Final Report for Syslab
Creating a Hand Solver for the Card Game Bridge

Nikhil Alladi

5/27/2025

Yilmaz - Period 4

Table of Contents

[

]

Abstract

Bridge is a 4-player trick-taking card game with a trump suit that overpowers the lead suit of a hand. This project presents a heuristic to model a full-game state from a hand, played cards, and a set of bids to determine the best move to play at any scenario. Testing with a 1S contract yields high accuracy on the opening trick.

I. Introduction

The game of Bridge is a 4-person card game involving 13 tricks. The game is a 2 vs. 2 game, with opposing players being partners. Each trick, the lead player chooses a card to play; all other players must pick a card from the suit of that card (the trick suit) if they have one (otherwise they may play any card of any suit), and the highest ranked card of that suit wins the trick. Bridge's novelty comes in its bidding phase. Before each round of play, each player bids two items: the number of tricks they will win that round, and a trump suit. For the remainder of the paper, these bids will be notated in the form [number of tricks made - 6][trump suit, or "NT" for no trump] (examples: 1C, 2NT, 3D). A trump suit can be played in the playing phase if no card of the trick's suit is held, and will count as more powerful than the trick's suit if the trick's suit is not the trump suit. Higher bids reward more points, incentivizing a more aggressive bid style to try and win the most amount of tricks between a set of partners. When a player makes the final bid of the bidding phase (marked by the next 3 bids being passed), the partner of that player is chosen as the "dummy" of the game, and their cards are shown to every player in the game.

From the perspective of a computer modeler, Bridge proves to be a uniquely difficult game to model. This is because of the game's lack of known information. Assuming you are not playing as the dummy player, the first card of the game still results in half the deck being unknown cards. Although the game's information rises with time, attempting to model the game is still highly restricted by the information yet known.

In this paper, I propose a solution to the incomplete information problem that Bridge presents by using known details about the game state to fill in the relevant patches of information needed to model the game in a complete information state. By using details about the bids that each player makes, affinities towards any suit, and cards already played in the game (if applicable), we can construct a representation of the game that allows for operations to determine the optimal move for any scenario. This solution also admits computational gains by both being generally non-intensive in terms of compute power and being highly customizable to the user's desires.

II. Background

Existing research on the playing phase of Bridge has mostly focused on searches of various kinds to eliminate possible failing branches of play. Examples include Paul Bethe's work on "claims" (claims being a player declaring that they can force the remainder of the game's outcome to cut the game short) in 1994 [1], although this is more focused on the later part of the game and offers no benefit for the complexities of the early game. More practical is the work of Ginsberg, developing a GIB solver that creates a perfect-information Monte Carlo search by generating many deals consistent with observed variables, then evaluating those deals and taking the best card on average across all deals to play [2]. An alternative to this approach was proposed in the form of an α-μ search to try and reduce the branching factor of the model, showing high success that situationally succeeded that of PIMC (implemented through GIB) [3]. Another approach is to deepen the Monte Carlo search, considering possibilities at deeper depths rather than just simulating depth 0 hands. Although effective in the case of no-trump bids, this solution leverages significant computational power and exits the realm of real-time evaluation that depth 0 is able to obtain.

While highly effective in no-trump domains, the solutions implemented here stumble upon being exposed to a trump suit deal, which comprises the vast majority of deals within the game. This paper aims to fill in that gap by focusing on a solution that is able to handle trump-suit gameplay by utilization of Bo Haglund's Double Dummy Solver module. This is a highly efficient full-information module that utilizes α-β search to identify optimal bids and cards for any point in the game [5].

III. Applications:

The primary application of this would be in the context of analyzing higher level bridge play. Little exists in the form of mass understanding of games, as most approaches are highly computationally intensive and cannot be upscaled in reasonable time complexity. My approach stands alone in computational speed, allowing it to be quickly applied to a game to exact its outcome without worry of time as a notable concern. Alongside this, it could be used in the standard form of most artificial intelligence bots for games, as opponents for play and improvement for players of all skill levels. This is particularly notable for the game of Bridge, as a set of opponent bots alleviates the need to find three players to set up a practice session with, a task notably more difficult than just finding one as is the case with chess, go, and other 1-on-1 games.

