

Applying Case-Based Reasoning to the Game of Bridge

Jacob Bellamy-McIntyre

**A dissertation submitted in partial fulfillment of the requirements for the
degree of Postgraduate Diploma of Science in Computer Science, University of
Auckland, 2008**

Abstract

Bridge provides a challenging problem for Artificial Intelligence research due to the game being stochastic (from the shuffling of the cards), hidden information (from not being able to see opponents cards) and from the general complexities of the game. Research into Computer Bridge is in its relative infancy, with the American Contract Bridge League holding the first World Championships Computer Bridge competition in 1997. With Bridge being a game that is more probabilistic and intuitive than Chess, it may be a better avenue of research for evaluating human-like intelligence.

This paper will explore the possibility of applying Case Base Reasoning to the game of Bridge and will discuss the problems that arise from trying to do so, while comparing Case Base Reasoning to other techniques used in Artificial Intelligence.

Table of Contents

1. Case-Based Reasoning

1.1 Introduction	5
1.2 The Domain and Adaptation	8
1.3 The Case and the Case Base	10
1.4 The Similarity Function	11

2. Games in AI

2.1 Introduction	14
2.2 Chess	15
2.3 Checkers	17
2.4 Poker	19
2.5 Bridge	25
2.6 Other Games	28

3. The Game of Bridge

3.1 Introduction to Bridge	30
3.2 The Bidding Phase	30
3.3 Play of the Hand	31
3.4 The Value of a Hand	32
3.5 Conventions	33

3.6 Online Play	34
3.7 Bridge Probabilities	35
4. First Attempt, CBridge1	
4.1 The Proposal	40
4.2 The CBridge1 Case Base	41
4.3 The CBridge1 Similarity Function	43
4.4 CBridge1 Initial Results	44
4.5 How To Improve CBridge1	48
4.6 CBridge1a	49
4.7 What if CBridge1 Had Been Successful?	53
4.8 Final Thoughts and Potential for Future Research ...	57

Chapter1:

Case-Based Reasoning

1.1 Introduction

Case-based reasoning (CBR) is a technique that attempts to solve new problems by looking at the solutions to similar problems and using those to construct solutions to new problems¹. The success of CBR is directly tied to the notion of similar problems having similar solutions. If a given task has similar problems that have wildly different solutions, then CBR is almost certainly doomed to fail for that task. But, if similar problems do have similar solutions, then CBR may provide a novel technique which is both simple and effective².

The first part of a CBR system is the case base. A case base is effectively a database with a series of past prototype cases. A case should accurately describe the problem, state the given solution, and then somehow describe the success of that solution to the given problem. Once it comes time to find a solution for a new problem, the problem is compared against all previous problems within the case base. A similarity function is then used to calculate how similar the current problem is to each of the cases in the case base and the most similar cases are retrieved. Once the most similar cases have been found, the results of their solutions are evaluated. At this stage the CBR system may apply an adaptation function which would attempt to take the existing solution and makes it applicable to the current problem.

¹ Mántaras et al, “Retrieval, reuse, revision, and retention in case-based reasoning”, in *The Knowledge Engineering Review* 20(03): 215 – 240 2006

² Riesbeck, C. and R. Schank. “Inside Case-Based Reasoning”. Hillsdale, NJ.: Lawrence Erlbaum 1989.

Once the solution has been applied to a new problem, the results can be recorded and a new case can be authored and placed in the case base. While not immediately obvious, this actually means that CBR is a machine learning technique. The more cases it encounters, the more prototypical cases it has to address new problems. Sometimes a proposed solution will not work, so when another similar problem is discovered a different solution used to the one previously can be attempted. Effectively this means that as the system attempts to solve more problems, the better it should become at solving them, although this will be dependant on a good application of CBR.

There are several motivations for wanting to use a system like this. One such motivation is that the system offers justification for the choices it makes. A justification is simply a similar case or a series of similar cases that were retrieved and used in determining the solution given. A rule based system's form of justification would have been to return back a series of executed rules which may happen to be far too elaborate or abstract for the user to follow. A system like an artificial neural network would not be able to offer any justification at all. Being able to look at a previous case and seeing that the solution used is straight forward and also helps in fine tuning and debugging. If the CBR system generates a response that is very inappropriate for a new problem, the justification case can be generated and the engineer dealing with the system can then discern why an inappropriate case was used.

Conceptually, it is possible to build up a database which contains every possible problem instance in a domain with optimal answers for each. This would result in the problem domain being optimally solved. A CBR system could achieve the same result by having a good representative set of cases and an adaptation function which together captures the essence of the important aspects of the domain. More commonly however, a CBR system does not set out to give optimum results every time, and instead settles on good rather than optimal solutions. In fact, in a lot of CBR applications such as imperfect information games it would be impossible to give optimum results every single time without clairvoyance as elements such as luck and chance make it impossible to get perfect results every single time.

Perhaps the biggest advantage of using a CBR system is its conceptual simplicity and the ease of which one can often be implemented. A rule based approach in most domains would require the input of a domain expert to achieve good performance. This is obviously prohibitive for any individual who wishes to create such a system who does not have access to a domain expert. Even if an expert is available, it is possible that they simply cannot express how to solve problems in their domain as a series of rules, either because they are unsure actually as to what rules they are using, or because they use something more abstract like pattern recognition. A CBR system can be developed by a novice to the domain, without a domain expert, with relatively few lines of code, and still achieve very good performance simply by developing a good case base.

In summary, a CBR system has the following components:

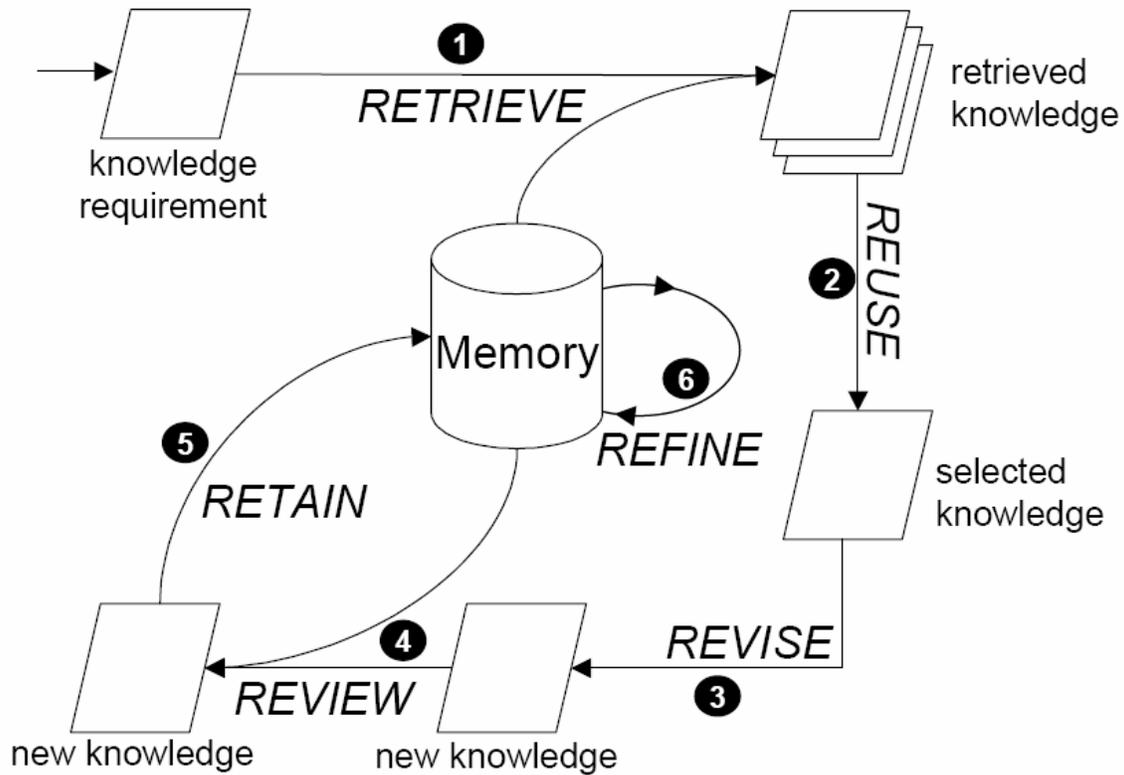
1. A domain over which it is attempting to solve problems.
2. A way to describe each case in a given domain
3. A collection of cases, known as a case base
4. A similarity function for computing similarity among cases, and a way of fetching similar cases
5. An adaptation function for applying similar solutions to the case base.

The CBR life cycle is usually a cyclical process, composed of the six 'REs' as defined in Watson(2003)³.

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.

The following figure helps illustrate this process.

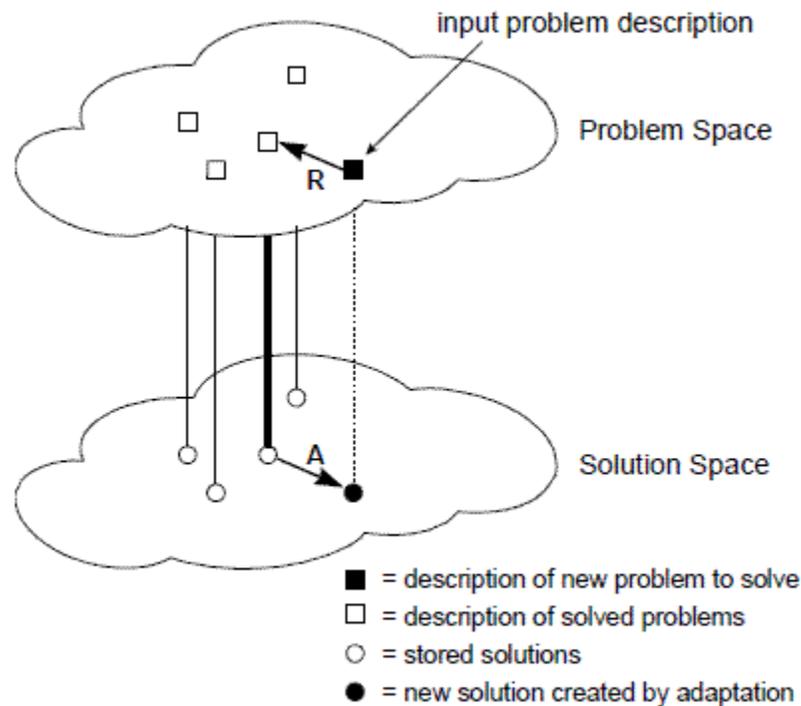
³ Watson, I. D. (2003). "Applying knowledge management: techniques for building corporate memories". Amsterdam ; Boston, Morgan Kaufmann, c2003.



1.2 The domain and adaptation

A domain comprises of a problem space and a solution space. The problem space is essentially every single possible problem situation in the domain, and the solution space is every possible solution that can be put forward. The problem space can be fairly small, for instance a CBR might be developed for a simple travel agent lookup system where a user chooses a country and a month of the year in which they would like to visit that country. The solution space would however be quite large, as it would include all the ways for a user to get to any place in the world at any given time using any mode of transport or accommodation in their system. In Bridge, the problem space is incredibly large as the possible number of deals is $52! / (13!)^4$, which evaluates to 53,644,737,765,488,792,839,237,440,000 different deals. The solution space is much smaller as there are only 113 possible contracts that can be established in any given deal.

The success of a CBR system hinges on similar problems having similar solutions⁴. The following diagram attempts to show this relationship in a CBR system.



