AI based Bridge playing program

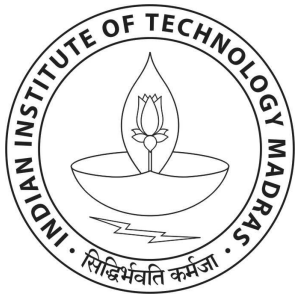
A Project Report

submitted by

VARSHITH BATHINI (CS16B034)

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **AI based Bridge playing program**, submitted by **Varshith Bathini (CS16B034)**, to the Indian Institute of Technology, Madras, for the award of the degree of **B.Tech**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Deepak Khemani
Research Guide
Professor
Dept. of CSE
IIT Madras, 600036

Place: Chennai

Date: 27th May 2020

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my guide Prof. Deepak Khemani for guiding me thoughtfully and efficiently through this project, giving me an opportunity to work at my own pace along my own lines, while providing me with very useful directions whenever necessary.

I would like to thank my friends/project-partners Joshua, Sri Harsha,Varun, Ashwin, Jithesh and Adarsh for being great sources of motivation and for providing encouragement and for all the discussions throughout the length of this project.

I would also like to thank Bo Haglund for allowing me to use his double dummy solver for the project and also guiding me on how to use it.

ABSTRACT

KEYWORDS: Declarer play; Double Dummy Solver; Cashing Winners.

Contract Bridge is a four-player partnership trick-taking game using a standard 52 card deck with thirteen tricks per deal. Bridge consists of two phases: Bidding phase and card playing phase. Compared to other games especially Chess and Alpha go, the research in computer bridge still remains immature. Although many expert level programs have been developed, bridge remains to be one of the very few games where computers are still not able to defeat human experts effectively.

This paper gives an overview of the research into the game of contract bridge, surveying the published literature on the card playing phase of the game and lastly discussing programs I Implemented for declarer play using double dummy solver and for cashing the top tricks/winners in the game of bridge. Also in this paper, we state our approach towards building a program to beat human experts effectively.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF FIGURES **v**

1 INTRODUCTION **vi**

1.1 Contract Bridge vii

1.1.1 Bidding viii

1.1.2 Card Playing viii

1.2 Problem Statement ix

1.2.1 Declarer play in Bridge ix

1.2.2 Cashing Winners/ Top Tricks x

1.3 Structure of the report x

2 LITERATURE REVIEW **xi**

2.1 Monte Carlo Simulation xi

2.2 Double Dummy Solver xviii

3 IMPLEMENTATION **xxii**

3.1 Declarer play in the game of bridge xxii

3.1.1 Program Inputs xxiii

3.1.2 Program Outputs xxiii

3.1.3 Double Dummy Solver Inputs xxiv

3.1.4 Double Dummy Solver Outputs xxiv

3.1.5 Representations xxv

3.1.6 Approach used to build the program xxvi

3.1.7 Algorithm used in the program xxvi
3.2 Cashing Winners/ Top tricks xxvii
3.2.1 Program Inputs xxvii
3.2.2 Program Output xxviii
3.2.3 Representations xxviii
3.2.4 Algorithm used in the program xxix

4 RESULTS **xxx**

5 SUMMARY **xxxiv**
5.1 Cashing Winners/ Top tricks xxxiv
5.2 Declarer play xxxiv

A Cashing Winners/ Top tricks examples **xxxvi**

B Declarer play examples **xxxix**

LIST OF FIGURES

2.1 Each entry is the number of deals successfully played by the programs in question (Ginsberg, 2001) xiv

2.2 GIB's performance as a declarer tested by Ginsberg (2001) xvi

2.3 GIB's performance as a defender tested by Ginsberg (2001) xvi

3.1 Encoding used in the program (Haglund, 2014) xxii

3.2 Playable cards and their respective scores(Program output) xxiv

3.3 Playable cards and their respective scores(Double Dummy output) . . . xxv

CHAPTER 1

INTRODUCTION

Cerebral games like Bridge, Chess, Go, Poker, Scrabble, Backgammon have always been a hallmark of Intelligence amongst humans and also as a benchmark testing beds for Artificial Intelligence. Building champion-level systems for each of these games have always been considered as milestones for Artificial Intelligence. A turning point in the history of Artificial Intelligence, so to speak, took place in 1996 and 1997 when IBM's famous supercomputer called DeepBlue, immortalised what computers can do when it played against then Chess GrandMaster Garry Kasparov. While Kasparov won the first six games in 1996, Deep Blue won the subsequent six games in 1997; which was the first defeat faced by a chess champion against a computer under competitive conditions. Deep Blue's win was symbolic of what is to come, it was a first indication that Artificial Intelligence could be a match for Human Intelligence and that brute computing force can challenge intellectual capability. Another more recent milestone being the DeepMind AlphaGo defeating Ke Jie, 2017 World Champion at the game of Go. The difference between DeepBlue and AlphaGo was mainly that DeepBlue evaluated millions of chess positions using the brute computing force but whereas AlphaGo on the other combined this with AI using Reinforcement learning and neural networks to closely mimic the human decision making behaviour.Mahalingam (2017)

But there is one frontier that still looms unconquered on the horizon. One game that humans still are superior at, One game that Artificial Intelligence still hasn't mastered yet to compete at that champion-level. This is Bridge, a game that brims with strategy, creative deceit and of course, tactics. It combines logic and reasoning with math. But what sets it apart from other games is the human communications, which can be an outright bluff or the truth. And also unlike other board games, bridge is the game of imperfect information which makes it even harder for a computer to play it.

So while Artificial Intelligence can beat Chess Grand masters and Go World Champions with ease it is nowhere close to outplaying the bridge champions, that's because computers can understand algorithms and AI but they are unable to understand underlying human behaviour in decision making and creative intent that comes to playing the game of Bridge. In this paper, we look at the game contract bridge.

1.1 Contract Bridge

Contract bridge, or simply bridge, is a trick-taking card game using a standard 52-card deck. In it's basic format, it is played by four players in two competing partnerships, with partners sitting opposite to each other around a table. North, East, South, West are the four positions with North-South being a team and East-West being in the other team.

