticeable differences were found using this similarity measure as opposed to the exponential decay function of equation (3.2). The final version of Casper used immediate hand strength, rather than effective hand strength as this appeared to improve the performance of the system.

Similarity for PPOT and NPOT was computed using the exponential decay function with a value of $k = 1$. Once again this was done to avoid situations where distinct cases were treated as very similar, when in fact there may have been considerable differences.

3.4 Case Retrieval

When it is Casper's turn to make a betting decision a target case is constructed by assigning each feature its required value. Once a target case has been constructed Casper needs to locate and retrieve the most similar cases it has stored in its case-base. The k-nearest neighbour algorithm is used to compute a similarity value for all cases in the case-base. Global similarity for all cases in the appropriate case-base is computed as a weighted linear combination of local similarity using the following equation:

$$s_j = \sum_{i=1}^{n} w_i x_i \Big/ \sum_{i=1}^{n} w_i \ , \qquad (3.6)$$

where s_j refers to the similarity value calculated for case j, x_i refers to the ith local similarity metric in the range 0.0 to 1.0 and w_i is its associated weight, in the range 0 – 100. Initially, all weights were hand-picked. A default value of 5 was used for most features, while features believed to hold more importance were assigned much higher values in the approximate range 50 – 80. The more salient features included items such as ‘hand strength’ and ‘positive and negative potential’. All hand-picked weights for the *preflop, flop, turn* and *river* case-bases are listed in Chapter 4 of this thesis. Later, an attempt to derive optimal feature weights was performed using evolutionary algorithms (also described in Chapter 4).

After computing a similarity value for each case in the case-base a descending quick sort of all cases is performed. The actions of all cases which exceed a similarity threshold of 97% are recorded. Each action is summed and then divided by the total number of similar cases which results in the construction of a probability triple *(f, c, r)* which gives the probability of folding, checking/calling or betting/raising in the current situation. If no cases exceed the similarity threshold then the top 20 similar cases are used. As an example, assume Casper looks at its *hole cards* and sees **A♥-A♠**, after a search of its *preflop* case-base Casper locates 30 cases that exceed the similarity threshold of 97%. Of the 30 cases retrieved none record a *fold* action, 3 cases record a *check/call* action and the remaining 27 cases all dictate a *bet/raise* action. Using the above information the following probability triple is generated: *(0.0, 0.1, 0.9).* This indicates that given the current situation Casper should never fold this hand, it should just call the bet 10% of the time (note, the option to check is not available here) and it should bet/raise 90% of the time.

A betting decision is then probabilistically made using the probability triple which was generated. For example, consider another probability triple, *(0.25, 0.25, 0.50).* A betting decision is made probabilistically by generating a random number between 0.0 and 1.0. For the triple above, if this random number is less than 0.25 then the decision to fold is taken. If however, the random number is between 0.25 and 0.50 Casper will check, if the option is available, otherwise it will call the minimum bet. Finally, if the random number generated was above 0.50 Casper will bet, if no other players have bet, otherwise it will raise the current bet. Figure 3.5 pictorially represents generating the random number 0.9, which results in the bet/raise action.

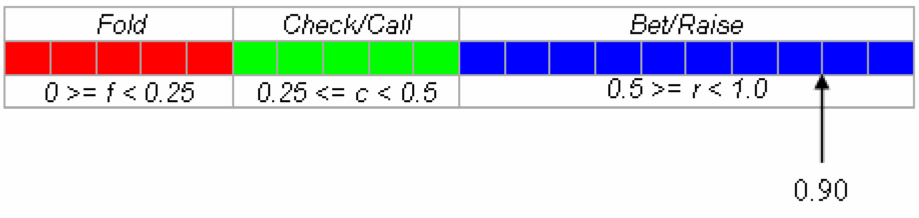


Figure 3.5: Representation of a probability triple. A bet/raise decision is being made.

The use of a probability triple for decision making means that given similar scenarios Casper will not always make the same betting decision. This has both positive and negative consequences. On the positive side it makes Casper's play less predictable and hence harder for its opponents to adapt to. On the other hand it may result in Casper making an 'incorrect' betting decision some of the time. As an example consider a situation where the correct decision is to bet/raise. Imagine 10 cases are retrieved that exceed the similarity threshold. If nine of these cases dictate a bet/raise decision and one indicates a fold action then there is a 10% chance that Casper will fold and make an incorrect decision. However, due to the nature of poker a 'correct decision' is often only known once all hidden information has been revealed.

A further issue is encountered when no cases in the casebase exceed the similarity threshold. This ensures that Casper now needs to make betting decisions based on less similar situations.

3.5 Implementation

Casper was designed to challenge several different types of opponents, including computer opponents such as the University of Alberta bots (*Pokibot* and *Simbot*) as well as real opponents on the internet. Casper was also required to challenge itself during self-play experiments (see Chapter 4). This required separate implementations of Casper to be performed.

3.5.1 Casebase Construction

Casper's casebase was constructed using the Java programming language to parse text files which contained instances of poker games played between *Pokibot* and

Simbot. All relevant feature information was then calculated and stored as cases in text files.

3.5.2 Computer Opponents

Initially, Casper was implemented in Java using the commercially available product Poker Academy Pro 2.5[9] and the Meerkat API which provides the appropriate libraries to program poker-bots. Poker Academy Pro includes the University of Alberta bots as well as other computer opponents who Casper was able to challenge. All testing against computer opponents was performed using Poker Academy Pro.

3.5.3 Self-play Experiments

As mentioned previously, an attempt to derive optimal feature weightings was performed as part of this thesis. This required the implementation of a genetic algorithm. Once again this was implemented using the Java programming language and the Meerkat API with Poker Academy Pro. Casper was able to perform self-play experiments within Poker Academy Pro in an attempt to improve its performance. These experiments are described in detail in Chapter 4.

3.5.4 Real Opponents

Finally, a separate implementation of Casper was required to be able to challenge real opponents across the internet. This was achieved using a product known as WinHoldEm[10], which provides the capability to interface into online poker sites. WinHoldEm allows programmers to write their own dynamically linked libraries using the C/C++ programming language. All testing against real opponents was performed on the Full Tilt Poker[11] online poker site using the WinHoldEm software.

[9] http://www.poker-academy.com/
[10] http://www.winholdem.net/
[11] http://www.fulltiltpoker.com/

Chapter 4

Improving CASPER: Investigating Optimal Feature Weights

The previous chapter described the design and implementation of Casper, where feature weight values were hand picked based on how important they were believed to be. This chapter examines the possibility of improving Casper by deriving *optimal* feature weight values algorithmically. Two attempts to derive optimal weights were investigated. The first involved the use of self-play experiments in an attempt to derive a set of general feature weights suitable for use against any competition. Once this was completed the derivation of weights directly suited to a specific set of opponents was investigated.

4.1 Hand Picked Weights

Each indexed feature that Casper uses to retrieve similar cases needs to be assigned a specific weight which indicates how much it contributes to the retrieval process. Casper used weights in the range from 0 to 100 where 0 indicates that the feature is not used in retrieval and 100 is the maximum possible value. Casper initially used the feature weightings detailed in Figure 4.1.

The values given in Figure 4.1 are hand picked weights which are thought to represent the importance of each particular feature. The greatest importance was assigned to features that represent the strength of Casper's hand. These features include **immediate hand strength**, **positive potential, negative potential** and **hand ranking**. All other features were then given much lower values and can roughly be classified into two subsets:

(1) – Features that control how many bets Casper has made or needs to make e.g. the **bets committed** and **bets to call** features, and

(2) – All other features.

Preflop	***Weighting***
Number of players	5
Relative position	10
Players in current hand	10
Players yet to act	5
Small bets committed	5
Small bets to call	10
Pot Odds	5
Hand ranking	50

Flop	***Weighting***
Number of players	5
Relative position	10
Previous round total bets	5
Players in current hand	5
Players yet to act	5
Bets committed	10
Bets to call	10
Small bets in pot	5
Pot Odds	5
Immediate hand strength	80
Positive potential	80
Negative potential	80

Turn	***Weighting***
Number of players	5
Relative position	5
Previous round total bets	5
Players in current hand	5
Players yet to act	5
Bets committed	20
Bets to call	20
Small bets in pot	5
Pot Odds	5
Immediate hand strength	80
Positive potential	80
Negative potential	80

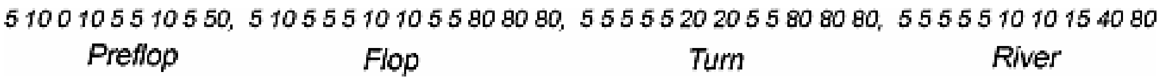
River	***Weighting***
Number of players	5
Relative position	5
Previous round total bets	5
Players in current hand	5
Players yet to act	5
Bets committed	10
Bets to call	10
Small bets in pot	15
Pot Odds	40
Immediate hand strength	80

Figure 4.1: Casper's initial feature weight values.

All features in category (2) (excluding **relative position**) were given a default weight value of 5. Features in category (1) were given slightly higher preference over features in category (2).

4.2 Self-play Experiments

Following the results obtained using the above hand picked feature weights an investigation into obtaining more optimal feature weights was conducted. An evolutionary computing approach was taken whereby a genetic algorithm was used together with self-play experiments to evolve a population of general optimal weight parameters. Each hypothesis in the search space was simply the concatenation of the *preflop*, *flop*, *turn* and *river* weights, for example, the hypothesis that would be used to represent the hand picked weights from the previous section would look as follows:



5 10 0 10 5 5 10 5 50, 5 10 5 5 5 10 10 5 5 80 80 80, 5 5 5 5 5 20 20 5 5 80 80 80, 5 5 5 5 5 10 10 15 40 80

Preflop Flop Turn River

Figure 4.2: Hand-picked weights hypothesis.

The details of the genetic algorithm used are as follows:

- Each generation consisted of nine instances of Casper. The only difference being the feature weights which were used.
- All nine instances of Casper play at the same table against each other and their profits/losses are recorded.
- Initially the first generation were each assigned random weight values for each feature.
- A fitness function was used to probabilistically select members of the current generation. The fitness function was simply the amount of money that each version of Casper had won or lost during the self-play experiments.
- Each generation consisted of playing approximately 5000 hands at the poker table.
- Selection, crossover and mutation were used to derive the next generation (explained below).

4.2.1 Selection

Rank selection was used to probabilistically select a number of instances from the current population to be used in the successor population.

After the completion of approximately 5000 hands all instances of Casper were ranked based on the profit which they had accumulated. Each instance was then given a number from 1 to 9 indicating their probability of selection. The instance that incurred the least profit (greatest loss) was assigned the value 1. The instance that achieved the second greatest loss was assigned the value 2 and so on, up until the instance that achieved the greatest profit was assigned the value 9. This generates a proportional bar graph as follows:

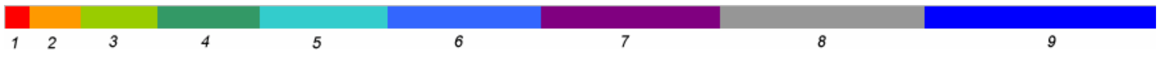


Figure 4.3: Proportional bar graph representing probability of selection.

The horizontal area in the graph above represents the probability of selection into the next population. As can be seen the instance that achieved the least profit (instance 1) has little probability of surviving to the next generation, whereas the better each instance did the higher the probability they will survive. The exact selection probabilities used are displayed in the follow pie chart:

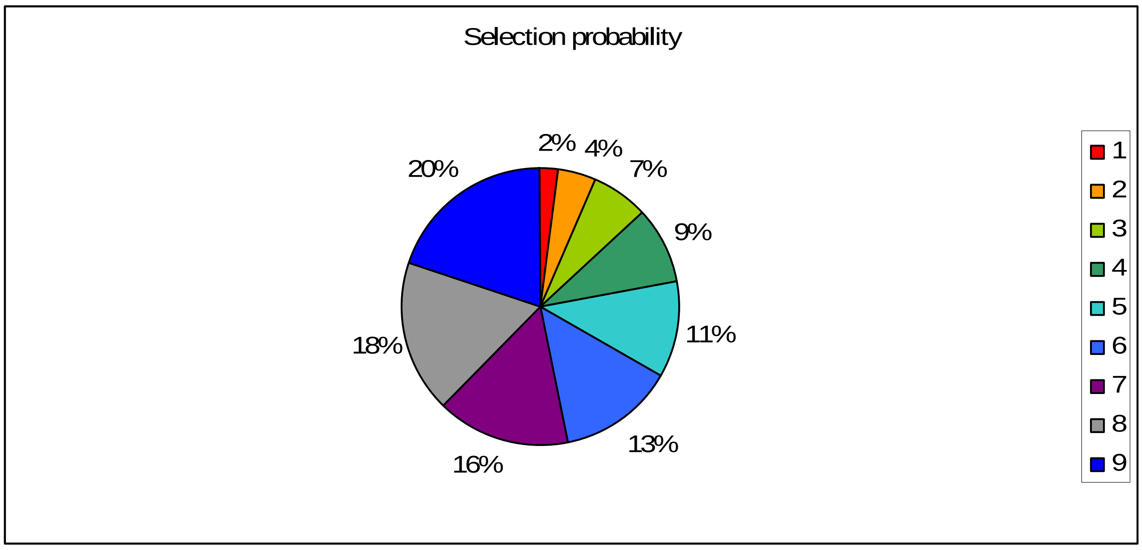


Figure 4.4: Pie chart with selection probabilities.

During initial experiments five instances were chosen for selection to be used in the next generation. Instances were chosen with replacement, meaning that multiple of the same instance could be carried forward to the next generation. Early investigation into this approach seemed to suggest a problem with *crowding* where instances that initially achieved a greater fitness value quickly began to dominate the population, leading to a lack of diversity (Mitchell 1997). The number of instances used in selection was then dropped down to three, which had the result of improving the *crowding* problem. Once again replacement was allowed.

4.2.2 Crossover

During selection hypotheses are selected from the current population and placed into the next generation unchanged. Crossover, however, attempts to select members of the current population to produce offspring and use these new hypotheses in the next generation. Once again rank selection is used to probabilistically select pairs of parent hypotheses from the current population. These parent hypotheses are then combined using a crossover mask to produce two new offspring. A crossover mask is simply a bit string composed entirely of 1's and 0's which are used to control which parent contributes which bit to each offspring. Single-point crossover was used for each stage of play (*preflop*, *flop*, *turn* and *river*) and the result is concatenated together to form a new hypothesis. As an example, consider combining the hand-picked weights from Figure 4.2 and another parent hypothesis given below (note: only the *preflop* stage is considered here).

Preflop weights for parent 1.

Weight	**5**	**10**	**10**	**5**	**5**	**10**	**5**	**50**
Index	0	1	2	3	4	5	6	7

Preflop weights for parent 2.

Weight	**30**	**5**	**15**	**5**	**20**	**80**	**5**	**100**
Index	0	1	2	3	4	5	6	7

A random number is generated between 1 and 7 (inclusive) to indicate the crossover point, e.g. using a value of 4 would produce the following crossover mask: 11110000. This crossover mask now controls which parent contributes which bit to each offspring and would produce the following output:

Preflop weights for offspring 1.

Weight	**5**	**10**	**10**	**5**	**20**	**80**	**5**	**100**
Index	0	1	2	3	4	5	6	7

Preflop weights for offspring 2.

Weight	**30**	**5**	**15**	**5**	**5**	**10**	**5**	**50**
Index	0	1	2	3	4	5	6	7

The same procedure is also separately applied to the *flop*, *turn* and *river* weights and the concatenation of the output produces two new offspring hypotheses. Three pairs of parents were used in the final version of the genetic algorithm to produce a total of six offspring. Parent hypotheses were allowed to be selected multiple times, but they were not allowed to crossover with themselves.

4.2.3 Mutation

Finally, once a new generation of nine hypotheses had been constructed, using selection and crossover, a single hypothesis was chosen at random and one of its weights replaced by generating a random number between 0 and 100.

4.2.4 Implementation

The genetic algorithm described above was implemented in Java and used the Meerkat API to interface into Poker Academy Pro 2.5[12] which allowed different poker bots to challenge each other at the poker table. As the table was limited to nine players each generation consisted of nine separate hypotheses. As stated above 5000 hands were played and then operations such as *selection*, *crossover* and *mutation* were performed to produce the next generation of nine players. After experimenting with various parameters such as the *selection* and *crossover* rates improvements were made to the *crowding* problem (described above). However, it was noticed that crowding still seemed to take place after approximately 7 – 9 generations into the algorithm. To overcome this problem it was decided to run several genetic algorithms and to combine the results, in an attempt to better search the hypothesis space. Nine separate GA’s were run, each consisting of ten generations. Each GA began by generating random

[12] http://www.poker-academy.com/

hypotheses. After ten generations had completed (50,000 poker hands) the final generation normally consisted of very similar hypotheses. Each of the best hypotheses (i.e. the hypothesis that maximised the fitness function) from the nine genetic algorithms run were collected and used as initial hypotheses in a final genetic algorithm. The final GA once again consisted of ten generations. After ten generations had completed the hypothesis with the highest fitness was once again chosen. This hypothesis was used as an approximation to the optimal feature weights desired. In total 500,000 poker hands were played over 100 generations to derive this set of weights. Figure 4.5 shows a pictorial representation of the genetic algorithm's design.