IV. Methods:

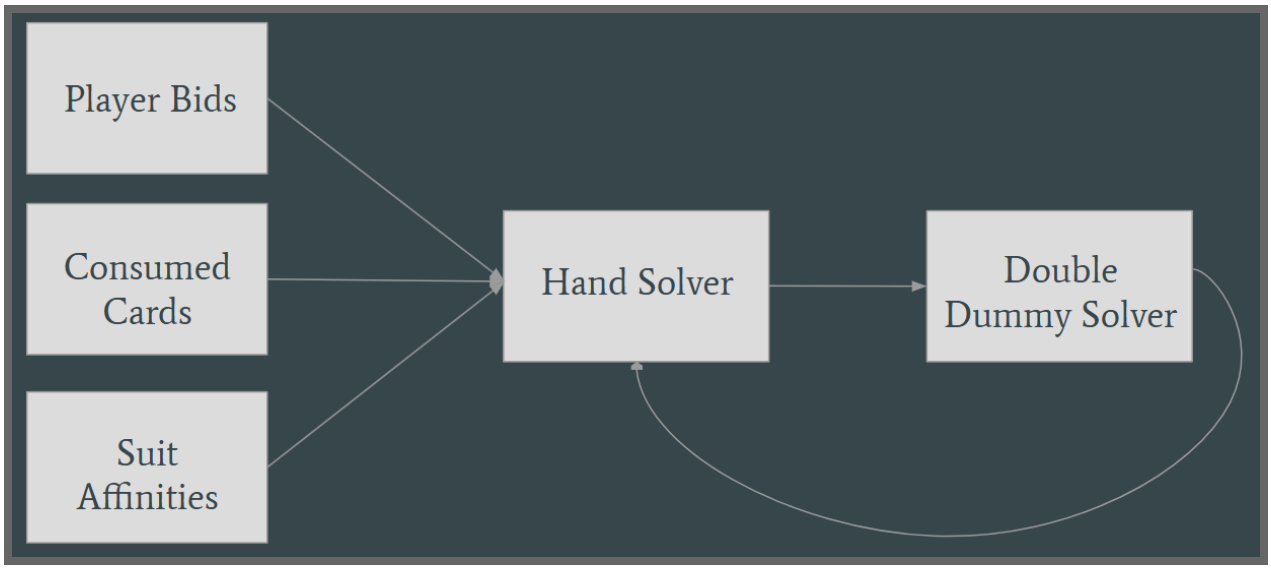

Fig. 1: A systems architecture for the hand solver and DDS loop that the model runs off of.

The core assumption of this hand solver is that the relevant cards are all that is needed to dictate the power of a hand. Relevant cards in this context refer to the cards of the trump suit, high cards (Jacks, Queens, Kings, and Aces), and cards of the suit that each player bid last, respective to that player. If we are able to model these, then the rest of the game is assumed to fall into place. This assumption is made under an analysis of the game's general state: many cards are simply discarded as waste cards or forced to lose from high cards and other forms of forced play. We group all of these states together and say that their specifics are not of particular relevance to the game, and make no attempt to model them, leaving it to random chance.

The input of this model is the game state, from both the bidding and playing phase. This includes all cards played in the game, and all bids made in the bidding phase of the game. Cards played in the game are pooled to their respective players, and the dummy's hand as well as the hand of the user that the perspective is taken from is filled out. From here, the best bids of each player are observed. The power of these bids have a correspondence value from bid power to amount of cards of suit in hand, outlined in the table below:

Bid Power	**Number of Cards in Suit**
1	4
2	4
3	5

4	6
5	7
6	7
7	8

Table 1: Correspondence relation between bid power and number of cards in suit used in the solver.

Rebids of a certain suit are also considered; if a player bids a certain suit and then rebids that same suit on a higher level later, it may be discerned that that player is stronger in that suit than otherwise indicated. To this effect, we add one card to the expected number of cards in a suit to each player that satiates this condition.

It is also important to note the importance of partner bids in defining a hand as well. For the purposes of this discussion, the "major" will be the player who made the final bid, and the "minor" will be their partner. The minor's cards will often influence the course of a bidding phase even if that minor's quantity of cards is substantially lower than that of the major. This is because the minor acts as a supplement, bolstering the power of the pair as a whole without needing as many cards to do so. In order to account for this, we perform two actions: grouping by pair, and normalization. When a bid suit is found that has that player's partner also bid the same suit, we can consider their lump sum of total cards as the number for the entire pair. We then normalize this value by noting that the algorithm will also trigger for the major's partner but without recognizing the current major as a minor in that case (since the bid was stronger), so it will naturally take care of the weaker case; we can then normalize the amount to the value provided in Table 1.

