Application to encode bidding system to facilitate the Bidding Phase in the Game of Contract Bridge

CS4900: UGRC Report

Kothapalli Sai Shre Laasya Raagyni CS20B043

Dr. Deepak Khemani

April 2023

1. Introduction

1.1 Contract Bridge:

Contract bridge, also known as bridge, is a trick taking game using a standard deck of 52 cards. It is played by 4 players with two partnerships each competing against the other. There are 4 positions, namely North, East, South and West with North & South being one team and East & West the other. The game consists of two phases: Bidding phase and Playing phase

1.1.1 Bidding phase:

In this phase the contract that governs the play of the hand is decided. Once the cards are dealt with all players getting 13 cards each, starting with the dealer (the one who dealt the cards) each player takes turns to make a bid in clockwise direction. A bid has 2 components - number of tricks the partnership is aiming and the trump suit. For Example if the bid is 1S then the number of tricks the partnership is aiming for would be 1+6 = 7 tricks and the trump suit will be Spades. So a bid will be <a number from 1 to 6> <suit>. A player can bid a no trump suit which implies that no suit will be a trump suit. There are other bids like Pass, Double or Redouble. Double and Redouble increase the penalties in case the number of tricks are not met but give higher rewards in case the contract is made. The bidding phase ends when 3 consecutive passes are made. The winning bid is called the contract for the deal. The bidding phase gives the players an opportunity to convey the information about their hand to their partners since no other communication is permitted. Most bids are natural with the suit choice indicating a preference for that suit and the number indicating hand strength. But there are bids that are used to convey information that is totally not related to the suit or the number of the bid. These bids are called Artificial bids. So the bids have three implications

1) Suggest an optional contract
2) Exchange information between partners
3) Interfere with opponents

1.1.2 Playing phase

The team that won the contract is called the declaring side and the other team is called the defending side. The contract specifies a particular number of tricks and a suit for the trumps.

The declaring side tries to win the number of tricks bid and the defending team, as the name suggests, defends against the other team winning.
The trump suit that has been decided in the auction phase will have the highest value among the other suits. Players have to still follow the suit if they can but if they have run out the chosen suit they can play a card from the trump suit. The trump suit card beats all the other suits. One member of the declaring side is called the dummy whose hands are visible to all the players. The declarer decides the cards to be played from his hand as well as the dummy's. The player who mentions the contract suit first in the auction phase is called the declarer and his/her partner is the dummy. A trick is a set of 4 cards, each card placed by each player in a clockwise manner. The player who places the highest ranking card among the 4 cards wins the trick. And he/she gets to start the next trick. The first card of the trick decides the suit. The players have to follow this suit unless no such card is available in which case they can use another card which does not contribute to the trick unless it is a trump card. When all the 13 tricks have been played, if the declaring side has made their bid or higher the contract is said to be successful. Otherwise the contract is defeated and the defending team scores.

1.1.3 Scoring

After the playing phase the points are awarded to the declaring side if they make the contract otherwise the defending side is awarded the points. A team is said to win a 'game' if they score above 100 points, otherwise it is called a 'part score'. A huge bonus is awarded to the team if they make a contract of 12 tricks, called the 'Small slam' and 13 tricks which is called the 'Grand slam'.

In the game of Contract bridge, the bidding phase is the key phase of the game, the objective is to decide on the trump suit of the game and the number of tricks to be won. During the bidding phase, the contract for the deal is set up and each player can deduce the constraints of other players' hands through the bids they make. Bidding system is an agreement among the partners about the meaning of the sequence of bids that helps them get an idea of the partner's hand and the contract they should reach during the bidding phase of the game.

1.2 Problem statement

This project is an extension of the work made by Naga Aneesh CS17B114 as part of his dual degree thesis[1]. The application was based on the work of two other seniors Joshua Rohit[5] and Ashwin[6] as part of their Btech projects. The application helps partners to improve their bidding skills by practicing the bidding phase of the game and fine tuning their bidding system along the way. They can encode their own bidding system in the application. The application consists of two parts - Creating an own budding system which is represented as a directed acyclic graph. Users can add a node (a bid is represented as an edge) with information about the hand stored in the node. The next phase involves a tool that generates hands based on some constraints and using the bidding DAG that was created before, the bids are shown to players and warns the users if they make a bid that is not according to their bidding system that was previously created. It also allows the users to modify the DAG. The application still lacks features like storing the information of hands of all the 4 players, comments if a bid is invalid especially if the partner's bid was Forcing or game forcing, trump suit fit, level of the contract. My project is the extension of this work by implementing the above mentioned details.

