A Program for Non Competitive Bidding in Bridge

A Project Report

submitted by

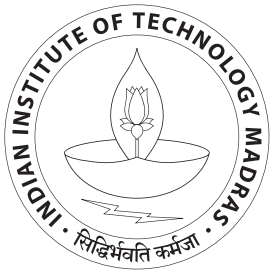
SRIHARSHA ANANTHOJU

CS16B002

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

BACHELOR OF TECHNOLOGY

under the guidance of
Dr. Deepak Khemani



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

May 2020

THESIS CERTIFICATE

This is to certify that the thesis entitled **A Program for Non Competitive Bidding in Bridge**, submitted by **Sriharsha Ananthoju (CS16B002)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out 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.

Dr. Deepak Khemani
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: June 2, 2020

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my guide Prof. Deepak Khemani for guiding me thoughtfully and efficiently through this project and teaching me the necessary concepts required for the project.

I would also like to thank my friends/project partners Joshua, Ashwin, Varshith, Varun and Jitesh for their help and discussions related to the project.

ABSTRACT

Bridge is considered as one of the most complicated card games and bidding is the most difficult phase of the game. Bridge bidding is one of the most challenging problems for game playing programs. Partial observability and incomplete information available during the game makes it impossible to predict the outcome of an action. Although several expert level programs have been created, Bridge is one of the very few games where computers were not able to defeat human experts and performance in the bidding phase is more unsatisfactory than in the playing phase. In this report, we state our approach towards building a program for bidding in Bridge.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Since the inception of artificial intelligence in the 1950s, we've been trying to find ways to measure progress in the field of AI. Many scientists believe that games are the ideal domain to test the extent of AI's perspicacity. With this ideology many game playing programs and systems were created and competed with human experts. Gradually AI's have been mastering the games and also defeating human players. Two of the most significant games where AI's brought a revolution by defeating human experts are 'Chess' and 'Go'. For a long time chess was believed to be the ultimate test of artificial intelligence. However in 1997 IBM's computer Deep Blue defeated the then world champion, Garry Kasparov, in a six-game match. This match's outcome became a headline worldwide and helped a broad audience understand the ability of high-powered computing. Years later in 2011, IBM introduced a new smart computer code-named " Watson" which defeated two human champions in the game Jeopardy. After the deep blue's victory in 1997 an astrophysicist claimed that it would take at least 100 years for computers and AI to defeat human experts in the ancient Chinese game of 'Go'. However, less than two decades later, in 2016, Deep Mind, a UK-based AI startup acquired by Google in 2014, made history when its AI Alpha-Go beat world Go champion Lee Sedol in a five-game match. [6]

Though AI's have shown significant performance in several games there are still a few games that are probably most challenging for computers to excel human players. One of those games is Bridge. Unlike other board games such as Chess and Go, Bridge is a stochastic game of incomplete information that makes computers hard to play. Many attempts were made to design an expert level bridge playing program, but none of them were able to defeat the top human players. This project describes the approach we used in designing such a program for the bidding phase of the game.

1.1 Contract Bridge

Contract Bridge, or simply Bridge, is a trick taking card game using a standard 52-card deck. In its basic format, it is played by four players in two competing partnerships, with partners sitting opposite each other around a table. North-South are partners of one team and East-West are partners of other team. The game consists of several deals each progressing through four phases. The four phases are *Dealing*, *Bidding*, *Playing* and *Scoring*. Initially each player is dealt 13 cards, then the auction (or) bidding begins where players call (or) bid to win the contract, specifying how many tricks their team must win to receive points for that deal. During bidding partners exchange information about their hands in the form of bids or sequence of bids. The player winning the auction is the declarer and his partner is the dummy, while the other team are the defenders. The declaring side try fulfilling the contract and the defenders try to stop the declaring side from achieving their 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 are the two main phases of the game, more details about them are discussed in subsequent sections. [1]

1.1.1 Bidding

Bidding is the language of bridge. Its purpose is to convey information about the strength and distribution of each player's cards to his partner. Generally a bid consists of a number and a suit (spades (♠), hearts (♥), diamonds (♦), clubs (♣) or notrump (NT), a designation indicating no trump suit). The suits are assigned value with notrump the highest and clubs the lowest. A one spade bid means the pair intends to take six tricks plus one, or seven tricks total, with spades as trump. Players can also make other bids such as:

- Pass - which means the player has nothing to say
- Double - which is made after an opponent has made a bid nS. Its contractual meaning is that it doubles the value of the contract. A doubled contract is denoted by nS*.
- Redouble - is a bid that can be made by the side that bid nS in response to a Double. Contractually it doubles the (already doubled) value of the deal. A redoubled contract is denoted by nS**.

In the bidding phase, the dealer makes the first call, either a pass or a bid, and the

auction proceeds clockwise until it is ended by three successive players saying "Pass." The final bid becomes the "contract." This means that one pair has contracted to make a certain number of tricks (six plus the number indicated in the bid) in a particular suit or in notrump. Spades ♠ and Hearts ♥ are considered as major suits, Diamonds ♦ and Clubs ♣ are considered as minor suits.[2]

1.1.2 Card play

The player from the declaring side who first bid the denomination(the suit) named in the final contract becomes declarer. The player left to the declarer leads the first trick. Dummy(partner of the declarer) then lays his/her cards face up on the table, organized in columns by suit. 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 trick leads to the next trick. The declarer has control of the dummy's cards and tells his partner which card to play at dummy's turn. For example, consider the following combination play called the finesse. Everyone can see the cards of North(dummy) along with their own cards. In the following card combination for one suit, say Spades,

North: A Q

West: ? East: ?