The game is scored based on several deals, each of the deals progresses through four phases. The four phases are dealing, bidding, playing and scoring. Each player is dealt 13 cards from a standard 52-card deck, then the bidding phase begins where players call the bids in the auctions to win the contract, specifying how many tricks their team must win to receive points for that deal. During the bidding there is an exchange of information between the partners, the information is about their hands, length of suits, strength of suits although conventions for use during play also exist. The side winning auction is the declarer and the its partner is the dummy, while the other team are the defenders. The declaring side try fulfilling the contract and the defenders try stopping the declarer side from achieving the goal. The score of the deal is based on the tricks won, the contract won and many other factors which depend on the variation of the game being played. Bidding and Playing being the main phases of the game, we are going to discuss them in more detail.

1.1.1 Bidding

Bidding is the language of bridge. It's purpose is to convey information, strengths and weaknesses of the cards between partners. A bid is of the form nS where n is the level and S is a suit, which can be Spades, Hearts, Diamonds, Clubs or No Trump with No Trump having the highest priority and Clubs having the least priority. The level n indicates the number of tricks bidder can make more than 6. For example if n is 2 then the bidder is conveying to his/her partner that he/her can make at least 2+6=8 tricks. The suit indicates the bidder's choice to have which suit as the trump suit during the playing phase.

In the bidding phase, the dealer makes the first call, the call can either be a pass or a bid and the auction continues in clockwise direction and ends when there are three consecutive pass calls. The final bid becomes the contract. This means that one pair has contracted to make a certain number of tricks in particular suit or no trump.The winner of the contract is called the declarer while it's partner is called a dummy. The hand to the left of the declarer is the opening lead which means that hand starts playing the card. The dummy hand puts down the cards and the declarer has control over what card to play for the dummy.

During the bidding phase, partners try to communicate their strengths and weaknesses of the cards they hold and also the other team tries to prevent the other team from reaching an optimal contract. There are some conventions such as Standard American Yellow Card (SAYC), etc which are used to understand what a player is conveying by his/her bid.

1.1.2 Card Playing

The player from the declaring side who first bid the denomination named in the final contract becomes the declarer. The player to the left of the declarer is the one who starts the game. The partner of the declarer is dummy. When the opening lead lands on the table the dummy then lays his/her cards face up on the table, organized in column by

suit. The play proceeds clockwise, with each player required to follow suit if possible. Tricks are won by the highest trump or if there were none played the highest card of the led suit. The player who won the previous tricks leads to the next trick. The declarer has control of the dummy's cards and tells his partner which card to play during the dummy's turn.

The opening lead hand determines which suit the other three players must play. Each player must follow the suit, meaning they must play the card from the led suit. But if the hand doesn't have any cards from the led suit then any card can be played. In the no trump scenario the highest card played in the lead suit is the winner and the next turn goes to the winner. But on the other hand if a trump suit card is played during the trick then it gets the highest priority and it is declared the winner. If more than one trump suit cards are played then the highest card among them is the winner.Yallamati (2019)

1.2 Problem Statement

1.2.1 Declarer play in Bridge

Bridge is an interesting area of AI research. One of the outstanding areas involves improving the card-play in the game of bridge. Bridge being an imperfect information scenario where only 26 card positions are visible, with a very large branching factor, so the state space of all the possible plays to evaluate is of the order of 2.5×10^{21}, which is way too large to compute.

Double Dummy Solvers are a class of functions that operate on a perfect information scenario to estimate the value of the current state. When a double dummy solver is input with the complete state of the bridge game (all four hands) it outputs a value along with each of the playable cards. This value is an estimate of number of tricks that can be won if the said card is played. Which in turn basically indicates the best card to play. A very efficient open source double dummy solver which uses alpha-beta pruning is available for use thanks to Haglund (2014).

With branching factor order being 2.5×10^{21} which is almost impossible to compute and the availability of an efficient double dummy solver leads us to the idea of Monte Carlo method to play as a declarer in the game of bridge. Monte Carlo method's concept is to use randomness to solve the problems that might be deterministic in principle.

Our motive was to develop a program which can play as a declarer in the game of bridge using a double dummy solver and a hand generator. The Algorithms, Representations etc, used to build this program will be explained in detail in Chapter 3.

1.2.2 Cashing Winners/ Top Tricks

The first question to tackle while building a program to play the game of bridge is "how can the program cash the winners?". Winners are basically the cards which guarantee the trick win when played no matter what cards the other team plays. For example, consider the following single suit deal of cards. Here, the winners are A, K, Q since when any of those cards are played it guarantees the trick no matter what the other team plays.

N: A Q 2 3

W:? E:?

S: K 7 6

Our motive was to develop a module which comes up with a line of play to cash the maximum winners possible. The Algorithms, Representations etc., used in building this module will be discussed in detail in chapter 3.

1.3 Structure of the report

Chapter 2 provides an extensive literature survey. All the algorithms, representations used to build the programs are mentioned in chapter 3. Chapter 4 is for results and analysis and Chapter 5 is for summary.

CHAPTER 2

LITERATURE REVIEW

Now-a-days almost any chess program can win against any player with ease and this has been the case for at least the last two decades. With recent improvements one can argue a similar case concerning the game of Go. But this isn't the case with Bridge. Bridge is a much more complex game compared to Chess or Go. Just like there is a Bridge competition for humans there is a Bridge championship for computer programs called World Computer-Bridge Championship established in 1997. It is a competition where computer programs compete against each other in the game of bridge. Bridge Baron is the first winner of this competition in 1997 whereas Micro Bridge is the recent one in 2019. GIB, Jack, Wbridge5 being other winners of this competition through the years. WBridge won this competition three years in a row from 2016 to 2018. While there are very few articles/papers published on these programs, most of the designers of these programs are reluctant to share details about their code, but the case seems to be that most of these programs use similar approaches with slight variations. A few of the approaches are discussed in this chapter.

2.1 Monte Carlo Simulation

This method has been the most successful till date.Ginsberg (2001) in his paper mentions that GIB and Wbridge5 use similar methods in their programs. Monte Carlo methods or Monte Carlo Experiments are a broad class of computational algorithms that rely on repeated random sampling to obtain a numerical value.

In order to understand the card play phase of the bridge which is an imperfect information scenario it is important to first understand the bridge's perfect information scenario. Perfect information scenario means that each of the cards are visible to all the

players irrespective of the team. So each player can see all the other cards each of the other 3 players hold. In this case the problem is straightforward, a game tree can be built and the best play can be found out using a minimax algorithm. Unlike general minimax trees here the min nodes can have min children, since there are cases where a side wins the trick in the last play and has to lead the next trick since it is a winner. Branching factor is about four.Ginsberg (1999) suggests the following improvements to decrease the branching factor. Alpha-Beta pruning and transportation table being one of the improvements it decreases the branching factor to 1.7. Move ordering heuristic and partition search (Ginsberg, 1996) are some of the other improvements which can reduce the search space to about 18000 nodes per deal which is computable.