2. Background and related work

2.1 Importance of bidding phase and motivation

Bridge bidding is considered to be one of the most difficult problems for game playing programs. Artificial intelligence researchers have traditionally utilized game-playing as a means of exploring decision-making in competitive multiagent settings. While most researchers have focused on two player games with complete information such as chess or checkers these games pose challenges in terms of decision making within large search spaces when resources are limited as well as accounting for an opponent agent. In contrast, games like poker present additional challenges, as they require decision making under conditions of partial information. Bridge is of particular interest to multi-agent researchers, as it encompasses all of these challenges, along with cooperative and competitive agents who have restricted communication capabilities. Bridge is divided into two phases: Bidding and playing. While computer programs have demonstrated excellent performance during the playing phase such as with Bridge baron, GIB, and Finesse, bidding remains a challenge. Some resources have even mentioned that the weakest part of GIB's game is bidding.

The encoding of the bidding system which is a part of the application has initially been designed with the motivation to help a program in this bidding phase by bridge the difficulty between the Bridge expert world and the programming world to the maximum extent. Later it evolved into an application for partners to practice by generating hands and to help partners communicate vital information with each other.

2.2 Bidding systems and conventions

Like mentioned earlier a bid can be used to convey the information of a hand. The bidding system is a set of agreements assigned to bids and sequence of bids used by a partnership. A bidding system assigns meaning to every possible bid and presents a language that allows players to exchange information about their hand.

A convention is an agreement about an artificial bid or set of artificial bids. Players fill out a convention card before the game to inform their opponents about their bidding conventions.

There is a lot of implicit information in the convention card that is not machine interpretable. An explicit bidding system with exact details of the information implied by the bids about the hand is essential. GIB[12] uses the following features to convey information about the hands

during the bidding phase:

- High Card points
- Total points
- Length of each suit
- Quality of each suit
- Stoppers
- High honors

2.3 Encoding bidding system in the form of a DAG

Players assign meaning to not just bids but to a sequence of bids as well so a single bid sometimes doesn't convey the entire information about the hand but a sequence of bids convey the meaning of the hand. Thus instead of using a hash table to store the meaning of the bids we use a DAG to store the meaning of the bid. The nodes of the DAG contain the information of hand that is being conveyed by the bid which is represented as an edge. Initially the application stored the bidding system in the form of a Directed acyclic graph to minimize the space, but to store more information (like the hands of all the 4 players) we converted this DAG into a tree so that more information can be stored. For example: 1S or P-1S convey the same information about the hand but due to this we cannot store information like all the 4 hands of the players in the node since it will be different for both the cases. We encode this bidding system through a python program that uses Qt framework for user interface and a graph database Neo4j for storing the nodes of the graph.

2.4 Graph database

We used graph database neo4j that has the native graph storage model to store the information in the node and the relationships among the nodes. Neo4j[8] uses cypher as a query language to update the nodes or retrieve information from the database. For our application we use py2neo driver in order to interact with the database using a python program. The py2neo provides a simple syntax to create, update or delete a node from the database. The graph stored in the database is viewed by running "http://localhost:7474/browser/" on the browser on the system where the database is being stored.

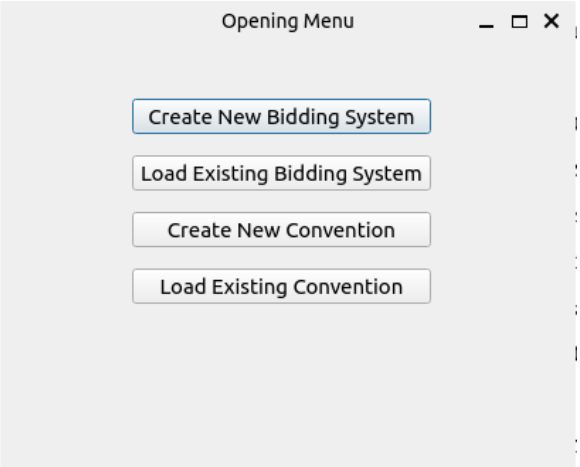
2.5 User interface using Qt platform

To input the information in a user-friendly manner we use a Python Qt framework[9] that allows us to use certain Qt libraries in python to create a user interface.