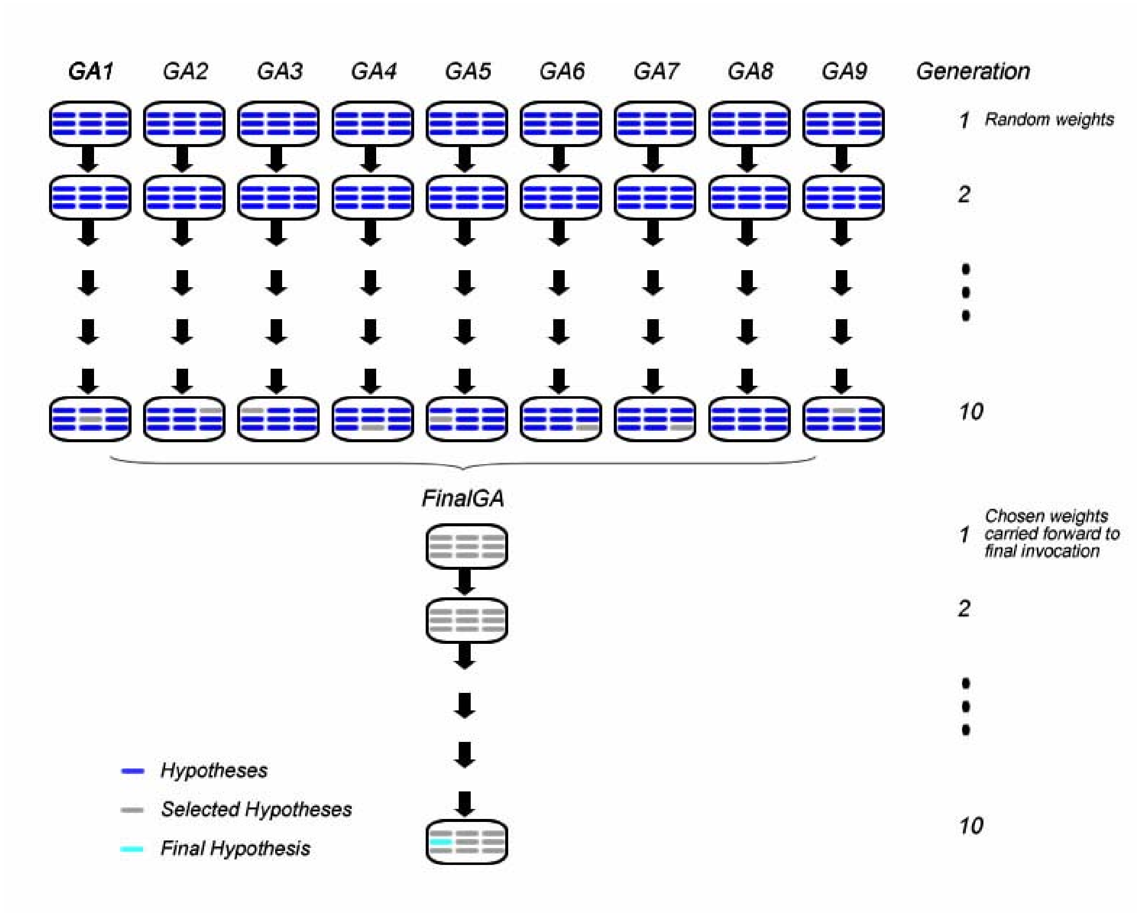


Figure 4.5: Pictorial representation of GA design.

4.2.5 Derived Weights

Figure 4.6 lists the optimal feature weights produced by the genetic algorithm for each separate stage of play. The hand picked weights are replicated in brackets for convenience.

Preflop	***Weighting***
Number of players	15 (5)
Relative position	61 (10)
Players in current hand	54 (10)
Players yet to act	62 (5)
Small bets committed	53 (5)
Small bets to call	83 (10)
Pot Odds	93 (5)
Hand ranking	94 (50)

Flop	***Weighting***
Number of players	9 (5)
Relative position	23 (10)
Previous round total bets	62 (5)
Players in current hand	45 (5)
Players yet to act	54 (5)
Bets committed	82 (10)
Bets to call	70 (10)
Small bets in pot	52 (5)
Pot Odds	39 (5)
Immediate hand strength	99 (80)
Positive potential	74 (80)
Negative potential	51 (80)

Turn	***Weighting***
Number of players	26 (5)
Relative position	14 (5)
Previous round total bets	22 (5)
Players in current hand	15 (5)
Players yet to act	16 (5)
Bets committed	55 (20)
Bets to call	7 (20)
Small bets in pot	35 (5)
Pot Odds	32 (5)
Immediate hand strength	96 (80)
Positive potential	72 (80)
Negative potential	71 (80)

River	***Weighting***
Number of players	38 (5)
Relative position	1 (5)
Previous round total bets	55 (5)
Players in current hand	37 (5)
Players yet to act	6 (5)
Bets committed	98 (10)
Bets to call	99 (10)
Small bets in pot	1 (15)
Pot Odds	26 (40)
Immediate hand strength	98 (80)

Figure 4.6: Casper’s feature weights derived using a genetic algorithm.

Perhaps what stands out most is that relatively high weights are assigned to features that refer to the strength of Casper's hand. For example, *preflop* **hand ranking** is given a value of 94 and *flop, turn* and *river* **immediate hand strength** are given values of 99, 96 and 98 respectively. Recall that features that control how many bets Casper has made or needs to make such as **bets committed** and **bets to call** were thought to have less significance than features that refer to Casper's hand strength, but were thought more important than other features such as **relative position**. The above weights appear to agree with this assumption, assigning values of 98 and 99 to **bets committed** and **bets to call** respectively during the *river* stage, as well as 82 and 70 during the *flop* stage. However, this is not the case for the values assigned during the *turn,* where **bets committed** was given a value of 55 and **bets to call** assigned a value of only 7.

4.2.6 Convergence

Perhaps an explanation for the discrepancies in weights described above is that the genetic algorithm has not converged upon the optimal solution. Due to the nature of the problem it is difficult to determine exactly whether the algorithm has converged or not, however we can possibly gain some insight into the results of the GA by examining the difference in player's profits over successive generations, also known as the variance.

4.2.6.1 Variance

Recall that each generation is made up of nine players. The only difference between each player are the weights they have assigned to each feature. At the beginning of a generation all individuals begin with $0 profit/loss. Once 5000 hands have been completed all individuals in the population have their profit/loss recorded. The difference between the highest profit earned and the greatest loss incurred during the generation is known as the variance. The variance was recorded over successive generations in an attempt to monitor possible convergence and track the progress of the genetic algorithm. Figure 4.7 below shows a plot of the variance for each generation for four of the ten genetic algorithm invocations that were conducted (only four GA's are shown to help improve the clarity, however their behaviour is relatively typical of the remaining GA's).

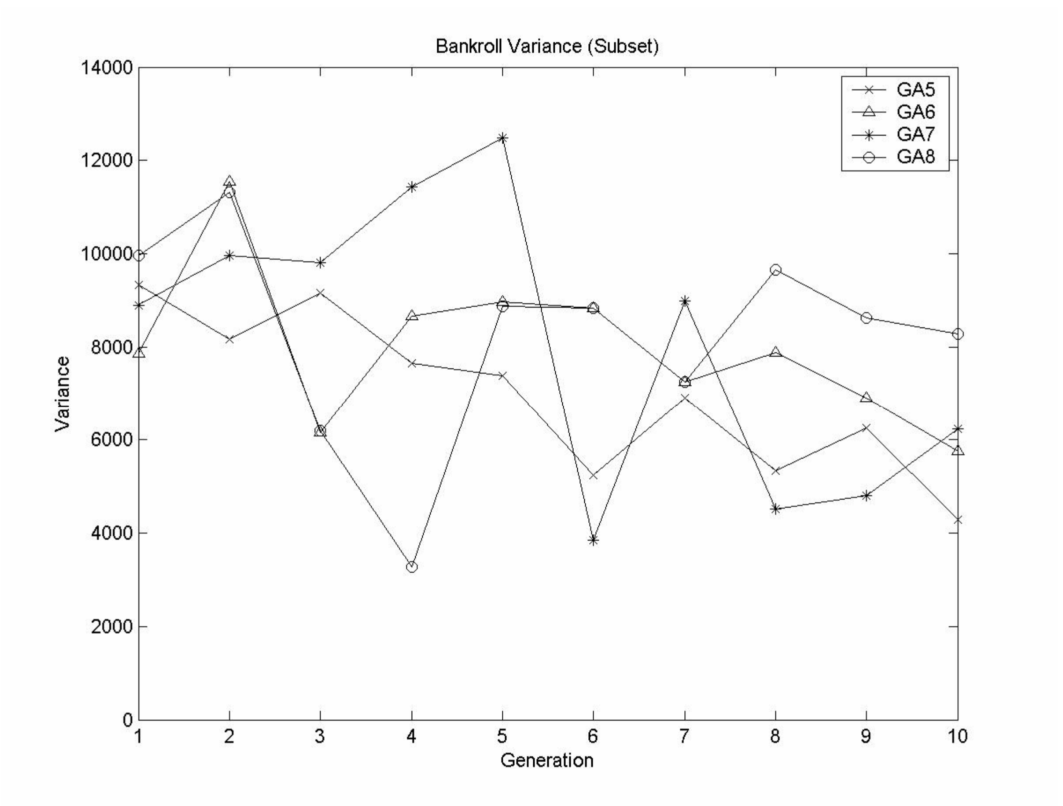


Figure 4.7: Bankroll variance recorded for each generation for a subset of four GA’s.

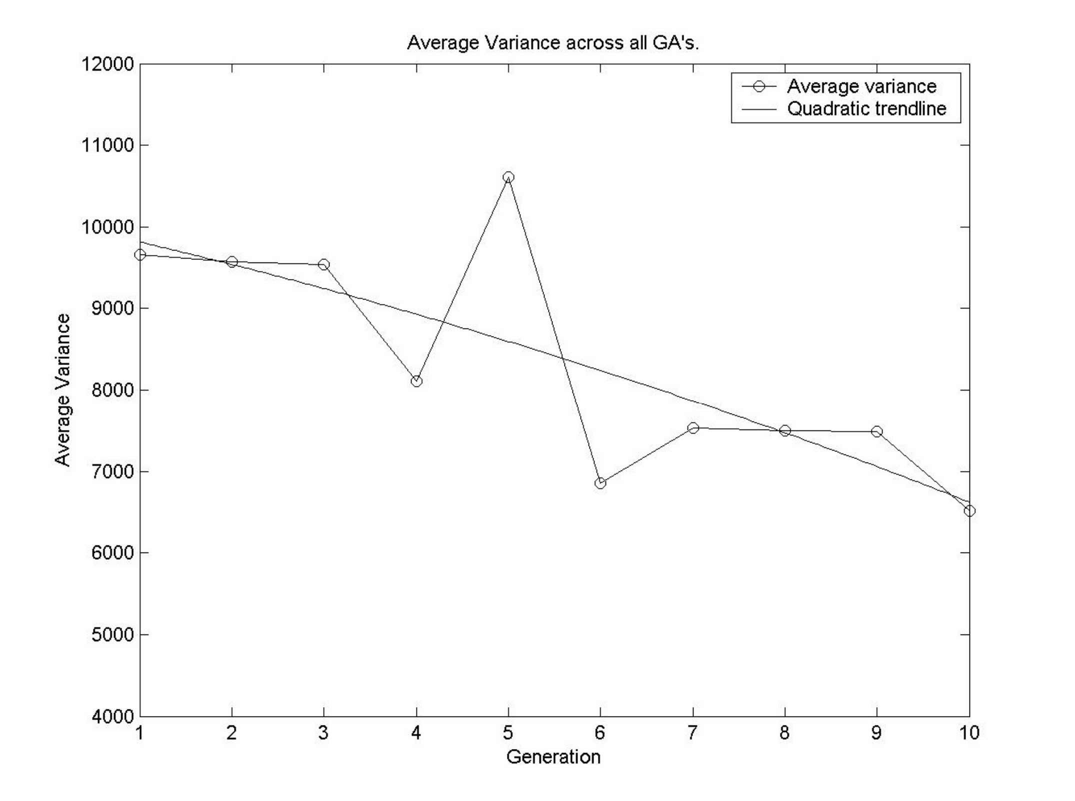


Figure 4.8: Average bankroll variance across all GA’s. The data indicates a downward trend.

While the data represented in Figure 4.7 indicates a lot of variability it does appear as though the variance is decreasing over successive generations. To examine this further the average variance over the first nine GA invocations were recorded. This information is displayed graphically in Figure 4.8.

Figure 4.8 seems to support the fact that as the number of generations increases the average variance decreases, in general. After the first generation, where all individuals in the population have been assigned random weights, the average bankroll variance is \$9,655.95. This figure decreases to \$6,518.19 after the last generation, where the weights should have now converged to some local minima.

Finally, the changes in variance observed for the final invocation of the genetic algorithm are displayed in Figure 4.9. Recall, that generation 1 was made up of the 9 best hypotheses obtained from the previous 9 GA's. While the variance values have decreased the overall downward trend remains the same.

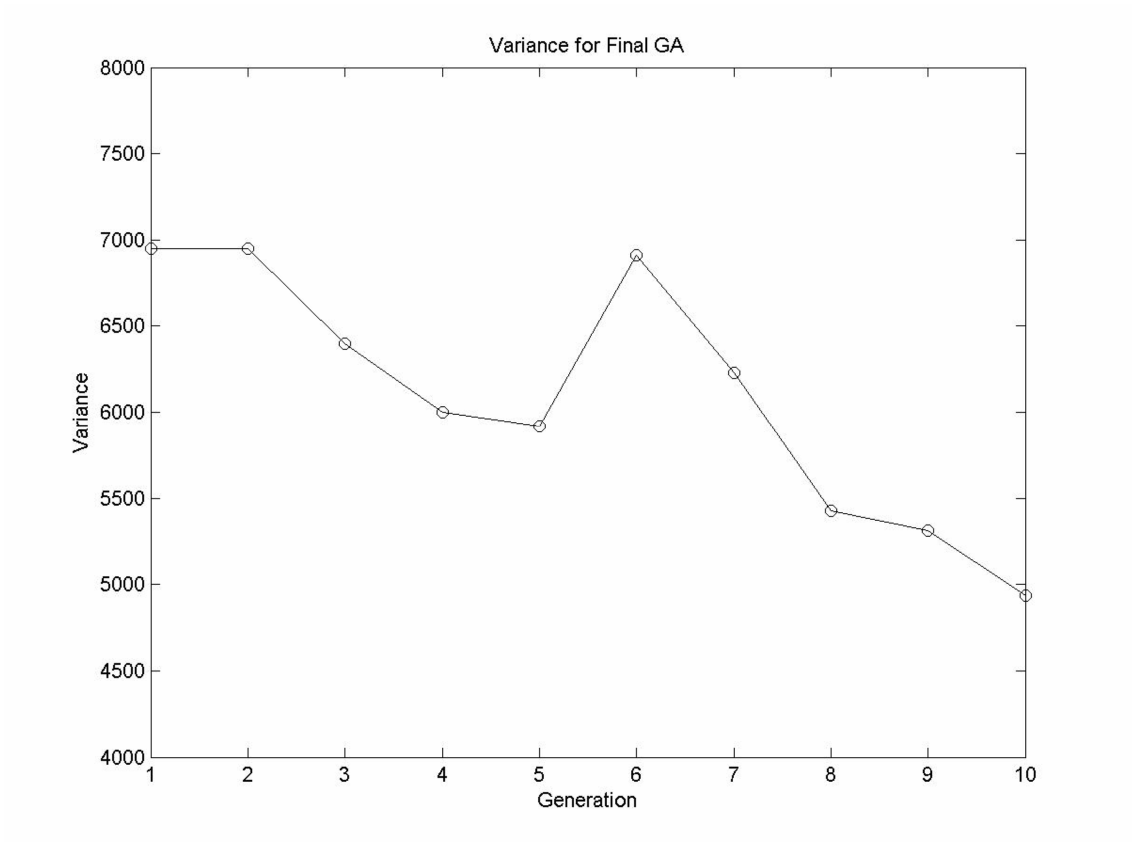


Figure 4.9: Bankroll variance recorded for the final invocation of the genetic algorithm.

From these observations it appears as though the overall genetic algorithm is possibly converging. One possible explanation for the decreasing trend is as follows:

Recall that the population that comprises each first generation is made up of random hypotheses, i.e. random weight vectors. Now, some of these hypotheses will be close to optimal solutions and others will be further away in the search space or, equivalently, some individuals in the population will be good players and others will be bad. It would be expected that after 5000 poker hands were played the better players will have extracted more money from the worse players than from other individuals. Therefore, the good players should have earned a considerable profit and the bad players incurred a considerable loss. As each generation progresses and operations such as selection and crossover are performed it becomes more likely that the bad players will be extinguished from the population and the better players will proceed into the next generation. So, overall the individuals in future generations should become better players and therefore it becomes harder to substantially increase profits (because bad players are no longer giving their money away to better players). As a result, the variance tends to decrease. In this sense the *variance* could possibly act as an approximate measure of the quality of the individuals that make up a generation.

4.3 Opponent-Based Experiments

The genetic algorithm described above attempted to derive a set of general optimal feature weights for use against all types of opponents. Further to this a separate genetic algorithm was designed to specifically take into account the opponents it was challenging. The opponents chosen were the University of Alberta *Pokibots*. Rather than conducting self-play experiments as in section 4.2, opponent-based experiments were performed where Casper was evolved by directly competing against a population of *Pokibots*. The aim of the genetic algorithm described below was to incorporate opponent modeling capabilities directly into the derivation of the feature weights.

Once again each generation consisted of playing 5000 poker hands, however as Casper had to challenge separate opponents the population size had to decrease. It was decided to use a population of three instances of Casper and six *Pokibots*. Profits/losses were recorded for all versions of Casper and this information used to influence selection and crossover operations.

4.3.1 Selection

Once again, rank selection was used to determine which hypotheses would proceed to the next generation. As the population size was so small, it was decided that only one hypothesis would be selected. Hypotheses were ranked by the amount of profit they had accumulated. The hypothesis that had achieved the greatest profit (or least loss) had a 50% probability of being selected for the next generation. This was followed by a 33% probability for the second best hypothesis and a 17% probability for the worst hypothesis in the population.

4.3.2 Crossover

Rank selection was again used to probabilistically select one distinct pair of parent hypotheses from the current population to be used for crossover. Single-point crossover was used for each stage of play (*preflop*, *flop*, *turn* and *river*) and the result concatenated together to form two new hypotheses.

4.3.3 Mutation

Once selection and crossover had completed each hypothesis in the new generation had a 1% chance of mutation. Mutation involved selecting a random weight within the hypothesis and replacing its value with a random number between 0 and 100.