Assuming the existence of a double dummy solver which is a perfect-information card-play engine brings us to the idea of usage of Monte Carlo Simulation. One way in which we might use this double dummy solver to proceed in a realistic situation would be to deal the unseen cards in random, distributing the deal in such a way that the deal is consistent with information from bidding as well as the information from already played cards. These full information sample deals obtained can then be used as an input to a double dummy solver to analyze the results and decide which of the possible cards to play is the best to play. Averaging this over a large number of sample deals created allows us to deal with the imperfect information nature of the game of bridge. Levy was the first to suggest this idea of Monte Carlo Simulation in bridge but according to Ginsberg (2001) this method appears to have problems in practice (see below).

Algorithm 1 Monte Carlo card selection algorithm

Ensure: To select a move from a candidate set of M such moves:

1: Construct a set D of deals consistent with both the bidding and play of the deal thus far.

2: For each move m M and each deal d D, evaluate the double dummy result of making the move m in the deal d. Denote the score obtained by making this move s(m, d).

3: Return that m for which P d s(m, d) is maximal.

The Monte Carlo approach has its drawbacks which have been pointed out by a variety of authors such as Koller (1995), Frank and Basin (1998). The most obvious drawback is that this approach never suggests making an information gathering play. After all, the perfect-information variant on which the decision is based invariably assumes that the information will be available by the time the next decision is made. Instead the tendency is for the approach to simply defer important decisions; in many situations this may lead to information gathering inadvertently, but the amount of information acquired will generally be far less than other approaches might provide.

Ginsberg (2001) suggests the following example to show the drawbacks, suppose that on a particular deal, GIB has four possible lines of play to make its contract:

1. Line of play 1 works if queen of spades is present with the West side.
2. Line of play 2 works if queen of spades is present with the East side.
3. Line of play 3 defers the guess until later.
4. Line of play 4 works independent of who holds the queen of spades.

Assuming either side has equal probability to hold the queen of spades, the Monte Carlo analyzer will correctly conclude the first line of play works half the time while line of play 2 works the other half time. The third line of play however will be presumed to work all the time since the contract can still be made since the guess is deferred. On the other hand the fourth line of play works no matter what because the position of the queen of spades does not decide the line of play, since it is independent.

Nevertheless basing the card play on the Monte Carlo gives strong results, which can be approximated to the level of a human expert. The first program to adopt this method was GIB. Since then almost all the programs which play the game of bridge have adopted this algorithm.

To get an idea of how good this method works practically GIB, the first program to adopt this was tested in three ways by Ginsberg (2001)

1. Testing GIB against the set available benchmarks

2. Performance in a human championship designed to highlight the card-play

3. Statistical performance measured over a large set of deals.

For the first test GIB's card-playing strength was evaluated against Bridge Master by Ginsberg (2001).Bridge Master currently known by the name Bridge Base Online. Bridge Master was developed by Fred Gitelman who was a Canadian international. Bridge Master had over 180 preset deals divided into 5 levels. If you misplay the hand, Bridge Master moves the defenders hands to ensure the loss.

Bridge Master does not randomly generate the deals, It uses already predetermined deals. The drawback with this is that these 180 predetermined deals may not represent all the cases that a bridge player generally faces in real life. Each of the deals shows a lesson to be learnt. Generally good plays and rewarded and bad plays are punished by the Bridge Master.

As a benchmark Bridge Baron 6 was used. Bridge Baron took 10 seconds for playing each card where GIB took 90 seconds to play the whole deal. The Monte Carlo sample size used by the GIB program was 50. This information is critical on a small number of deals.

Here is how the two systems performed:

Level	**BB**	**GIB**
1	16	31
2	8	23
3	2	12
4	1	21
5	4	13
Total	33	100
	18.3%	55.6%

Figure 2.1: Each entry is the number of deals successfully played by the programs in question (Ginsberg, 2001)

GIB's mistakes are shown clearly,According to Ginsberg (2001) most of them are due to the fact that it fails in gathering information and problems which combine multiple chances.

GIB was invited to participate in an invitational event at the 1998 world bridge championships in France. The human participants were given 90 minutes to play each deal, although they were penalized slightly for playing slowly. GIB played each deal in about ten minutes, using a Monte Carlo sample size of 500; tests before the event indicated little or no improvement if gib were allotted more time. Michael Rosenberg, the eventual winner of the contest and the pre-tournament favorite, in fact made one more mistake than did Bart Bramley, the second place finisher. Rosenberg played just quickly enough that Bramley's accumulated time penalties gave Rosenberg the victory. The scoring method thus favors GIB slightly

Finally GIB was tested against the records from actual play by Ginsberg (2001). These records are available from high level competitions, these records help us to judge the frequency at which humans make errors on a double dummy basis. The following figure shows the frequency with which the player commits errors from the first trick leading up to the last trick. X-axis is the trick and Y-axis is the logarithm of errors committed.

According to Ginsberg (2001) this method of analysis favours GIB slightly and also failing to make an information gathering play is reflected in the figure because a double dummy solver shows the best play based on the information gathering.