By comparing a new problem instance with similar problems, a solution can be hypothesized by estimating the solution based on similar solutions to similar problems. The adaptation function is used to create an estimation that takes into account the differences between the existing cases and the new problem. This allows results from cases with distant similarity to still be informative. Failing to include an adaptation function can mean that trying to get an answer from a case less than a certain threshold such as 80% will be completely fruitless. But care needs to be made when designing an adaptation function. It often requires a fairly extensive level of domain knowledge to adequately change and adapt an answer to similar cases with any degree of accuracy. Making a perfect adaptation function would be virtually the same as creating a rule base

⁴ Leake, D. B. "Case-Based Reasoning: Experiences, Lessons, & Future Directions". Cambridge, MA, AAAI Press / MIT Press 1996.

program that automatically knows what to do for any given hand. Besides being difficult to do, using a complicated adaptation function can slow down the system too much on a very large case base.

Some CBR systems forego the use of an adaptation function. A CBR system may decide not to use adaptation because the creation of such a function is too difficult to do without extensive knowledge elicitation, or because of the time required to compute an adapted answer. These systems may instead prefer to use a larger case base to receive a larger number of almost identical cases, providing an almost identical answer. Some CBR systems may be working with problems simple enough that no adaptation function is required.

1.3 The Case and the Case Base

As the domain can be quite large case design can be crucial to the success of the CBR system. If the system for authoring cases is lacking in detail, then it cannot adequately differentiate between cases. Missing out key detail in the case representation can lead to cases being matched as being identical when there are actually fundamental differences between them. Unless these differences are evident within the case representation, the system will never be able to tell the difference. Conversely however, a case representation that is too obtuse may make the task of creating a good CBR more difficult than it should be. A more complicated case representation would very well likely result in a more complicated similarity or adaptation function, which would slow down case retrieval and potentially result in worse performance than a system using a simpler case representation.

The cases themselves are not distilled knowledge. They simply describe actual events. This is one of the biggest advantages of using a CBR system over a rule based system. Usually, describing and authoring a case is far simpler than generating an elaborate set of rules that attempt to cover every possible problem that can be thrown at

it. Sometimes there may already be an established convention for representing cases in that domain.

Once a way of describing a case has been decided upon, the case base has to be generated. Sometimes such a database may already exist, and all that is required is developing a way to interface that database with the CBR system. When it comes to applying CBR to games, this could mean simply taking a series of cases from the historical play of actual experts. If the domain lends itself well to CBR, and if the system has been well designed, this can mean the CBR system makes expert decisions without any training time, without having to elicit knowledge from domain experts, with relatively little source code and while still offering justifications for decisions made.

Sometimes however, no existing database exists for populating the case base for the CBR system. In this case a case base may be generated just through attempting random actions and recording the results. Even cases which apply bad logic are informative if there is a way to measure how successful each action was. If the system can tell that one particular decision was a bad one, then it can decide to perform some other type of action next time it encounters a similar case. This approach would mean a certain amount of time would have to be used in training the CBR system so that it gave good or even adequate results, but it means a case base can be generated even in the absence of any existing prototypical cases.

1.4 The Similarity Function

Creating a case base can be an incredibly straight forward affair. Sometimes all it requires is taking an existing database and using it. However, for the system to work it needs a similarity function. The goal of this function is that for any two cases, it can give a value to represent how similar the two cases are. Usually this is represented as a percentage. The most common way of computing similarity is through using the K-Nearest Neighbor (KNN) algorithm. Informally, KNN will compute the similarity between the current hand and every other hand in the case base, and then the K most

similar cases are retrieved, and the most commonly successful action from those cases is the one used. More formally, global similarity is computed as follows:

$$\text{Similarity}(T, S) = \sum_{i=1}^N f(T_i, S_i) \times W_i$$

Here, T refers to the target case for which we wish to compute the similarity, and S refers to all the cases that we are comparing it with in our case base. The function f refers to the similarity function, the variable i refers to each individual attribute in the case, the constant n refers to the size of the case base and W_i refers to a weighting for attribute i that is used for evaluating the relative weight or value that any individual attribute will hold while calculating similarity.

Creating a similarity function is usually not as straightforward as obtaining a case base. Decisions need to be made that may be difficult to make without the input of a domain expert. For instance, what should the size of K be? If it is too large, then it may retrieve cases that are too dissimilar to be of use. If K is too small, then it may not look at enough cases to give a well informed answer. Another decision needs to be the exact weightings for each attribute in the case base. If these weightings do not accurately reflect the attributes actual importance in the domain, then it will incorrectly assess the cases similarity. The way in which similarity should be calculated for each attribute is also highly important, and not always immediately obvious. A given similarity function may well give a very accurate and useful result for similarity, but if it is too complicated then the system will struggle to compute the similarity for all cases in the case base. Having a poor similarity function will also mean that cases retrieved may not be the most relevant or useful.

It is difficult to create a similarity function with any real certainty about the weightings. The person designing the system will be required to make judgment calls on what they should be. Knowledge of how the similarity function should be implemented and what weightings to give to individual attributes can be obtained via knowledge elicitation. At the very least, that kind of information would be easier to obtain than a complex series of rules on the domain. Without access to a domain expert, the software

engineer can simply attempt to guess the similarity function and weightings. If the case base that is being used has large numbers of very similar cases it will not actually matter too much so long as the similarity function and weightings are good enough to recognize very similar or identical cases. If that is the case, then fine tuning the weightings or similarity function will not necessarily make much difference at all, as the same identical or very similar cases will be retrieved regardless. But as the case base is more sparse and diverse, the more important it is to have an accurate similarity function to find cases that are informative.

Chapter 2

Games in AI

2.1 Introduction

Serious attempts at making computer programs capable of playing games began in the middle of the twentieth century. Several papers in the 1950s on chess and checkers started decades of interesting research on how to make a program that could play these games at a competitive level⁵. One big motivation for the popularity in research for game Artificial Intelligence is the possibility of using common reproducible environments suitable for testing new search, learning, and pattern recognition algorithms. As the level of these programs increases, often they become contenders at the highest level of human play, making for several historical Man versus Machine showdowns, such as Tinsley and Chinook, and Kasparov and Deep Blue. Some games, such as Checkers, are now completely solved with optimum play⁶. Others, such as Go, still struggle to play at the novice level even with elaborate techniques⁷. Games where human play is significantly above that of computer play has prompted unique research into the psychology of players in an attempt to create programs that can mimic human cognition.

Once a given game playing AI has reached world champion status, the question remains as to whether that is the ultimate satisfying goal? Is there any more need for research into AI that can play that game? Perhaps the AI could be improved further to be more efficient, or to perform slightly better than the current computer champion. But improved efficiency would not be a particularly exciting goal, and nor for the most part would it be a sufficient one. Perhaps the more interesting aspect of game AI is seeing

⁵ C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41 (7th series)(314):256–275, 1950.

⁶ Schaeffer et al, “Checkers is Solved” *Science* 14 September 2007: 1518-1522

⁷ Jacek Mandziuk “Computational Intelligence in Mind Game”, Springer Berlin / Heidelberg 2007, Page 413

what techniques work for what kinds of games. For instance, chess and checkers have had the largest success using brute force search algorithms, whereas the top computer backgammon player was developed using neural networks⁸. By experimenting with different techniques, and analyzing their success and failure, the problems unique to each game have been more thoroughly defined and analyzed. Some of the most interesting tasks involve creating solutions to problems that do not require domain specific knowledge, with training techniques such as neural networks or with case base reasoning.

2.2 Chess

Chess is perhaps the most widely researched game in Artificial Intelligence. The first well known Chess playing automaton was *The Turk*, created by Baron Wolfgang von Kempelen and revealed in 1770. *The Turk* seemed to be an automaton capable of expert play, defeating famous individuals such as Napoleon and Empress Catherine Russia. For eighty-four years *The Turk* was demonstrated throughout Europe and America before being destroyed in a fire in 1854. Three years later, it was revealed that *The Turk* had been a hoax, and there had been a Chess expert sitting inside the machinery the entire time, controlling the moves played from within. It would take until the 20th century until machines were actually capable of performing the same kind of feats as *The Turk*.

Perhaps the first working automaton to play Chess was *El Ajedrecista*⁹ in 1912, although it only played one end game situation where it controlled King and Rook against the players King. The automaton was guaranteed victory every single game it played, although it would not necessarily win the game in the smallest number of moves. By the standard of today, the automaton is not particularly remarkable, but at the time it was considered revolutionary. Still, it was a long way off from performing the kinds of miracles that *The Turk* seemed to be able to perform more than a hundred years earlier. This level of play would have to wait until the creation of computers.

⁸ Ibid, page 411

⁹ http://en.wikipedia.org/wiki/El_Ajedrecista

In the last fifty years many attempts at creating a potent Chess playing computer bots have been made including *Kaissa*, *Mac Hack*, *Chess 4.6*, *Belle*, *Cray Blitz*, *Hitech* and *Deep Thought*¹⁰. This research and development of chess playing programs cumulated in the creation of *Deep Blue*. *Deep Blue* was created by IBM and in 1997 managed to defeat Gary Kasparov in a highly televised chess match. After his defeat, Gary Kasparov accused *Deep Blue* of being another Turk. Kasparov claimed that IBM had pitted a Master chess player against him who used the computational power of *Deep Blue* to make informed choices¹¹ . To demonstrate the plausibility of this, Gary Kasparov has since created a variant of chess known as Advance Chess where players would partner up with analytical computer bots to improve their level of play.

Kasparov pointed out several instances within the match where *Deep Blue* played like it was a human. First was in game two, where Kasparov attempted to sacrifice a pawn for a long term positional advantage. Kasparov reasoned that computer chess players are generally greedy and would take pieces for a short term advantage. It also turned out that one of *Deep Blue*'s moves was a terrible error, and Kasparov could have threatened a perpetual check. The mystery was how a brute force algorithm could miss such a disastrous outcome within its own horizon? But regardless of whether Gary Kasparov's accusations were correct, the algorithm employed by *Deep Blue* is a highly effective one.

Deep Blue relied on brute force searching of game trees, specifically minimax trees with alpha-beta pruning. Using brute force to search many moves ahead is a simple but effective solution to the problem, with the most complicated portion being the heuristics which state whether or not a board position is a good one to be in. With this algorithm, a more powerful computer will actually play better than a slower one as it would allow the faster system to search more moves within the allocated time. With the increase in computational power, modern day chess playing programs employing brute force enjoy even more powerful searching capability than *Deep Blue* with modern programs like *Fritz*, *Hydra*, and *Deep Junior* continuing to play Grand Masters though with significantly less media attention than the Kasparov and *Deep Blue* match. While

¹⁰ http://www.sciencenews.org/sn_arc97/8_2_97/bob1.htm

¹¹ Hal Vogel "Game Over: Kasparov and the Machine", 23 January 2004 (UK), Documentary

this approach is certainly effective, and would be considered a milestone from an engineering and programming point of view, there is still room to explore other methods which may closer map human cognition.

David Sinclair in a paper in 1998¹² applied Case Base Reasoning to Chess. It involved 16,728 games from current Chess grandmasters using a variety of strategies, and used Principal Component Analysis and the notion of “chunks” for encoding cases. A case-base would be retrieved for each individual board position, and the move made was recorded within the system. When new board positions are encountered the case base is searched for the most similar cases, as is usual in a CBR system. Sinclair found that with a strict similarity metric in place, the quality of cases retrieved is very high, but fewer of them are retrieved. When relaxing the similarity metric, more cases are retrieved, but that they were of less quality.