4.3.4 Implementation

As stated above, the design of the opponent-based GA differed from the general GA in that each table no longer consisted of nine instances of Casper. Each table now consisted of six *Pokibots* and three instances of Casper. Once again the first generation began by initialising all instances of Casper with random weights. After 5000 hands were completed selection, crossover and mutation operations (described above) were performed and a new generation created. Due to the decrease in population size it took fewer generations before *crowding* became an issue, therefore only five generations were conducted rather than ten. After five generations were completed the hypothesis which maximized the fitness function was recorded.

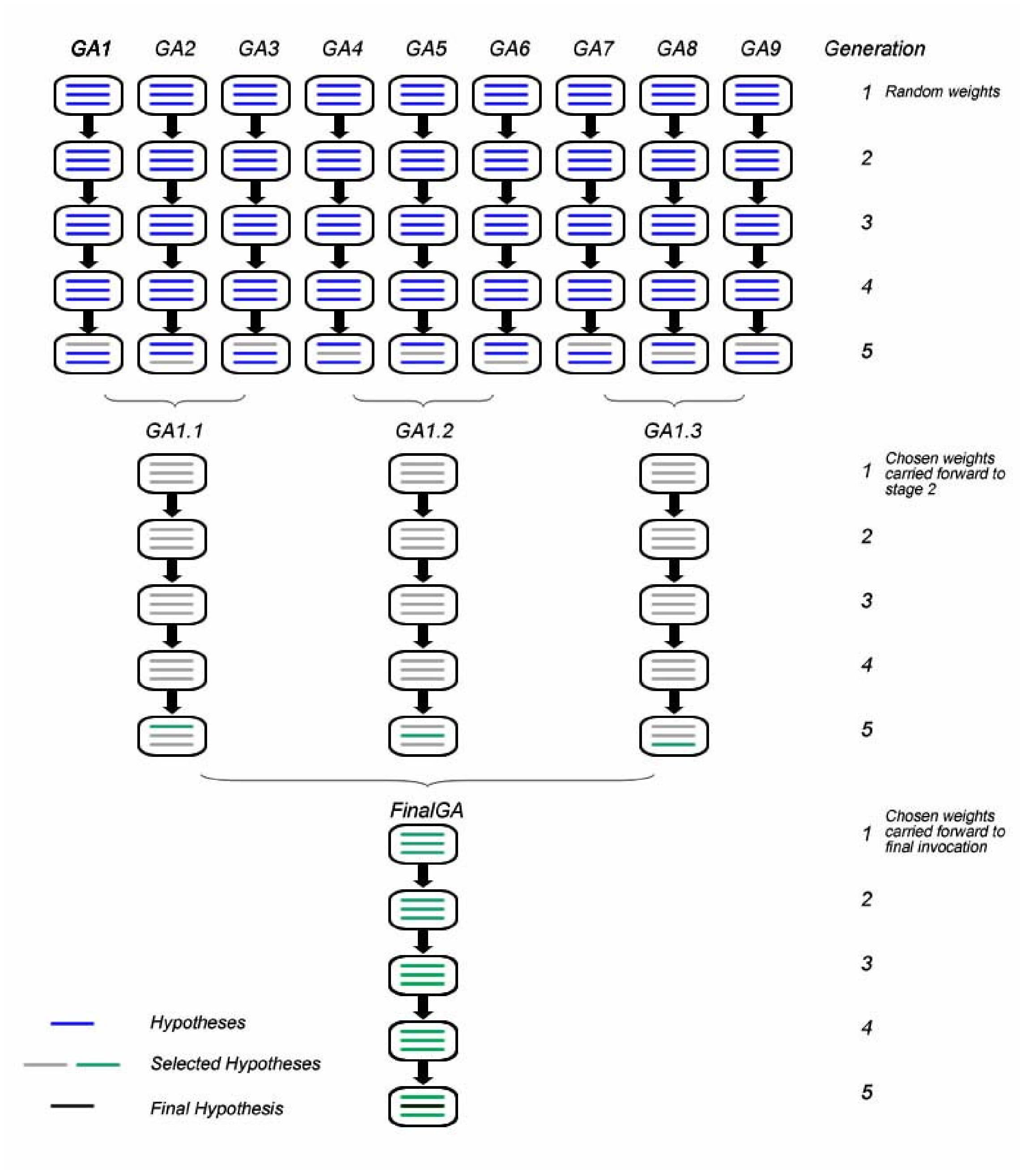


Figure 4.10: Pictorial representation of GA design.

Once again multiple invocations of the genetic algorithm were performed in an attempt to fully explore the hypothesis space. Initially nine invocations were performed (*GA1 – GA9*) which resulted in nine separate hypotheses being generated. These hypotheses were split into three groups of three and used as the weights in the first generation for a further three invocations (*GA1.1 – GA1.3*). Once again, after five generations the best hypothesis was retained for each invocation. This resulted in a final three hypotheses which were used as the initial weights in a final run of the genetic algorithm (*FinalGA*) to produce one set of opponent-based feature weights. In total 325,000 poker hands were played over 65 generations to produce this set of weights. This design is represented pictorially in Figure 4.10.

4.3.5 Derived Weights

Preflop	***Weighting***
Number of players	97 (15)
Relative position	62 (61)
Players in current hand	82 (54)
Players yet to act	78 (62)
Small bets committed	49 (53)
Small bets to call	3 (83)
Pot Odds	42 (93)
Hand ranking	73 (94)

Flop	***Weighting***
Number of players	71 (9)
Relative position	56 (23)
Previous round total bets	32 (62)
Players in current hand	93 (45)
Players yet to act	53 (54)
Bets committed	5 (82)
Bets to call	34 (70)
Small bets in pot	85 (52)
Pot Odds	3 (39)
Immediate hand strength	39 (99)
Positive potential	45 (74)
Negative potential	18 (51)

Turn	***Weighting***
Number of players	46 (26)
Relative position	1 (14)
Previous round total bets	73 (22)
Players in current hand	22 (15)
Players yet to act	41 (16)
Bets committed	55 (55)
Bets to call	100 (7)
Small bets in pot	93 (35)
Pot Odds	27 (32)
Immediate hand strength	32 (96)
Positive potential	33 (72)
Negative potential	13 (71)

River	***Weighting***
Number of players	36 (38)
Relative position	7 (1)
Previous round total bets	6 (55)
Players in current hand	4 (37)
Players yet to act	79 (6)
Bets committed	94 (98)
Bets to call	98 (99)
Small bets in pot	55 (1)
Pot Odds	52 (26)
Immediate hand strength	69 (98)

Figure 4.11: Casper's opponent-based feature weights derived using a genetic algorithm.

Figure 4.11 highlights some significant differences between the weights derived for the first, general genetic algorithm (listed in brackets) compared to the opponent-based genetic algorithm. Most notably, features that relate to Casper's hand strength and potential, such as **immediate hand strength**, **positive potential** and **negative potential** have been assigned relatively lower values when compared to the other available features. Once again, there are discrepancies between salient features during the *flop* and the *turn.* According to the derived weights the most important features during the *flop* appear to be the amount of opponents also in the hand. Features such as **bets committed** and **bets to call** have had their values significantly reduced. However, the *turn* seems to indicate that **bets to call** is the most important feature (as does the *river*).

4.3.6 Convergence

Once again we would like to gain some insight into whether the genetic algorithm described above is actually converging or whether it is just producing random weight vectors. As the population in the opponent-based GA experiments consisted of competition from *Pokibots* it is possible to track the profits/losses recorded of the hypotheses in the population over successive generations in an attempt to try and establish whether the population is improving over time or not.

4.3.6.1 Average Profit

Recall that each generation consisted of three hypotheses and six instances of *Pokibots*. In order to establish whether the population is improving over time, the average profit for each of the three hypotheses was recorded for each generation. This information is graphed in Figure 4.12 for four out of the original nine genetic algorithms used. Only four instances are plotted to help improve clarity, however their behaviour is believed to be representative of the entire population.

While some of the GA's represented do exhibit improvements in profit over successive generations no obvious trend is evident in the data represented in Figure 4.12.

Next, the average for all of the GA's (*GA1 – GA9*) was computed to try and determine the overall behaviour of the genetic algorithms. (Note: this is actually the average of the average for each individual GA). Figure 4.13 depicts this overall average profit for all initial GA's.

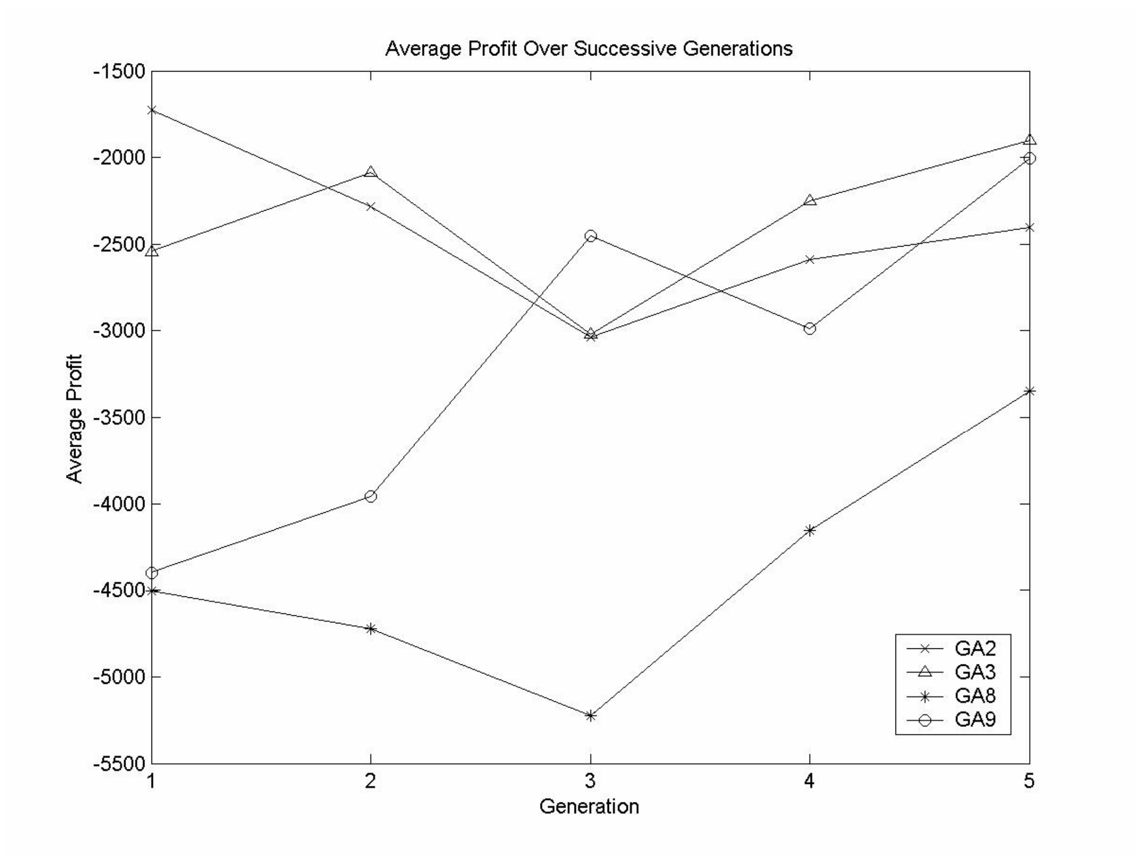


Figure 4.12: Average profit for a subset of GA's recorded over each generation.

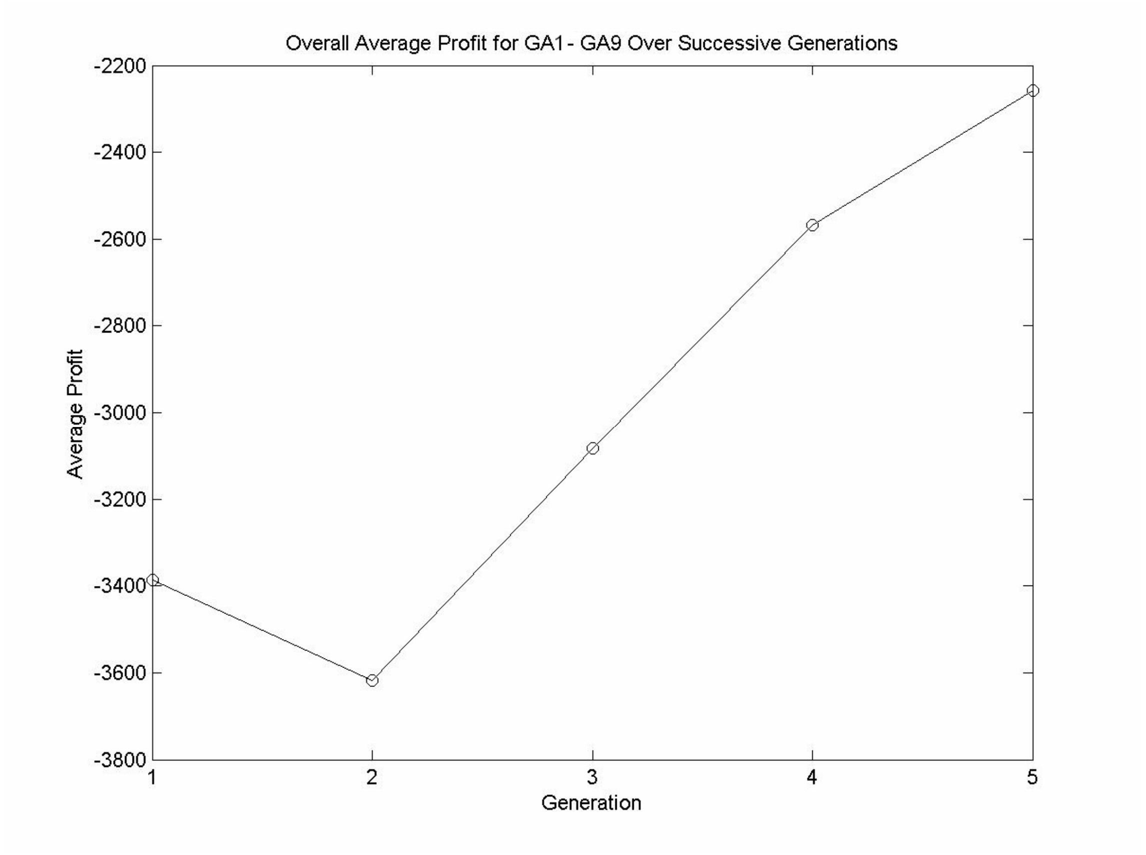


Figure 4.13: Overall average profit for all initial GA's recorded over each generation.

Now the trend is much more apparent, over each successive generation the overall average profit appears to improve, or more correctly, the average loss steadily decreases. This appears to indicate that the overall pattern for each of the initial nine GA's is an increase in profits (or decrease in losses). While each generation may be improving upon the preceding generation it is far from clear whether the GA is converging towards an optimal solution.

The final invocation of the GA (*FinalGA*) involved establishing a population of three of the best hypotheses obtained from *GA1.1*, *GA1.2* and *GA1.3* (see Figure 4.10). This formed the initial generation for one final run of the GA, where once again another 5 generations were processed. After 5 generations were completed the hypothesis which maximized the fitness function (i.e. obtained the greatest profit) was chosen as the derived set of weights. Figure 4.14 tracks the changes in average profit for all three hypotheses in the population over all 5 generations.

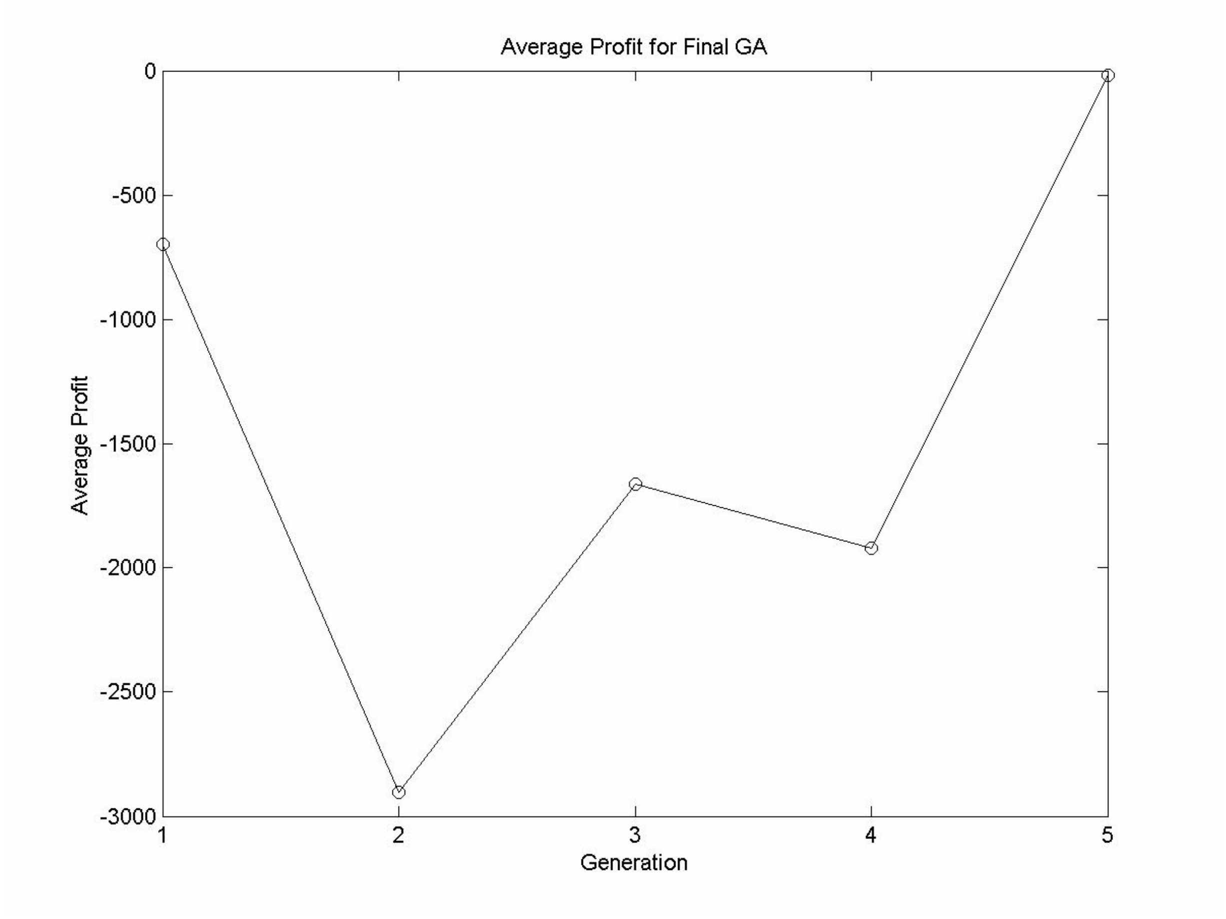


Figure 4.14: Average profit recorded for each generation for the final GA.