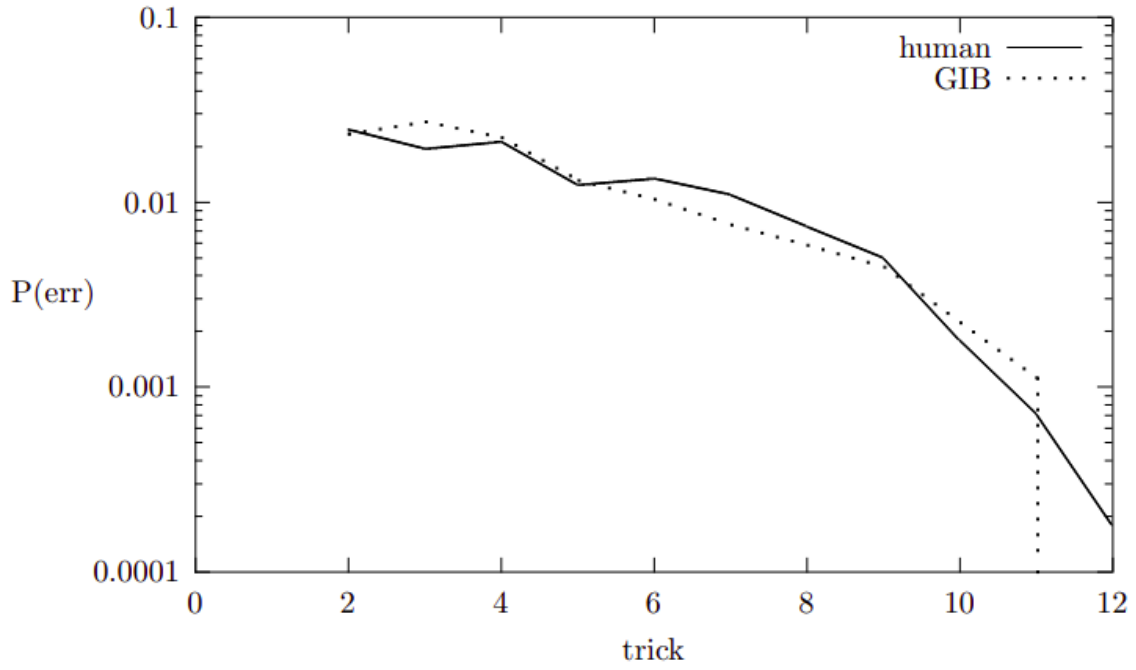


Figure 2.2: GIB's performance as a declarer tested by Ginsberg (2001)

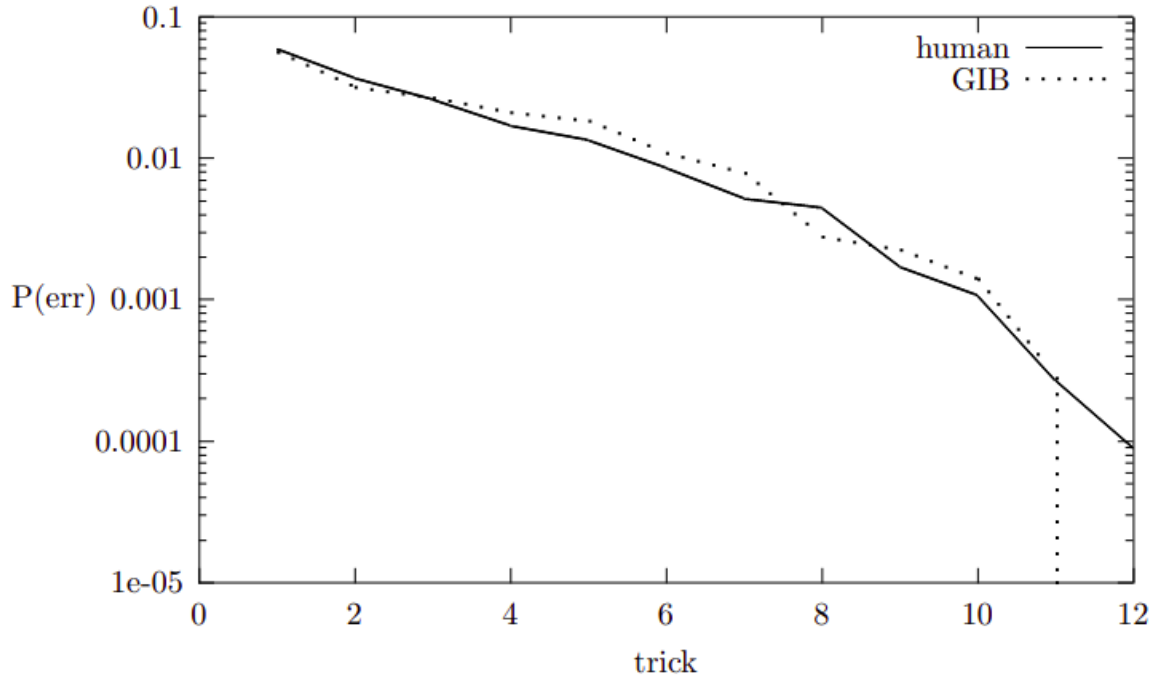


Figure 2.3: GIB's performance as a defender tested by Ginsberg (2001)

A modification was suggested to the monte carlo simulation algorithms that it is impractical to test each hypothetical decision against the module itself. Instead, GIB uses probabilities to give more weight to some deals. As an example, suppose GIB plays the 5♠.The analysis indicates that 80% of the time that the next player(say West) holds the K♠, it is a mistake for West not to play it. If West in fact does not play K♠, the Bayes' rule is used to adjust the probability of West holding the K♠. The probabilities are then modified further to include information revealed, and the adjusted probabilities are finally used to bias the Monte Carlo sample, replacing the evaluation ds(m, d) with wdds(m, d) where wd is the weight assigned to deal d. More heavily weighted deals thus have a larger impact on GIB's eventual decision.

2.2 Double Dummy Solver

Although the size of the game tree for searching the best line of play is huge as mentioned previously in the problem statement, using various search algorithms and some efficient moves ordering and pruning heuristics, most of the double-dummy deals can be solved in a reasonable time. Many other improvements are also suggested by Chang (1996).

Cazenave and Ventos (2019) have suggested a way on how to apply $\alpha\mu$ to the game of Bridge. $\alpha\mu$ is an anytime heuristic search algorithm for incomplete information games that assumes perfect information for the opponents.$\alpha\mu$ addresses the strategy fusion and non-locality problems encountered by Perfect Information Monte Carlo sampling.

There have been efforts to develop a double dummy solver using Artificial Neural Networks as well by Mossakowski and Ma´ndziuk (2004) but as mentioned in the paper published by them the results are not very positive and that their solver is not as efficient as the solver developed by Haglund (2014).

The algorithm presented below is the basic search algorithm used in the Haglund (2014) double dummy solver. The inputs are posPoint, Target, Depth (described in detail below). The algorithm uses minimax and backtracking. When it's our turn to play it uses value to be false when it's opponents chance to move it uses true. Using this with basic backtracking. It makes a move recursively till the game ends and backtracks with the answer to current move. It returns TRUE if the target is reached and returns FALSE if it doesn't reach the target.

The search parameters are:

posPoint: a pointer to a structure containing state information for the position (deal) to be searched eg. leading hand, played cards, trump suit etc.,

Target: the number of tricks the team must take.