Application to encode the bidding system

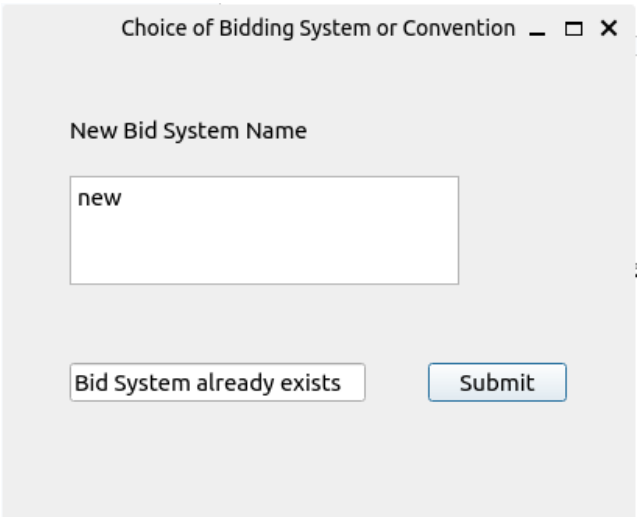
3.1 Overview of the application

When the application is run, a Qt object that asks for creating or loading a new bidding system appears.

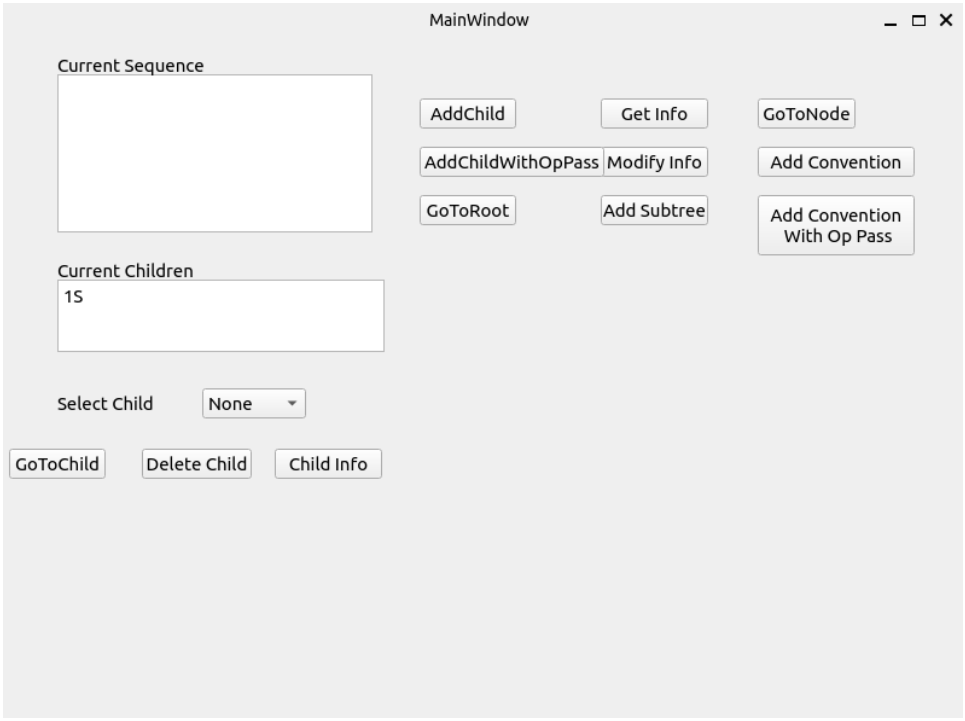


When the user clicks on create a new bidding system, the name of the bidding system has to be entered. We use a cypher query to check if the bidding system already exists(otherwise we display an error message saying the bidding system already exists) and if the bidding system doesn't exist we create a new root node in the graph and display the next interface that would allow the user to add nodes and their information. To add a node to the node created the user has to use the dropdown to select the current node (child of the present node) and then click on go to the child node. Each time a node is created a new neo4j object is created that is updated with necessary information from the user and added to the parent node in the graph