Apart from the first generation, Figure 4.14 suggests an overall increase in profits (decrease in losses). After an initial, relatively high average profit (generation 1) then profit seems to steadily increase over generations 2 – 5, with the final ‘average profit’ being calculated at just under $0 (i.e. breaking even).

It should also be mentioned that one of *Pokibots* main abilities is its opponent modelling capabilities. When *Pokibot* meets a new opponent it has no information about that opponent’s strengths or weaknesses or style of play. However, as the number of hands played increases *Pokibot* consistently gathers data about its opponent to try and model their style of play. This information is then used to inform *Pokibot’s* betting strategy against that particular opponent. This aspect of *Pokibots* play needs to be kept in mind when analysing the results of the genetic algorithm, as it becomes much harder to increase profits after playing a large number of hands against *Pokibot*. At the start of Generation 1 *Pokibot* has no information about Casper’s style of play, so no opponent modelling is possible. However, after approximately 10,000 hands (Generation 3) *Pokibot* would have gathered a large amount of data on modelling Casper’s play and would be using this data to determine how to act in certain situations. Therefore, *Pokibot* would most likely be playing differently then it was at the beginning of the GA. Casper has no such dynamic opponent modelling capabilities, rather the only thing that changes during the execution of the GA are Casper’s relative weights.

Chapter 5

Results

Casper has been evaluated by challenging many different opponents on various poker tables. To begin with Casper was evaluated by playing other poker bots provided through the commercial software product Poker Academy 2.5. Initially Casper was tested at two separate poker tables on Poker Academy 2.5. The first table was made up of the University of Alberta *Pokibots* and *Simbots*. This table consisted of strong, adaptive poker bots that model their opponents and try to exploit weaknesses. As Casper has no adaptive qualities of its own it was also tested against non-adaptive, but loose/aggressive opponents. A loose opponent is one that plays a lot more hands, whereas aggressive means that they tend to bet and raise more than other players. Finally, Casper was also tested against real opponents on the internet, offering a range of wildly differing styles of play. In total, three separate versions of Casper were evaluated. The first version of Casper used the hand-picked feature weights. The next version of Casper used the derived set of general weights from self-play experiments and the final version used the opponent-based feature weights specifically tailored for challenging the University of Alberta *Pokibots*. Each of these are presented below.

5.1 Hand-picked Weights

This section will describe the results obtained for Casper, using hand-picked weights, challenging computerised opponents. Recall that initially two separate versions of Casper were investigated that used the hand-picked weights: Casper01 and Casper02. Casper02 improved upon Casper01 by using a larger case-base generated from approximately 20,000 hands. A poker bot that makes totally random betting decisions was also tested separately against the same opponents as a baseline comparison. All games were \$10/\$20 limit games which consisted of 9 players. All players began with a starting bankroll of \$100,000.

5.1.1 Strong/Adaptive Competition

As stated above, the strong/adaptive competition was composed of various *Pokibots* and *Simbots* provided by the University of Alberta. Figure 5.1 plots the amount of *small bets won* for Casper01, Casper02 and the Random poker bot, over a period of approximately 20,000 hands. The *small bet* in this case is $10, therefore if a player has won one small bet they have made a profit of $10. Also listed are the amount of *small bets won per hand* (sb/h), this is a measure of how (un)profitable a particular player is. If a player makes +0.5 sb/h this means that they make a profit of half the small bet for every hand that they play. If the small bet is $10 this means they make $5 for every hand played. Say that player plays 40 hands per hour; this works out to be a profit of $200 per hour.

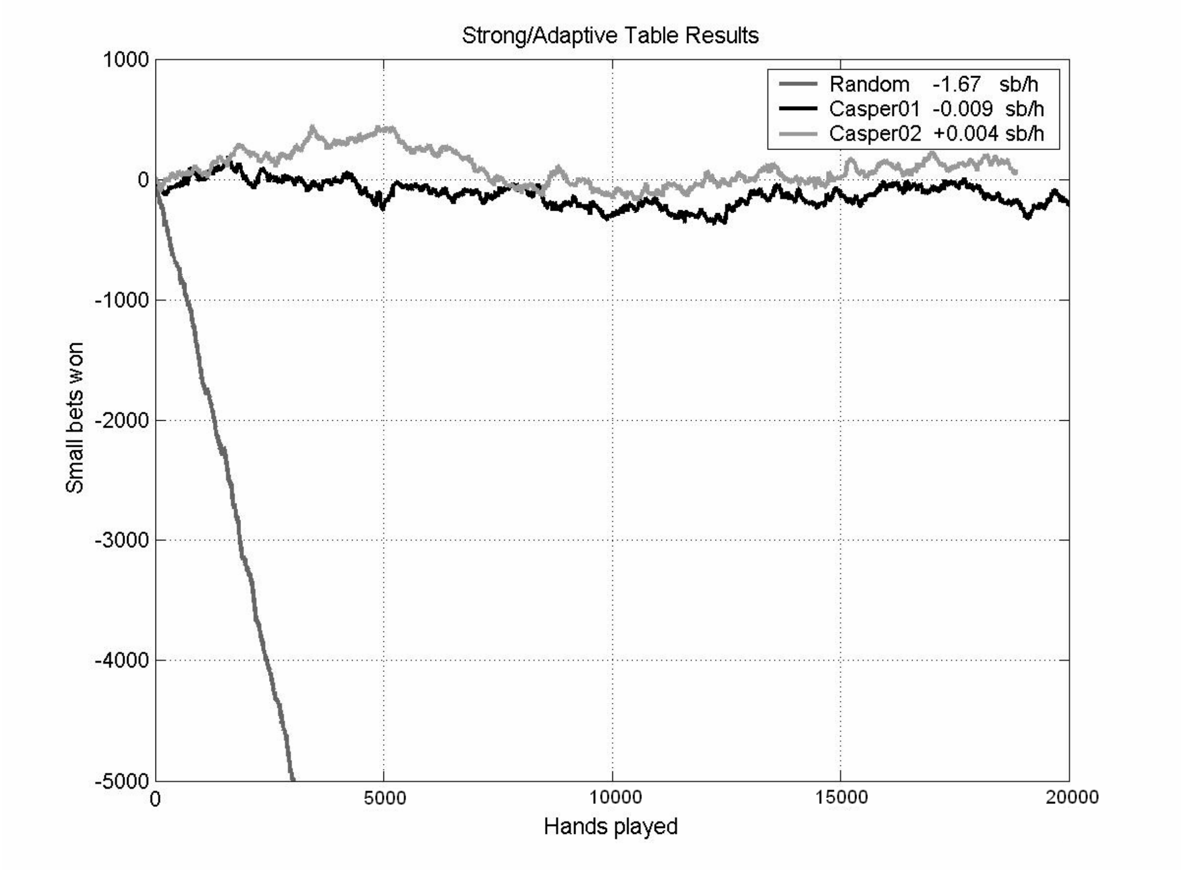


Figure 5.1: Results obtained at the Strong/Adaptive Table.

While Casper01 concludes with a slight loss and Casper02 concludes with a slight profit, Figure 5.1 suggests that both versions approximately break even against strong competition, whereas the random player exhausted its bankroll of $100,000 after approximately 6,000 hands[13]. Casper01's small bets per hand (sb/h) value is -0.009 which indicates that Casper01 loses about \$0.09 with every hand played. Casper02 slightly improves upon this by winning approximately \$0.04 for every hand played. Random play against the same opponents produces a loss of \$16.70 for every hand played.

5.1.2 Aggressive/Non-Adaptive Competition

The next table that Casper was tested on consisted of different versions of another computerised opponent known as Jagbot (also available with Poker Academy 2.5). Jagbot is a loose/aggressive rule-based player. Its style of play is very different than that of *Pokibot* and *Simbot* in that it will play a lot more hands and bet very aggressively, this requires that Casper make a lot more decisions than previously.

Figure 5.2 records the amount of small bets won over a period of approximately 20,000 hands. Once again, a bot which makes completely random decisions was also tested separately against the same competition as a baseline comparison for Casper.

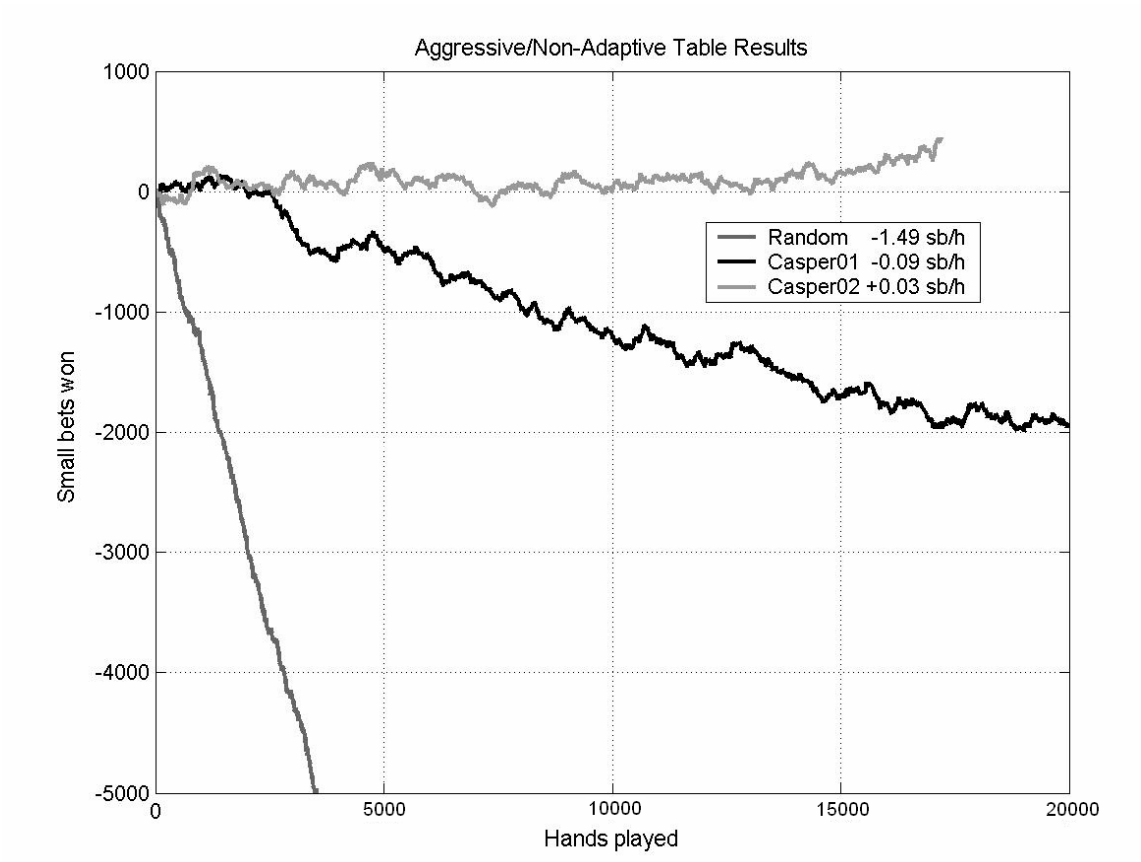


Figure 5.2: Results obtained at the Aggressive/Non-Adaptive Table.

[13] Not all data points are shown to improve clarity.

Figure 5.2 indicates that the first version of Casper tested is unprofitable against the aggressive/non-adaptive players. Casper01 loses approximately $0.90 for each hand played. Casper02, however, shows a considerable improvement in performance compared to Casper01. The addition of extra cases sees Casper02 produce a profit of +0.03 sb/h, or $0.30 for each hand played. Once again, the random player exhausted its initial bankroll after approximately 7000 hands, losing on average $14.90 for each hand played.

It is also of interest to compare how Casper performs against the aggressive/non-adaptive opponents compared to how the University of Alberta *Pokibots* and *Simbots* perform. Therefore, one *Pokibot* was selected to be tested against a table of Jagbots. As was one *Simbot*, tested against a separate table of Jagbots. The results obtained are presented in Figure 5.3.

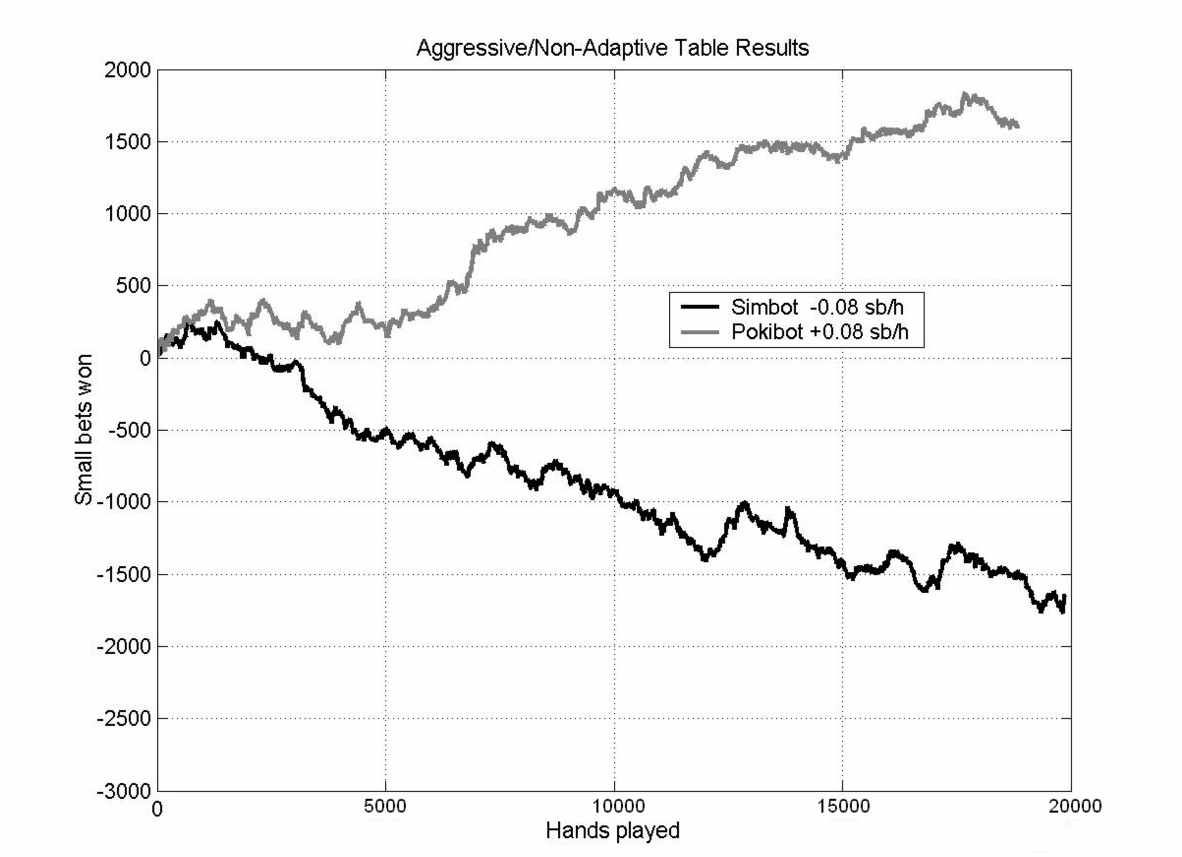


Figure 5.3: University of Alberta bots tested at the Aggressive/Non-Adaptive Table.

Figure 5.3 indicates a dramatic difference between the results obtained for the *Pokibots* and for the *Simbots*. The *Pokibot* tested makes a consistent profit of +0.08 sb/h, compare this to Casper02 from Figure 5.2 which achieves a slightly lesser profit of

+0.03 sb/h. However, *Simbot* is unprofitable against the aggressive competition losing at a rate slightly less than the first version of Casper (i.e. Casper01). Therefore, while the final version of Casper (Casper02) is not as profitable as *Pokibot* is against aggressive competition, it appears as though it out performs *Simbot* against this particular set of opponents.

5.2 Evolutionary Derived Weights

This section describes the results obtained against computerised opponents for the versions of Casper that used evolutionary derived weights. *CasperGeneral* refers to the use of general weights derived using self-play experiments, whereas *CasperOppBased* refers to the opponent-based set of weights. Finally, a version of Casper that used random weights, *CasperRandom*, was also tested in an attempt to determine whether there were any real performance differences between the separate sets of weights. Once again all games were $10/$20 limit games which consisted of 9 players and all players began with a starting bankroll of $100,000.

5.2.1 Strong/Adaptive Competition

The results obtained against the University of Alberta bots are recorded below in Figure 5.4. Using the weights derived for challenging general opponents Casper achieves a consistent loss against the strong/adaptive competition of -0.06 sb/h or -$0.60 per hand. Using random weights Casper consistently loses -0.11 sb/h or -$1.10 per hand. If we compare this with the results obtained using hand-picked weights from section 5.1.1 we notice that using the derived weights for general competition has not improved Casper's performance. Recall that using hand-picked weights Casper achieved a slight profit of +0.004 sb/h.

CasperOppBased on the other hand does appear to improve upon CasperGeneral. CasperOppBased uses weights that were derived specifically for this type of competition. While the results show a lot of variability, CasperOppBased concludes with a slight profit of +0.006 sb/h, marginally improving upon the hand-picked weights.