Depth: the current search depth.

```
int Search(posPoint, target, depth) {
    if (depth==0) {
        tricks=Evaluate;
        return (tricks >= target ? TRUE : FALSE);
    }
    else {
        GenerateMoves;
        if (player_side_to_move) {
            value=FALSE;  moveExists=TRUE;
            while (moveExists) {
                Make;
                value=Search(posPoint, target, depth-1);
                Undo;
                if (value==TRUE) // Cutoff, current move is the best move
                    goto searchExit;
            }
        } // Opponents to move
        else {
            value=TRUE; moveExists=TRUE;
            while (moveExists) {
                Make;
                value=Search(posPoint, target, depth-1);
                Undo;
                if (value==FALSE) // Cutoff, current move is the best
                    goto searchExit;
            }
        }
    }
    searchExit: return value;
}
```

To find the maximum number of tricks the team can make with this deal. A binary search over the above algorithm from 1 to 13 is run to find the maximum no.of tricks for which the algorithm returns true.

GenerateMoves generates a list of alternative moves that can be played in the initial position whose state data is pointed to by posPoint.

```
g = guessed number of tricks for side of the player
iniDepth = number of cards to play minus 4
upperbound = 13;
lowerbound = 0;

do {
    if (g==lowerbound)
        tricks=g+1;
    else
        tricks=g;
    if ((Search(posPoint, tricks, iniDepth)==FALSE) {
        upperbound=tricks-1;
        g=upperbound;
    }
    else {
        lowerbound=tricks;
        g=lowerbound;
    }
}
while (lowerbound < upperbound);

g=maximum tricks to be won by side of player.
```

The double dummy solver that Haglund (2014) developed is basically a lot of improvements over this basic search algorithm. The improvements are using Ordering algorithm, Transportation table etc,. This version of Double Dummy Solver is very efficient, In Memory, Correctness and also time. It also uses multi-threading to make the computation even faster.

CHAPTER 3

IMPLEMENTATION

3.1 Declarer play in the game of bridge

Using the publicly available efficient Double Dummy Solver developed by Haglund (2014) and a Hand Generator based on Monte Carlo Simulation I developed a program which can play as a declarer in the game of bridge.

Encoding	Element	Value
Suit	Spades	0
	Hearts	1
	Diamonds	2
	Clubs	3
	NT	4
Hand	North	0
	East	1
	South	2
	West	3
Side	N-S	0
	E-W	1
Card	Bit 2	Rank of deuce
	...	
	Bit 13	Rank of king
	Bit 14	Rank of ace
Holding	A value of 16388 = 16384 + 4 is the encoding for the holding "A2" (ace and deuce). The two lowest bits are always zero.	
PBN	Example: W:T5.K4.652.A98542 K6.QJT976.QT7.Q6 432.A.AKJ93.JT73 AQJ987.8532.84.K	

Figure 3.1: Encoding used in the program (Haglund, 2014)

3.1.1 Program Inputs

Declarer : A number(0-3) based on encoding indicating which hand is the declarer.
Example: 0 if North is the Declarer.

Lead Hand: A number(0-3) based on encoding indicating which hand starts the play
Example: 1 if East is starting the play.

Trump Suit: A number(0-4) based on the encoding indicating what is the trump suit for the game
Example: 4 if there is no trump suit.

Deal: A string indicating the deal on which the game is played. This is in PBN format. PBN stands for Portable Bridge Notation.
Example: .63.AKQ987.A9732 A8654.KQ5.T.QJT6 J973.J98742.3.K4 KQT2.AT.J6542.85

3.1.2 Program Outputs

The final output of the program is the number of tricks the program won. This is considered to be optimal based on Monte Carlo simulation.

While each card is played by the program, it also outputs a score next to all the playable cards. The figure below shows the output from the program, the first column indicates the suit of the card (Spades(S),Hearts(H),Diamonds(D),Clubs(C)), whereas the second column indicates the card number and the final column indicates the score. The score here for a specific card is a number which is cumulative sum of all the double dummy solver outputs while performing Monte Carlo simulation. These scores indicates how good it is to play the said card.

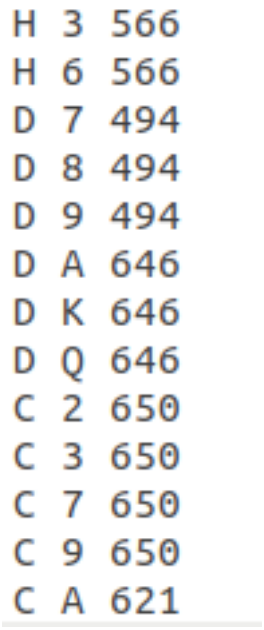


Figure 3.2: Playable cards and their respective scores(Program output)

3.1.3 Double Dummy Solver Inputs

Current Hand : A number(0-3) based on encoding indicating which hand has the current turn to play
Example: 1 if East is starting the play.

Deal: A string indicating the deal on which the game is played. This is in PBN format. PBN stands for Portable Bridge Notation.
Example: .63.AKQ987.A9732 A8654.KQ5.T.QJT6 J973.J98742.3.K4 KQT2.AT.J6542.85

Cards played in the current trick : The cards played before the current hand must be input.

3.1.4 Double Dummy Solver Outputs

Double Dummy Solver outputs each of the playable cards along with a score. In the figure below the first column indicates the suit of the card, the second column indicates the card number, whereas the third indicates the equivalent cards i.e, the card in the

second column and the ones in this column have the same score and the last column indicates the score. This score indicates the number of tricks the hand can make playing the said the card. Which in turn shows us which is the best card to play in the current situation.

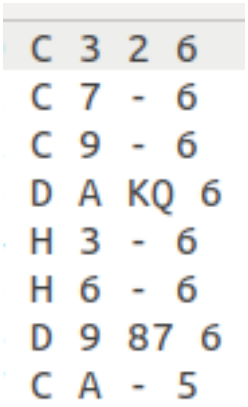


Figure 3.3: Playable cards and their respective scores(Double Dummy output)

3.1.5 Representations

All the code is written in C++, So all the variables used are C++ data types.
Suits: Suits are represented by an int data type. The value of this int data type is either 0 or 1 or 2 or 3.

0 indicates Spades

1 indicates Hearts

2 indicates Diamonds

3 indicates Clubs

Cards: Cards are represented using a pair of int and char data types. The int data type represents the suit the card belongs(0 - 3) to and the char data type represents the card

value(2 - A).