South: 5 2

South plan the play for both partners, and they need two tricks from the spade suit.The finesse play is as follows South plays a small card, say the 2 If West plays the King then North plays the Ace Else if West plays a small spade then North plays the Queen[1]

[1]This description is taken from Joshua's report

1.2 Problem Statement

Given a bidding sequence and a hand the program should output the next appropriate bid consistent with the auction. This can also be extended as the program playing and bidding whenever it is it's turn. The following is a representation of sample input and output data for the program.

Input:

Hand:

♠ A 4 3 2	// cards in spade suit
♥ K 7 6 5 4	// cards in heart suit
♦ T	// cards in diamond suit
♣ 9 8 6	// cards in club suit

Sequence:
1N 2C 2D ?
Every alternate bid in the sequence is made by our partner therefore in this case partner has started the auction.

Output:

bid: 2H; zone: {p,g}; trumps: {N H}
The program has output the bid 2H as well as the zone and trump of the combined hand. More details about input and output format will be discussed in Chapter 4.

1.3 Structure of the Thesis

Chapter 2 introduces few preliminaries that are required to understand the project. Chapter 3 provides an extensive literature survey. All the representations and implementation details of the programs are mentioned in chapter 4. Chapter 5 discusses results and analysis and finally in Chapter 6 we summarize the work and discuss the scope of future work.

CHAPTER 2

Preliminaries

In this chapter we introduce few definitions and terminologies that are used in our approach.

2.1 Rule Based Systems

Rule-based systems (also known as production systems or expert systems) are the simplest form of artificial intelligence. A rule based system uses rules as the knowledge representation for knowledge coded into the system. These systems mimic the reasoning of human experts in solving a knowledge intensive problem. Instead of representing knowledge in a declarative as a set of things which are true, rule based systems represent knowledge in terms of a set of rules that tells what to conclude in different situations.[10]

A rule-based system is a way of encoding a human expert's knowledge into an automated system. A rule-based system can be created by using a set of assertions and a set of rules that specify how to act on the assertion set. Rules are expressed as a set of if-then statements

2.2 Elements of Rule Based Systems

Any rule-based system consists of a few basic and simple elements as follows:

1. A set of facts. These facts are actually the assertions.
2. A set of rules. This contains all actions that should be taken within the scope of a problem to specify how to act on the assertion set. A rule relates the facts in the IF part to some action in the THEN part.
3. A termination criterion. This is a condition that determines that a solution has been found or that none exists

Facts can be seen as a collection of data and conditions. Data can be seen as set of variables for which values can be assigned and conditions represent the relation of these variables with constants or other variables.

2.3 Game Terminologies

Definition 2.1. A ***Hand*** is the 13 cards held by a player in the game.
Ex:

♠ A 4 3 2	// cards in spade suit
♥ K 7 6 5	// cards in heart suit
♦ T 4	// cards in diamond suit
♣ 9 8 6	// cards in club suit

Definition 2.2. A ***Bid*** is an element in the following set B = {1C, 1D, 1H, 1S, 1N, 2C, 2D, 2H, 2S, 2N,....., 7N, Pass, Double, Redouble}.

Except for the bids Pass, Double and Redouble, a bid is usually of the structure nS where n is the level of bid and S is the trump suit of the bid. The player who bids nS is conveying that he will win at least (6 + n) tricks with S as the trump suit.

Definition 2.3. A ***Contract*** is a bid with structure nS mentioned in the above set B.

The contracts 3NT, 4♠, 4♥, 5♣, 5♦ are considered to be Game contracts.When a team achieves these contracts in the playing phase, they get a bonus called as 'Game' bonus. All the 6-level contracts i.e., 6♣, 6♦, 6♥, 6♠, 6NT are called Small Slam contracts. When a team achieves these contracts in the playing phase, they get a bonus called a 'small slam' bonus. This bonus is higher than the Game bonus. All the 7-level contracts i.e. 7♣, 7♦, 7♥, 7♠, 7NT are called Grand Slam contracts. They promise to make all 13 tricks. When a team achieves these contracts in the playing phase, they get a bonus called a 'Grand Slam' bonus. This bonus is higher than both 'Small Slam' and 'Game' bonuses. These bonuses vary based on the scoring system.

Definition 2.4. A ***zone*** is the range of points where combined strength of partnership lies.

We have described four zones in our program game namely *partscore* (p), *game* (g), *small slam* (s) and *grand slam* (S). If a partnership identifies that they could make a game contract then they are in the game zone, if they could make a small slam contract then they are in the small slam zone and if they could make a grand slam contract then they are in the grand slam zone. Generally a partnership requires 25+ points to be in the game zone and 33+ points to be in the small slam zone and 37+ points to be in the grand slam zone.

2.4 The Bidding Graph

A bidding system in contract bridge is the set of agreements and understandings assigned to bids and sequences of bids used by a partnership, and includes a full description of the meaning of each treatment and convention. Therefore before the start of the game players should agree upon a bidding system. There are several bidding systems in the world of bridge like SAYC, Acol, precision etc.. We have used the Standard American Yellow Card (SAYC) bidding system in our program as it is the most common and most used bidding system.