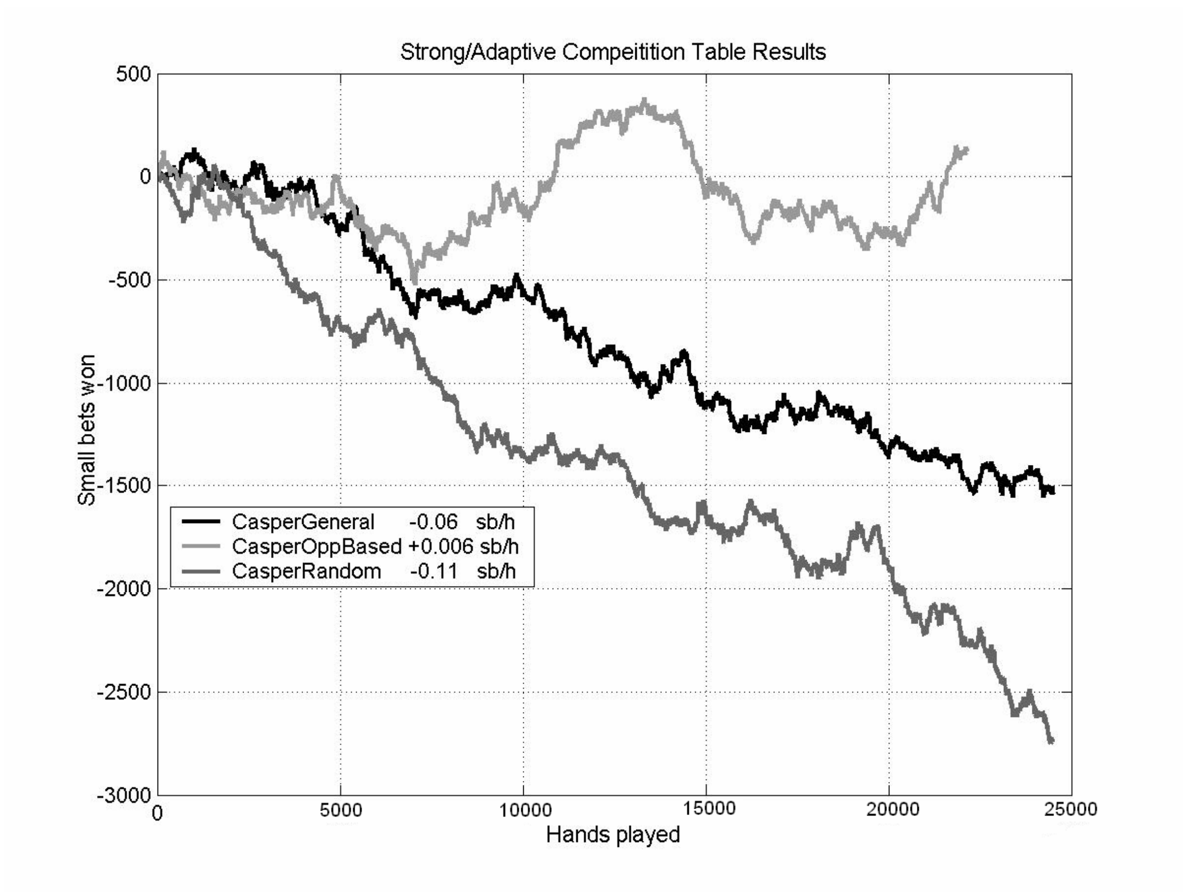


Figure 5.4: Results obtained at the Strong/Adaptive Table.

5.2.2 Aggressive/Non-Adaptive Competition

Figure 5.5 displays the results of CasperGeneral and CasperRandom at the aggressive/non-adaptive table. CasperRandom was used as a baseline comparison for CasperGeneral.

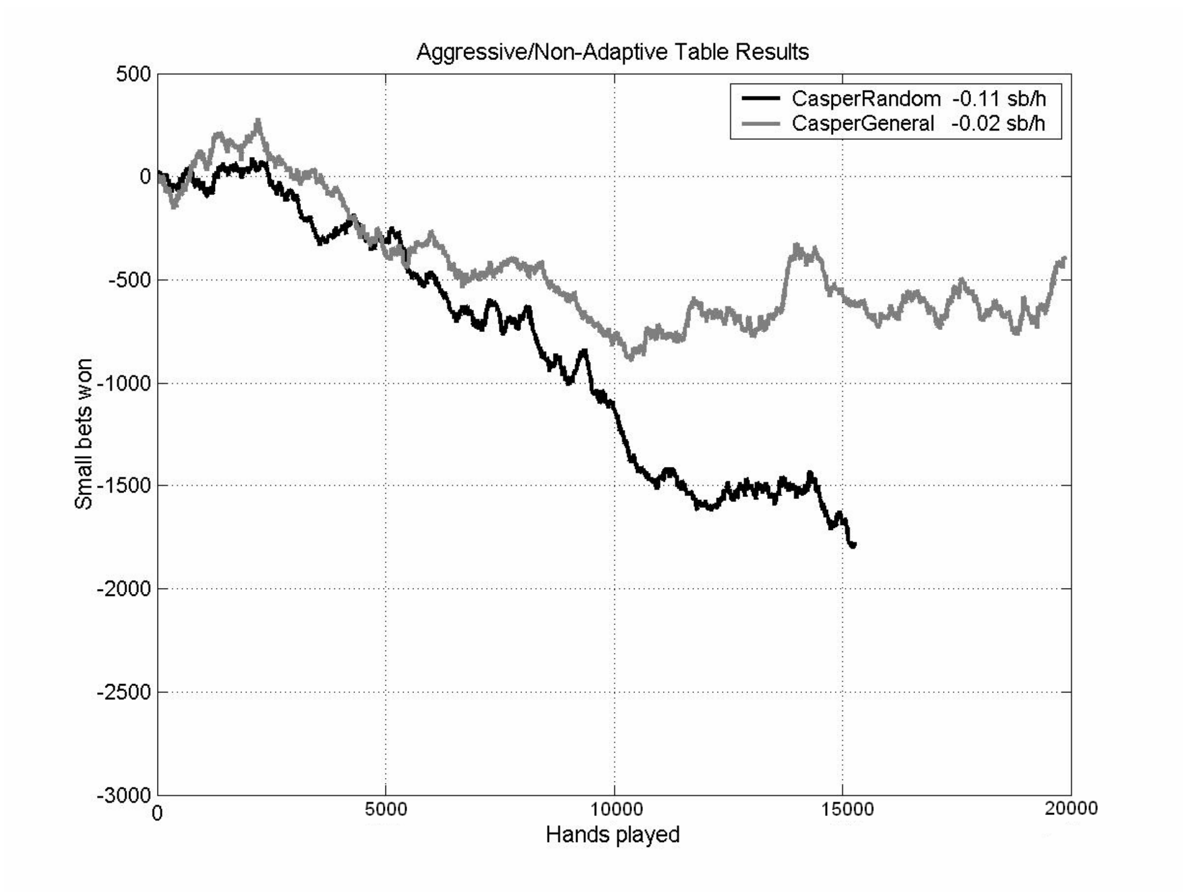


Figure 5.5: Results obtained at the Aggressive/Non-Adaptive Table.

Figure 5.5 shows that the derived set of general weights has not improved upon the performance of using the initial hand-picked weights. Recall from section 5.1.2 that using hand-picked weights Casper achieved a profit of +0.03 sb/h, whereas CasperGeneral now achieves a loss of -0.02 sb/h, or $0.20 lost for each hand played.

On the other hand it does appear that the evolutionary generated weights do actually perform better than randomly chosen weights. Using random weights Casper only managed to achieve -0.11 sb/h, or a loss of $1.10 for each hand played. Therefore, it would appear that while the evolutionary generated weights have not converged upon an optimal solution their performance is better than a random solution.

5.3 Real Opponents

Sections 5.1 and 5.2 above only discuss results obtained when Casper challenged various computerised opponents. However, Casper was also tested against real opponents on the internet. All instances of Casper now used the entire case-base to make their betting decisions. The results obtained are detailed below.

5.3.1 Play money

Casper was initially tested against real opponents by playing on the ‘play money’ tables of internet poker websites. Here players from all over the world can participate in a game of poker using a bankroll of play money. Initially all players begin with a starting bankroll of $1000. In the event that this bankroll is exhausted, a player can top their bankroll back up to a maximum of $1000. All games played at the ‘play money’ table were $10/$20 limit games. At each table a minimum of two players and a maximum of nine players could participate in a game of poker. Casper was tested by playing anywhere between one opponent all the way up to eight opponents. Figure 5.6 displays the results recorded at ‘play money’ tables for Casper (using hand picked weights) and CasperGeneral (using the derived general weights) as well as a random opponent which makes random decisions (used as a baseline comparison).

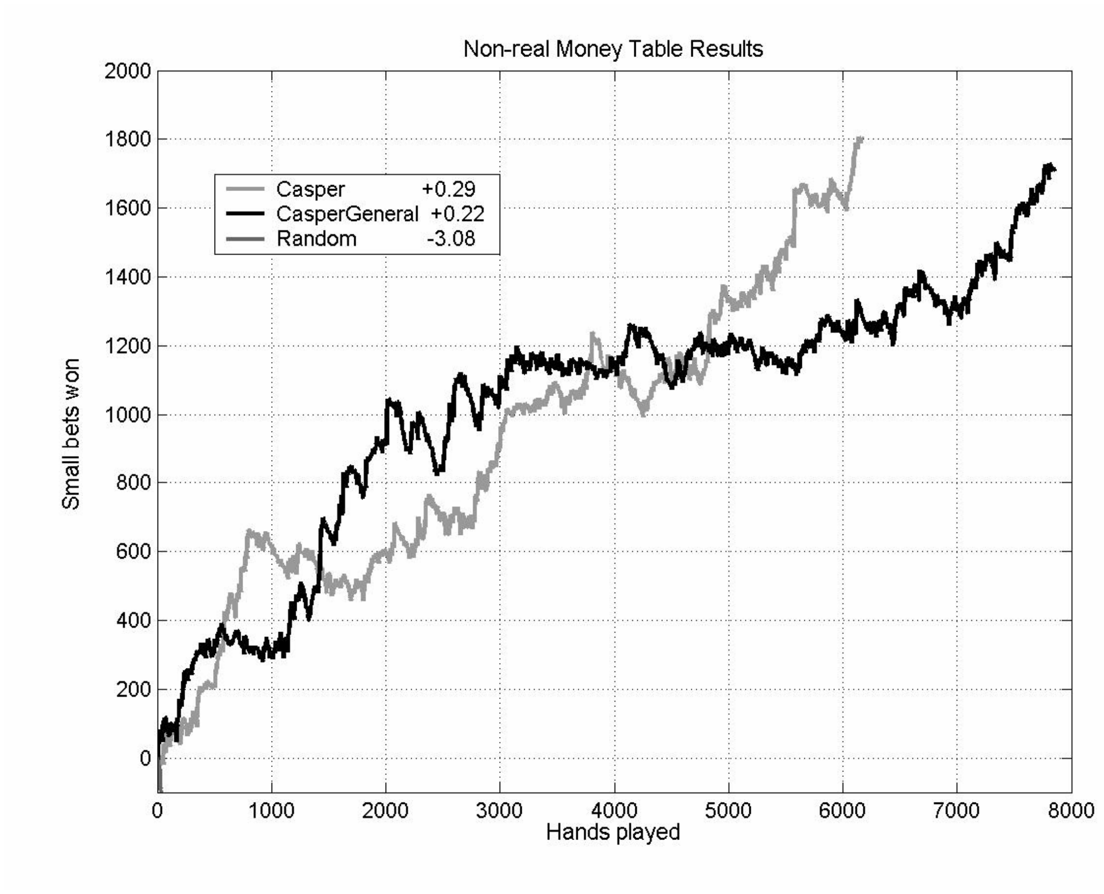


Figure 5.6: Casper vs. Real Opponents at the Play Money tables.

Both Casper and CasperGeneral earn consistent profit at the ‘play money’ tables. The results suggest that the use of Casper with hand picked weights outperforms CasperGeneral, which uses the evolutionary derived weights. Casper earns a profit of \$2.90 for every hand played, followed by CasperGeneral with a profit of \$2.20 for each hand. Random decisions resulted in exhausting the initial \$1000 bankroll in just over 30 hands, losing approximately -\$30.80 for each hand played.

Both *Pokibot* and *Simbot* have also been tested against real opponents by playing on Internet Relay Chat (IRC). The IRC server allows bots and humans to challenge each other online using “play money”. Results reported by (Davidson 2002) indicate that *Pokibot* achieves a profit of +0.22 sb/h, i.e. a profit of \$2.20 per hand, and *Simbot* achieves a profit of +0.19 sb/h or \$1.90 profit per hand. These results are very similar to those obtained by Casper, when challenging real opponents for play money. As Casper used *Pokibot* and *Simbots* playing style to build its casebase this result would be expected.

Because Casper was playing against real opponents the time taken to record the results is longer than when challenging computerised opponents. For this reason, fewer hands were able to be played against real opponents. Casper is also mainly suited to playing poker at a full table, i.e. with nine or ten players present, however the results recorded above consist of anywhere between two and nine players at a table. While we need to take caution in analysing the above results, it is safe to say that Casper is consistently profitable at the ‘play money’ tables.

5.3.2 Real money

Because there is normally a substantial difference in the type of play at the ‘play money’ tables compared to the ‘real money’ tables it was decided to attempt to get an idea of how Casper would perform using real money against real opponents. Only one instance of Casper was tested. Because using hand-picked weights had achieved the best performance at the ‘play money’ tables it was decided to test this instance of Casper at the ‘real money’ tables. The betting limit used was \$0.25/\$0.50, i.e. a small bet of \$0.25 and a big bet of \$0.50. The results are given in Figure 5.7.

Casper achieves a small bet per hand value of -0.07. Therefore, Casper now loses on average \$0.02 per hand. The results indicate that while Casper loses money very slowly it is now, nonetheless, unprofitable against this set of opponents. Due to the fact that real money was being used, much fewer hands were able to be recorded. Unfortunately, no results exist for *Pokibot* or *Simbot* challenging real opponents using real money. Therefore, it is not possible to evaluate how Casper performs using real money compared to *Pokibot* or *Simbot*.

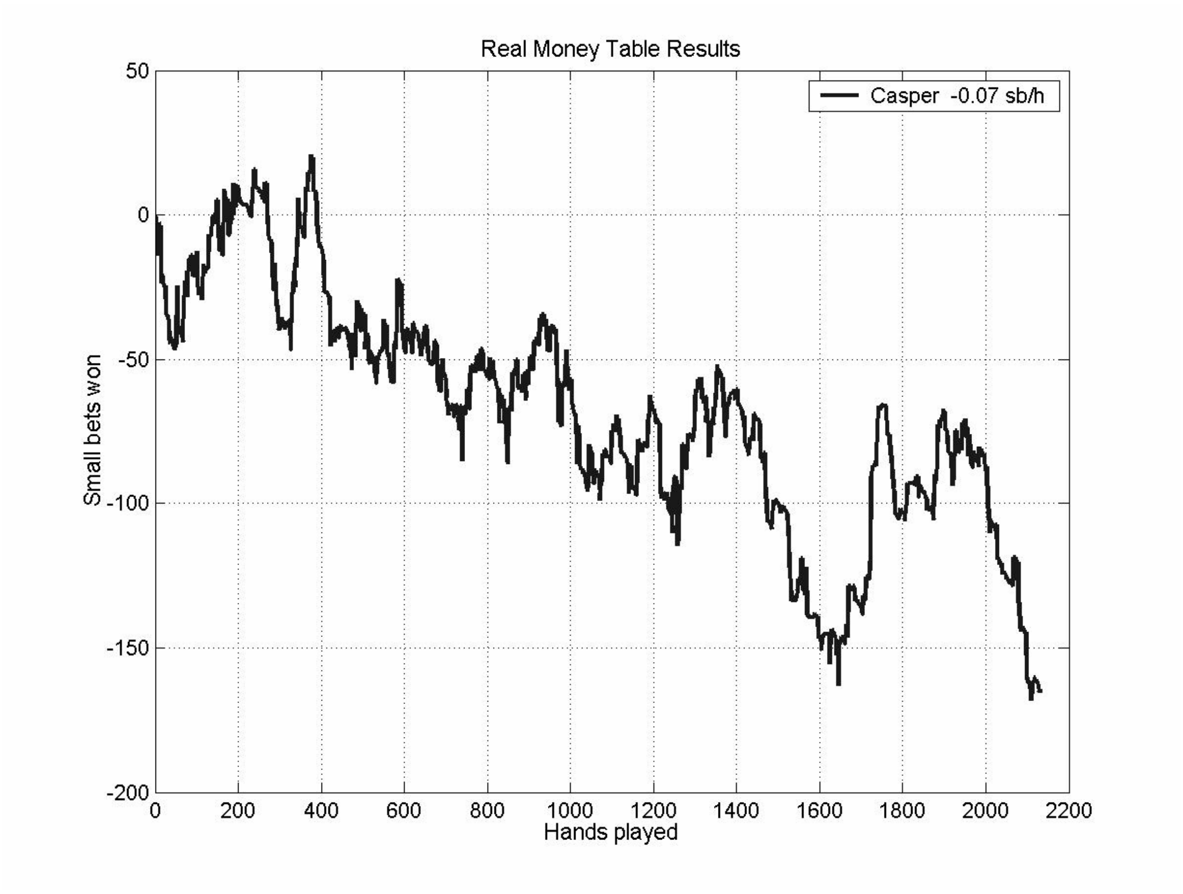


Figure 5.7: Casper vs. Real Opponents at the Real Money tables.

5.4 Case Similarity and Retrieval

From the results presented above it appears as though Casper's final outcome is quite different depending on the type of opponents being challenged. The strength of a particular set of opponents would most likely be the main factor that influences how Casper achieves against these opponents. Another factor that would affect Casper's overall results would be the particular playing style of the opposition. Casper may have encountered very dissimilar scenarios at the different tables that it played on because the playing style of the opponents is different.