Example: 0,A in a pair of int and char and this data type indicates Ace of Spades.

Hands: Hands are represented using an int variable. The value of this int data type is either 0 or 1 or 2 or 3.

0 indicates North

1 indicates East

2 indicates South

3 indicates West

Cards held by each hand: The cards held by each hand as a whole are represented using a string. This string is in PBN format.

Example: AQ96.AQ7.A8.T753 , Cards in each suit starting from spades separated by '.'

Cards played by each hand: It is similar to Cards held by each hand, a string in PBN format.

3.1.6 Approach used to build the program

Double Dummy Solver can be used to find the best card to play if the state of the game is known i.e, if the whole deal is known. But bridge being an imperfect scenario i.e, only declarer and dummy hands are visible. So, we Monte Carlo simulate the defense hands with a sample size of 100 and feed the double dummy solver with these 100 possible deals and find the best card to play which has the best cumulative sum over these 100 hands.

3.1.7 Algorithm used in the program

Inputs: Deal, Lead hand, Declarer hand and Trump suit.

Algorithm 2 Monte Carlo Simulation using DD solver

Ensure: Maximize the number of tricks won

While there are left cards to play:
    If the hand to play is dummy or declarer:
        Call hand generator to generate 100 consistent deals based on constraints**
        For each of the generated deals call Double Dummy Solver
        Accumulate all the Double Dummy Solver outputs and play the card with the highest score
    Else if the hand to play is defender:
        Take the card to play as input
    If four cards are played in the trick:
        Find the winner among four hands to play and change the hand to play to the winner.
END: Output the number of tricks won by Declarer-Dummy team.

** Constraints - The inputs we get from bidding and previously played cards

3.2 Cashing Winners/ Top tricks

As mentioned before in the problem statement winners are basically the cards which guarantee the trick win when played no matter what cards the other team plays.

Using a basic complete search algorithm I built a program which takes the declarer and dummy hands as inputs and outputs a line of play to cash the maximum number of Winners.

3.2.1 Program Inputs

Declarer and Dummy hands: Declarer and Dummy hands in the form of a string in PBN format.
Example: .63.AKQ987.A9732 A8654.KQ5.T.QJT6

3.2.2 Program Output

The program outputs a line of play to cash the maximum number of Winners.

Example: Q4.K54.KJ54.A654 J32.A32.AQ32.K32 max top tricks: 8

South plays ♣2 North plays ♣A

North plays ♣4 South plays ♣K

South plays ♦2 North plays ♦J

North plays ♦4 South plays ♦Q

South plays ♦3 North plays ♦K

North plays ♦5 South plays ♦A

South plays ♥2 North plays ♥K

North plays ♥4 South plays ♥A

More examples with better visualization are presented in Appendix A.

3.2.3 Representations

All the representations used are same as used in the declarer play program(3.1.5).

3.2.4 Algorithm used in the program

Algorithm 3 Cash Top tricks in declarer play

Ensure: Maximize the number of top tricks made

1: Construct a Game tree between declarer and dummy with the following conditions.

2: During a trick, if one player plays a low card(not rank-1), the other player must play a rank-1 card.

3: During a trick, if one player plays a rank-1 card, the other player can play any card at their disposal

4: A Node is considered to be a leaf if there are no rank-1 cards with both declarer and dummy.

5: The longest path from the root to a leaf is considered to be the line of play to maximize the number of top tricks.

CHAPTER 4

RESULTS

The declarer play program I built doesn't have a bidding system included which makes it harder to test it against other bridge programs in the market. Bridge Master is an online bot available on Bridge Base online. There are 5 levels in the bridge master spanning from beginner to expert level. Each level has about 60-80 deals. As there is no automation tool available the testing had to be done manually by feeding the cards played by my program into bridge master manually and feeding back the cards played by bridge master into my program. So the sample set is small. In Bridge Master there is no bidding involved, there is a predetermined bid which the program has to play on. The following are results of some of the deals played. In all the deals played South hand is the declarer and West hand was the lead hand. So North-South was played by the program where as East-West team was played by bridge Master

Level	Deal	Contract	Result
1	N-S: Q4.K54.KJ54.A654 J32.A32.AQ32.K32 E-W: A65.QJT.987.9876 KT987.9876.QJT.10	3NT	Contract Made
1	N-S: 432.AK4.5432.KQ4 AKQ.Q32.KQJT.A32 E-W: 765.765.765.9876 JT98.JT98.JT98.A	6NT	Contract Made
1	N-S: 87.43.A76543.432 AKQJT9.AK2.K2.AK E-W: 65.QT5.J765.QJT9 432.J9876.QT98.8	7S	Contract Made
1	N-S: KQ.JT98.K987.AKJ 32.AK32.AQJT.QT9 E-W: A7654.Q654.32.32 JT98.7.654.87654	6H	Contract Made
1	N-S: 65.432.32.AKQJT9 A432.AK8765.AK.2 E-W: KQ7.9.7654.87654 JT98.QJT.QJT98.3	6H	Contract Made
2	N-S: A.AK.5432.QJT987 KQJ.J432.AQJT.32 E-W: T5432.765.AK.976 9876.QT98.654.K8	3NT	Contract Made

2	N-S: AJ9.KQJ.AQJ.5432 432.A32.K32.AKQJ E-W: K65.T987.T987.T9 QT87.654.654.876	6NT	Contract Failed (11-2)
2	N-S: A5432.765.3.K432 QJT98.432.AK2.A5 E-W: 7.AT98.QJT.T9876 K6.KQJ.987654.QJ	4S	Contract Made
2	N-S: 65.A65432.QJT.32 432.QJT98.K2.AKQ E-W: QJT9.K.A987.JT98 AK87.7.6543.7654	4H	Contract Failed (1-4)
2	N-S: QJT.65.KJ5432.65 A32.A432.AQ.A432 E-W: 54.K87.T987.K987 K9876.QJT9.6.QJT	3NT	Contract Made
3	N-S: 987.543.K5432.K6 AKQJT.A2.A5432.A E-W: 65.876.QJT98.876 432.KQJT9.7.QJT9	6S	Contract Failed (8-2)
3	N-S: A32.AK.KQT876.43 KQJ.J9876.A2.AK2 E-W: T98.54.765.J9543 7654.QT32.QJT98.	6NT	Contract Failed (8-2)
3	N-S: KQ.432.65.AK5432 A2.AKQJT.A432.76 E-W: 76543.765.KQJT.8 JT98.98.987.QJT9	6H	Contract Failed (6-2)
3	N-S: 876.765432.A2.A2 A5432..KQJT.KQJT E-W: KQJ9.QJT98.98.98 T.AK.76543.76543	4S	Contract Failed (8-4)
3	N-S: 432.432.Q432.KQJ AQJ.QJ9876.5.A32 E-W: 876.A.A876.T7654 KT95.KT5.KJT9.98	4H	Contract Failed (4-4)
4	N-S: K432.A432.32.K43 AQ8765.K987.A.A2 E-W: .QJT5.QJT5.87654 JT9.6.8765.KQJT9	6S	Contract Made
4	N-S: JT9.5432.AKQ.AKQ AKQ32.K.432.JT98 E-W: 8765.A9876.32.JT 4.QJT.7654.98765	6S	Contract Made