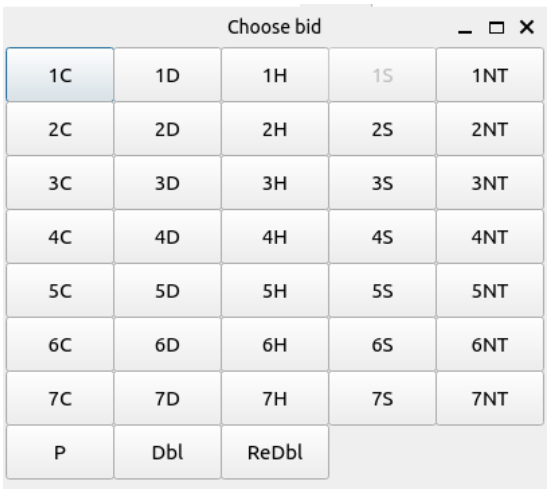
If the user clicks on load existing bidding system, the name of the bidding system has to be entered. Using a cypher query we check if the bidding system is already present in the graph, otherwise we display an error message saying the bidding system doesn't exist. If the bidding system already exists we run a cypher query to get the root node and start displaying the data using the next interface.



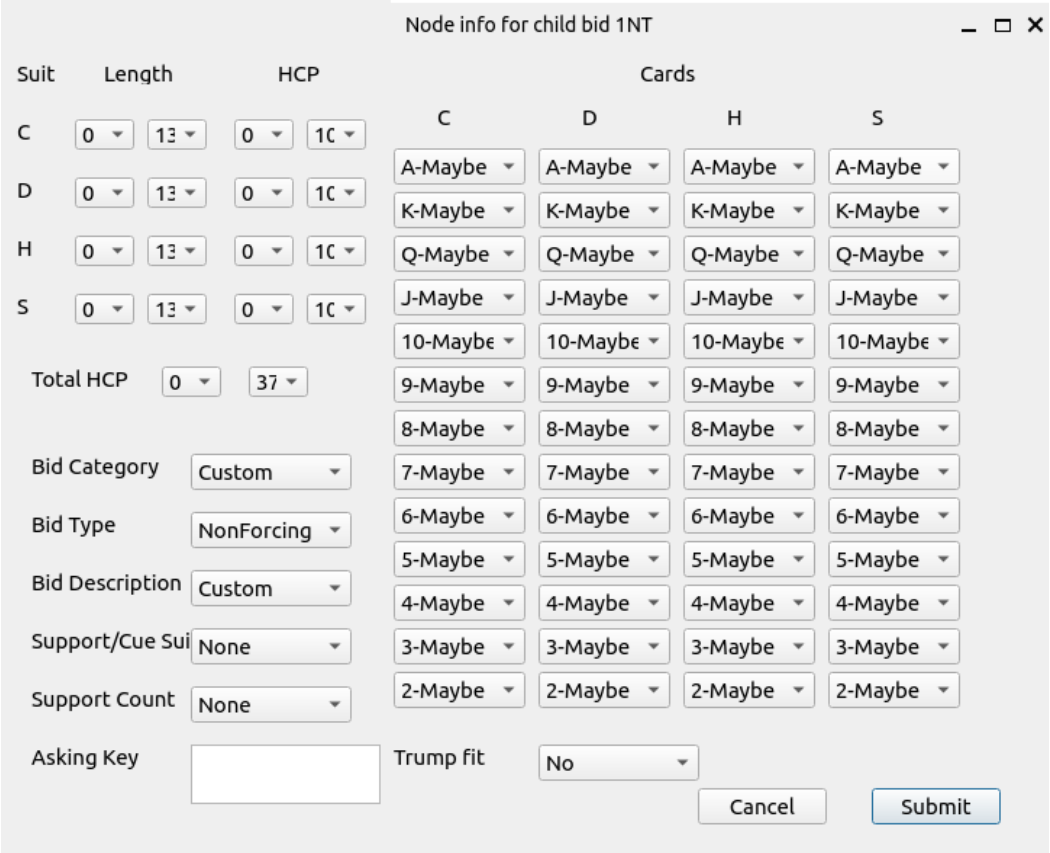
Once the bidding system name is entered a main window is displayed that contains a feature to add a child, get information about the child, go to a particular node by entering the sequence of that node from the root, go to the root, add convention etc. To add a node to say a node 1, the user has to go to that particular node either by entering the sequence of that node from the root node or by selecting the child node and going to the child



When add child is clicked a window displaying all the allowed bids to that node is displayed



When a particular bid is selected the application displays a window that lets the player enter the information of the hand to be conveyed by the bid