For these reasons it was decided to investigate the effect that challenging separate sets of opponents had on Casper's case retrieval and similarity. Table 5.1 summarises the average number of retrieved cases used to make a betting decision and the average similarity value for these retrieved cases[14]. The average number of retrieved cases is shown in parenthesis beside the average similarity.

	Strong/Adaptive	***Aggressive/Non-Adaptive***	***Real Opponents Play Money***	***Real Opponents Real Money***
Preflop	0.98850364 *(92.4)*	0.98845772 *(90.1)*	0.976867 *(69.5)*	0.917778 *(46.4)*
Flop	0.98220273 *(94.7)*	0.98166614 *(85.3)*	0.979374 *(59.0)*	0.971213 *(44.4)*
Turn	0.97887567 *(89.3)*	0.97826279 *(88.7)*	0.971135 *(62.6)*	0.964151 *(49.2)*
River	0.98286971 *(93.2)*	0.98237579 *(90.2)*	0.982322 *(63.4)*	0.980000 *(69.0)*

Table 5.1: Average similarity and number of retrieved cases (shown in brackets) for the different sets of opponents that Casper challenged.

Table 5.1 indicates high similarity and retrieval rates at the Strong/Adaptive table. As Casper's casebase was constructed by recording instances of games played at this table this result is to be expected. While similarity and retrieval values drop slightly at the Aggressive/Non-Adaptive table they are, nonetheless, still quite high. At the Real Opponents – Play Money table average similarity is still over the similarity threshold value, but case retrieval has dropped considerably. Finally, at the real money table the number of retrieved cases has fallen again and for the *preflop* and *turn* stages average similarity is now below the similarity threshold.

Therefore, it appears as though the different playing style of the various sets of opponents does have an effect on Casper's decision making and therefore final result at that table. Take for example the *preflop* stage at the Real Money table. Casper now needs to make betting decisions based on fewer more dissimilar cases that it has stored in its casebase

Appendix D lists several sample target cases at each table as well as the number of retrieved cases and their average similarity.

[14] Recall that the similarity threshold was 97% and that at most 100 cases could be used for retrieval and when no cases exceeded the similarity threshold the top 20 cases were used regardless of their similarity value.

5.5 Results Summary

In summary, Casper manages to approximately breakeven when challenging the University of Alberta poker-bots and Casper achieves a profit comparable to *Pokibot* and *Simbot* when challenging real opponents using play money. As Casper used data obtained from *Pokibot* and *Simbot* to build its casebase these results would be expected. Casper was also profitable when challenging a separate set of aggressive, computerised opponents. However, Casper failed to make a profit when challenging real opponents using real money.

Results using the evolutionary generated weights were somewhat mixed. While a slight improvement was made using the opponent-specific weights, using the general weights usually resulted in degradation in performance compared to hand-picked values. However, all evolutionary generated weights performed better than the use of random weights.

Chapter 6

Conclusions

In the past little research has been presented on applying the tools and techniques of Case-Based Reasoning to the game of Texas Hold'em. One instance of a case based poker player was described in (Sandven and Tessem 2006) which used case-based learning to play poker. The results of this research indicated a lot of room for improvement. Other attempts at applying CBR to Poker have mainly focused on one particular aspect of the game e.g. (Salim and Rohwer 2005) which attempted to use CBR for opponent modelling purposes. This thesis has tried to add to and improve current research in the area of applying CBR principles to the game of Texas Hold'em.

A case-based poker player was developed, nicknamed Casper, that uses CBR to make all of it's betting decisions. The results of the Casper system suggest that it is possible to record instances of poker games played between strong players and then reuse these to obtain a similar performance. This approach bypasses the need for any initial, intensive knowledge engineering effort, such as that required for both *Pokibot* and *Simbot*.

A further outcome of this work was the investigation of improving feature weights by deriving weights algorithmically, rather than relying on hand-picked values. The results for this, reported in Chapter 5, were somewhat mixed. In total, two separate genetic algorithms were designed and implemented. The first focused on deriving a set of general weights that Casper could use against any type of competition. While the second algorithm focused on specifically tailoring Casper's weights for use against a specific set of opponents (in this case the University of Alberta *Pokibots*). The results represented suggest that using the Opponent-Specific weights slightly improved performance compared to using hand-picked weights (Figures 5.1 and 5.4). The overall results for the set of general opponent weights show that performance actually degraded compared to using hand-picked weights. This suggests that the algorithm used has not converged upon an 'optimal' solution. One possible reason for this may come down to the actual design of the genetic algorithm or the fact that more generations may have been needed. On the other hand, using algorithmically generated weights always outperformed the use of random weights (Figures 5.4 and 5.5). So, even though an optimal set of weights may not have been derived, it appears as though some improvement is taking place.

The results represented in this thesis show that by using a case-based approach Casper is able to play evenly against the University of Alberta poker-bots and profitably against other computerised opponents and real opponents using ‘play’ money. Casper was not, however, profitable against real opponents when using real money. In fact, a substantial difference was observed when Casper challenged real opponents using real money, compared to using play money. Casper makes consistent profit when playing with play money, however using real money Casper only manages to slowly lose. This difference is mainly thought to occur due to the change in opponent strength. It is believed that opponents play quite differently when real money is at stake as opposed to play money. In general, players at the real money table seem to normally only play their better hands. It was observed that when Casper won a pot its winnings would be a lot less at the real money table than at the play money table, indicating that players at the real money table had a better idea of the actual strength of Casper’s hand.

Another possible cause for the difference in performance may be because of an overall decrease in similarity of retrieved cases at the real money tables. As an example, when examining similarity for the preflop case-base it was discovered that against computerised opponents Casper does not exceed the similarity threshold (97%) for approximately 2% of total target cases. However, when compared to real opponents using real money this amount increases to approximately 37% of total target cases. Hence, there is a dramatic increase in the number of times that Casper fails to retrieve similar cases during the preflop stage at the real money tables, resulting in Casper having to make betting decisions based on less similar cases. The reason for this drop in similarity may be because the case-base is not complete and fails to describe many of the scenarios that can occur at the poker table.

Finally, it should be noted that due to the high variance of the game of Texas Hold’em it is normally common for at least 10,000 hands to be played to get an accurate indication of performance. However, this was not possible for the results obtained at the real money table. In total, approximately 2200 hands were played using real money. So any results reported from this data needs to be interpreted with caution.

6.1 Future Work

While Casper does well using play money against both computerised and real opponents, it is evident that improvements need to be made for Casper to be profitable when real money is at stake. Listed below are a few improvements that may aid in this goal.

- Recall that Casper's case-base is derived by recording actions observed during play between both *Pokibot* and *Simbot*. While these two poker-bots have proven to be profitable in the past (once again using play money), they are far from expert players. If expert data were available it would most likely improve Casper's case-base and hence its level of play. At the present there is no known source for this expert data, however another option may be to build Casper's case-base by recording actions observed at real money tables. It seems likely that by recording this data and then playing at these same tables would result in an improvement in performance compared to using data obtained at separate poker tables. A problem with this approach is that the *hole cards* of a player need to be known before a case can be entered into the case-base. Therefore, it would only be possible to record a case when a particular hand has reached a showdown and the *hole cards* have been revealed. This would drastically slow down the generation of the case-base.
- At the present Casper simply reuses the cases stored in its case-base. A possible improvement would be to allow Casper to learn by recording and storing new cases observed during play. This approach may allow Casper to adapt better to separate poker tables where the type of opponents may vary.
- Casper's betting decision is currently made by recording all the actions for all the retrieved cases which exceed a similarity threshold of 97%. This strategy simply reuses these decisions and doesn't consider the actual outcome of each decision. Therefore, at present Casper is simply making the most popular decision recorded by *Pokibot* and *Simbot*. It

seems a more beneficial approach would be to actually allow Casper to evaluate the outcome of these decisions (based on losses incurred or profits made) and then choose the most effective action.

- Finally, Casper currently has no opponent modelling capabilities. A key skill in the game of poker is being able to read your opponents and attempt to determine the actual strength of their hand. Adding the ability to infer the hand strength of a particular opponent based on how that opponent has been playing in the past would most likely improve Casper’s performance at the poker table.

Appendix A

Investigating the Effectiveness of Applying Case-Based Reasoning to the game of Texas Hold'em

The following paper was published in the proceedings of the 20th Florida Artificial Intelligence Research Society Conference (FLAIRS), Key West, Florida, May 2007. AAAI Press.

http://www.cise.ufl.edu/~ddd/FLAIRS/flairs2007/

Investigating the Effectiveness of Applying Case-Based Reasoning to the game of Texas Hold'em

Jonathan Rubin[1] and Ian Watson[2]

Department of Computer Science, University of Auckland, New Zealand
[1]jrub001@ec.auckland.ac.nz
[2]ian@cs.auckland.ac.nz

Abstract

This paper investigates the use of the case-based reasoning methodology applied to the game of Texas hold'em. The development of a CASe-based Poker playER (CASPER) is discussed. CASPER uses knowledge of previous poker scenarios to inform a betting decision. CASPER improves upon previous case-based reasoning approaches to poker and is able to play evenly against the University of Alberta's Pokibots and Simbots and profitably against other competition.

1. Introduction

The game of poker provides an interesting environment to investigate how to handle uncertain knowledge and issues such as dealing with chance and deception in a hostile environment. Games in general offer a well suited domain for investigation and experimentation due to the fact that a game is usually composed of several well defined rules which players must adhere to. Most games have precise goals and objectives which players must meet to succeed. For a large majority of games the rules imposed are quite simple, yet the game play itself involves a large number of very complex strategies. Success can easily be measured by factors such as the amount of games won, the ability to beat certain opponents or, as in the game of poker, the amount of money won.

Up until recently AI research has mainly focused on games such as chess, checkers and backgammon. These are examples of games which contain *perfect information*. The entire state of the game is accessible by both players at any point in the game, e.g. both players can look down upon the board and see all the information they need to make their playing decisions. These types of games have achieved their success through the use of fast hardware processing speeds, selective search and effective evaluation functions (Schaeffer, Culberson et al. 1992).

Games such as poker on the other hand are classified as *stochastic*, *imperfect information* games. The game involves elements of chance (the actual cards which are dealt) and hidden information in the form of other player's *hole cards* (cards which only they can see). This ensures that players now need to make decisions with uncertain information present.

The focus of this paper is to investigate the application of CBR to the game of poker. We have developed a poker playing robot, nicknamed CASPER (CASe-based Poker playER), that attempts to use knowledge about past poker experiences to make betting decisions. CASPER plays the variation of the game known as limit Texas Hold'em and has been tested against other poker bots.

The remainder of this paper is structured as follows, section two will detail related previous research, section three gives a brief introduction to the game of Texas hold'em. Sections four, five and six describe the design and implementation of CASPER. This is followed by the experimental results obtained (section seven) and a conclusion and discussion of future work in section eight.

2. Related Work

Over the last few years there has been a dramatic increase in the popularity of the game of Texas hold'em. This growing popularity has also sparked an interest in the AI community with increased attempts to construct poker robots (or bots), i.e. computerised poker players who play the game based on various algorithms or heuristics. Recent approaches to poker research can be classified into three broad categories:

Heuristic rule-based systems: which use various pieces of information, such as the cards a player holds and the amount of money being wagered, to inform a betting strategy.
Simulation/Enumeration-based approaches: which consist of playing out many scenarios from a certain point in the hand and obtaining the *expected value* of different decisions.
Game-theoretic solutions: which attempt to produce optimal strategies by constructing the game tree in which game states are represented as nodes and an agents possible decisions are represented as arcs.

* With special thanks to the University of Alberta Computer Poker Research Group.

The University of Alberta Computer Poker Research Group[1] are currently leading the way with poker related research, having investigated all of the above approaches. Perhaps the most well known outcome of their efforts are the poker bots nicknamed *Loki/Poki* (Schaeffer, Billings et al. 1999; Billings, Davidson et al. 2002).

Loki originally used expert defined rules to inform a betting decision. While expert defined rule-based systems can produce poker programs of reasonable quality (Billings, Davidson et al. 2002), various limitations are also present. As with any knowledge-based system a domain expert is required to provide the rules for the system. In a strategically complex game such as Texas hold'em it becomes impossible to write rules for all the scenarios which can occur. Moreover, given the dynamic, nondeterministic structure of the game any rigid rule-based system is unable to exploit weak opposition and is likely to be exploited by any opposition with a reasonable degree of strength. Finally, any additions to a rule-based system of moderate size become difficult to implement and test (Billings, Peña et al. 1999).

Loki was later rewritten and renamed *Poki* (Davidson 2002). A simulation-based betting strategy was developed which consisted of playing out many scenarios from a certain point in the hand and obtaining the *expected value* (EV) of different decisions. A simulation-based betting strategy is analogous to selective search in perfect information games.

Both rule-based and simulation-based versions of *Poki* have been tested by playing real opponents on an IRC poker server. *Poki* played in both low limit and higher limit games. *Poki* was a consistent winner in the lower limit games and also performed well in the higher limit games where it faced tougher opposition (Billings, Davidson et al. 2002). More recently the use of game theory has been investigated in the construction of a poker playing bot. The University of Alberta Computer Poker Research Group have attempted to apply game-theoretic analysis to full-scale, two-player poker. The result is a poker bot known as *PsOpti* that is:

> able to defeat strong human players and be competitive against world-class opponents (Billings, Burch et al. 2003).

There have also been numerous other contributions to poker research outside the University of Alberta Poker Research Group. Sklansky and Malmuth have detailed various heuristics for different stages of play in the game of Texas hold'em (Sklansky 1994; Sklansky and Malmuth 1994). The purpose of these rules, however, has been to guide human players who are looking to improve their game rather than the construction of a computerised expert system. (Korb, Nicholson et al. 1999) have produced a Bayesian Poker Program (BPP) which makes use of Bayesian networks to play five-card stud poker. (Dahl 2001) investigated the use of reinforcement learning for neural net-based agents playing a simplified version of Texas hold'em.

Finally, we have encountered relatively few attempts to apply the principles and techniques of CBR to the game of poker. (Sandven and Tessem 2006) constructed a case-based learner for Texas hold'em which they nicknamed Casey. Casey began with no poker knowledge and builds up a case-base for all hands that it plays. Sandven and Tessem report that Casey plays on a par with a simple rule-based system against three opponents, but loses when it faces more opponents. (Salim and Rohwer 2005) have attempted to apply CBR to the area of opponent modeling, i.e. trying to predict the hand strength of an opponent given how that opponent has been observed playing in the past. While CBR seems inherently suited to this particular type of task they report better performance by simply relying on long-term averages.

3. Texas Hold'em

Texas hold'em is the variation used to determine the annual World Champion at the World Series of Poker. This version of the game is the most strategically complex and provides a better skill-to-luck ratio than other versions of the game (Sklansky 1994).

The game of Texas hold'em is played in four stages, these include the *preflop*, *flop*, *turn* and the *river*. During each round all active players need to make a betting decision. Each betting decision is summarised below:

Fold: A player discards their hand and contributes no money to the pot. Once a player *folds* they are no longer involved in the current hand, but can still participate in any future hands.

Check/Call: A player contributes the least amount possible to stay in the hand. A *check* means that the player invests nothing, whereas a *call* means the player invests the least amount required greater than $0.

Bet/Raise: A player can invest their own money to the pot over and above what is needed to stay in the current round. If the player is able to *check,* but they decide to add money to the pot this is called a *bet*. If a player is able to *call*, but decides to add more money to the pot this is called a *raise*.

All betting is controlled by two imposed limits known as the small bet and the big bet. For example, in a $10/$20 game the small bet is $10 and all betting that occurs during the *preflop* and the *flop* are in increments of the small bet. During the *turn* and the *river* all betting is in increments of the big bet, $20. The number of bets made within each stage of the game is capped at a maximum of 5. All results detailed in this paper refer to a $10/$20 limit game[2]. Before the hand begins two forced bets are made, known as the *small blind* (half the small bet) and the *big blind* (one full small bet), to ensure that there is something in the

[1] http://www.cs.ualberta.ca/~games/poker/

[2] In no limit there is no restriction on the amount a player can bet.

pot to begin with. Each of the four game stages are summarised below:

Preflop: The game of Texas hold'em begins with each player being dealt two *hole cards* which only they can see. A round of betting occurs. Once a player has made their decision play continues in a clockwise fashion round the table. As long as there are at least two players left then play continues to the next stage. During any stage of the game if all players, except one, fold their hand then the player who did not fold their hand wins the pot and the hand is over.
Flop: Once the *preflop* betting has completed three community cards are dealt. Community cards are shared by all players at the table. Players use their hole cards along with the community cards to make their best hand. Another round of betting occurs.
Turn: The *turn* involves the drawing of one more community card. Once again players use any combination of their *hole cards* and the community cards to make their best hand. Another round of betting occurs and as long as there are at least two players left then play continues to the final stage.
River: During the *river* the final community card is dealt proceeded by a final round of betting. If at least two players are still active in the hand a *showdown* occurs in which all players reveal their *hole cards* and the player with the highest ranking hand wins the entire pot (in the event that more than one player holds the winning hand then the pot is split evenly between these players).

4. Casper System Overview

CASPER uses CBR to make a betting decision. This means that when it is CASPER's turn to act he evaluates the current state of the game and constructs a *target case* to represent this information. A target case is composed of a number of features. These features record important game information such as CASPER's hand strength, how many opponents are in the pot, how many opponents still need to act and how much money is in the pot. Once a target case has been constructed CASPER then consults his case-base (i.e. his knowledge of past poker experiences) to try and find similar scenarios which may have been encountered. CASPER's case-base is made up of a collection of cases composed of their own feature values and the action which was taken, i.e. fold, check/call or bet/raise. CASPER uses the k-nearest neighbour algorithm to search the case-base and find the most relevant cases, these are then used to decide what action should be taken.