Since the program requires the knowledge of the bidding system, the bidding system will be represented as a directed acyclic graph (DAG). We call this the bidding graph. The task of creating the bidding graph and an application based on it where users can populate the graph by giving the input is done by my project mate Joshua. This graph has all the information such as rules and conventions of the bidding system i.e. meaning of bids and sequence of bids. Each node is a bid, and the path from the root to the node captures the bidding history. The meaning of the bidding sequence from root to a particular node is stored in the form of features in this node.

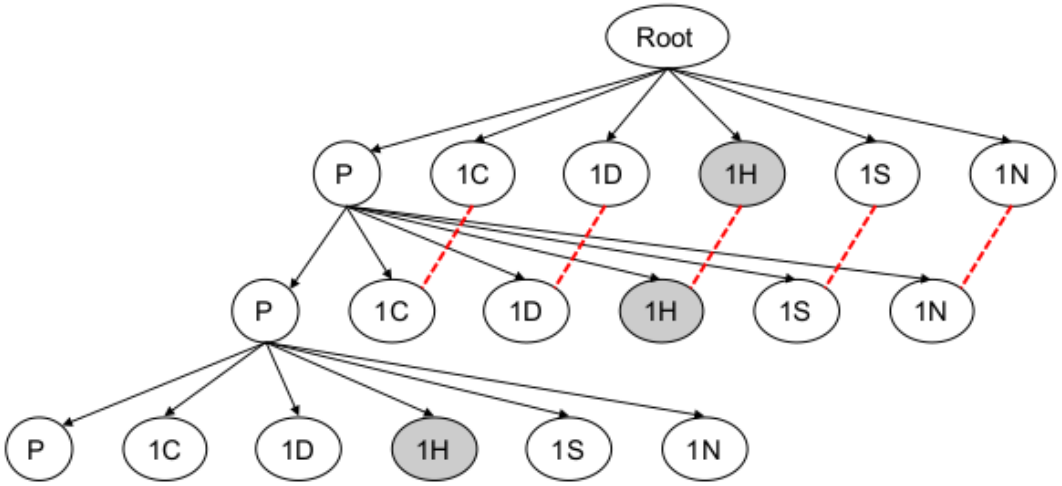


Figure 2.1: bidding graph

The fig 2.1 is a representation of the graph. Every alternate level represents bids made by the opponents. The first level contains all possible opening bids. Suppose if we have sequence {1C P 1H P 1S} then we find a path from root to 1S where the nodes 1C,P,1H,P are traversed in order. Now the descriptive meaning of this sequence in stored in the form of features or variables (will be discussed in Chapter 4)in the node 1S.

Formally if we have a bidding sequence B = $\{b_1, b_2, b_3 ... b_k\}$ then the meaning of this sequence is stored in the form of set of variables V = $\{v_1, v_2, .. v_n\}$ in the node containing bid b_k. Therefore if we query the graph with the sequence B then the output will be a set of values assigned to these set of variables (these variables are also known as state variables which will be discussed later).

$$Q(G, B) = V_a$$

Where $V_a = \{v_{a_1}, v_{a_2}, ... v_{a_n}\}$ and v_{a_i} is a value assigned to the variable v_i.

CHAPTER 3

Literature Survey

Every year since 1997, all computer bridge playing programs have been participating at World Computer-Bridge Championships. Usually eight or more programs are represented, with their developers. The format is teams-of-four, with four identical copies of each program competing. The four top teams from a round-robin of all possible matches play semi-final matches, with a final match determining the winner. The winner of the first championship was BridgeBaron. There are several other programs which have shown their strength by winning the championship in subsequent years such as Jack, GIB , Shark Brige and the current winner being WBridge5. There have been many methods used by different programs to excel in bridge, but most of these methods are mainly focused on the playing phase of the game. In this chapter we will look at few methods which have shown considerable results in the bidding phase of the game.[11]

3.1 Monte Carlo Simulation

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches. Famous programs like GIB and wbridge5 are known to use these methods for bidding. This is often referred to in the bridge-playing community as a Borel simulation.[9]

Algorithm 1: Monte Carlo Bidding

Output: To select a bid from a candidate set B, given a database Z that suggests bids in various situations.

1. Construct a set D of deals consistent with biddings thus far.
2. For each bid $b \in B$ and each deal $d \in D$, use the database to project how the auction will continue if the bid b is made (if no bid is suggested by the database, the player in question is assumed to pass.) Compute the double dummy result of the eventual contract, denoting it by $s(b, d)$.
3. Return that b for which $\sum_d s(b, d)$ is maximum

In the paper [3] the authors have presented a new algorithm called Partial Information Decision Making (PIDM). The algorithm allows models to be used for both opponent agents and partners, while utilizing Monte Carlo sampling method to overcome the problem of hidden information. The paper also presents a learning framework that uses the above decision-making algorithm for co-training of partners. The agents refine their selection strategies during training and continuously exchange their refined strategies. The refinement is based on inductive learning applied to examples accumulated for classes of states with conflicting actions.

3.2 Learning Algorithms

Learning algorithms haven't been adopted in the bridge programs mentioned above but research was done by people to use Learning Algorithms in bridge. Some of them are mentioned below.

- (Chih-Kuan Yeh et.al., 2018) [17]

In this paper , the authors took the help of the Q-learning algorithm and Deep Neural Networks and proposed a model that automatically learns to bid. They claimed in their paper that their model outperforms champion-winning programs and state-of-the-art models by a considerable margin. A team in TCS implemented this model, but no promising results were shown by it.