3.2 Features added

3.2.1 Propagating constraints on the hands

When a player bids, some information about the hand of that player is revealed (like generally the suit length, high card points of each suit, total high card points, yes or no or maybe for each of the 52 cards). This would mean that knowing the variables of the player's hand would impose constraints on the hand of other players. For example: A 1S opening in the 2/1 bidding system conveys that the player has 5 or more spades and 12 to 21 HCP. Since the total no of HCP over all the players are constant this would impose a lower bound of (40-21) = 19 and upper bound of (40-12) = 28 for all the other players. The total cards in a spade suit is also constant i.e. 13 cards therefore the range of spades will be (0 to 8) for all the other players.
We can impose these constraints on the hands and propagate over all the players.
Constraints:

- **Sum of cards of all suits for a single player = 13**
    - Lower bound of a suit length >= 13 - upper bound of suit length of other suits
    - Upper bound of a suit length <= 13 - lower bound of suit length of other suits
- **Sum of total cards of a suit among all players = 13**
    - Lower bound for that suit length >= 13 - sum of upper bound of that suit length of other players
    - Upper bound for that suit length <= 13 - sum of lower bound of suit length of other players
- **Sum of HCP of a suit for all players = 10**
    - Lower bound for that suit HCP >= 10 - sum of upper bound of that suit HCP of other players
    - Upper bound for that suit HCP <= 10 - sum of lower bound of suit HCP of other players
- **Sum of total HCP across all players = 40**
    - Lower bound for a player HCP >= 40 - sum of upper bound of other players HCP
    - Upper bound for a player HCP <= 40 - sum of lower bound of other players HCP
- **Yes, No for a particular card**
    - Yes for a particular card for a particular player is No for the rest of the players for that particular card
    - No for a particular card for a particular player is Yes for the rest of the players for

that particular card

A list of all the hands of the players are maintained and these constraints are being continually updated till there are no improvements. This final list of hands is stored in the node.

This feature was present only in the next part of the application - in the generating the hands phase and giving additional information and comments about the bids but this is not added in the encoding the bidding system part. Adding the feature in this part would help the player's in accessing the hand in O(1) without additional computation in the next phase of the application.This would help decrease the time by a large amount if the player's use a single bidding system for their practice in the next part of the application. If the players practice bidding again and again using the same bidding system, even if they make similar bids the application would have to compute all the hands every time they start but encoding it in the bidding system would reduce this time.
This feature has been added by adding additional information about all the hands of the players (instead of just the one making the bid) in the node in the database.

3.2.2 Comments about valid and invalid bids

A forcing bid is a bid that requires the partner to bid again, even if they have a minimum of hand. This bid is used to keep the bidding process going and to obtain more information about the partner's hand.
For example, if a player opens the bidding with 1 club and their partner responds with 1 heart the opening player makes a forcing bid of 2 clubs. This bid indicates that the player has a strong hand and wants to continue the bidding process to gather more information about their partner's hand. So if the partner's bid is a forcing bid the player cannot pass, in case the player still passes (it not illegal though) we comment saying that the bid is not valid since the partner has made a forcing bid.
Example: In the 2/1 bidding system in bridge, a bid of 3S following a response of 2S to a 1S opening bid is typically an invitational bid, indicating that the responder's hand is strong enough to potentially win the contract at the 3-level, but not strong enough to make a game-level bid.
It invites the opening bidder to bid for a game if they have extra strength or a favorable distribution or to pass if their hand is minimum or unsuitable for a game-level contract. So in this case a comment is stored in the bid specifying if the invitation has been accepted or declined by checking the partner's bid (which is stored in the parent node's information; the grandparent node is the partner's bid).

Similarly if the partner makes a game forcing bid the player has to keep bidding till the contract is reached. So in case a player passes after the partner making a game forcing bid, the program would comment that it's not valid. This feature helps in the next part of the application.

3.2.3 Trump fit:

This is a new feature added in the user interface that helps the player select if they agree with the partner's suit. When the user enters the bid conveys a trump fit but selecting the dropdown for trump fit as yes, the application gets the trump suit by checking the partner's bid (grandparent of this node) and adding this info in this node.

3.2.4 Displaying the information about the contract level

A feature that mentions the contract level is also added to the node. It describes the score in part score, game, small slam or grand slam level for a particular bid. Here since we used a dictionary to describe the contract level for each bid it won't be accommodating doubles and redoubles.