2.3 Checkers

The approach for checkers has historically been the same as chess. However, it was a much simpler game. Only half the tiles on the board are used, and all pieces are identical before they turn into kings. As such, the game tree generated has a much lower branching factor and far more moves ahead can be evaluated by an Artificial Intelligence which wishes to brute force its way through the game. Jonathan Schaeffer of The University of Alberta created *Chinook*, which would become the first computer winner of a human world championship through the same method of brute force search that would later be adopted by *Deep Blue*¹³. The basic design was the same, with opening and end game databases, efficient search trees, and a well tuned board evaluation function. When *Chinook* became the winner of the checkers world championship it utilized an impressive end game database where any situation where there were seven pieces left on the board could guarantee either a win or a draw for *Chinook*, so long as victory had not already been decided. By 2007 Schaeffer managed to solve the game of Checkers entirely with a

¹² David Sinclair “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)*: 1998, pages 126–135.

¹³ J. Schaeffer, R. Lake, P. Lu, and M. Bryant. “Chinook: The world manmachine checkers champion”. *AI Magazine*, 17(1):21–29, 1996.

game database that would guarantee at the very least a tie in every match it played, from start to finish.

Blondie24 was a checker playing computer program created by David Fogel with a very different approach to the game¹⁴. It utilized a machine learning algorithm that combined a genetic algorithm with a neural network. *Blondie24* initially trained just by playing repeated games against itself. Then, *Blondie24* was unleashed onto the Internet Gaming Zone masquerading as a 24 year old blonde woman. *Blondie24* played against 165 people online and managed to achieve a rating of 2048, ranking her higher than 99.61% of the player population on The Zone. *Blondie24* only ever managed to beat Chinook on its novice difficulty so in regards to playing ability *Blondie24* was inferior to Chinook. However, *Blondie24* did not have any expert knowledge on how to play checkers at all. Programs like Deep Blue and Chinook had to get expert knowledge for programming their heuristic functions, while *Blondie24* could theoretically have been programmed by a complete checker novice. The ability to play checkers was drawn from the experience of the games it played.

Case-Based Reasoning has also been applied to checkers. *CHEBR* (CHEckers case-Based Reasoning) was created by Jay Powell, and used Automatic Case Elicitation¹⁵. As the game is played in real time cases are added to the case base. The system has a series of observations, or states, and a set of actions, which is a design similar to that used in planners. When the system comes across a new problem, it will first check in the case base for similar problems in the past to decide on which action to take, and in the lack of guidance it will perform a random action. It then evaluates whatever action was taken, and records firstly whether it was valid, and secondly how successful the move was. It was shown that despite the domain being defined so loosely, it could still get to a sufficient playing level just by playing more games.

¹⁴ David B. Fogel "Blondie24: Playing at the Edge of AI" , Morgan Kauffman, 2001

¹⁵ Powell, J. H., B. M. Hauff, J. D. Hastings. "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*. 2004

2.4 Poker

Poker is a very different game to those mentioned previously. One of the biggest difficulties in creating an artificial intelligence that plays poker at a competitive level is that Poker is an impartial information game because other cards are hidden. Because of the stochastic nature of the game, it is impossible to solve Poker in the same manner that checkers was solved. It also means that a game tree search algorithm used in *Deep Blue* and *Chinook* simply do not lend themselves to this kind of game play. Poker is not a game just about probabilities, as it is a competitive game and players are expected to deal with bluffing and deception. This means that such programs will typically have to implement features such as opponent modeling, risk management techniques, and probabilistic functions.

Poker is a game of deception. With poker, the notion of objectively playing the game optimally is hard to define because of the uncertainty of the opponents cards. Due to this hidden information, ‘optimal play’ would have to be derived from probabilities, but it will also have to depend on the play of the opponent. For example, if a player never bluffs even in extreme circumstances, then it would be wise to assume they have a strong hand when they continue to raise, whereas no such assumption should be made about a player who wildly bluffs whenever their hand is weak. One possibility for a program to deal with this is to make no assumptions about the player and to develop a statistical model of their style of play as more games are played against that opponent. Another possibility would be to train a neural network to achieve this task, with the same learned function to be used against all people it plays against.

The University of Alberta Poker Research Group headed by Jonathan Schaeffer, who also worked on Chinook, has been working on a number of poker playing programs for a number of years¹⁶. Two programs that have come out of this research are *Poki*¹⁷ and *PsOpti*¹⁸. *Poki* was designed to play against a full table of players on the poker table, and

¹⁶ <http://www.cs.ualberta.ca/~games/poker/> as of 20/10/2008

¹⁷ Davidson, A. “Opponent modeling in poker: Learning and acting in a hostile and uncertain environment.” Master's thesis, University of Alberta 2002.

¹⁸ Billings, D., N. Burch, et al. “Approximating game-theoretic optimal strategies for full-scale poker”. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. 2003

PsOpti was designed with play against only one opponent in mind. *Poki* was tested extensively against real people using fake money, and was found to generate a profit on average, and was estimated to be at intermediate playing strength. *Poki* was developed as a rule based program. To create such a program and have it be effective, those creating the program either have to be domain experts or have access to a domain expert from which they can elicit knowledge from. Programs like *Poki* operate by having a series of ad hoc rules which represent codified knowledge elicited from an expert on how to play well. The problem with such programs is that their performance is often limited by the ease of which a domain expert can actually come up with a series of rules by which to play the game. With most reasonably complex games simply using a set of rules will not be enough to make a player an expert. There will be too many situations for an expert to realistically cover completely. An expert will tackle new problem instances using reasoning, logic, and past experience on similar experiences. The process of eliciting knowledge can take a very long time, and additionally can require large amounts of written code.

A variant of *Poki* and another poker playing program from the University of Alberta called *Loki* use a technique known as simulation¹⁹. Simulation works by randomly considering different possible assignments of cards they may have, then performing a game tree search on each of those possibilities, and finally using the results of these game trees to inform as to which move should be taken. To determine in these game trees whether a player would fold, call, or raise, a probability triple (f, c, r) was generated and used to guess what their action would be. The way in which this probability triple would be constructed would be through the use of rules, or game-specific information. *Loki* and *Poki* both used opponent modeling techniques to help inform the betting decisions that would be made by each player.

CBR has been applied to poker in Jonathan Rubin's poker playing program *CASe-based Poker player (Casper)*²⁰. *Casper* is the main influence in this project's attempt at

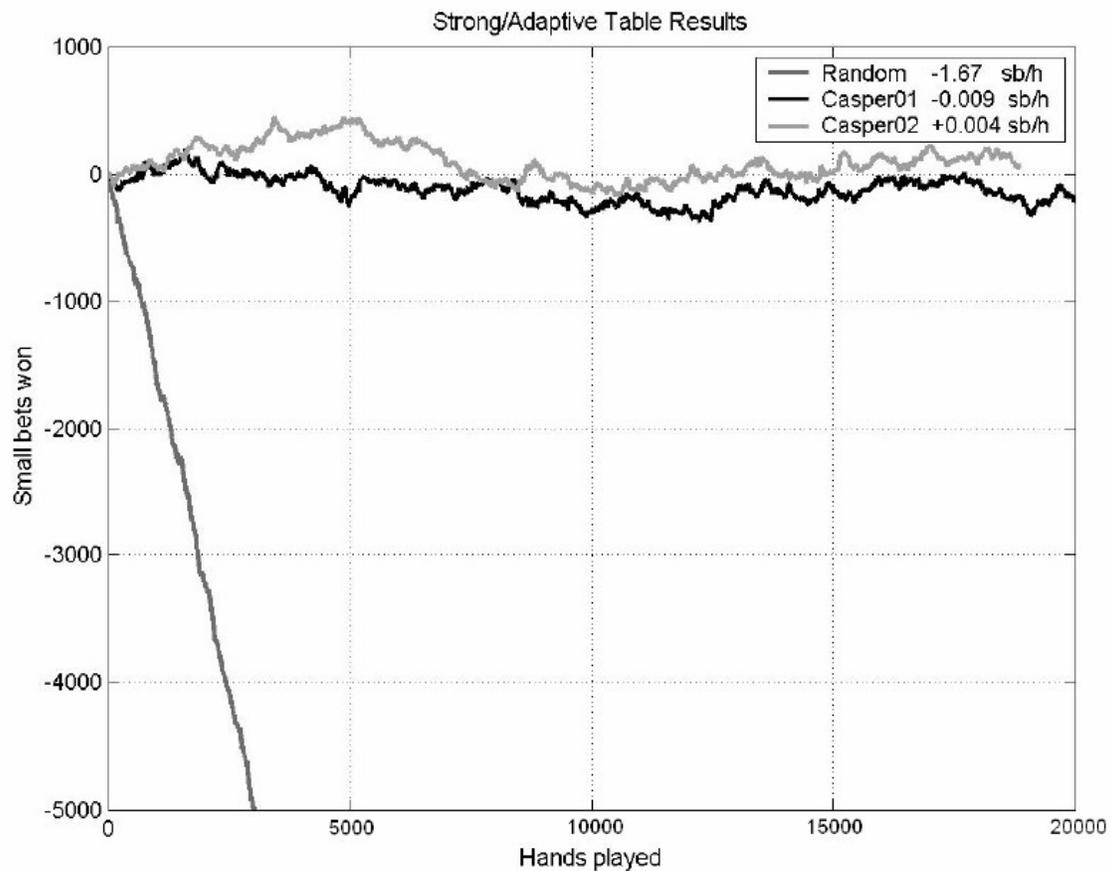
¹⁹ Billings, D., L. Peña, et al. "Using probabilistic knowledge and simulation to play poker". *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*: 697-703. 1999

²⁰ Jonathan Rubin, "Casper: Design and Development of a Case-Based Poker Player", Masters Thesis, University of Auckland 2007

applying CBR to Bridge. *Casper* generated its case base from games played by *Pokibot* and *Simbot*, both created from the University of Alberta Poker Research Group, and both of which had been shown to be fairly successful playing against human opponents for money. Both had been developed through extensive knowledge elicitation from a domain expert. By using a case base developed from the play of both these programs, *Casper* effectively bypassed the knowledge elicitation phase.