• (Yegnanarayana et.al., 1996) [16]

In the paper , they used a multilayer feedforward network with the backpropagation algorithm for training. They concluded that a network to only do the first bid (1-level network) seemed to have performed well, but the 2-level network didn't learn well because of the lack of data(rare cases). But for a program to actually go through the complete auction, they suggested that a modular approach where specialized modules can be trained individually might be worth to take a look at.

• (DeLooze and Downey, 2007). [5]

In this paper authors have used a special form of neural networks called self organizing maps (SOM) to effectively bid notrump hands. The input vector is a series of discrete values to show the distribution of cards in each suit and quality of the cards. They have generated a set of training instances that represent card distribution by suit and the total number of high card points (HCP). Each distribution is then mapped to an appropriate bid according to the guidelines published by the ACBL.

Algorithm 2: Kohonen algorithm

1. Initialize each node's weights.
2. Choose a vector at random from the set of training data and present it to the lattice.
3. Examine every node and determine which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. Calculate the radius of the neighborhood of the BMU. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.
5. Adjust the weights of each neighboring node to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.
6. Repeat step 2 for N iterations.

They have limited their approach till only four phases of bidding as most of the contracts end here and they have determined only 6 SOMs are required for an accurate bidding scheme: opening bids, no trump responses, primary responses, overcalls, re-bids by opener, and re-bids by responder.

3.3 Search Tree

In the paper [8] the authors have tried using a combination of search based and genetic algorithm for bidding. In particular, to apply genetic algorithms to the Bridge bidding problem they require a fitness function appropriate for the problem. Therefore, in this paper, they first solve the optimization problem of finding the maximum number of tricks that can be taken in a bridge hand with optimal play and with complete information. Solutions from this optimization problem are subsequently used as fitness functions in applying genetic algorithms to the bidding problem. For solving the optimization problem they have used a search based algorithm like α-β algorithm on multiple game trees.

Algorithm 3: Modified alpha beta

```
α = num of max wins;
β = num of min wins;
γ = current wins;
MD = Max Depth 13 for bridge;
D = Actual depth;
if (the node is a MAX node) then
    if α > (γ + (MD – D) then
        Prune;
    end
end
if (the node is a MIN node) then
    if β < (γ – (MD – D) then
        Prune;
    end
end
```

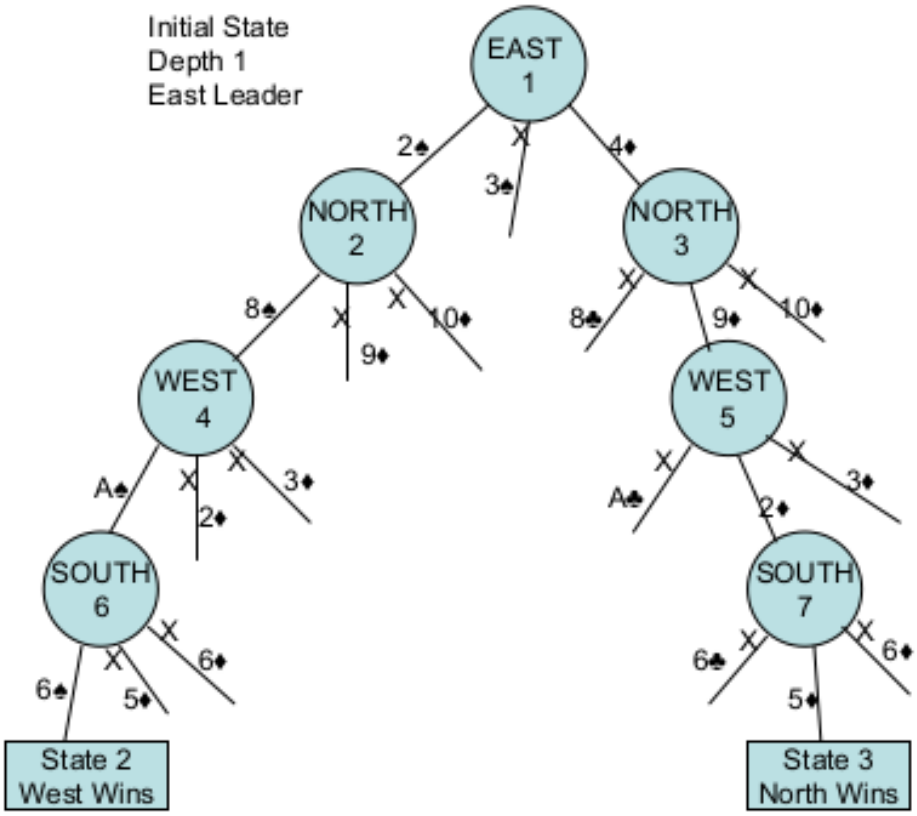


Figure 3.1: search tree

The initial population is generated by taking each player's hand and randomizing the order in which the cards are played. The hands are then played, validating them against the rules of bridge play. If the hand passes the validation, it is included in the bridge population. This processor is called the validator. The validator will also calculate the number of tricks that will be won for this particular hand.

This approach has several drawbacks first one being the size of the tree. Though in the program the author have restricted to 8 cards per hand the size of tree is still very large so they used several powerful processors to actually run this program. The other draw back being the feasibility to use a specific bidding system.

CHAPTER 4

Our Approach

We have implemented the program using a rule based approach. Since bidding in bridge is done adhered to several rules which is conveyed by the bidding system and almost every rule has an exception affiliated to it. Therefore any non-deterministic approaches like a search algorithm or a learning algorithm will most likely fail at such exceptions and should include them with certain conditions. Moreover rule based systems have several advantages such as explanation of output, versatility, speed etc. So this approach helps not only for an AI to play a game but also explain why certain move has been played during the game. I would like to quote the papers [15] and [13] for being an inspiration for this approach. In this chapter we shall present a formal description and structure of the program.