Casper was implemented using the commercially available product Poker Academy Pro 2.5[3] and the Meerkat API. The University of Alberta Poker Research Group provides various poker bots with the software including instantiations of *Pokibot* and the simulation based bot *Simbot*. Both *Pokibot* and *Simbot* are the result of an intensive knowledge engineering process. These poker bots have been used to generate the training data for CASPER. Approximately 7000 hands were played between various poker bots and each decision witnessed was recorded as a single case (or experience) in CASPER's case-base. Both bots have proven to be profitable against human competition in the past (Davidson 2002) so it is believed that the data obtained is of greater quality then it might be from other sources, such as free money games on the internet composed of real players. CASPER then reuses these recorded instances to make decisions at the poker table and therefore bypasses the intensive knowledge engineering effort required of other poker-bots.

5. Case Features

CASPER searches a different case-base for each separate stage of a poker hand (i.e. *preflop*, *flop*, *turn* and *river*). The features that make up a case and describe the state of the game at a particular time are listed and explained in Table 1. The features listed were chosen by the authors because they are believed to capture important information needed to make a betting decision. These are the indexed features, which means that they are believed to be predictive of a case's outcome and by computing local similarity for each feature they are used to retrieve the most similar cases in the case-base. The first eight features are used in all case-bases, whereas the last four features are only used during the postflop stages. Each case is also composed of one outcome, which is the betting decision that was made.

The 'hand strength' feature, listed in table 1, differs somewhat for *preflop* and *postflop* stages of the game. During the *preflop* there exists 169 distinct card groups that a player could be dealt. These card groups were ordered from 1 to 169 based on their hand ranking, where 1 indicates pocket Aces (the best *preflop* hand) and 169 indicates a 2 and a 7 of different suits (the worst *preflop* hand). *Preflop* hand strength was then based on this ordering, whereas *postflop* hand strength refers to a calculation of immediate hand strength based on the *hole cards* a player has and the current *community cards* that are present. This value is calculated by enumerating all possible *hole cards* for a single opponent and recording how many of these hands are better, worse or equal to the current player's hand. *Positive potential* and *negative potential* are also calculated in this way with the addition that all possible future community cards are considered as well. For more details on hand strength and potential consult (Billings, Davidson et al. 2002; Davidson 2002).

6. Case Retrieval

Once a target case has been constructed CASPER needs to locate and retrieve the most similar cases it has stored in its case-base. The k-nearest neighbour algorithm is used to compute a similarity value for all cases in the case-base.

[3] http://www.poker-academy.com/

Each feature has a local similarity metric associated with it that evaluates how similar its value is to a separate case's value, where 1.0 indicates an exact match and 0.0 indicates entirely dissimilar.

Two separate similarity metrics are used depending on the type of feature. The first is the standard Euclidean distance function given by the following equation:

$$s_i = 1 - \left(\frac{|x_1 - x_2|}{MAX\_DIFF} \right) \qquad (1)$$

where x_1 refers to the target value, x_2 refers to the case value and *MAX_DIFF* is the greatest difference in values, given by the range in table 1.

The above Euclidean similarity metric produces smooth, continuous changes in similarity, however, for some features, minor differences in their values produce major changes in actual similarity, e.g. the 'Bets to call' feature. For this reason an exponential decay function, given by equation (2), has also been used for some features:

$$s_i = e^{-k\,(|x_1 - x_2|)}, \qquad (2)$$

where, x_1 refers to the target value and x_2 refers to the source value and k is a constant that controls the rate of decay.

Feature:	Type:	Range:	Explanation:
Number of players	int	2 - 10	Number of active players at the beginning of the round (preflop, flop, turn or river).
Relative position	double	0.0 - 1.0	What order the player acts relative to other players at the table. 0.0 means the player is first to act in the round, 1.0 means the player is last to act.
Players in current hand	int	0 - 9	The number of players that have already acted and are still currently in the hand, i.e. players that have checked, bet, called or raised.
Players yet to act	int	0 - 9	The number of players that still need to make a future betting decision.
Bets committed	double	0.0 - 5.0	A multiple of the current bet size a certain player has committed to the pot. Small bets are used during the preflop and flop and big bets are used during the turn and river.
Bets to call	double	0.0 - 5.0	A multiple of the current bet size a certain player has to commit to the pot to stay in the hand. Small bets are used during the preflop and flop and big bets are used during the turn and river.
Pot Odds	double	0.0 - 0.5	The amount to call divided by the amount currently in the pot plus the amount needing to be called, a risk/reward measure.
Hand strength	double	0.0 - 1.0	A numerical measure of the strength of a player's hand. 0.0 represents the worst possible hand whereas 1.0 represents an unbeatable hand ("the nuts").
Positive potential[4]	double	0.0 - ~0.40	A numerical measure which represents the chance that a player who does not currently hold the best hand will improve to the best hand after future community cards are dealt.
Negative potential[4]	double	0.0 - ~0.30	A numerical measure which represents the chance that a player currently holding the best hand no longer holds the best hand after future community cards are dealt.
Small bets in pot	double	0.0 - ~300.0	The total amount in the pot divided by the value of the small bet size.
Previous round bets	int	0 - 5	How many bets or raises occurred during the previous betting round.
Action	char	{f, k, c, b, r}	A character representing the decision which was made. f = fold, k = check, c = call, b = bet, r = raise.

Table 1. *Preflop* and *postflop* case features.

[4] Not used during the *river* as there are no further betting rounds.

Global similarity is computed as a weighted linear combination of local similarity, where higher weights are given to features that refer to a player's hand strength as well as positive and negative potential. All weights were hand picked by the authors and fell in the range of 0 – 100. A default value of 5 was used for most features, while features we felt were more salient, such as 'hand strength' and 'positive and negative potential', were assigned values in the approximate range of 50 - 80. The following equation is used to compute the final similarity value for each case in the case-base:

$$\sum_{i=1}^{n} w_i x_i \Big/ \sum_{i=1}^{n} w_i \ , \qquad (3)$$

where x_i refers to the *i*th local similarity metric in the range 0.0 to 1.0 and w_i is its associated weight, in the range 0 – 100.

After computing a similarity value for each case in the case-base a descending quick sort of all cases is performed. The actions of all cases which exceed a similarity threshold of 97% are recorded. Each action is summed and then divided by the total number of similar cases which results in the construction of a probability triple *(f, c, r)* which gives the probability of folding, checking/calling or betting/raising in the current situation. If no cases exceed the similarity threshold then the top 20 similar cases are used. As an example, assume CASPER looks at his *hole cards* and sees **A♥-A♠**. After a search of his *preflop* case-base the following probability triple is generated: *(0.0, 0.1, 0.9).* This indicates that given the current situation CASPER should never fold this hand, he should just call the small bet 10% of the time and he should raise 90% of the time. A betting decision is then probabilistically made using the triple which was generated.

7. Results

CASPER was evaluated by playing other poker bots provided through the commercial software product Poker Academy Pro 2.5. CASPER was tested at two separate poker tables. The first table consisted of strong, adaptive poker bots that model their opponents and try to exploit weaknesses. As CASPER has no adaptive qualities of his own he was also tested against non-adaptive, but loose/aggressive opponents. A loose opponent is one that plays a lot more hands, whereas aggressive means that they tend to bet and raise more than other players. All games were $10/$20 limit games which consisted of 9 players. All players began with a starting bankroll of $100,000.

The adaptive table consisted of different versions of the University of Alberta's poker bots: Pokibot and Simbot. Figure 1 records the amount of small bets won at the adaptive table over a period of approximately 20,000 hands. Two separate versions of CASPER were tested separately against the same competition. Casper02 improves upon Casper01 by using a larger case-base, generated from approximately 13,000 poker hands. A poker bot that makes totally random betting decisions was also tested separately against the same opponents as a baseline comparison.

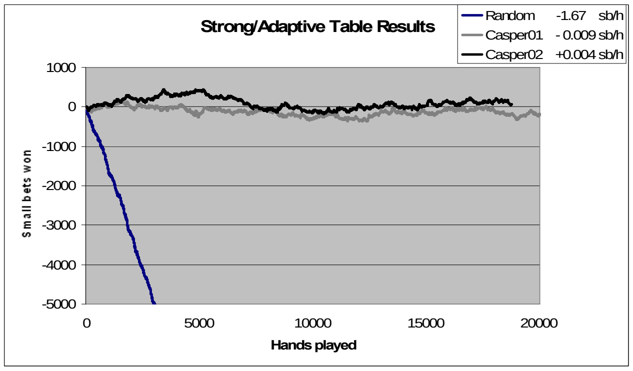


Figure 1: Casper's results at the adaptive table.

While Casper01 concludes with a slight loss and Casper02 concludes with a slight profit, figure 1 suggests that both versions approximately break even against strong competition, whereas the random player exhausted its bankroll of $100,000 after approximately 6,000 hands[5]. Casper01's small bets per hand (sb/h) value is -0.009 which indicates that Casper01 loses about $0.09 with every hand played. Casper02 slightly improves upon this by winning approximately $0.04 for every hand played. Random play against the same opponents produces a loss of $16.70 for every hand played.

The second table consisted of different versions of Jagbot, a non-adaptive, loose/aggressive rule-based player. Figure 2 records the amount of small bets won over a period of approximately 20,000 hands. Once again a bot which makes random decisions was also tested separately against the same competition as a baseline comparison for CASPER.

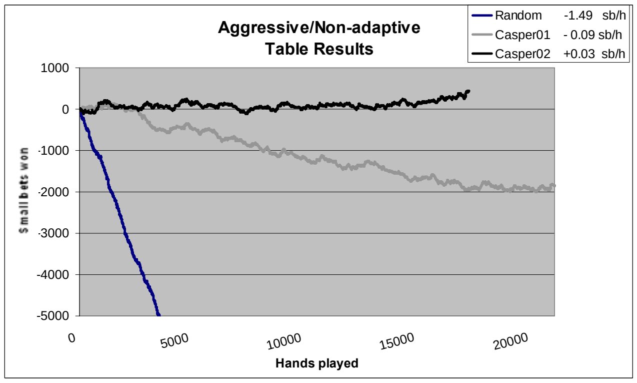


Figure 2: Casper's results at the non-adaptive table.

[5] Not all data points are shown to improve clarity.

Figure 2 indicates that the first version of Casper is unprofitable against the non-adaptive players, losing approximately $0.90 for each hand played. Casper02 shows a considerable improvement in performance compared to Casper01. With more cases added to the case-base, Casper02 produces a profit of +0.03 sb/h, or $0.30 for each hand played. Once again the random player exhausted its initial bankroll after approximately 7000 hands, losing on average $14.90 for each hand played.

8. Conclusions and Future Work

In conclusion, CASPER, a case-based poker player has been developed that plays evenly against strong, adaptive competition and plays profitably against non-adaptive opponents. The results suggest that it is possible to record instances of games from strong players and reuse these to obtain similar performance without the need for an intensive knowledge engineering effort.

Two separate versions of CASPER were tested and the addition of extra cases to the case-base was shown to result in improvements in overall performance. It is interesting to note that CASPER was initially unprofitable against the non-adaptive, aggressive opposition. One possible reason for this is that as CASPER was trained using data from players at the adaptive table it perhaps makes sense that they would play evenly, whereas players at the non-adaptive table tend to play much more loosely and aggressively. This means that while CASPER has extensive knowledge about the type of scenarios that often occur at the advanced table, this knowledge is weaker at the non-adaptive table as CASPER runs into situations which he is not familiar with. For example, for a random sample of 10,000 hands there were 86 occurrences where Casper01 didn't exceed the similarity threshold during the *preflop* stage at the adaptive table, whereas this figure increases to 222 occurrences at the non-adaptive table. CASPER must therefore make a betting decision at the non-adaptive table using less similar scenarios. With the addition of extra cases these values drop to 38 and 45 for the *preflop* stage at the adaptive and non-adaptive table respectively.

While CASPER does achieve slight profits against various opposition, it appears as though there are still many improvements necessary to ensure that CASPER improves its profitability. Some areas of possible future work are:

- Investigating optimal feature weightings using a genetic algorithm and self play experiments.
- Further investigation of CASPER's level of play based on the size of its case-base.
- Improving the case-representation by considering the actual outcome that occurred rather than simply the action that was taken.
- Testing CASPER against real opponents.
- Adding opponent modeling capabilities.

References

Billings, D., N. Burch, et al. 2003. *Approximating game-theoretic optimal strategies for full-scale poker*. Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence.

Billings, D., A. Davidson, et al. 2002. *The challenge of poker*. Artificial Intelligence **134**(1-2): 201-240.

Billings, D., L. Peña, et al. 1999. *Using probabilistic knowledge and simulation to play poker*. Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence: 697-703.

Dahl, F. A. 2001. *A Reinforcement Learning Algorithm Applied to Simplified Two-Player Texas Hold'em Poker*. Proceedings of the 12th European Conference on Machine Learning Springer-Verlag.

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

Korb, K. B., A. E. Nicholson, et al. 1999. *Bayesian poker*. UAI'99 - Proceedings of the 15th International Conference on Uncertainty in Artificial Intelligence, Sweden: 343-350.

Salim, M. and P. Rohwer 2005. *Poker Opponent Modeling*. Indiana University**:** Personal communication.

Riesbeck, C. and R. Schank 1989. *Inside Case-Based Reasoning*. Hillsdale, NJ.: Lawrence Erlbaum.

Sandven, A. and B. Tessem 2006. *A Case-Based Learner for Poker*. The Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006), Helsinki, Finland.

Schaeffer, J., D. Billings, et al. 1999. *Learning to play strong poker*. Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing.

Schaeffer, J., J. Culberson, et al. 1992. *A world championship caliber checkers program*. Artificial Intelligence **53**(2-3): 273-289.

Sklansky, D. 1994. *The Theory of Poker*. Las Vegas, NV.: Two Plus Two Publishing.

Sklansky, D. and M. Malmuth 1994. *Hold'em Poker for Advanced Players*. Las Vegas, NV.: Two Plus Two Publishing.

Watson, I. 1997. *Applying case-based reasoning: techniques for enterprise systems*. San Francisco, CA.: Morgan Kaufmann Publishers Inc.

Appendix B

Preflop Hand Rankings

The following list was used to determine a ‘hand ranking’ feature value during preflop play. It lists the preflop hand rankings from highest to lowest. A preflop hand is made up of two cards and is described by three characters. The first two characters consist of each card’s rank. This is followed by either an ‘o’ for offsuit (meaning both cards are of different suits) or an ‘s’ for suited (meaning both cards have the same suit).

1. AAo
2. KKo
3. QQo
4. JJo
5. KAs
6. QAs
7. TTo
8. QKs
9. JAs
10. KAo
11. JKs
12. TAs
13. JQs
14. TKs
15. QAo
16. 99o
17. TQs
18. TJs
19. QKo
20. 9As
21. JAo
22. 9Ks
23. 88o
24. JKo
25. 8As
26. 9Qs
27. 9Ts
28. JQo
29. 9Js
30. TAo
31. 7As
32. 5As
33. TKo
34. 4As
35. 6As
36. TQo
37. 3As
38. 8Ks
39. TJo
40. 77o
41. 2As
42. 8Ts
43. 8Qs
44. 7Ks
45. 8Js
46. 89s
47. 6Ks
48. 66o
49. 5Ks
50. 9Ao
51. 78s
52. 4Ks
53. 7Qs
54. 7Ts
55. 79s
56. 9Ko
57. 7Js
58. 3Ks
59. 9To
60. 6Qs
61. 8Ao
62. 55o
63. 9Jo
64. 9Qo
65. 2Ks
66. 67s
67. 5Qs
68. 68s
69. 4Qs
70. 7Ao
71. 44o
72. 56s
73. 69s
74. 5Ao
75. 6Ts
76. 6Js
77. 3Qs
78. 2Qs
79. 57s
80. 4Ao
81. 45s
82. 5Js
83. 6Ao
84. 8Ko
85. 33o
86. 8To
87. 58s
88. 4Js
89. 3Ao
90. 89o

91. 22o
92. 46s
93. 8Jo
94. 8Qo
95. 3Js
96. 59s
97. 7Ko
98. 5Ts
99. 2Js
100. 35s
101. 2Ao
102. 4Ts
103. 47s
104. 3Ts
105. 34s
106. 6Ko
107. 48s
108. 78o
109. 2Ts
110. 36s
111. 5Ko
112. 79o
113. 49s
114. 7To
115. 25s
116. 7Qo
117. 39s
118. 7Jo
119. 4Ko
120. 37s
121. 29s
122. 24s
123. 67o
124. 3Ko
125. 38s
126. 6Qo
127. 26s
128. 68o
129. 23s
130. 28s
131. 2Ko
132. 56o
133. 5Qo
134. 69o
135. 27s
136. 6To
137. 4Qo
138. 45o
139. 57o
140. 6Jo
141. 3Qo
142. 5Jo
143. 58o
144. 46o
145. 2Qo
146. 4Jo
147. 59o
148. 35o
149. 3Jo
150. 5To
151. 47o
152. 4To
153. 2Jo
154. 34o
155. 3To
156. 48o
157. 36o
158. 2To
159. 49o
160. 25o
161. 39o
162. 37o
163. 24o
164. 29o
165. 38o
166. 26o
167. 23o
168. 28o
169. 27o

Appendix C

Glossary of Poker Terms

The following is a brief glossary of the poker terms used throughout this thesis. For a more detailed list of poker terms consult:

http://www.pokertips.org/glossary/glossary.php

Or,

http://www.worldseriesofpoker.com/learn/terms.asp

5-card draw: A poker variation where each player is given five cards which they keep hidden and no community cards are dealt. Players have one chance to replace any number of their five cards with cards from the undealt portion of the deck.

Ante: Each player is forced to contribute a certain amount of chips/cash to the pot.

Bet: When a player bets they willingly add chips/cash to the current pot total.

Big Blind: A forced bet made by one player each round to ensure there is something in the pot to play for. The big blind amount is equal to the amount of one small bet.

Blinds: A general term that covers the big and small blinds, which are forced bets.

Call: When a player calls a bet they invest the least amount possible to stay in the current hand.