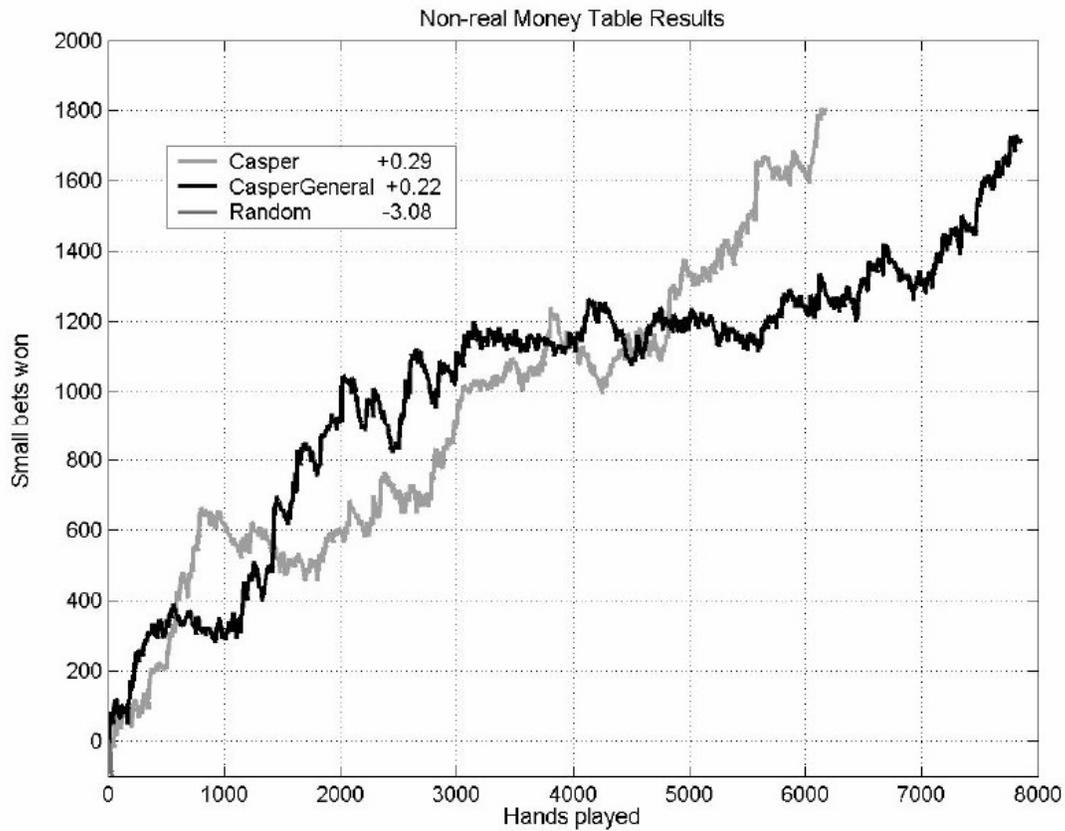
The different rounds of poker, preflop, flop, turn, and river, were all listed as separate cases within *Casper*'s case base. Each case has a set of attributes which are unique to that particular stage of the game, such as number of players, hand strength, pot odds, and so forth, and each attribute has its own similarity metric to be calculated using a simple ratio.

The results were very favourable for *Casper*. It was first played against the University of Alberta poker playing programs, using a set of hand picked weights. The results are as follows.



Casper01 and *Casper02* both managed to break roughly even in their play against the University of Alberta programs, despite not having elicited knowledge. *Casper01* and *Casper02* differed only in that *Casper02* had a larger case base. *Casper* was tested against a variety of these bots, including *Jagbot* an aggressive non-adaptive player, using a variety of weights, including ones generated from play. Then *Casper* was put online and played against real human opponents.

²¹ *ibid*



In this case, *Casper* was the version of *Casper* with hand picked weights and enlarged case base, and *CasperGeneral* was using derived weights. When playing with non-real money, *Casper* seems to have won a constant profit on average. The next obvious step was to put *Casper* online against real people, for real money, and to hope that riches followed.

²² ibid



23

Unfortunately, when played for real money, *Casper* made a constant loss. There could be a number of reasons why the results differed so much from when it was played with false money. One obvious possibility, though not expressed in Rubin's Thesis, is that it was not playing on a fair playing field. It could be that there was an agent on the server side who *Casper* was playing against that had some kind of unfair advantage, such as being able to dictate the deals, or having perfect knowledge as to the hands of their opponents. The shape of this downward trend does look like one that would trap an individual with a gambling problem. A constant downward trend is usually partnered with an upward trend that makes up for most of the losses incurred, but which is then replaced by an even more severe downward trend until the losses are even greater, before starting another upwards trend. However, there is another reason for this downward trend proposed by Rubin that is also very plausible.

²³ *ibid*

One observation made by Rubin is that people who play poker online for real money are likely to be stronger players than those that play for fake. Additionally, the way they play the game will be quite different. When playing for fake money, a player might be more ready to make very large bids because they are not losses they actually have to incur. When playing against real people for real money, the average similarity of cases for preflop, flop, and turn, were less than 50%, and was only 69% for the river, compared to the roughly 90% average that *Casper* enjoyed against the University of Alberta programs.

In summary, *Casper* enjoyed excellent performance considering it used no knowledge elicitation in its creation. It has also shown that CBR can do very well in an imperfect information game such as Poker. This leaves the possibility of it being applied to other games, which may also be played online against real people, and perhaps for real money, using the same techniques.

2.5 Bridge

Bridge has more in common with Poker than it does with Chess or Checkers in that it is also a stochastic imperfect information game. However it also has a few extra complexities. It is a game played in partnership, and is far more complex and difficult for a new player to start and learn how to play. Chapter 3 will go more into the rules of Bridge. There has been less research into Bridge than there have been into other games discussed up to this point, including Poker, and most of the progress in Bridge has happened within the last two decades. One of the earliest papers on Bridge was authored by Berlekamp in 1976²⁴. This paper was not interested in how to actually play Bridge, but rather it was on how he had come up with a bidding scheme which could tell precisely which player had exactly what card, for every card in the deck, though this process was artificial as it required all four players to cooperatively work together, and the final bid was made without any regard to its viability for the play of the hand. In Zia Mahmood's book, *Bridge, My Way* (1992), Mahmood offered a one million pound bet that no four person team of his choosing would be beaten by a computer. In 1996 with the improved play of Computer Bridge players, Mahmood was forced to withdraw his bet²⁵.

²⁴ Berlekamp, E. "Cooperative bridge bidding " *Information Theory, IEEE Transaction On* , Volume 2, Issue 6, Nov 1976 Page(s): 753-756

²⁵ http://www.imp-bridge.nl/articles/The_GIB2Zone.html

Since 1997 the World Computer-Bridge Championship has been organized by the American Contract Bridge League, and have since then has had a number of recurring entries to the competition. The regular entrants are *Bridge Baron*, *WBridge5*, *Jack*, *Micro Bridge*, *Q-Plus Bridge*, and *Blue Chip Bridge*. The first champion in 1997 was *Bridge Baron*.

Bridge Baron developed by Stephen Smith, Dana Nau, and Thomas Throop apparently consists of “fifty-thousand lines of C code dedicated solely to the Bridge engine which calculates which bid to make and what cards to play²⁶.” The way in which *Bridge Baron* wrote its rules for bid play is typical of most of the Bridge playing programs. They utilize a set of ad hoc rules to find out what bids should be made. Most of the focus on improving the play of Bridge programs is on their performance during the play of the hand, not the bidding phase. Research into the bidding phase of Bridge is far more limited. The reason is likely because with elicited knowledge an effective bidding strategy can be put in place through a series of ad hoc rules, whereas the play of the hand is a complex game tree problem all in itself which lends itself to a variety of different search techniques. For instance, the papers written on *Bridge Baron* focus almost exclusively on it’s Hierarchical Task-Network planning techniques used for the play of the hand. However, this is only half the game, and the play of the hand can be futile if a good contract is not reached during the bidding phase.

GIB created by Mathew Ginsberg however is a Bridge playing program that attempts to apply search to both the bidding and play of the hand phase. On bidding Ginsberg has said “It is possible to use search-based techniques here also, although there is no escaping the fact that a large database of bids and their meanings is needed by the program. (Bidding is, after all, a communicative process; the meanings of the bids need to be agreed upon.)”²⁷

For the bidding phase, *GIB* uses a Borel simulation. Given a bidding situation, the Borel simulation generates a number of different deals consistent with the information taken so

²⁶Smith Et Al, “Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge”, *IAAI-98/AAAI-98 Proceedings*, pp. 1079–1086, 1998

²⁷ Matthew L. Ginsberg “GIB: Steps Toward an Expert-Level Bridge-Playing Program” *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*: 584-593, 1999

far from the players hand and current bidding. Then for each possible bid that can be made, paired with each deal generated, a database is queried which will project the likely outcome of the auction if the bid is made, and the bid with on average the best outcome is picked. Ginsberg notes several problems with this approach. Firstly, it does not take into account double dummy techniques where the bids are intentionally leaving out information about the hand. Human experts often decide not to reveal too much information about their hands to their partner because that information is also declared to the opposition who may attempt to use it during the play of the hand. The database being used also can create difficulties. For instance, if the database assumes a more conservative play from the partner, then it will require a more aggressive approach from the player to compensate. Lastly, there may be gaps in the database for rare outlying cases, such as a 7D bid. Because the case is so rare, the problem will be encountered so infrequently that it may escape notice for a very long time.

There have also been a couple of attempts at applying artificial neural networks to Bridge. DeLooze and Downey²⁸ identified four different types of bids a player can make.

- 1. Asserting Bids.** "By making this bid, I assert that my hand has these properties"
- 2. Denying Bids.** "By making this bid, I deny that my hand has these properties"
- 3. Asking and Answering Questions Bids.** "If your hand is of type 1, make bid 1; if it is of type 2, make bid 2, etc. (e.g. asking how many aces your partner has)"
- 4. Interrupting Bids.** "The primary purpose of this bid is to stop the opponents communicating"

DeLooze and Downey set up an artificial neural network implementing a Kohonen Map, and trained it using a convention system suggested by the American Contract Bridge League. The end result was a program capable of making Bridge bids named *BridgeSom*, however it was only developed to make asserting bids. The experimental method described in the paper however was not vigorous. It played twenty four distinct games against *Jack*, each followed up with a mirror match where the board positions were swapped to compare how the two programs bid on identical hands. The actual play of the hand was handled by *Jack* in both situations. All of the special bidding features of *Jack*

²⁸DeLooze, L, L.; Downey J "Bridge Bidding with Imperfect Information", *Computational Intelligence and Games*, 2007, CIG 2007, IEEE Symposium On 1-5 April 2007 Pages 368-373

were disabled, and each board was setup for contracts in No Trumps rather than in specific suits. The evidence from this rather limited number of games played showed *Jack* outperformed *BridgeSom*, even with its features turned off, but not by a large margin. There are issues with using *Jack* to measure whether a bid was a good one made or not, as it uses a technique called simulation which generates a set of random deals that are consistent with the information provided and performs a game tree search on each. This simulation technique however is not deterministic since it depends on random deals. Also, *BridgeSom* did not use a large enough sample size to tell whether or not it really could make Bridge bids on par with *Jack*. But it does show a technique capable of making bids in Bridge, without knowledge elicitation, and the results of which can be expanded upon by extending its functionality to denying bids, asking and answering question bids, and interrupting bids.

2.6 Other Games

The most well known Backgammon playing program is *TD-Gammon* written by Gerald Tesauro of IBM, which implements temporal difference learning and is the most archetypal example of success of machine learning applied to games²⁹. The evaluation function for *TD-Gammon* is implemented using a multilayer perceptron neural network trained using backpropagation. According to assessments by two champion Backgammon players, Bill Robertie and Kit Woolsey, *TD-Gammon*'s level of play is at , or possible even above, the level of the worlds top Backgammon players.

Another game where humans are outperformed by computer players is Othello. In 1997 Michael Buro's program *Logistello* beat world champion player Takeshi Murakami with a score of 6-0³⁰. *Logistello*'s evaluation function is based on disc patterns and utilized a forward pruning method which cut short most of the irrelevant subtrees in the overall game tree.

²⁹ G. Tesauro. "Temporal Difference Learning and TD-Gammon". *Communications of the ACM*, 38(3):58–68, March 1995.

³⁰ M. Buro. "Improving heuristic mini-max search by supervised learning." *Artificial Intelligence*, 134:85–99, 2002.

The game of Go is considered one of the most demanding and challenging areas of research in game AI. Despite the game having simple rules, and only one kind of playing piece, no one has yet managed to create a program capable of playing the game well. Even the most advanced Go playing programs can be defeated by amateur players. One of the biggest reasons for this is that Go has an incredibly large branching factor which is so large it effectively eliminates any brute force search method. Several distinct features of the game also make analyzing a single static board arrangement several orders of magnitude slower than other games.