4.1 State of the auction

The state of the auction represents the information available about the combined hand after each bid is made. The state of any particular deal includes information and inferences about the distribution of suits and specific cards between the partners. Formally, A state of auction is a set of variables i.e. $S = \{v_1, v_2, v_3 \ldots\ldots v_n\}$, where $v_1, v_2, ...v_n$ are state variables. These state variables are nothing but some common bridge parameters which store the information of the combined hand (i.e. our and partner's hand) and they are required to analyze and output an appropriate bid. The following are the state variables which are used in the program.

Player points range, Partner points range, Player suit length range, Partner suit length range, Player previous bids, Partner previous bids, Player balanced, Partner balanced, Possible trumps, Possible zone, Stayman, Jacoby, Available bids.

These state variables are possibly dependent on each other so a value update of one state variable will affect the values of other state variables. A hand changes state every time a bid is issued. After the dealer begins the auction, the state of auction transforms with each subsequent bid until a final contract is decided.

4.2 The state updater

Given a state of the auction and current bidding sequence a state analyzer function analyzes the bidding sequence and updates the state variables. The updation of state variables is done in two stages:

(i) The bidding graph is queried by the given bidding sequence and based on the outputs of the query certain state variables are updated.

(ii) An inference function will be called and based on these new values of few state variables all other state variables are also modified using few basic constraint satisfaction processing and game inferences.

Formally,
Let S denote set of all state variables $\{v_1, v_2, ...v_n\}$ and let S_a be a set of values assigned to these state variables $S_a = \{v_{a_1}, v_{a_2}, v_{a_3}...v_{a_n}\}$ i.e. value of state variable v_i is v_{a_i}. Now S_a defines the current state of the auction.
Let B = $\{b_1, b_2, ...b_k\}$ be the current bidding sequence where each b_i represents a bid made by us (or) our partner (alternate bids are partner bids).
Let a function Q: $\{G, B\} \rightarrow P$ be the query function where G denotes bidding graph which has bidding system encoded in it and P denotes a set of variables which is a subset of state variables i.e. $P \subset S$. $P^{'}$ denotes set of values assigned to these variables, then

$$Q(G, B) = P^{'}$$

$$I(S_a, P^{'}) = S_b$$

where $S_b = \{v_{b_1}, v_{b_2}, v_{b_3}...v_{b_n}\}$ denotes the new state of auction.
The function I here is the Inference function. It performs a basic constraint processing and few common game inferences and update the variables accordingly. The following describes few of the inference methods used in the program.

Algorithm 4: inference function for responder's first bid

input: bid b;

if $b == P$ **then**

  $partner.points \leftarrow (0, 12)$;

end

if $b ==$ *1m* **then**

  $partner.points \leftarrow (12, 22)$;

  $partner.suit[m_1].length \leftarrow (3, 13)$;

  $partner.suit[m_2].length \leftarrow (0, 10)$;

  $partner.suit[M].length \leftarrow (0, 4)$;

end

if $b ==$ *1M* **then**

  $partner.points \leftarrow (12, 22)$;

  $partner.suit[M_1].length \leftarrow (5, 13)$;

  $partner.suit[M_2].length \leftarrow (0, 8)$;

  $partner.suit[m].length \leftarrow (0, 8)$;

end

if $b ==$ *1N* **then**

  $partner.points \leftarrow (15, 17)$;

  $partner.balanced \leftarrow true$;

  $partner.suit[.].length \leftarrow (2, 5)$

end

Note: 'm' is a minor suit, 'M' is a major suit and '.' denotes all suits

After the updation of state variables the function *nextbid* is called and it takes the state of the auction as input and outputs the appropriate bid by following the predefined rules.

$$nextbid(S) \leftarrow b$$

Algorithm 5: inference function for opener's second bid

```
input: bids b1, b2;
if b2 == 'N' then
    partner.balanced = true;    // A notrump bid indicates a balanced hand
end
if b1.suit == b2.suit then
    trump = b1.suit;            // partner has supported the suit
end
if b2.level-b1.level == 1 then
    partner.range = 1;          // non jump bid indicates minimum hand
end
if b2.level-b1.level == 2 then
    partner.range = 2;
end
if b2.level-b1.level == 3 then
    partner.range = 3;
end
b1 is opener's first bid and b2 is responder's first bid
```

Each of these inference functions are called at a particular stage of bidding. For example a method *inference1* for opener is called when the opener has to rebid and respond to responder's first bid.

4.3 Finding an optimum contract

The goal of the partners in bidding is to identify an optimum contract and drive the bidding towards it. To identify an optimum contract partners should first identify the zone they are in and find a trump suit they could agree upon. There are four zones in bidding and they are represented as {p, g, s, S}, The power set lattice of this set represents the zone-space in which the bidders search. i.e.