3.3 Description of the classes and functions used in the program

A node in the graph contains the following information:

- Bid
- Allowed bids from that node
- HCP range
- Each player's hand that consists of
    - Suit cards - (Yes, No or Maybe for each of the 52 cards with Maybe being the default)
    - HCP range
    - Total points range
    - Each Suit HCP range
    - Each Suit Length range
- Bid Category - Opening, asking, responding to ask, custom, response, overcall
- Bid Description - preemptive, support, cue, cue first round, cue second round, custom
- Bid Impact - non forcing, forcing, game forcing, invitational, game invitational, slam invitational, signoff, custom

- Bid System name
- Is Convention - boolean to check if the node is part of a convention
- Is Game forcing - boolean to check if the bid is a game forcing bid
- Is parent Gameforcing - boolean to check if the parent is a game forcing bid
- Is Invitational - boolean to check if bid is Invitational
- Is Parent Invitational - boolean to check if the parent is an invitational bid
- Is Parent forcing - boolean to check if the parent is a forcing bid
- Trump fit - boolean if the bid agrees to the trump suit as mentioned by the partner
- Comments - comments if the bid is invalid (in case if the partner's bid is game forcing or forcing) or if the bid has been accepted or declined if the partner's bid is invitational
- Trump suit - If agreed to the partner's trump suit (assuming that the partner's bid is not an artificial bid)
- Is Root - boolean that checks if the node is a root node or not

We use a DAG function object that takes input as a pointer to the neo4j graph and initializes this object. This class contains the following functions that used to make changes to the graph

- Get Hand info - this takes a neo4j node as an object and returns the hand of the player
- Create Node - using a python node it creates a neo4j node and updates all the information accordingly and returns the node
- Create root - creates a neo4j node as the root node
- Create Bidding system - queries whether the bidding system exists and creates and returns the root node if it doesn't exist
- Find root - using the name of the bidding system gets the root of the bidding system
- Find Node - Using the sequence form the root node and the bid system name returns the node
- Insert child - takes a python created object node, parent node, bidsystem and creates a neo4j node using the python created object node and adds it to the parent node
- Insert child with opponent pass - insert's the child to the parent node with a pass node in between
- Get children - using the parent node and the bidding system, gets the list of nodes from the parent node
- Delete child
- Delete all node
- Add convention
- Add convention with opponent pass

- Constraint propagation

A current state object is used to add information to the graph, it connects the user interface with the graph database. It contains the following members:

- Current sequence - This contains the sequence from the root node to the current node
- Convention list - List of all the conventions present that can be added
- Convention bid dictionary
- Current node (a neo4j object) - This is a neo4j node
- Bid system name
- Path length of the current node from the root - This is used to check which player's hand information the current node contains
- Current node hand list - list of all the player's hands after updating the current player's hand and propagated the hands among all the players
- Previous node hand list - list of all the player's hands
- Current children List of the current children (in form of neo4j node object)
- Graph - neo4j graph object
- Main window object - to display the user interface window

Member functions:

- Update node - If case the information in the node is modified, this functions gets the node pointer from the sequence to reach the node as the input and updates the node
- Update children - when this function is called the current children member of this class is updated, it also updates the current children text box in the main window

3.4 Future work

We haven't accounted for modification of info in the next stage of the application where the player's while using the application can modify the bidding system. In this case if the user modifies the information in the node, the information in its children nodes also have to change. Thus some more work along the lines of using a DFS algorithm starting with that node and changing all the information in it's children's nodes is necessary in order to modify the information in the child nodes accordingly.

References:

[1] Naga Aneesh Mylavarapu. An application to facilitate the bidding phase in the game of contract bridge. DDP thesis, IIT Madras, 2022.

[2] Contract bridge rules of the game
Available at https://www.bridgebum.com/how_to_play_contract_bridge.php.

[3] Bridge scoring system. Available at https://gathertogethergames.com/bridge.

[4] Bidding systems. Available at https://bridgebum.com/bridge_bidding_systems.php.

[5] Y Joshua Rohit. An application to encode a bidding system. Btech thesis, IIT Madras, 2020.