Finally, we return each unused card to the hands in random fashion, invoking the assumption made at the beginning. This crafts a full-game state, which in turn allows usage of the Double Dummy Solver module to pick the best card to play given this mock state.

V. **Results**:

The efficacy of this model was evaluated off of the number of tricks made in a sample bid. To test this, I took a sample hand of the following form:

Player	**Hand**
North	TS, 5S, AH, KH, JH, AD, TD, 3D, 2D, 9C, 5C, 4C, 3C
East	QS, 6S, 2S, QH, 7H, 6H, 2H, QD, 6D, 5D, AC, JC, 7C
South	AS, KS, 9S, TH, 9H, 8H, 5H, 4H, JD, 9D, 8D, KC, 6C
West	JS, 8S, 7S, 4S, 3S, 3H, KD, 7D, 4D, QC, TC, 8C, 2C

Table 2: Card distributions of the test deal.

I tested 1,000 hand simulations of this state from a bid sequence of (starting North):
1D, PASS, 1S, PASS, PASS, PASS, which ultimately evaluates to a contract of 2H by south.

I used the hand solver to evaluate the best opening card for West, who would have first bid after a contract from South. All 1000 test returned 4S as the best card, with varying degrees of effectiveness:

Predicted Tricks Won	**Number of Games**	**Percentage**
6	9	0.9%
7	989	98.9%
8	2	0.2%

Table 3: Predicted tricks won per game. 7 tricks won (or a made contact at the 1 level) was far and away the most predominant of the three ranges, with almost no representation of the 6 or 8 trick-won brackets.

The hand solver identifies 7 tricks won as the best possible scenario in the vast majority of cases, with 4S being its card of choice. A test with the true hands shows that 4S is one of a couple possible ideal choices here, with 7 tricks being won in that path.

VI. Limitations:

One of the primary limitations of this project was the lack of empirical development of the heuristic for the hand solver. Analyzing in the case of suited contracts is a uniquely difficult task due to the considerations of the trump suit and its power in a bid. That trump suit is also a lot more volatile in spread, with minute changes causing game-wide strategy shifts. Further refinement of this could highly increase the accuracy of the hand solver and allow it to model higher contracts better. Additionally, not much consideration was placed on no-trump bids in this model, which while covered by other models already could offer room for improvement.

VII. Conclusion:

This paper presented a suit-focused approach to the playing stage of the card game Bridge. I used a heuristic approach to craft a complete-information mock state of the deal, then used a double dummy solver module to solve that deal. Empirical results on a test hand show high effectiveness matching that of the game state.

VIII. Future Work:

Future expansion of this project could involve further development of the hand solver to address the problems mentioned in VI, or a visual representation of the output of the project to consolidate it into an easy-to-view UI.

IX. Materials:

The Double Dummy Solver module can be found here:
https://github.com/dds-bridge/dds/tree/develop
Code for this project can be found here:
https://github.com/nikhil-alladi/syslab-final

References:

[1] DTAC: A method for planning to claim in Bridge. 2010. New York U, MA thesis. cs.nyu.edu/web/Research/MsTheses/bethe_paul.pdf. Accessed 3 Jan. 2025.
[2] Ginsberg, M. L. "GIB: Imperfect Information in a Computationally Challenging Game." *Journal of Artificial Intelligence Research*, vol. 14, 1 June 2001, pp. 303-58, https://doi.org/10.1613/jair.820.
[3] Cazenave, Tristan, and Véronique Ventos. "The α μ search algorithm for the game of bridge." *Monte Carlo Search International Workshop*. Cham: Springer International Publishing, 2020.
[4] Bouzy, Bruno, Alexis Rimbaud, and Véronique Ventos. "Recursive monte carlo search for bridge card play." *2020 IEEE Conference on Games (CoG)*. IEEE, 2020.
[5] Haglund, Bo. *Search algorithms for a bridge double dummy solver*. Technical report, 2010.

APPENDIX:

I. **CODE:**

ContractBridgePlayer