{p , g , s , S}
{p , g , s }, {p , g , S}, {p , s , S}, {g , s , S}
{p , g}, {p , s}, {p , S}, {g , s}, {g , S}, {s , S}
{p}, {g}, {s}, {S}

So after every bid issued the partnership might change from one set to other based on their combined strength. The goal of the bidders is to eventually identify the singleton set of their zone. The process begins with the top level with maximum uncertainty, and travels down the lattice (of subset relations) as more information accrues.
A similar lattice exists for the trump suit possibilities with the top level being {C, D, H, S, NT}. Again, the two partners need to narrow these possibilities down. So, one can imagine the cross product of the two lattices, and the need to discover and bid the best makeable contract. The program works in a similar fashion i.e. after every bid it analyzes plausible zones and trump suits and outputs them along with an appropriate bid.[1]

4.4 Implementation details

The program was written in C++ because of it's ability to model object oriented paradigm, since the work was divided into several classes and sub classes. The input to the program is a hand and bidding sequence and the output is a bid, zone of the combined hand and possible trump suits. Note that the bidding graph is not a user input rather it is accessed by the program during execution. The representation is as follows

[1] This description was written from personal communication with prof. Deepak Khemani

Input:

South hand:

♠ A 4 3 2	// cards in spade suit
♥ K 7 6 5 4	// cards in heart suit
♦ T	// cards in diamond suit
♣ 9 8 6	// cards in club suit

Sequence:
1N P 2C P 2D ?

Assume North has started the bidding and computer plays South, then East and West become opponents, we assume opponents always pass, hence in this scenario program's partner has started the bidding and three bids have been made between partners, now it's south's (i.e. program's) turn to bid.
(Note: bidding goes in a clock wise direction N-E-S-W)

Output:

bid: 2H; zone: {p,g}; trumps: {N H}

In this example the program bids 2H. It analyzes that the combined hand is in the zone {p, g} i.e. the combined hand could have a final contract in either partscore (or) game zone, also referred as game invitational zone. It also says that the ideal trump suit of final contract could be either N (Notrumps) or H (Hearts).

All the input data is taken in a string format and stored in their respective classes after processing it. while inputing the hand first line always denotes cards in spade suit, second line denotes heart suit, third line denotes diamond suit and last line denote club suit. Since we assume opponents always pass as we are considering non competitive scenario we exclude opponents bids in the sequence while taking input. Ex: the above sequence should actually be like 1N 2C 2D.

There are several classes created to represent the information of two hands, Class hand stores the information of a particular hand i.e. points and suit lengths and is defined as follows

```
Class Hand {
    int hcp;
    int dp;
    map <char, pair> lengths;
}
```

The program contains two main classes opener and responder. Opener class is initiated when the computer starts the auction and responder class is initiated when the computer's partner starts the auction. Each of the class contain several methods:

```
Class Opener {
    string firstbid();        //opening bid
    string rebid();           //second bid
    string staymanbid();      //artificial stayman bid
    string jacobybid();       //artificial jacoby transfer
    string subsequentbid(); //further bids
}

Class Responder {
    string firstbidsuit();       //response for partner's suit opening
    string firstbidnot();        //response for partner's notrump opening
    string rebid();              //second bid
    string staymanrebid();       //artificial stayman rebid
    string jacobyrebid();        //artificial jacoby transfer rebid
    string subsequentbid();      //further bids
}
```

A particular method is called based on the state of the auction the program is in. Ex: if the program has not made at least one bid yet, then the method *firstbid* is called, similarly for every subsequent state the corresponding methods are called. The values of state variables decide which method to initate. These methods return the bid in string format. A method will look something like this

```
if(inRange(Points,0,24))
    if(suitfit)
        return raisebid();
    else if(balanced)
        return notrump();
    else
        return getfourcard();
else:
    ....
```

There are several other methods (few of them mentioned below) outside these classes which update other required variables and evaluate the zone and trump suits. For example a method *prunespace* is called after every bid so that it will remove all the bids which are no more available in the bidding space and showcase only those bids which are legal. After every bid is made the program also analyzes the zone and possible trump suits for the team.

```
bool inRange(int,int,int);        //checks if a value is in the range
void init();                      //initializes all state variables
char sixcardsuit();               //returns a six card suit
void prunespace(string);          //prunes bidding space after every bid
char getfourcard(char c);         //returns a four card or above suit
char getminor(char);              //returns a longer minor suit
bool balanced(int,int,int,int);   //checks if a hand is balanced
bool perfect();                   //checks if the hand is perfect
bool checkfit();                  //checks for a fit in partners suit
string bid_new_suit();            //returns a new suit bid
void possible_trumps();           //finds all possible trump
void outputdata();                //output the data in particular format
string notrump();                 //returns lowest notrump bid avaialble
```

Algorithm 6: Responder's response to notrump opening

```
if ((A.hcp >= 8) and (A.suit['M'].first >= 4) and !perfect(A)) then
|  return "2C";
end
if (A.suit['H'].first >= 5) then
|  return "2D";
end
if (A.suit['S'].first >= 5) then
|  return "2H";
end
if (A.balanced and inRange(A.hcp,8,9)) then
|  return "2N";
end
if (A.suit['C'].first >= 6 and inRange(A.hcp,7,8)) then
|  return "3C";
end
if (A.suit['D'].first >= 6 and inRange(A.hcp,7,8)) then
|  return "3D";
end
if (A.hcp > 16) then
|  return "4C";
end
if (inRange(A.hcp,10,INF)) then
|  return "3N";
end
if (A.suit['C'].first >= 6 or A.suit['D'].first >= 6) then
|  return "2S";
end
return "P";
```

CHAPTER 5

Results and Analysis

The current program works for first four phases of bidding i.e. opening bid, responding bid, openers rebid and responder rebid. Therefore it can bid till game zone perfectly (as it was the primary target) and in the case of slam possibility it will bid a slam investigating bid to ensure that we don't sign off at game level. It doesn't really use the bidding graph yet as it is still under development. Hence most of the rules and game inferences are hard coded in the program. We have tested the program with several hands and most of the times we have got desired results. The following are results of few test cases [4]

Hand:	Sequence:
♠ A Q 3	?
♥ A K 3	
♦ K Q T 9 3	Output:
♣ K 6	bid: 2N;

Since we have 21 HCP and a balanced hand the program chose to bid 2N.