[6] Ashwin L, Bidding practice application and hand generator, Btech project, IIT Madras 2020

[7] learn to bid in bridge
Amit, A., Markovitch, S. Learning to bid in bridge. *Mach Learn* 63, 287–327 (2006). https://doi.org/10.1007/s10994-006-6225-2

[8] Some resource for installing and using Neo4j
https://neo4j.com/docs/getting-started/current/
https://neo4j.com/docs/getting-started/current/languages-guides/neo4j-python/
https://neo4j.com/docs/getting-started/current/cypher-intro/
https://neo4j.com/docs/getting-started/current/graph-visualization/graph-visualization/

[9] Qt framework for python: Available at https://www.qt.io/qt-for-python

[10] py2neo driver: Available at https://py2neo.org/2021.1/

[11] Bridge scoring system. Available at https://gathertogethergames.com/bridge.

[12] Gib features. Available at https://www.bridgebase.com/doc/gib_descriptions.php.

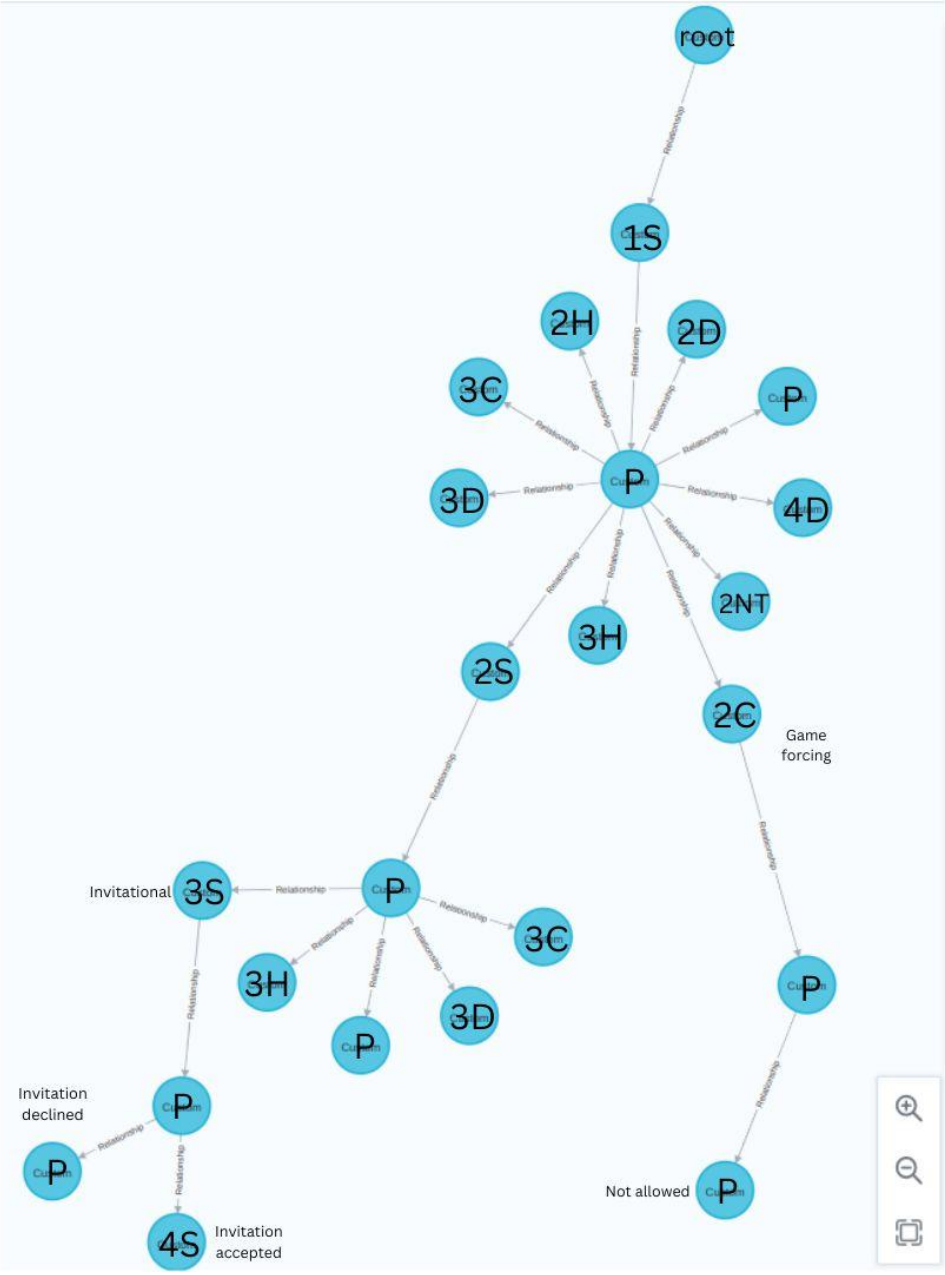
Acknowledgements

I would like to thank Prof. Deepak Khemani for constant guidance in this project and senior Naga Aneesh who set up the foundation for this application. I would also like to thank our faculty advisor Prof. Raghavendra Rao B. V and Prof. Narayanaswamy for coordinating the UGRC.