4	N-S: 432.K3.KQ54.K432 QJT9.A2.A32.AJ98 E-W: AK5.Q7654.76.865 876.JT98.JT98.T7	3NT	Contract Failed (8-5)
4	N-S: AK76.432.K432.32 JT9832.AK.AJ8.AQ E-W: Q5.T987.65.K7654 4.QJ65.QT97.JT98	6S	Contract Failed (3-2)
4	N-S: 543.K54.6543.987 AKQ2.A32.2.AKQJT E-W: 76.J98.AQJT9.432 JT98.QT76.K87.65	5C	Contract Made
5	N-S: K3.A65.Q7.AK6543 A2.432.A65432.J2 E-W: 7654.JT98.T98.KJ QJT98.KQ7.Q7.T98	3NT	Contract Made
5	N-S: 5432.KT98.Q432.6 A.QJ2.KT.AQJ5432 E-W: K6.543.A98765.K7 QJT98.A76.J.T98	5C	Contract Made
5	N-S: AQ54.Q95.8765.T4 32.AKJ432.KJ.J32 E-W: 76.T76.AQ42.AKQ9 KJT98.8.T93.8765	4H	Contract Failed (8-4)
5	N-S: KQJ.K765.Q43.A43 A32.AQJT98.K2.Q2 E-W: T98765.2.765.765 4.43.AJT98.KJT98	6S	Contract Failed (11-2)
5	N-S: AK54.765.654.432 32.AK432.AQ32.KQ E-W: QJT.QJT.KJT9.A65 9876.98.87.JT987	4H	Contract Made

Table 4.1: Five deals played by Program vs Bridge Master in each level

Examples with line by line play with better visualization are presented in Appendix B.

Level	No.of deals played	No.of deals won	Win Percentage
1	5	5	100
2	5	3	60
3	5	0	0
4	5	3	60
5	5	3	60
Total	25	14	56

Table 4.2: Level-wise comparison

The sample size is quit small to be conclusive but the programs' mistakes are illuminating. While some of them involve failing to gather information, most are problems combining multiple chances. As the level increases in the Bridge Master the deals involve combining a variety of possibly winning options and that is why the program performs poorly in the level 3. As the levels progress further most of the deals involve complex end positions which the program seems to solve better.

CHAPTER 5

SUMMARY

5.1 Cashing Winners/ Top tricks

The implemented program to cashing winners uses a basic complete search algorithm which implies a total correctness. But the problem with the implemented program is the time efficiency. For example consider the case where the south hand has all cards in a single suit, let it be spades and the north hand too has all cards in a single suit, let it be Hearts. This is the worst case for time. The time taken in this case is of order 13^{26} which is way too large. So for future work decreasing the search sample size should be the main focus. Some ideas which can help in decreasing the search sample size are Adding the idea of suit breaks, Instead of doing a complete search adding a part of logic that a bridge player might use to decrease the search sample size. With a reduced time taken by the program it can be used as a very efficient module in the bridge program to be built in future to give a line of play to cash the maximum winners.

5.2 Declarer play

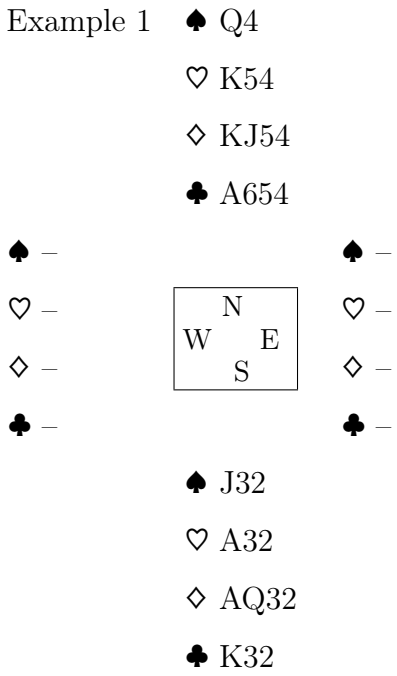
The implemented program to play as a declarer in a game of bridge is still far from being perfect, being at that human expert level because it lacks that thought process that human bridge players possess. It doesn't reason its plays which is a major drawback for retrospection. It just gives the best card to play. This program can be used as a practice bot from people learning bridge. It will give a tough competition but it is just not that expert level. There are some novice cases that a Double Dummy Solver can't catch but any bridge player would.

This program can be used as a benchmark for bridge programs that are going to be built in future. With slight changes in the code this program can be able to tell the efficiency of the plan that other modules developed by my friends come up with.

Lack of having an UI makes it harder for a bridge player to play with the program, So building an UI to port this program to and also for bridge programs to be built in future must be the priority. As far as efficiency of the program is concerned it can be improved by using a better hand generating technique. Using more,better constraints from bidding helps to generate better hands which represent the non visible hands in a better way. Currently all the hands generated have equal weight, exploring a bit more on using different weights for the hands generated might produce better results. Also exploring a bit more on how to tackle the case where multiple cards have the same score then which card to play.