Hand:	Sequence:
♠ Q J 7 6	1S ?
♥ J 4 3	
♦ T 7 4	Output:
♣ A T 8	bid: 2S; zone: {p, g}; trumps: {S}

Since we have three spades and partner has at least five spades we got a fit in spade suit and hence program chose to raise partners bid in minimum level as we got only 8 points.

Hand:	Sequence:
♠ J 3	1C 1S ?
♥ A K T	
♦ K J 9 7	Output:
♣ A Q T 7	bid: 2N; zone: {g}; trumps: {N S}

Since we have 18+ points and balanced hand the program chose to bid 2N.

Hand:	Sequence:
♠ Q 9	1C 1D 1H ?
♥ K 8 7	
♦ A T 6 3	Output:
♣ Q J 4 2	bid: 2N; zone: {g}; trumps: {N}

Since we have 11 HCP, balanced hand with no major suit fit, the program chose to bid 2N.

Hand:	Sequence:
♠ A 4 3 2	1N ?
♥ K 4	
♦ T 5 4	Output:
♣9 8 6 4	bid: P; zone: {p}; trumps: {N}

We don't have enough points to bid a new suit hence the program chooses to pass.

The following table describes 29 test cases where each test case has a hand and bidding sequence as input. (Note that these sequences contain opponents 'P' bids, hence we modify the sequence by removing these bids and input to program). This list includes bids like responses to 1NT, Stayman and Blackwood. For the testcases highlighted in bold the output was not as expected because the program bids in lower level than desired. For example the case where program chose to bid 3N there is a possibility of slam and can bid 4N investigating slam and the case where program chose to bid P there is a possibility of bidding 3H and try for game. These differences mainly occurred because of the point ranges that are used in the program which slightly vary from what expert players use. (More details regarding this will be discussed in Chapter 6)

S.No	HAND	SEQUENCE	OUTPUT
1.	AQ3, AK3, KQT93, K6	?	bid: 2N;
2.	AQ753, A73, KQ3, J6	?	bid: 1N;
3.	AKQ753, AK7, KQ3, Q	?	bid: 2C;
4.	A432, K4, T54, 9864	1N P ?	bid: P; zone: {p}; trumps: {N}
5.	AQ32, K4, 654, 9864	1N P ?	bid: 2C; zone: {p,g}; trumps: {N S}
6.	AQ32, K4, K54, 9864	1N P ?	bid: 2C; zone: {g}; trumps: {N S}
7.	A432, K432, T54, 98	1N P ?	bid: P; zone: {p}; trumps: {N}
8.	A432, KQ32, T54, 98	1N P ?	bid: 2C; zone: {p,g}; trumps: {N H S}
9.	A432, KQ32, K54, 98	1N P ?	bid: 2C; zone: {g}; trumps: {N H S}
10.	A432, KQ32, AK4, 98	1N P ?	bid: 2C; zone: {g,s}; trumps: {N H S}
11.	**A42, KQ3, AK4, 9843**	**1N P ?**	**bid: 3N; zone: {g,s}; trumps: {N}**
12.	Q4, AK974, KT4, 765	1H P 2H P ?	bid: P; zone: {p}; trumps: { H }
13.	**Q4, AK974, KQT4, 76**	**1H P 2H P ?**	**bid: P; zone: {p}; trumps: { H }**
14.	AQ, AK974, KQT4, 76	1H P 2H P ?	bid: 4H; zone: {g}; trumps: {H }
15.	**AQ, AK974, KQT4, A**	**1H P 2H P ?**	**bid: 4H; zone: {g,s}; trumps: {H }**
16.	Q4, AK974, KT4, 76	1H P 1S P ?	bid: 1N; zone: {p}; trumps: {H N S }
17.	AQ43, AK974, T4, 87	1H P 1S P ?	bid: 2S; zone: {p}; trumps: {S }
18.	AQ43, A74, KT94, 87	1D P 1S P ?	bid: 2S; zone: {p}; trumps: {S }
19.	**AQ43, A74, KT94, 87**	**1D P 1H P ?**	**bid: 1N; zone: {p}; trumps: {H N }**
20.	AQ3, A74, KT94, 876	1D P 1H P ?	bid: 1N; zone: {p}; trumps: {H N }
21.	**AQ3, A73, KQT983, 6**	**1D P 1H P ?**	**bid: 2D; zone: {p,g}; trumps: {D H }**
22.	A432, K7654, T, 986	1N P 2C P 2D ?	bid: 2H; zone: {p,g}; trumps: {N H}
23.	AQ32, K4, 654, 9864	1N P 2C P 2D ?	bid: 2N; zone: {p,g}; trumps: {N}
24.	AQ32, K4, K54, 9864	1N P 2C P 2D ?	bid: 3N; zone: {g}; trumps: {N}
25.	A432, K432, T54, 98	1N P 2C P 2H ?	bid: 3H; zone: {p,g}; trumps: {N H S}
26.	A432, KQ32, T54, 98	1N P 2C P 2H ?	bid: 3H; zone: {p,g}; trumps: {H S}
27.	A432, KQ32, K54, 98	1N P 2C P 2H ?	bid: 4H; zone: {g}; trumps: {H S}
28.	A432, KQ32, AK4, 98	1N P 2C P 2H ?	bid: 4N; zone: {g,s}; trumps: {N H S}
29.	AK32, KQ32, AK4, 98	1N P 2C P 2H ?	bid: 4N; zone: {s}; trumps: {N H S}

p – partscore
g – game
s – small slam
S – grand slam

CHAPTER 6

Conclusion & Discussion

Bridge is considered as one of the most complex card games and bidding is the most difficult part of the game. A program to play bridge like a human expert is a difficult task because bridge is a game of incomplete information. Unlike Chess, Go and other games, bridge requires heavy expert knowledge.

