AI based Bridge playing program

B.Tech Project Jan-May 2020

Varshith B (CS16B034)

Indian Institute of Technology Madras

June 2, 2020

A brief history of AI in board games

- 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.

A brief history of AI in board games

- 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 brief history of AI in board games

- 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.
- Chess - Deep Blue

A brief history of AI in board games

- 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.
- Chess - Deep Blue
- Go - Alpha Go

A brief history of AI in board games

- 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.
- Chess - Deep Blue
- Go - Alpha Go
- But unlike Chess and Go, Bridge doesn't have a program which can outperform humans.

Why is there no program which can outperform humans at the game of Bridge?

- In games like Chess and Go the player gets to access the full information which is not the case in Bridge. Hence the computer can't work out what happens if a card is played in Bridge.

Why is there no program which can outperform humans at the game of Bridge?

- In games like Chess and Go the player gets to access the full information which is not the case in Bridge. Hence the computer can't work out what happens if a card is played in Bridge.

- Another problem is that the play of the cards is influenced by the player's own style and knowledge.

Why is there no program which can outperform humans at the game of Bridge?

- In games like Chess and Go the player gets to access the full information which is not the case in Bridge. Hence the computer can't work out what happens if a card is played in Bridge.

- Another problem is that the play of the cards is influenced by the player's own style and knowledge.

- A third significant problem comes with the bidding. It is presumably possible (although not useful) to formulate a perfect bidding system – but only if opponents don't interfere. And existence of a range of different bidding systems and styles would make this very hard.

Motivation

- Previous efforts to duplicate human Bridge playing methodology had limited success which was also the case with Chess(Paradise).

Motivation

- Previous efforts to duplicate human Bridge playing methodology had limited success which was also the case with Chess(Paradise).
- So instead of modeling the play on techniques used by humans, A brute-force search approach to analyze the situation was taken.

Motivation

- Previous efforts to duplicate human Bridge playing methodology had limited success which was also the case with Chess(Paradise).
- So instead of modeling the play on techniques used by humans, A brute-force search approach to analyze the situation was taken.
- But for brute force approach to work the scenario has to be perfect information which is not the case with Bridge.

Motivation

- Previous efforts to duplicate human Bridge playing methodology had limited success which was also the case with Chess(Paradise).
- So instead of modeling the play on techniques used by humans, A brute-force search approach to analyze the situation was taken.
- But for brute force approach to work the scenario has to be perfect information which is not the case with Bridge.
- So using Monte Carlo simulation, Bridge is reduced to its perfect-information variant by dealing a variety of hands for the opponents based on information from bidding and cards played previously and then analyzing each hand given that information.

Motivation

- Previous efforts to duplicate human Bridge playing methodology had limited success which was also the case with Chess(Paradise).
- So instead of modeling the play on techniques used by humans, A brute-force search approach to analyze the situation was taken.
- But for brute force approach to work the scenario has to be perfect information which is not the case with Bridge.
- So using Monte Carlo simulation, Bridge is reduced to its perfect-information variant by dealing a variety of hands for the opponents based on information from bidding and cards played previously and then analyzing each hand given that information.
- Using this perfect information scenario a Double Dummy Solver is used to evaluate the situation and in turn to decide which card to play.

Double Dummy Solver

- Double Dummy Solvers are a class of functions that operate on a perfect information scenario to estimate the value of the current state.

Double Dummy Solver

- 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.

Double Dummy Solver

- 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.

Double Dummy Solver

- 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.

Double Dummy Solver

- 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 Bo Haglund.

Encoding used in the program

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	

Inputs

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

Inputs

- **Declarer** : A number(0-3) based on encoding indicating which hand is the declarer.
- **Lead Hand** : A number(0-3) based on encoding indicating which hand starts the play.

Inputs

- **Declarer** : A number(0-3) based on encoding indicating which hand is the declarer.
- **Lead Hand** : A number(0-3) based on encoding indicating which hand starts the play.
- **Trump Suit** : A number(0-4) based on the encoding indicating what is the trump suit for the game

Inputs

- **Declarer** : A number(0-3) based on encoding indicating which hand is the declarer.
- **Lead Hand** : A number(0-3) based on encoding indicating which hand starts the play.
- **Trump Suit** : A number(0-4) based on the encoding indicating what is the trump suit for the game
- **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

Output

- The final output of the program is the number of tricks the program won.

Output

- The final output of the program is the number of tricks the program won.

- While each card is played by the program, it also outputs a score next to all the playable cards.

Output

- The final output of the program is the number of tricks the program won.
- While each card is played by the program, it also outputs a score next to all the playable cards.
- 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.

Output

- The final output of the program is the number of tricks the program won.
- While each card is played by the program, it also outputs a score next to all the playable cards.
- 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.

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.

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.
- **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).

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

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

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.

Algorithm used in the program

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

Algorithm 1 Monte Carlo Simulation using DD solver

Ensure: Maximize the number of tricks won

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

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.

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.

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.
- In Bridge Master there is no bidding involved, there is a predetermined bid which the program has to play on

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.
- In Bridge Master there is no bidding involved, there is a predetermined bid which the program has to play on
- 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 of results is small.

Results

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

Analysis

- The sample size is quite small to be conclusive but the programs' mistakes are illuminating.

Analysis

- The sample size is quite small to be conclusive but the programs' mistakes are illuminating.
- While some of the mistakes involve failing to gather information, most are problems combining multiple chances.

Analysis

- The sample size is quite small to be conclusive but the programs' mistakes are illuminating.
- While some of the mistakes 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.

Analysis

- The sample size is quite small to be conclusive but the programs' mistakes are illuminating.
- While some of the mistakes 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.

Conclusions

- 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.

Conclusions

- 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.

Conclusions

- 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 benchmark for bridge programs that are going to be built in future.

Conclusions

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

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.
- Using more and better constraints from bidding helps to generate better hands which represent the non visible hands in a better way.

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.
- Using more and 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.

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.
- Using more and 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.
- Exploring a bit more on how to tackle the case where multiple cards have the same score then which card to play.

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.
- Using more and 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.
- Exploring a bit more on how to tackle the case where multiple cards have the same score then which card to play.
- Exploring with different Monte Carlo sample size to improve efficiency.

Further Work

- Explore using a better hand generating technique to see if the efficiency is improved.
- Using more and 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.
- Exploring a bit more on how to tackle the case where multiple cards have the same score then which card to play.
- Exploring with different Monte Carlo sample size to improve efficiency.
- 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.