APPENDIX A

Cashing Winners/ Top tricks examples



Output: max top tricks: 8

South plays ♣2 North plays ♣A

North plays ♣4 South plays ♣K

South plays ♦2 North plays ♦J

North plays ♦4 South plays ♦Q

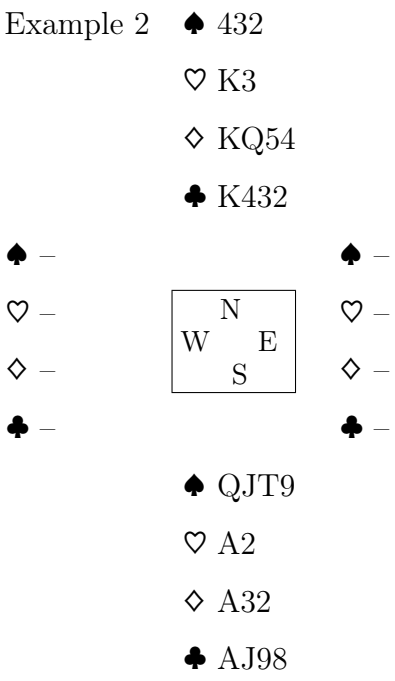
South plays ♦3 North plays ♦K

North plays ♦5 South plays ♦A

South plays ♥2 North plays ♥K

North plays ♥4 South plays ♥A

Example 2



Output: max top tricks: 7

South plays ♣8

North plays ♣K

North plays ♣2

South plays ♣A

South plays ♦2

North plays ♦Q

North plays ♦4

South plays ♦A

South plays ♦3

North plays ♦K

North plays ♥3

South plays ♥A

South plays ♥2

North plays ♥K

Example 3

	♠ QJT ♡ 65 ◇ KJ5432 ♣ 65	
♠ – ♡ – ◇ – ♣ –	N W E S	♠ – ♡ – ◇ – ♣ –
	♠ A32 ♡ A432 ◇ AQ ♣ A432	

Output: max top tricks: 6

South plays ♣A

North plays ♣5

South plays ♦A

North plays ♦2

South plays ♦Q

North plays ♦K

North plays ♦J

South plays ♣2

North plays ♥5

South plays ♥A

South plays ♠A

North plays ♠T

APPENDIX B

Declarer play examples

In the following examples North and South hands are played by my declarer program, East and West hands are played by Bridge Master.

Example 1	♠ Q4	Level
	♡ K54	1
	◇ A654	
	♣ KJ54	
♠ KT987		♠ A65
♡ 9876	N W E S	♡ QJT
◇ QJT		◇ 987
♣ T		♣ 9876
	♠ J32	
	♡ A32	
	◇ K32	
	♣ AQ32	

Declarer: South

Openeing Hand: West

Contract: 3NT

Line by line play:

1. W: ♠T ♠4 ♠A ♠2 || 2. E: ♠6 ♠3 ♠9 ♠Q

3. N: ♡4 ♡T ♡A ♡6 || 4. S: ◇2 ◇T ◇A ◇7

5. N: ◇4 ◇8 ◇K ◇J || 6. S: ♣2 ♣T ♣J ♣6

7. N: ♣4 ♣7 ♣Q ♡7 || 8. S: ♣3 ♡8 ♣K ♣8

9. N: ♣5 ♣9 ♣A ♡9 || 10. S: ♡2 ♠7 ♡K ♡J

Contract Made!

Example 2	♠ 65	Level
	♡ A65432	2
	♢ QJT	
	♣ 32	
♠ AK87		♠ QJT9
♡ 7	N W E S	♡ K
♢ 6543		♢ A987
♣ 7654		♣ JT98
	♠ 432	
	♡ QJT98	
	♢ K2	
	♣ AKQ	

Declarer: South

Openeing Hand: West

Contract: 4H

Line by line play:

1. W: ♠K ♠5 ♠Q ♠2
2. W: ♠A ♠6 ♠J ♠3
3. W: ♢6 ♢T ♢A ♢2
4. E: ♣J ♣Q ♣4 ♣2
5. S: ♡8 ♡7 ♡2 ♡K

Contract failed! (1 - 4)

Example 3	♠ K3	Level
	♡ A65	5
	◇ AK6543	
	♣ Q7	
♠ QJT987		♠ 654
♡ 87	N W E S	♡ KQJT9
◇ T987		◇ Q
♣ 8		♣ KJT9
	♠ A2	
	♡ 432	
	◇ J2	
	♣ A65432	

Declarer: South

Openeing Hand: West

Contract: 3NT

Line by line play:

1. W: ♠Q ♠3 ♠4 ♠A
2. S: ◇2 ◇7 ◇3 ◇Q
3. E: ♠6 ♠2 ♠7 ♠K
4. N: ◇4 ♣9 ◇5 ◇8
5. S: ♡2 ♡7 ♡A ♡9
6. N: ◇K ♣T ♣2 ◇9
7. N: ◇A ♠5 ♡3 ◇T
8. N: ◇5 ♡T ♣3 ♣8
9. N: ◇6 ♡5 ♡4 ♡8
10. N: ♣7 ♣J ♣A ♠8

Contract made!

REFERENCES

1. **Cazenave, T.** and **V. Ventos** (2019). The $\alpha\mu$ search algorithm for the game of bridge.

2. **Chang, M.-S.** (1996). *Building a Fast Double-Dummy Bridge Solver*. Ph.D. thesis, Department of Computer Science, Courant Institute of Mathematical Sciences, NYU, New York.

3. **Frank, I.** and **D. Basin** (1998). Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence*, **100**.

4. **Ginsberg, M. L.** (1996). Partition search. *Innovative Applications of Artificial Intelligence Conference*, **1**.

5. **Ginsberg, M. L.** (1999). Gib: Steps toward an expert-level bridge-playing program. *International Joint Conference on Artificial Intelligence*.

6. **Ginsberg, M. L.** (2001). Gib: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, **14**.

7. **Haglund, B.** (2014). Search algorithms for a bridge double dummy solver. URL `http://privat.bahnhof.se/wb758135/bridge/index.html`.

8. **Koller, D.** (1995). Generating and solving imperfect information games. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*.

9. **Mahalingam, S.** (2017). Why 'bridge' a cerebral game that even ai can't conquer deserves more attention. URL `https://www.dailyo.in/variety/bridge-ai-card-game-chess-go-sport/story/1/19380.html`.

10. **Mossakowski, K.** and **J. Ma´ndziuk** (2004). Artificial neural networks for solving double dummy bridge problems. *International Conference on Artificial Intelligence and Soft Computing*.

11. **Yallamati, J.** (2019). *An AI program for Contract Bridge*. UGRC thesis, under Dr. Deepak Khemani, Department of Computer Science and Engineering, IIT-Madras, Chennai – 600036.