Bidding in the actual game is a very complicated task, since in our project we have focused on only four phases of a non competitive bidding the task became a little simpler. Even with so many rules encoded in the program we might expect few ambiguous outputs this is because several players have different understanding and approach to follow a bidding system. For example few might consider a hand with 13 to 15 points as minimum hand and few might consider a hand with 12 to 15 points as minimum. Therefore there is this little bit of ambiguity in the nature of the game itself. Nevertheless ending up in an ideal contract is the main aim of the auction.

Coming to the program implemented, currently the bidding graph which stores the entire information of a bidding system is not functional yet, utilizing the bidding graph will make the code even more efficient. So in future we hope we can get a better utilization of this graph. The other advantage of the graph is that we can always have a choice from choosing different bidding systems. Currently we are using the SAYC system but for suppose if we want to use a different system say Acol, we can always create a different graph which stores the rules and conventions of this new bidding system and allow users to decide the bidding system they want the computer to use. The output of the program is the bid, zone and trump suit but one can always change this format and say output any of the state variables. This program is part of a bigger project to create a complete bridge playing AI. This program will also be used by my project mate Aswin for his application which allows partners to practice bidding.

The reason for choosing a rule based system is that the first four phases of bidding are generally deterministic i.e. a typical response for a bid almost always conveys a similar meaning. Since most of the information will already be exchanged between partners by first four bids I thought an exhaustive set of rules to drive the bidding will make this exchange process more effective. This method has its own disadvantages like hardcoding all sets of rules and exceptions could be a tedious task making the code very vulnerable and since with every rule there is some uncertainty associated with it, multiple rules might actually increase this uncertainty causing an ambiguous or an erroneous output.

As part of future work there are several improvements that can be done related to the program. After the first four phases of bidding one can implement a heuristic method to output the next bid. This shouldn't require huge processing of data because most of the information is already conveyed. An abstract approach could be 'output any bid consistent with the auction and driving the auction to the identified final contract'. One can also extend this program to work in competitive bidding. Competitive bidding or contested auction is a completely different scenario; one should consider opponents interference , infer about opponents hands based on their bids etc. therefore a rule based system might not be an ideal approach. We have also not considered the scoring system in the program which describes the payoff and penalty for every bid made.

Since there hasn't been any program which has excelled in this game, especially in the bidding phase, one can try different approaches to decide which would actually work. Since our program hasn't been tested intensively with various expert players and there is no proper metric to measure how good the program is, we can't evaluate its relative performance with existing bridge programs, hope this can also be done as part of future work. Hopefully this program will lay a foundation for a better bidder.

REFERENCES

[1] https://en.wikipedia.org/wiki/Contract_bridge.

[2] ACBL. https://www.acbl.org/learn_page/how-to-play-bridge.

[3] Asaf Amit and Shaul Markovitch. Learning to bid in bridge. *Machine Learning*, 63(3):287–327, 2006.

[4] Lee Asher-Simpson and Dennis Asher. https://www.60secondbridge.com/quizzes/sayc-quizzes.htm.

[5] Lori L DeLooze and James Downey. Bridge bidding with imperfect information. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 368–373. IEEE, 2007.

[6] Ben Dickson. All the important games artificial intelligence has conquered. https://bdtechtalks.com/2018/07/02/ai-plays-chess-go-poker-video-games/, 2018.

[7] By Ned Downey and Ellen ÒCaitlinÓ Pomer. *Standard Bidding with SAYC*. 2005.

[8] John Galatti, Sung-Hyuk Cha, Michael L Gargano, and Charles C Tappert. Applying artificial intelligence techniques to problems of incomplete information: Optimizing bidding in the game of bridge. In *IC-AI*, pages 385–392, 2005.

[9] Matthew L Ginsberg. Gib: Steps toward an expert-level bridge-playing program. In *IJCAI*, pages 584–593. Citeseer, 1999.

[10] Crina Grosan and Ajith Abraham. Rule-based expert systems. In *Intelligent Systems*, pages 149–185. Springer, 2011.

[11] Pete Matthews Jr. Bridge playing and simulation software review. 2013.

[12] Richard Pavlicek. http://www.rpbridge.net, 2019.

[13] Gregg Silveira. Bidding a bridge hand: a thesis on knowledge acquisition and application. 1991.

[14] Jeff Tang. https://www.bridgebum.com.

[15] Anthony I Wasserman. Realization of a skillful bridge bidding program. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 433–444, 1970.

[16] B Yegnanarayana, Deepak Khemani, and Manish Sarkar. Neural networks for contract bridge bidding. *Sadhana*, 21(3):395–413, 1996.

[17] Chih-Kuan Yeh, Cheng-Yu Hsieh, and Hsuan-Tien Lin. Automatic bridge bidding using deep reinforcement learning. *IEEE Transactions on Games*, 10(4):365–377, 2018.