Chapter 3

The Game of Bridge

3.1 Introduction to Bridge

A game of bridge requires four players, a deck of cards, and a table. The four players will form two partnerships. Partners sit opposite each other on the table. It is usually customary to refer to the players by the compass positions with North and South being partners against East and West. The dealer distributes the shuffled cards amongst all four players, so each will have thirteen cards face down. Each player can look at their own cards, but they cannot see their partners or their opponents. The game itself has two phases, a bidding phase, and then the play. The bidding phase is a section of the game where the partnerships try to establish a contract to be used in the play of the hand. This phase is the main focus of this dissertation. The play of the hand follows, and the contract states which suit is trumps (if any), and the number of tricks that need to be won for the declarers partnership to establish their contract. At the end of play, the players count up the number of tricks won by each side to determine whether or not the contract had been made, and the number of any overtricks or undertricks. There are two main scoring systems used to calculate the score once the play of the hand has finished, one is Rubber Bridge, and the other is International Match Points.

3.2 The Bidding Phase

The objective in a game of Bridge is to bid and make a contract. Each bid a player makes has a number and a denomination. Each player has thirteen cards, and there are thirteen rounds in the play of the hand. The number on the bid states how many rounds of play that the partnership will need to win. Regardless of which bid a player has established, they will still need to win a majority of the tricks. The numbering has this in mind. The number of tricks they will need to win is six plus the number on the bid, so in the case of the contract 1S, the partnership will have to win seven tricks. If the bid was 7S, the partnership would have to win all thirteen tricks. The denomination can be any of the suits, or 'NT' which stands for No Trumps. The denomination of the final bid made at the end of bidding will establish which suit should be trumps.

When a bid is made it must be a higher suit at the same level, or any suit at a higher level than one made previously. The bidding order is clubs, diamonds, hearts, spades, then no trump. So for instance, the bidding could go 1C, 1H, 1NT, 2D. Also, the lowest bid that can be made is 1C, and the highest is 7NT. Players also have the option to bid double and redouble. A double has the effect of increasing the trick score if the contract makes and the increasing the trick penalty if it fails. A redouble increases the bonus or penalty even further. A player can only give a double bid if the last bid made other than pass was their opponents. Also, a player can only redouble if the opponent attempted to make a double bid on their previous bid. A double is usually used by a player if they do not believe their opponents can make their contract. Finally, if all four players pass initially with no bids, the cards are reshuffled and a new game is played.

3.3 The Play of the Hand

At the end of the bidding, one player will become declarer. Declarer is the first member of the partnership that bid whatever suit the contract is now in. The declarer has control of both their hand and that of their partner during the play. Their partner is then on referred to as Dummy, and they simply follow the instructions of the declarer. The player sitting on the declarers left makes the opening lead, placing a card on the table so all the other players can see it. For all the other tricks, the person who won the previous trick has the lead.

After the opening lead, the dummy displays their hand to the entire table, arranging the cards in suits with trumps on the right hand side. Their duty is simply to comply with whatever declarer's instructions are as to what they should play. On any lead, the other players will have to follow the same suit as the one that has already bid so long as they can. If they do not have any card in that suit, they may play any card in their hand. Unless the card they play is in the trumps suit, it will automatically be discarded. If the card played is in trumps, then it can only be defeated by a card higher than it in the trump suit. When a player is leading, they can play any card in their hand.

The card that wins the trick is the highest card dealt in the suit lead, unless a card in trumps is dealt. On any trick containing a trump, the highest trump wins the trick. At the end of each trick, if the trick is won for the partnership then the players put their card vertically face down. If the trick was lost, then they place it horizontally face down. At the end of all three rounds, the cards are counted up. A contract is made perfectly if they won exactly the number of tricks they said they would. A partnership wins an overtrick for every trick they won on top of the established contract. The number of undertricks is the number of tricks the partnership was short of making the contract by.

3.4 The value of a hand

To be able to know what to bid when information about the other players' hands is limited, players need a way of evaluating the strength of their own hand. As such, there is a points system used for assessing the strength of a hand. Simply, points are given to each of the high cards. An Ace is worth four points, King is three, Queen is Two, Jack is 1, and the other cards are not worth any points. As such, there are forty high card points in the entire deck.

Additionally, the length of suits helps the trick taking potential of the hand. Some players choose to adjust their score based on suit length. If they have a void suit (one with no cards) they add three, if they have a suit with only one card, they add two, and a suit with only two cards adds one. However, players do not count a short suit in partner's main suit as an asset, and they do not count short suits as assets for no-trump bids. This scoring scheme only comes into play once it has been established there is a good fit on one of the suits for the two partners. Points are then used to judge how high to bid.

3.5 Conventions

The most complex portion of the bidding phase is the conventions. To reach an optimum contract without any undertricks or overtricks, partners have to exchange information. Without conventions, bids would just give a vague inkling of overall hand strength and suit preference. Conventions are used so that the partner will know unambiguously what a player's hand looks like. Each player can even adopt their own set of conventions, and can potentially come up with their own if they so wish. The only requirement is that they need to inform their partner what their bidding conventions are, and also have to say what they are if their opponents ask. A rule based Bridge playing program would have its own set of conventions encoded in, and have documentation explaining the rules used. Without knowing the conventions, it is difficult to tell what another player's hand means. One example is two commonly used conventions which are mutually exclusive, the Strong Twos and Weak Twos. Strong Twos is a convention where players with a particularly strong hand in a particular suit will open at the two level instead of the one level to indicate extra strength in their hand. Weak Twos are used when a player actually has a hand too weak to open at the one level, but they have extra length in a particular suit. A Strong Two seems like it would be more intuitive, but the Weak Two is still very commonly used. As the player is unable to make a bid at the one level, it is likely that the opposing partnership will be able to make a contract, and opening at the two level will make it harder for them to exchange information about their hands and to reach an optimum contract. Both conventions are useful, but are obviously exclusive. They are identical looking bids that will mean very different things.

A single partnership can have two players using different conventions. All that is required is that each player knows how they should interpret their partner's bids. Some conventions, such as Gerber or Blackwood, ask for specific responses from the partner. If the partner does not use that convention, then that bid should not be made as there is a possibility that the partner does not know how to properly respond.

3.6 Online Play

There are several free and subscription based Bridge servers available to play on over the internet. As mentioned previously Jonathan Rubin's Casper was the main driving motivation for attempting this project. In that project, after Casper had been trained against Poker playing programs developed at the University of Alberta, it was then moved online and played against real people for both fake and real money. One long term goal in developing a CBR based Bridge playing program would be to also take it online.

Currently, the largest online Bridge service would be Bridge Base Online³¹. It has over a hundred thousand active and registered users, and generally has several thousand people online at any given time³². Bridge Base Online allows for a variety of scoring systems and tournament modes, and also offers the facility for players to play online for real money.

Bridge Base Online encourages casual play between strangers. To accommodate the different bidding systems used by different players, players are required to fill out a convention card detailing the list of conventions used.

³¹ <http://www.bridgebase.com/>

³² <http://online.bridgebase.com/intro/introduction.php>

Convention cards - Stock - SAYC - Standard American Yellow Card			
List CCs New CC Delete CC Save changes Close			
My Favorite <input type="checkbox"/>		Partner <input type="text"/>	Title SAYC - Standard American Yellow Card
Doubles	Notrump Overcalls	System Summary	
Negative->2IS	1NT=15-18 (system on) 2NT=unusual (2 lowest unbid)	SAYC - Standard American Yellow Card	
Simple Overcalls	Over 1NT Openings	Notrump Openings	
8-16+ HCP; new suit=non-forcing; jump raise=weak; cuebid=limit raise+	2IC/D/H/S=natural; DBL=penalty	1NT=15-17: Stayman, Jacoby, 2IS->3IC, 3IC/D=6+ inv, 3H/3IS=6+ slam try; 2NT=20-21; 3NT=25-27	
Jump Overcalls	Over Takeout Doubles	Major Suit Openings	Minor Suit Openings
Weak	New suit forcing at 1-level; RDBL=10+ HCP; 2NT=limit raise+; jump raise=weak; jump shift=weak;	1H/S=5+; Jacoby 2NT; 3NT=16-18	1IC/D=3+; 2NT=13-15; 3NT=16-18
Direct Cuebid	Slam Bidding	2-Level Openings	
Michaels	Blackwood; Gerber; Grand slam force; DOPI; DEPO	2IC=strong; 2ID=waiting 2ID/H/S=weak; 2NT=forcing; new suits forcing	
Defensive Carding	Other important notes		
Attitude: high=encouraging; low=discouraging Count: high-low=even; low-high=odd; 4th best leads; Ace from AKx versus suits	4th suit forcing		

As this figure shows, the conventions that players use are heavily detailed, and can be easily modified by any player who wishes to use a different system.

3.7 Bridge Probabilities

An important part of Bridge is mathematical probabilities. To decide on which strategy is going to have the highest likelihood of success, a declarer requires at least a rudimentary knowledge of bridge probabilities. The following are some a priori probabilities about how the deck can be dealt four ways.

High Card Points:

The Following table shows the probability (as a percentage) of a freshly dealt hand having a certain number of high card points. The actual maximum number of high card points a player could possibly have is thirty seven, but this table only goes up to twenty three as the possibility of getting any higher than that is so incredibly small.

hcp	Probability	hcp	Probability	hcp	Probability
0	0.4%	8	8.9%	16	3.3%
1	0.8%	9	9.4%	17	2.3%
2	1.3%	10	9.4%	18	1.6%
3	2.5%	11	8.9%	19	1.1%
4	3.3%	12	8.0%	20	0.6%
5	5.2%	13	7.0%	21	0.4%
6	6.6%	14	5.6%	22	0.2%
7	8.0%	15	4.5%	23	0.1%

³³

The table above shows that the probability of falling within the point range of 7-12 points is around 52%. That is only five possible point scores out of forty seven.

Hand Shape:

The following table shows the probability (as a percentage) of a fresh hand having a certain pattern. The # column corresponds to the number of different suit permutations there can be to give that particular shape.

³³ Table taken from www.wikipedia.org/Bridge_Probabilities/

Patter n	Probability	#	Patter n	Probability	#	Pattern	Probability	#
4-4-3-2	21.55%	12	5-5-3-0	0.90%	12	9-2-1-1	0.018%	12
5-3-3-2	15.52%	12	6-5-1-1	0.71%	12	9-3-1-0	0.010%	24
5-4-3-1	12.93%	24	6-5-2-0	0.65%	24	9-2-2-0	0.0082%	12
5-4-2-2	10.58%	12	7-2-2-2	0.51%	4	7-6-0-0	0.0056%	12
4-3-3-3	10.54%	4	7-4-1-1	0.39%	12	8-5-0-0	0.0031%	12
6-3-2-2	5.64%	12	7-4-2-0	0.36%	24	10-2-1-0	0.0011%	24
6-4-2-1	4.70%	24	7-3-3-0	00.27%	12	9-4-0-0	0.0010%	12
6-3-3-1	3.45%	12	8-2-2-1	00.19%	12	10-1-1-1	0.0004%	4
5-5-2-1	3.17%	12	8-3-1-1	00.12%	12	10-3-0-0	0.00015%	12
4-4-4-1	2.99%	4	7-5-1-0	00.11%	24	11-1-1-0	0.00002%	12
7-3-2-1	1.88%	24	8-3-2-0	00.11%	24	11-2-0-0	0.00001%	12
6-4-3-0	1.33%	24	6-6-1-0	00.072%	12	12-1-0-0	0.0000003%	12
5-4-4-0	1.24%	12	8-4-1-0	00.045%	24			

13-0-0-

These thirty nine entries show all the different possible hand shapes that can occur. Five of them have more than a ten percent chance of occurring, and the most common is 4-4-3-2. As there are twelve different permutations for this shape, there is about a 1.7% chance that any given two hands will have the 4-4-3-2 shape and share the same hand distribution. This figure is much lower for most other hand shapes, for instance the chance of having two hands that are 5-4-3-1 with the same suit distribution is 0.54%.