Appendix

Visualization of the graph in the database and information contained in the nodes

This is a partial bidding system of 1S opening in a standard bidding system



Features added -

<id>	178
AllowedBids	[false,false,false,false,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,false,true,true]
Forcing	false
HcpLb	12
HcpUb	21
Player_0SuitCards	[0,0,0,0,0,0,0,0,0,0,0,0, Integer 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Player_0Suit_C_HCPLb	0
Player_0Suit_C_HCPUb	10
Player_0Suit_C_lenLb	0
Player_0Suit_C_lenUb	8
Player_0Suit_D_HCPLb	0
Player_0Suit_D_HCPUb	10
Player_0Suit_D_lenLb	0
Player_0Suit_D_lenUb	8
Player_0Suit_H_HCPLb	0
Player_0Suit_H_HCPUb	10
Player_0Suit_H_lenLb	0
Player_0Suit_H_lenUb	8
Player_0Suit_S_HCPLb	0
Player_0Suit_S_HCPUB	10
Player_0Suit_S_lenLb	5
Player_0Suit_S_lenUB	13
Player_0TotalPointsLb	0
Player_0TotalPointsUb	40
Player_0_HCP_LB	12
Player_0_HCP_UB	21
Player_1SuitCards	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Player_1Suit_C_HCPLb	0
Player_1Suit_C_HCPUb	10
Player_1Suit_C_lenLb	0
Player_1Suit_C_lenUb	13
Player_1Suit_D_HCPLb	0
Player_1Suit_D_HCPUb	10
Player_1Suit_D_lenLb	0
Player_1Suit_D_lenUb	13
Player_1Suit_H_HCPLb	0
Player_1Suit_H_HCPUb	10
Player_1Suit_H_lenLb	0
Player_1Suit_H_lenUb	13
Player_1Suit_S_HCPLb	0
Player_1Suit_S_HCPUB	10
Player_1Suit_S_lenLb	0
Player_1Suit_S_lenUB	8
Player_1TotalPointsLb	0
Player_1TotalPointsUb	40
Player_1_HCP_LB	0
Player_1_HCP_UB	28
Player_2SuitCards	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Player_2Suit_C_HCPLb	0

Hand of all 4 players (only 2 are shown here)

Trump fit	No
bid	P
bidCategory	Custom
bidDescription	Custom
bidImpact	NonForcing
bidSystem	1Sopening
comments	Valid bid. Partner's invitation has been declined.
highestIncomingBid	13
isConvention	false
isGameForcing	false
isInvitational	false
isParentForcing	false
isParentInvitational	false
isRoot	false
nodeId	189

Comments about the invitation for 1S-P-2S-P-3S-P-P

Trump fit	No
bid	P
bidCategory	Custom
bidDescription	Custom
bidImpact	NonForcing
bidSystem	1Sopening
comments	Not valid since the partner's bid is Game forcing.
highestIncomingBid	5
isConvention	false
isGameForcing	true
isInvitational	false
isParentForcing	false
isParentInvitational	false
isRoot	false
nodeId	208

Integer

Comments about game forcing 1S-P-2C-P-P

Trump fit	Yes
Trump suit	S
bid	2C
bidCategory	Custom
bidDescription	Custom
bidImpact	NonForcing
bidSystem	new
comments	Valid bid.
highestIncomingBid	5
isConvention	false
isGameForcing	lse
isInvitational	false
isParentForcing	false
isParentInvitational	false
isRoot	false
nodeId	39

Boolean

Trump fit and trump suit for 1S-P-2C bid

AllowedBids	[false,false,false,false,false,false,true,true,true,true,true,tr e,true,true,true,true,true,true,true,true,true,true,true,true
Contract level	Part score
Forcing	false

Contract level for bid of 2C (Part score)

AllowedBids	[false,false,false,false,false,false,false,false,false,false,false,fals e,false,false,false,false,false,false,false,true,true,true,true,true, true,true,t... Show all]
Contract level	Game

Contract level for bid of 4S (Game)