Cash game: A game where players wager with their own funds during each hand, as opposed to a tournament where players wager with chips that don't represent their actual cash value. The blind values don’t change and are usually much lower than the average stack of each player. Players may leave the table any time with their winnings/losses.

Check: When a player checks they do not invest any funds into the pot, but they are still active in the hand.

Chip stack: The amount of chips/cash that a certain player has to wager with at the poker table.

Chips: Tokens that represent monetary value. The usual form of currency used for betting at the poker table.

Community cards: Cards which are visible to all and all active players share to make their best poker hand.

Dealer: During each round one player takes on the status of the dealer. The position of the dealer controls the betting order around the table. The dealer is the last player to act for all *postflop* stages in Texas Hold’em.

Flop: The flop is the second stage in the game of Texas Hold’em. During the flop three community cards are dealt face up on the table and players combine these cards with their two hole cards to make their best five hand combination. This is then followed by a round of betting.

Fold: When a player folds they discard their hand and are no longer active in the round.

Heads up: A term used to describe a game of poker where there are only two players.

Hole cards: A player’s private cards which only they can see.

Limit: A term used to indicate that betting is limited to certain amounts.

No Limit: In no limit there are no betting restrictions. A player can bet up to the total amount they have in their current chip stack.

Postflop: In Texas Hold'em the game is broken up into four stages: the *preflop*, *flop*, *turn* and *river*. *Postflop* refers to the three stages after the *preflop*, where community cards are drawn during each stage.

Pot: The total amount of chips/cash that is being contested by each active player.

Preflop: The first stage in the game of Texas Hold'em where each player is dealt two *hole cards* and a round of betting occurs.

Raise: When a player raises they invest over and above what is required to stay in the hand. All other active players must now match the amount that was raised to continue to play.

Ring game: see *Cash game*

River: The final stage in Texas Hold'em where one final community card is drawn followed by a final round of betting.

Showdown: After all betting has ceased all players that are still active in the hand reveal their private cards and the player with the highest ranking hand wins the entire contents of the pot. If players have equal ranking hands then the pot is split between those players.

Small blind: A forced bet made by one player each round to ensure there is something in the pot to play for. The small blind amount is equal to half the amount of one small bet.

Stack: See *chip stack*.

Turn: The third stage in the game of Texas Hold'em where one community card is dealt followed by a round of betting.

Appendix D
Sample Target Cases

This appendix provides a collection of target cases that were constructed for various hands played by Casper against different types of opponents. Each case lists all indexed features and their values as well as the probability triple that was constructed, the action taken, the number of cases retrieved and the average similarity of the retrieved cases.

D.1 Computerised Opponents

D.1.1 Strong/Adaptive Competition

Stage: PREFLOP	
Number of players:	9
Relative position:	1.0
Players in pot:	2
Players to act:	3
Small bets committed:	0.0
Small bets to call:	3.0
Pot Odds:	0.315789474
Hand ranking:	106 (6Ko)
Probability Triple:	***(1.0, 0.0, 0.0)***
Action:	*Fold*
Cases retrieved:	*83*
Average Similarity:	0.975320915

Stage: PREFLOP	
Number of players:	9
Relative position:	0.375
Players in pot:	0
Players to act:	7
Small bets committed:	0.0
Small bets to call:	1.0
Pot Odds:	0.4
Hand ranking:	70 (7Ao)
Probability Triple:	***(0.65, 0.34, 0.01)***
Action:	*Call*
Cases retrieved:	*100*
Average Similarity:	1.0

Stage: PREFLOP	
Number of players:	9
Relative position:	0.375
Players in pot:	1
Players to act:	0
Small bets committed:	1.0
Small bets to call:	1.0
Pot Odds:	0.181818182
Hand ranking:	70 (7Ao)
Probability Triple:	***(0.0, 0.876, 0.12)***
Action:	*Raise*
Cases retrieved:	*97*
Average Similarity:	0.980626787

Stage: FLOP	
Number of players:	2
Relative position:	0.0
Preflop bets:	3
Players in pot:	0
Players to act:	1
Bets committed:	0.0
Bets to call:	0.0
Small bets in pot:	7.5
Pot Odds:	0.0
Hand strength:	0.692414431
Positive potential:	0.070175439
Negative potential:	0.143754175
Probability Triple:	***(0.0, 0.51, 0.49)***
Action:	*Bet*
Cases retrieved:	*100*
Average Similarity:	0.993235666

Stage: TURN	
Number of players:	2
Relative position:	0.0
Flop bets:	1
Players in pot:	0
Players to act:	1
Bets committed:	0.0
Bets to call:	0.0
Small bets in pot:	9.5
Pot Odds:	0.0
Hand strength:	0.576328502
Positive potential:	0.109697315
Negative potential:	0.171569001
Probability Triple:	***(0.0, 0.46, 0.54)***
Action:	*Bet*
Cases retrieved:	*100*
Average Similarity:	0.996149977

Stage: RIVER	
Number of players:	2
Relative position:	0.0
Turn bets:	1
Players in pot:	0
Players to act:	1
Bets committed:	0.0
Bets to call:	0.0
Small bets in pot:	13.5
Pot Odds:	0.0
Hand strength:	0.584343434
Probability Triple:	***(0.0, 0.82, 0.18)***
Action:	*Check*
Cases retrieved:	*100*
Average Similarity:	0.997685402

Stage: PREFLOP	
Number of players:	9
Relative position:	0.875
Players in pot:	2
Players to act:	4
Small bets committed:	0.0
Small bets to call:	2.0
Pot Odds:	0.307692308
Hand ranking:	152 (4To)
Probability Triple:	***(1.0, 0.0, 0.0)***
Action:	*Fold*
Cases retrieved:	*100*
Average Similarity:	0.993812552

D.1.2 Aggressive/Non-Adaptive Competition

Stage: PREFLOP	
Number of players:	9
Relative position:	0.875
Players in pot:	1
Players to act:	3
Small bets committed:	0.0
Small bets to call:	2.0
Pot Odds:	0.363636364
Hand ranking:	10 (KAo)
Probability Triple:	***(0.22, 0.45, 0.33)***
Action:	*Call*
Cases retrieved:	*100*
Average Similarity:	0.99660311

Stage: FLOP	
Number of players:	4
Relative position:	0.667
Preflop bets:	2
Players in pot:	2
Players to act:	1
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	10.5
Pot Odds:	0.086956522
Hand strength:	0.400018667
Positive potential:	0.080843585
Negative potential:	0.128395062
Probability Triple:	***(0.46, 0.46, 0.08)***
Action:	*Call*
Cases retrieved:	*100*
Average Similarity:	0.979265465

Stage: TURN	
Number of players:	4
Relative position:	0.667
Flop bets:	1
Players in pot:	2
Players to act:	2
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	14.5
Pot Odds:	0.121212121
Hand strength:	0.241461401
Positive potential:	0.093760913
Negative potential:	0.147048452
Probability Triple:	***(0.44, 0.48, 0.08)***
Action:	*Call*
Cases retrieved:	*100*
Average Similarity:	0.978368517

Stage: TURN	
Number of players:	4
Relative position:	0.667
Flop bets:	1
Players in pot:	2
Players to act:	0
Bets committed:	1.0
Bets to call:	1.0
Small bets in pot:	22.5
Pot Odds:	0.081632653
Hand strength:	0.387761908
Positive potential:	0.093760913
Negative potential:	0.147048452
Probability Triple:	***(0.14, 0.84, 0.02)***
Action:	*Call*
Cases retrieved:	*45*
Average Similarity:	0.975647445

Stage: RIVER	
Number of players:	4
Relative position:	1.0
Turn bets:	2
Players in pot:	2
Players to act:	1
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	26.5
Pot Odds:	0.070175439
Hand strength:	0.263822314
Probability Triple:	***(0.18, 0.77, 0.05)***
Action:	*Call*
Cases retrieved:	*44*
Average Similarity:	0.975160008

Stage: RIVER	
Number of players:	4
Relative position:	1.0
Turn bets:	2
Players in pot:	2
Players to act:	0
Bets committed:	1.0
Bets to call:	1.0
Small bets in pot:	34.5
Pot Odds:	0.054794521
Hand strength:	0.263822314
Probability Triple:	***(0.25, 0.75, 0.0)***
Action:	*Call*
Cases retrieved:	*4*
Average Similarity:	0.974936377

D.2 Real Opponents

D.2.1 Play Money

Stage: PREFLOP	
Number of players:	8
Relative position:	1.0
Players in pot:	2
Players to act:	2
Small bets committed:	0.0
Small bets to call:	1.0
Pot Odds:	0.222222222
Hand ranking:	21 (JAo)
Probability Triple:	***(0.07, 0.27, 0.66)***
Action:	*Call*
Cases retrieved:	*91*
Average Similarity:	0.976823051

Stage: PREFLOP	
Number of players:	8
Relative position:	1.0
Players in pot:	4
Players to act:	0
Small bets committed:	1.0
Small bets to call:	1.0
Pot Odds:	0.1
Hand ranking:	21 (JAo)
Probability Triple:	***(0.0, 1.0, 0.0)***
Action:	*Call*
Cases retrieved:	*1*
Average Similarity:	0.971734645

Stage: FLOP	
Number of players:	5
Relative position:	1.0
Preflop bets:	2
Players in pot:	4
Players to act:	1
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	13.0
Pot Odds:	0.071428571
Hand strength:	0.117203166
Positive potential:	0.093249102
Negative potential:	0.150092227
Probability Triple:	***(0.54, 0.45, 0.0)***
Action:	*Fold*
Cases retrieved:	*11*
Average Similarity:	0.978143700

D.2.2 Real Money

Stage: PREFLOP	
Number of players:	6
Relative position:	1.0
Players in pot:	3
Players to act:	2
Small bets committed:	0.0
Small bets to call:	2.0
Pot Odds:	0.210526316
Hand ranking:	62 (55o)
Probability Triple:	***(0.06, 0.35, 0.05)***
Action:	*Call*
Cases retrieved:	*20*
Average Similarity:	0.836749143

Stage: FLOP	
Number of players:	5
Relative position:	1.0
Preflop bets:	2
Players in pot:	4
Players to act:	0
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	16.0
Pot Odds:	0.058823529
Hand strength:	0.444177014
Positive potential:	0.046011755
Negative potential:	0.115127479
Probability Triple:	***(0.0, 0.0, 1.0)***
Action:	*Raise*
Cases retrieved:	*1*
Average Similarity:	0.975627976

Stage: TURN	
Number of players:	5
Relative position:	1.0
Flop bets:	2
Players in pot:	4
Players to act:	0
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	30.0
Pot Odds:	0.0625
Hand strength:	0.23745975
Positive potential:	0.046909091
Negative potential:	0.135136835
Probability Triple:	***(1.0, 0.0, 0.0)***
Action:	*Fold*
Cases retrieved:	*1*
Average Similarity:	0.975157071

Stage: PREFLOP	
Number of players:	8
Relative position:	0.714
Players in pot:	2
Players to act:	4
Small bets committed:	0.0
Small bets to call:	2.0
Pot Odds:	0.266666667
Hand ranking:	5 (KAs)
Probability Triple:	***(0.0, 0.18, 0.82)***
Action:	*Raise*
Cases retrieved:	*65*
Average Similarity:	0.978264896

Stage: PREFLOP	
Number of players:	8
Relative position:	0.714
Players in pot:	4
Players to act:	0
Small bets committed:	3.0
Small bets to call:	1.0
Pot Odds:	0.047619048
Hand ranking:	5 (KAs)
Probability Triple:	***(0.0, 0.1, 0.0)***
Action:	*Call*
Cases retrieved:	*3*
Average Similarity:	0.977896181

Stage: FLOP	
Number of players:	5
Relative position:	0.75
Preflop bets:	4
Players in pot:	3
Players to act:	2
Bets committed:	0.0
Bets to call:	1.0
Small bets in pot:	23.0
Pot Odds:	0.041666667
Hand strength:	0.294740859
Positive potential:	0.083655536
Negative potential:	0.129734254
Probability Triple:	***(0.2, 0.8, 0.0)***
Action:	*Fold*
Cases retrieved:	*5*
Average Similarity:	0.974186519

Bibliography

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

Billings, D., N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, D. Szafron (2003). Approximating game-theoretic optimal strategies for full-scale poker. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence.*

Billings, D., A. Davidson, J. Schaeffer, D. Szafron (2002). The challenge of poker. *Artificial Intelligence Journal* **134**(1-2): 201-240.

Billings, D., D. Papp, J. Schaeffer, D. Szafron (1998). Opponent modeling in poker. *In AAAI National Conference*: 493-499.

Billings, D., L. Peña, J. Schaeffer, D. Szafron (1999). Using probabilistic knowledge and simulation to play poker. *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*: 697-703.

Buro, M. (1997). The Othello match of the year: Takeshi Murakami vs. Logistello. *International Computer Chess Association Journal* **20**(3): 189-193.

Campbell, M., J. A. J. Hoane, F. Hsu (2002). Deep Blue. *Artificial Intelligence Journal* **134**(1-2): 57-83.

Dahl, F. A. (2001). A Reinforcement Learning Algorithm Applied to Simplified Two-Player Texas Hold'em Poker. *Proceedings of the 12th European Conference on Machine Learning* Springer-Verlag.

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

Findler, N. V. (1977). Studies in machine cognition using the game of poker. *Communications of the ACM* **20**(4): 230-245.

Fogel, D. B. (2000). Evolving a checkers player without relying on human experience. *intelligence Journal* **11**(2): 20-27.

Ginsberg, M. L. (1999). Steps Toward an Expert-Level Bridge-Playing Program. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*: 584-593.

Ginsberg, M. L. (2001). GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research* **14**: 303-358.

Harrington, D. and B. Robertie (2004). *Harrington on Hold'em: Expert Strategy for No-Limit Tournaments. Volume 1: Strategic Play.* Las Vegas, Nevada, Two Plus Two Publishing.

Hsu, F., M. S. Campbell, J. A. Hoane (1995). Deep Blue system overview. *Proceedings of the 9th international conference on Supercomputing*: 240-244.

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

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

Kuhn, H. W. (1950). A simplified two-person poker. *Contributions to the Theory of Games I* Princeton University Press: 97-103.

Leake, D. B. (1996). *Case-Based Reasoning: Experiences, Lessons, & Future Directions.* Cambridge, MA, AAAI Press / MIT Press.

Ramon López de Mántaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael Cox, Kenneth Forbus, Mark Keane, Agnar Aamodt, and Ian Watson (2005). Retrieval, reuse, revision, and retention in case-based reasoning. *The Knowledge Engineering Review* **20**(03): 215 - 240.

Mitchell, T. M. (1997). *Machine Learning.* New York, McGraw-Hill Higher Education.

Nash, J. F. and L. S. Shapley (1950). A simple three-person poker game. *Contributions to the Theory of Games I* Princeton University Press: 105-116.

Papp, D. (1998). *Dealing with Imperfect Information in Poker*. Master's Thesis, University of Alberta.

Powell, J. H., B. M. Hauff, J. D. Hastings (2004). Utilizing Case-Based Reasoning and Automatic Case Elicitation to Develop a Self-Taught Knowledgeable Agent. *Proceedings of the Workshop on Challenges in Game AI, Nineteenth National Conference on Artificial Intelligence.*

Rasmusen, E. (2001). *Games and information : an introduction to game theory.* Malden, MA : Blackwell(3rd ed).

Riesbeck, C. and R. Schank (1989). *Inside Case-Based Reasoning*. Hillsdale, NJ, Lawrence Erlbaum.

Rubin, J. and I. Watson (2007). Investigating the Effectiveness of Applying Case-Based Reasoning to Texas Hold'em. *FLAIRS-20: The 20th International FLAIRS Conference. Key West, Florida.*

Salim, M. and P. Rohwer (2005). *Poker Opponent Modeling*. Indiana University**:** Personal communication.

Sandven, A. and B. Tessem (2006). A Case-Based Learner for Poker. *The Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006), Helsinki, Finland.*

Schaeffer, J., D. Billings, L. Peña, D. Szafron (1999). Learning to play strong poker. *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing.*

Schaeffer, J., J. Culberson, N. Treloar, B. Knight, P. Lu, D. Szafron (1991). Reviving the Game of Checkers. *Programming in Artificial Intelligence; The Second Computer Olympiad* **119-136**.

Schaeffer, J., J. Culberson, N. Treloar, B. Knight, P. Lu, D. Szafron (1992). A world championship caliber checkers program. *Artificial Intelligence Journal* **53**(2-3): 273-289.

Schaeffer, J., R. Lake, P. Lu, M. Bryant (1996). Chinook: The World Man-Machine Checkers Champion. *AI Magazine* **17**(1): 21-29.

Schauenberg, T. (2006). *Opponent Modelling and Search in Poker.* Master's thesis, University of Alberta.

Shih, J. (2001). Sequential instance-based learning for planning in the context of an imperfect information game. *Fourth International Conference on Case-Based Reasoning (ICCBR-01):* 483–501.

Sinclair, D. (1998). Using example-based reasoning for selective move generation in two player adversarial games. *Proceedings of the Fourth European Workshop on Case-Based Reasoning (EWCBR-98):* 126–135.

Sklansky, D. (1994). *The Theory of Poker.* Two Plus Two Publishing Las Vegas, NV, 1994 3rd ed.

Sklansky, D. and M. Malmuth (1994). *Hold'em Poker for Advanced Players.* Two Plus Two Publishing, Las Vegas, NV, 2nd ed.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM* **38**(3): 58-68.

Tesauro, G. (2002). Programming backgammon using self-teaching neural nets. *Artificial Intelligence Journal* **134**(1-2): 181-199.

Waterman, D. A. (1970). Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence Journal* **1**(1-2): 121-170.

Watson, I. (1997). *Applying case-based reasoning: techniques for enterprise systems.* San Francisco, CA, Morgan Kaufmann Publishers Inc.

Watson, I. D. (2003). *Applying knowledge management: techniques for building corporate memories*. Amsterdam ; Boston, Morgan Kaufmann, c2003.