Number of Possible Deals:

The next table shows the number of possible deals that can exist. As the difference between two suits that differ by one card, we might consider them indistinguishable if one has the two of hearts and the other has the three of hearts. An x corresponds to where we no longer care about the actual value of cards that low, but just their suit. So the first entry is where the only low card is those cards numbered with a two, and the last entry is where we do not care about any of the cards, but only their suits.

To consider just how big the possible number of deals is, if you were to ask how much area someone would need to display all possible deals where each deal only took up one square millimeter, the answer would be more than a hundred million times than the size of the earth.

³⁴ Table taken from www.wikipedia.org/Bridge_Probabilities/

Suit composition	Number of deals
AKQJT9876543x	53,644,737,765,488,792,839,237,440,000
AKQJT987654xx	7.811,544,503,918,790,990,995,915,520
AKQJT98765xxx	445,905,120,201,773,774,566,940,160
AKQJT9876xxxx	14,369,217,850,047,151,709,620,800
AKQJT987xxxxx	314,174,475,847,313,213,527,680
AKQJT98xxxxxxx	5,197,480,921,767,366,548,160
AKQJT9xxxxxxx	69,848,690,581,204,198,656
AKQJTxxxxxxx	800,827,437,699,287,808
AKQJxxxxxxx	8,110,864,720,503,360
AKQxxxxxxx x	74,424,657,938,928
AKxxxxxxx	630,343,600,320
Axxxxxxx	4,997,094,488
xxxxxxx	37,478,624

Chapter 4

First Attempt, *CBridge1*

4.1 *The proposal*

The intention of this project is to create a CBR system that applies itself to the game of Bridge. Jonathan Rubin's CBR system Casper mentioned in Chapter 2 showed that CBR can work effectively for imperfect information games such as Poker. The intention was to try and extend this to another card game Bridge using the same methodology.

The first attempt was *CBridge1*. The test was to see whether CBR could be applied to the simplest case of bidding, the opening bid. If it could not work here, then attempts to extend the system would be futile. The desire was to eventually have a full Bridge playing program that uses CBR successfully. This desire came out of the success of Casper, and seeing an overall lack of research into AI for Bridge compared to other games. Bridge Baron, one of the world's top Bridge playing programs is reportedly 50,000 lines of C code, which are mostly ad hoc rules for the bidding and play of the hand written by a domain expert, and still only plays at the Intermediate level. With this in mind, a bridge playing program based on CBR would be an achievement even if it played at a lower level, due to a CBR implementation having a significantly shorter development time.

The intention of *CBridge1* was to show how effective CBR could be with a simple implementation. As such, it uses a very straight forward application of CBR. *CBridge1* uses an existing case base and does not revise it with its own additions. The similarity function is simple, favouring an implementation that is quick to compute and

³⁵ Table taken from www.wikipedia.org/Bridge_Probabilities/

logically straight forward without the need for knowledge elicitation from a domain expert. *CBridge1* also forgoes the use of an adaptation function as at this point of the game, all bids are valid, and many of the bids are made under very specific conventions and are difficult to adapt in terms of the running time required to do so, and the amount of code required to implement such a function. To adequately create an adaptation function for Bridge bidding would almost certainly require domain expertise.

4.2 The CBridge1 Case Base

As *CBridge1* was designed with only the opening bid in mind, a fundamental problem arose. If *CBridge1* only made the opening bid, and did not play the play of the hand, how could you measure success? Without such a way to quantify the success of the opening bid it would be impossible to generate a case base from scratch by making random bids. So, the idea was to get an established case base. One thought would be to try and find a repository of actual games played by actual experts. However there are two problems with this even if such a repository was found, one would be trying to find a way to efficiently automate the extraction of the information from the source, the other is a problem to do with conventions. As mentioned in the previous chapter, each player adopts a series of conventions which is meant to tell their partner what they have in their hand. However, players typically adopt any set of conventions that they want. The only requirement is that their partner knows what conventions they use beforehand. This is relevant even on the opening bid. Some players state that 1NT means that they have 12-14 high card points, while others would have it at 14-16. If a case base was used from a large variety of different experts, then it is highly likely they would operate under different conventions, and the case base would be highly inconsistent as to what to bid in different situations. Additionally, if this program was to play against any humans, there would be a requirement to tell the partner what conventions are being used. If the case base is populated by games using a large variety of different conventions, then its play would be inconsistent, and it would be impossible to give a list of conventions being used. Because of this, all the cases must come from sources using identical conventions. The easiest way to go about this would be to take all the cases from the same source.

The source that was chosen for the case base is *WBridge5*. *WBridge5* is a Bridge playing computer program that has won the World Championships Computer Bridge in 2005, 2007, and 2008, and is freely available online³⁶. *WBridge5* has facilities for playing games against itself and keeping logs. This became the source of the case base for *CBridge1*. As there was no other measure that could be used for evaluating how good a move is, *CBridge1* makes the assumption that any decision made by *WBridge5* is a good one. When deciding on an opening bid, *CBridge1* looks at its database of *WBridge5* games and performs KNN to retrieve the most similar cases. Then, *CBridge1* just picks the most commonly made bid from the retrieved cases.

This method is clearly an attempt to make a program that mimics *WBridge5* without knowing its implementation. As it has no other way of evaluating how good a bid is, it will never consistently outperform *WBridge5* while only having *WBridge5* cases in the case base. In the best case scenario, it would make the same bids as the World Computer Bridge Champion. The reason for this is because it is very difficult to come up with a good way to measure how good a bid is. One alternative would be to see whether the opening bid made could have resulted in a contract the partnership could win during the play of the hand. This would prove problematic on a number of conventions, where stating certain bids demands a response from the partner regardless of the strength of their hand. Usually though in response to an opening bid a player can pass if they have less than five points. This would mean that the initially bid contract would have to be played. While this would indicate when a bid should certainly not be made, it is not a very good measure of when it should be. A bid is meant to convey information to the partner. Opening with 3S would tell the partner that the player has a very strong hand with spades as the strong suit. The partnership may in fact have 3S as their perfect contract, but this could be because the partner happens to hold a strong spade suit, not the player. Going by this information which tells partner of a very strong spade suit, they would likely try to raise the bidding further which would result in a failed contract. As such, simply going by the bid that *WBridge5* would suggest is simpler and likely more

³⁶ <http://perso.chello.fr/yvescostel/>

accurate. Although this may seem like it intends to copy *WBridge5*, if a CBR program such as *CBridge1* had access to a more varied case base that took cases from multiple sources but which stuck to a set series of conventions, then it could potentially play quite differently to *WBridge5*.

4.3 The *CBridge1* Similarity Function

As the *CBridge1* system was only playing the first opening bid, the only information available to the player is the cards in their own hand. As this project lacked the input of a domain expert, the implementation of the similarity function is fairly simplistic. As such, it can calculate the similarity between two cases faster than if there was a complex implementation. Additionally, if *CBridge1* succeeded with a relatively straightforward similarity function it would highlight the ability of CBR to work well with a minimum amount of domain expertise.

The focus of the similarity function implemented in *CBridge1* takes into account the two most important factors used by human bridge players when deciding on bidding. The first is high card points. Internally, all of the cases are read from the case base, and a list of Hand objects are created for each one. In the constructor of the Hand class, the number of high card points is calculated and stored as an instance variable. Each Hand object will have a value for high card points somewhere between zero and forty. A simple ratio of the two is calculated, and the weighting given to the attribute is fifty-one out of a possible one hundred. As such, the number of high card points is the attribute with the largest overall weight in the similarity function. This is because in a game of Bridge, high card points is the most important factor in deciding on whether or not to bid in the first place, and at which level. Most conventions have very strict requirements on the number of points that the player must have to make a certain bid, with usually only about one or two points of leeway.

The second factor is the shape of the hand. This factor takes into account the length of each individual suit. While high card points will usually determine whether or

not to bid, and at which level, the shape of the hand will usually state which suits should be bid. To do this, a simple ratio is calculated based on the lengths of each individual suit. Each ratio for each suit is given a weighting of nine out of a possible one hundred, so in total this gives shape of the hand a weighting of thirty-six.

The final factor taken into account in calculating similarity is the actual specific cards within each hand. Surprisingly, this is a factor not strongly taken into account by actual Bridge players. Generally, bids made do not describe particular cards within the hand and instead describe high card points and suit length. Some bids however, do count on specific cards, for instance under the Blackwood convention a bid of 4NT is used to ask how many aces the partner has, and the bid replied would contain this information. Additionally, players will still look at the specific cards in the suit to find things such as trick winners, or sequences. This factor is still fairly small compared to the overall number of high card points and the shape of the hand however. As such, for every card which belongs in both hands, 1% is added to the similarity score, for a maximum of 13%.

4.4 CBridge1 Initial Results

The initial *CBridge1* test ran with 2091 cases in the case base generated by *WBridge5*. The K value in the KNN algorithm was set to ten. *CBridge1* was tested by going through each individual case, removing it from the case base, and then seeing whether *CBridge1* could give the original bid based on the remaining cases in the case base.

Of the 2091 cases in the case base, 1094 of them returned the correct original bids, which is around 52% of the time. In an attempt to try and improve these results, different values of K were attempted, and the results are shown in the following table.

K	Bids made correctly	Percentage correct	Average Similarity
1	773	36.97%	85.20%
2	773	36.97%	83.85%
3	894	42.75%	82.88%
4	955	45.67%	82.11%
5	993	47.49%	81.45%
6	1034	49.45%	80.89%
7	1054	50.41%	80.39%
8	1069	51.12%	79.93%
9	1082	51.75%	79.50%
10	1094	52.12%	79.10%
11	1100	52.60%	78.72%
12	1109	53.04%	78.38%
13	1117	53.42%	78.06%
14	1111	53.13%	77.75%
15	1115	53.32%	77.47%
16	1114	53.28%	77.19%
17	1110	53.08%	76.92%
18	1112	53.28%	76.67%
19	1113	52.22%	76.43%
20	1114	53.28%	76.20%

These results are both disappointing and informative. It was expected that the first two values would share the same percentage correct, as when the two cases differed as to what bid to be made, the decision of the case with higher similarity is always favoured. What is surprising though is how cases even with 85% similarity use a different bid the majority of the time. Additionally, a very noticeable improvement can be observed very quickly by increasing the value of K, although these improvements plateau after a peak of 53.42% bids being made correctly where K is set to 13.

So what can be observed? The average similarity of cases retrieved starts dropping quite quickly. This would indicate that the problem space is quite diverse, and is expected as the number of possible deals is $52!/(13!)^4$. What seems most peculiar is that the information being provided by less similar cases is actually increasing the

performance of *CBridge1*. But then, the results plateau out at around 50%. But as the average similarity of retrieved cases continues to drop, the performance does not get any worse. In fact, this does not change even setting when the value of K is as high as sixty.

It is about this point that a painful hypothesis is realized. A bid of PASS is the correct bid roughly half the time. As the value of K increases, a larger number of irrelevant cases are looked at. Roughly half of these will pass, and since the remaining hands will be on a variety of different bids, the pass will become the single majority. This hypothesis was tested, and the following table shows the results:

K	Bids made correctly	Percentage correct	Number of Pass Bids
1	773	36.97%	1127
2	773	36.97%	1127
3	894	42.75%	1391
4	955	45.67%	1517
5	993	47.49%	1610

6	1034	49.45%	1672
7	1054	50.41%	1722
8	1069	51.12%	1755
9	1082	51.75%	1790
10	1094	52.12%	1825
11	1100	52.60%	1845
12	1109	53.04%	1866
13	1117	53.42%	1880
14	1111	53.13%	1883
15	1115	53.32%	1900
16	1114	53.28%	1906
17	1110	53.08%	1921
18	1112	53.28%	1934
19	1113	52.22%	1936
20	1114	53.28%	1943

From this it is clearly evident the bid of PASS becomes more frequent as the value of K increases. For 1127 of the cases, PASS is the correct bid, which is around 53% of the cases, and which also happens to be the peak of *CBridge1*'s performance. In short, when K reaches a sufficiently high level, *CBridge1* will bid PASS 100% of the time.

Further investigation shows that the number of bids made correctly that are not pass bids is exceedingly low. When K is 1, the number of correct non-pass bids is 142, which is a mere 14.6% of the 970 cases that require a bid other than PASS. As K increases, the number of non-PASS cases solved decreases.

4.5 How to improve CBridge1?

Unfortunately, *CBridge1* performance is not at an acceptable level even for just the opening move. A Bridge program that simply passes every move regardless of hand would not be classed as an intelligent opponent. There are a number of ways in which the program could be modified which may improve performance. First already shown and discussed is changing the value of K. By increasing K there was an obvious increase in the number of correct bids made, but in reality this was no improvement at all as it became more and more inclined just to pass. The next obvious solution would be

increasing the size of the case base. It would be expected that the performance of *CBridge1* would increase with a larger case base with more coverage of the problem space, but then query time for the case base will be an issue. Even with 2091 cases, the time it takes to perform leave one out testing is noticeable. To perform leave one out testing requires going over the case base of length n , n times over. An increase in the size of the case base will make meaningful experimentation difficult. Additionally, it took several weeks of automated game playing for the case base to be generated, as it took roughly five minutes for a single game to be played, and *WBridge5* crashed frequently. While it is likely that 2091 cases is simply too few for *CBridge1* to work competently, it is unknown whether the case base could ever be large enough while still being manageable to get adequate case coverage. As noted previously, the size of the domain is exceedingly large with $52!/(13!)^4$ different possible deals, and a case base with 2091 cases is unlikely to catch many of the outlying cases. However, the probable distribution of the hands indicate that even with 2091 cases there is still likely to be a lot of very similar cases. About half the cases fall between the point range 7-12, and roughly twenty percent will have the distribution 4-4-3-2. As such, 2091 cases should be able to cover at least the more common hand distributions.

If we are to ignore increasing the size of the case base, this leaves two main ways to improve the performance of *CBridge1*. One would be modifying the similarity function to give a more accurate account of what factors are important in bidding, and the other would be to incorporate an adaptation function. Both would certainly need a degree of domain expertise to implement well. One possibility for the adaptation function would be to see whether the proposal is a suit bid and the suit matches the longest suit of that hand, and then adapting that response to bid the longest suit of the current hand instead. In the majority of bids for the opening hand this would be a correct adaptation, but under a lot of conventions this would be a misinterpretation of the intention of the bid, particularly if it took place somewhere other than the opening bid.

A change in the similarity function could improve performance. One observation is that while simple ratios can be used to tell the cases apart, it is not the way actual

Bridge players would compare cases. Rather, they would say two hands are similar if they fell within a certain high card point range and have the same or similar distribution between the suits. What this range is will depend on the convention. For instance, a 1NT bid might mean a balanced hand between 12-14 points. Under the current similarity function, a balanced hand with thirteen points and a balanced hand with eleven points would be counted just as similar to a hand with twelve high card points. A 1NT bid would be correct for a bid at twelve of thirteen high card points, but not for one at eleven. Perhaps then rather than using a ratio, different ranges dependant on the conventions being used, such as 12-14, should become well defined in the similarity function. Then the similarity function would take into account exactly the factors used by conventions in determining the bid. This may also help compensate for a smaller case base.

4.6 *CBridge1a*

CBridge1a is the first attempt at improving *CBridge1*. *CBridge1a* takes *CBridge1* and adds a basic adaptation function. This function is the one mentioned previously, where all it does is it changes the suit bid if in the original hand it was the longest suit, and that suit differs from the longest suit in the current hand. This is in an attempt to try and break the majority that the PASS bid frequently got because the other bids were divided up into different suits. This function consolidates several of them so that more of the retrieved bids will be identical, and something other than PASS.

K	Bids made correctly	Percentage correct	Number of Pass Bids
1	779	37.25%	990
2	779	37.25%	990
3	907	43.38%	1223
4	968	46.29%	1330
5	1004	48.02%	1418
6	1054	50.41%	1476

7	1076	51.45%	1516
8	1098	52.51%	1545
9	1103	52.75%	1577
10	1116	53.37%	1602
11	1123	53.76%	1619
12	1131	54.09%	1639
13	1138	54.42%	1650
14	1136	54.32%	1651
15	1143	54.66%	1661
16	1142	54.62%	1667
17	1141	54.57%	1677
18	1146	54.81%	1684
19	1146	54.81%	1688
20	1146	54.81%	1694

The results show a slight improvement in the number of bids made correctly. Also, while the PASS bid still quickly dominates when K increases, on average there are a few hundred less pass bids than before. The decrease in PASS bids does not evenly match the number of correct non-PASS bids however the number of correct bids is still slightly improved.

Attempts at modifying the similarity function however did not meet with similar success. No other combination of weighted attributes matched the number of correct bids where K was low, and all had the problem of constantly bidding PASS when K was high. The first attempt was to set the similarity function to return 100% where the number of points was the same in both cases, and where they both had the same longest suit. This would certainly cause problems for any cases that were outliers with especially long suits or a high number of points, as the closest cases would return 0% similarity to them. But the idea was that perhaps in the area where there were a large number of cases, the bids

were being inferred from cases that were highly similar, but were just different enough to bid in an entirely different suit or give a pass. What occurred however was around 16% of correct bids where K was low, which steadily increased as once again *CBridge1* would start bidding PASS more frequently as K was increased. As K increased the average similarity of cases retrieved plummeted, and cases with 0% similarity were being used to inform the decision. Since roughly half of those cases would be a pass, it became significantly more likely for *CBridge1* to pass.

Other attempts were made trying to cluster cases into different point ranges based on conventions. These ranges were 0-5, which typically should always result in a pass, 6-10 which will result in a pass unless one suit is particularly long in which case it would be bid at the two level, 11-12 where suit bids could be made at the one level if they had good length, 13-15 where a bid should definitely be made either as a suit bid if one has good length or as a NT bid, and sixteen and above where either the suit should be bid at the one level, or an overcall to the three level is made to indicate a particularly strong hand. This unfortunately met with similar results. Even where K was low similarity was also low, around 70%, and the number of correct bids was around 18%. As K increased, average similarity decreased quickly, and the percentage of correct bids increased as *CBridge1* would increase the number of PASS bids until the number of correct bids reached 53%, which is also the percentage of cases where PASS is the correct bid to make. Attempts at changing the weightings, making distinctions in the case base about minor and major suits, and even putting a bias against all cases that suggest a PASS resulted in the same outcome. Low correct bids were made where K is low, and where K is high every bid becomes a PASS bid, allowing for 53% of the cases to bid correctly.

4.7 Conclusions on CBridge1

The poor performance of *CBridge1* can be attributed to several things. The first is poor case coverage. Simply, 2091 cases of Bridge games is not enough to provide even acceptable coverage of what bids should be made in the opening bid. On the same coin, increasing the size of the case base was difficult. The amount of time it took to generate

each case was prohibitive, as was the time it took to evaluate the entire case base. Despite this though, while the similarity function could find cases which did indeed look very similar, the bids which they could make would vary differently. Using that measure of similarity, it would be possible to have three hands, each differing from each other by one card, and still have them bid entirely differently. For instance, if one hand had eleven points and another twelve, the eleven point hand may likely pass, while the twelve point hand would bid. Two hands which differed in only one card may end up bidding in entirely different suits because the suit lengths were different. This means that cases even over 90% similar will give bids which are completely wrong for each other. Attempts at tightening the similarity function so that it strictly only gave value to those hands with the same number of points and the same suit length actually decreased performance due to poor case coverage. It is still possible perhaps for someone with good domain knowledge and a much larger case base than was used here to achieve results for the opening bid, but these initial results do cast a shadow of doubt over this.

The adaptation function did help improve the bidding of *CBridge1* slightly, but not enough to be the solution to the problems associated with the lack of case coverage. Its use may even have been inappropriate as it likely interfered with several bids made that was following a specific convention. Intuitively, adaptation should not be applied to Bridge in this manner. Many conventions are so situation specific that they should not be adapted at all, whereas others would be adapted in different ways. This adaptation function would only work on the opening hand, and even then the increased performance was negligible.

It is possible the reason why the results look so poor is because the way in which they are being tested was too harsh. For any given hand there might be several possible bids that could reasonably be made on it. It is likely that at least some of the bids proposed would have been acceptable as well, but were recorded as incorrect because it was not the same choice that *WBridge5* made. However, in Bridge quite often there is only one bid that could reasonably be made, and retrieving any other cases would be completely wrong.

Perhaps the most ironic thing about this project was that a CBR system for Bridge was supposed to take less time to develop and code than a rule based system. Developing a rule based Bridge playing program for the opening bid would actually be trivial and would consist of a series of very simple rules, based off of the conventions being used. The difficulty that these programs face in writing rules for the bidding phase is writing them for later rounds of bidding than the opening. These later rounds actually pose an even greater difficulty for CBR systems trying to solve the problem.

4.8 What If CBridge1 Had Been Successful?

A good question to ask would be what would have happened if *CBridge1* could acceptably play the opening hand? It is not impossible that if someone was to take the time to develop a much more extensive case base, and used a highly insightful similarity metric that they could have had greater success. However the next problem they would have is extending their CBR system to cover the entire bidding phase, and not just the opening bid. This would be exceedingly difficult for the following reasons.

Firstly, the number of cases required to cover differing hands is already staggering. When the program has to consider on top of this a chain of bidding the complexity becomes even greater. For each additional round of bidding that the CBR system has to consider, the total number of possible cases multiples thirty seven fold, one for each possible bid. The number of rounds that the bidding could go on for is not fixed either. Bidding ends after three players have passed in a row, and everyone must have had to have a chance to bid once.

Bidding could take the following form:

PASS PASS PASS PASS 1C PASS PASS DOUBLE PASS PASS DOUBLE PASS
PASS 1D PASS PASS DOUBLE PASS PASS DOUBLE PASS PASS 1H

This style of bidding could go on until 7NT is reached, which would take 311 bids to reach. Although this style of bidding would never seriously occur, and usually bidding finishes in far fewer bids, it means Bridge is not divided up nicely into different bidding

phases like Poker. For a CBR program to be useful it should still be able to perform even with prolonged bidding.

Second is the problem of knowing how to actually measure similarity on cases based on a chain of bidding. One (bad) possibility would be to just measure the hand similarity, ignore the bidding chain, and make the bid which the similar cases made in that particular round of bidding. This would completely ignore the bids made by both partner and opponent, and could not possibly perform well. Another way to deal with the problem might be to only compare the current problem with cases that have an identical chain of bidding. As pointed out previously, the share number of ways in which the cards can be dealt and the number of ways in which the bidding could go would mean this solution would only rarely be able to give any real answer at all. Perhaps some kind of ratio could be performed like was used in the opening bid? A straight line function that compared both the relative order of the suit bid, and the level on which it was bid could possibly be used. But the problem with this is that conventions often state specifically the exact suit which might be bid. For instance, the Gerber convention requires that the player bids in clubs. Comparing the Gerber bid with a bid at the same level in diamonds would make no sense at all. As it stands, there does not seem to be any sensible way to compare the similarity between chains of bidding.

The next problem is finding a way to actually evaluate the bids made by the chain of bidding. In *CBridge1* the measure was to just compare the bid made by *CBridge1* with that of *WBridge5*. This cannot work once a chain of bidding has been implemented. This is because subsequent bids are in reaction to the bids made by the partner and the opposition. Even if the player had the exact same cards in their hand as *WBridge5* did, there is no guarantee that the opposition would have the same hand distribution, and also whether they would make the same bids. Another possibility might be to play through the bidding phase, see what contract is made in the end, and then seeing whether the contract was won during the play of the hand. The problem with this is that it would then favour smaller bids with a greater chance of success, although the higher bids get more points. It would also ignore the chain of bids that were made to get there. If for instance

the program decided to jump bid to 4S even though it had neither points nor spades, the contract might be reached regardless because the partner was very strong in spades. So then the CBR system would measure that opening bid as a good answer, when it was actually nonsensical. The system could see another case in which the bidding was done flawlessly, however the game was lost due to inferior play during the play of the hand. That bidding then would be seen by the CBR system as having been incorrect. It is also possible that the chain of bidding was perfect up to a point, but it was the final bid, which was possibly established by the partner, which was incorrect. Then the system would mistakenly regard the entire chain of bidding as incorrect when it was only a small portion. What if the player has a particularly weak hand and cannot bid or make any contract at all? Would it be measured incorrect if the player passed and formed no contract? If not, then how would it react if the player passed when they could have made a contract?

To avoid misjudging the entire chain of bids based on a bad final contract, one possibility might be to try and evaluate the merit of each bid independently. Once again, this begs the question on how they could be evaluated? Well, the most obvious way might be to test whether each individual bid made would be able to form a winning contract in its own right. This would be acceptable if bids were made in a manner where each bid made up to the contract bid should be able to win in its own right. Conventions will often throw away this notion entirely. One example convention where this would result in a disaster is the Splinter bid. This actually shows a void or singleton in the suit bid! It is possible that the suit bid is one that the opposition is very strong in, in which case a contract of that nature would not be expected to win. But the bid made is still highly informative to the partner and should not be considered incorrect.

The next problem is conventions. If the case base is developed against different opponents and with different partners, then the conventions they are using will differ. This will result in inconsistencies on how the CBR system should actually respond. The case base would have to be developed with consistency over the conventions used. A larger issue however is trying to get the CBR system to actually have any sort of

knowledge as to what the conventions mean. If the CBR player bids a 4NT under the Blackwood convention, the partners response bid will reply indicating how many aces they have. This is useful not only for figuring out what contract should be reached, but is also highly informative for the play of the hand, so this information should be stored. This will be a problem in general with a CBR system playing bridge, because it would lazily attempt to mimic the actions taken in the case base, without having any real knowledge as to what conventions are and what they mean.

Lastly, because of the share size of the problem space for Bridge, the case base will probably have big difficulty dealing with any outliers. When it encounters one, the most similar cases will probably be too significantly different to give correct bidding. This will be an ongoing issue for a CBR system even if solutions were found for the other problems. Whenever an outlier is encountered, the bidding will likely suffer.

4.8 Final Thoughts and Potential for Future Research

In hindsight, a CBR system is inappropriate for the game of Bridge. It has already been noted that similar hands often had very different bids, and that the size of the domain was too prohibitive for developing a case base. But besides that, there was a more fundamental problem to developing this system. Real players do not actually play the opening hand by attempting to remember past hands. Rather, they have a set series of rules, based on their set of conventions, which should describe their hand to the other player. The inspiration for CBR is the way some people approach problems by simply thinking of past situations, and how they explain things with cases and not rules. But when it comes to Bridge, the logic is different. Even if this project had worked for the opening bid, it would have been very difficult to implement properly. Bidding is not about taking a hand as input, then using a function to determine which bid should be

used. Rather, the point of it is for two partners to describe their hands to each other so they can make a decision about what sort of contract they should go with.

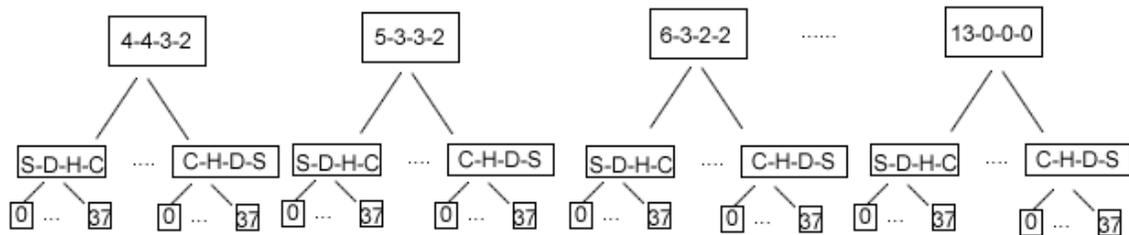
Besides the opening bid, doing so is not a trivial matter of following rules. Players trying to form a contract have to deal with opponent players making bids which are there to disrupt communication between the players. Players also sometimes have to attempt to communicate to each other and make a contract while the opposition are also bidding to attempt to establish a contract. Sometimes choices need to be made such as whether a player should support their partners suit, or declare their own. These strategies are a higher level abstraction than merely the bids themselves, and are based heavily on their set of conventions.

Someone attempting to apply CBR to Bridge again in the future will have a lot of difficulty, as this project has shown. But there are a number of things that a person could do that may, or may not, result in a CBR based program that can play good Bridge. The first observation is that the case base needs to be both large and diverse. The approach used in CBridge1 was to generate all the cases into a single case base. The problem with this is that it is difficult to find out how diverse the case coverage is by looking at this single case base. It also meant that testing was difficult. A more ideal situation would be to use a variety of different case bases, such that similar cases are all in the same case base. This way, each case base represents a different region of the problem space. Observing how many cases are in each case base would give a good indication of how complete the case base is.

The similarity function needs to be properly elicited by a domain expert. Ideally, it should be a well defined hierarchy which takes into account precisely what an expert will consider. It should work slightly differently than what is typical for a CBR program. Each node of the tree should represent each of the case bases referred to in the previous paragraph. The tree should only be used to find out which node the current case should be compared with. If the nodes are highly defined with very little room for cases to differ, then either a random subset or all the cases at that node could be retrieved for the KNN

comparison. Else, a second similarity function could be applied, such as the straight line function used in CBridge1, to return the most similar of the set of already highly similar cases in the node. If a similarity function used is similar to the one used in CBridge1, this means that each node could have around 2000-3000 cases and still be queried with leave one out testing.

To get complete coverage of the problem domain, cases could be generated for each of the nodes independently. So if a program like WBridge5 has the facility to generate random cases with certain constraints, then domain coverage for even outliers could be achieved. This is a hard thing to guarantee if the cases are not generated without any constraints put on them. The tougher the restraints placed, the more nodes will exist in the hierarchy. For instance, the following figure highlights one possible way a hierarchy could be made.



As there are thirty nine possible hand shapes, with twenty four ways to split the four suits, and each node having up to thirty seven points, this would mean 3552 different nodes in the tree. Populating this would definitely be very tedious without automation, But if someone managed to populate each node with around 2000 cases, then it would have excellent case coverage. For the opening hand, a similarity function would only need to retrieve the similar cases. As each node could be tested independently from the rest, leave one out testing would still be plausible. The biggest issue would be in generating a case base big enough to populate it, as this would require 7,104,000 cases to completely fill. More realistically many of these nodes would be combined. For instance, there could be a single node [0-5] instead of five different nodes. It is crucial however that the way these divisions are made is informed by experts.

This would likely manage to solve the problems of applying CBR to the opening hand in Bridge. But there are other more fundamental issues that would need to be addressed if CBR was used to try and play the game completely. One criticism of *CBridge1* is that it relied fully on *WBridge5*'s case base. While convenient, it may be better for *CBridge1* to generate its own case base so that it could play distinctly from *WBridge5*. If *CBridge1* is to put its own cases into the case base, then there needs to be a way for it to measure the success of a bid. As mentioned previously, this is a very difficult thing to do. In fact, the only bid which could be tested fairly would be the final contract making bid. The other bids are used to inform what that bid should be. Anyone interested in making a CBR based program will face this problem.

Additionally they will have to deal with the chain of bidding. The previous solution of tackling the overly large problem space of using the tree hierarchy will not straight forwardly work for a bidding chain. Given that there are thirty eight different bids that can be made, and the bidding can go on for quite some time, the possible number of bidding chains is hugely prohibitive. This cannot be solved with a straight forward application of CBR.

To get over these issues, an individual looking to solve this problem with CBR really should try to use a hybrid-CBR system that is informed by both cases and rules. To get over the huge branching factor of the possible bids that can be made, this information needs to be abstracted. For example, instead of choosing a suit and a bid, the choice might be something like "Show Partner strongest suit", and then a series of rules would dictate how they should go about that. The hard part of bidding for players is a choice between abstract choices such as supporting partner's suit, or forcing their own suit. As bids are received, rules can be used to interpret them. For instance, if the partner responds to a 1S bid with 2S, it might be established they have at least 5 points, and at least 3 spades. So rather than a chain of bidding, a case could be a list of information shared, some information about partners hand, and the current level of bidding. This level of information could perhaps be extracted from an existing case base, such as one from

WBridge5, but may very well be easier if it is generated solely by CBridge1. This would mean generating a case base by having CBridge1 play against itself. Because the bids made are partially informed by rules, they would not be entirely random. For instance, it may bid PASS too soon, or may press a suit too far, but it would not bid in a suit which neither it nor partner were interested in. Rather attempting at guessing on a contract, rules would be used to suggest a contract based on the information from players hand and from what is received from partner, and the case base would be used to decide on how best to inform partner about the hand, how far a contract should be pushed, and on whether to accept the contract proposed by partner. Such a program would no longer be purely CBR, but it would use the CBR methodology to inform on the sorts of things real Bridge players would actually look to past experience to decide on, while still preserving the knowledge of the game required to play at even a novice level.

Such a program will of course require a set of conventions. If it is generating its own case base, then by rights it can use any set of conventions the author so chooses, whether it be one of the standard convention lists, or whether it follows the conventions of a domain expert from which knowledge is being elicited. It also means that the two working in partnership will have to use the same conventions.

In conclusion, this project met with little success. It could have potentially worked with a larger case base or with a tree hierarchy as described above, but time constraints leading up to the conclusion of this project meant these possibilities could not be physically explored. This project does highlight the difficulties anyone attempting to apply CBR to Bridge will have, and also hopefully has put forward some suggestions as to how to solve these. It would be interesting if someone with access to a domain expert, or expert knowledge, could actually attempt a system as described here on possibility for further research.

