Agent for Bazar Blot

Project in Artificial Intelligence and Decision Support
American University of Armenia

Grigor Hovhannisyan Aram Abrahamyan
Meruzhan Khachatryan Mikayel Davtyan

December 8, 2024

Contents

Abstract

The game Bazar Blot is a very popular game originating in France and making it's way into Armenia with heavy modifications. In this paper we will thoroughly explain the rules of the game, discuss how one might try to implement an agent for the game and show some examples of how we managed to do it. The game in itself is very complex involving enough rules to barely fit in 4-5 pages and requires a lot of effort to understand. The agents for the game available now generally have poor performance, due to the lack of available research material. In order to demonstrate the agents we also made the game using Python that entirely runs inside a terminal which is available in our GitHub Page.

1 Introduction

1.1 The Origin of the Game

Belote is a 32-card game played primarily in France and in some other countries like Armenia, Belgium, Bulgaria, Croatia, Cyprus, and Georgia. In each country, the game has encountered some modifications; however, the fundamental rules are mostly identical everywhere. Firstly, it would be more accurate to introduce the original game, Belote, and then add the Armenian modifications to get the overall understanding of Bazar Blot's rules and gameplay. The game appeared around 1900 in France and is one of the most popular card games in France and other European countries. The rules were first published in French in 1921. The name of the game *"Belote"* has possibly originated from the term used to describe a pair of *King* and *Queen* of a trump suit, which gives 2 bonus points to the pair of players when one of them has this combination. This name also varies from region to region. In Saudi Arabia, it is called *Baloot*, in Cyprus, the name is *Pilotta*, and in Armenia, it is known as *Blot*.

1.2 General Rules

Belote is played with a *Piquet* deck, which is a standard deck stripped from 2 to 6's, meaning it has 4 suits: *Spades, Hearts, Clubs, Diamonds* and 8 ranks: *A, K, Q, J, 10, 9, 8, 7*. The game is played with 2 teams of 2 players and playing in the turn counterclockwise. The deck is typically shuffled and offered the player preceding the dealer to cut the deck. The cutter may cut or just tap on the top of the pack in which case no cutting is done. The first dealing is done by the winners of the previous game, which in Armenia sometimes is called "Pativ tal". The cards are dealt counterclockwise, starting from the dealer's successor. In the original version, each player receives a pack of 3 cards, then another set of 2. The remains faced down until the contract is being agreed. The remaining cards are dealt after the bidding – a group of three for each player except the player who got the card that was in the middle, who gets two. In the Armenian *"Bazar Blot"* version, each player gets a pack of 4 cards, then another equal pack, after which the bidding process starts.

1.3 Bidding

The bidding process starts with the player on the left of the dealer. The aim of this procedure is to decide the trump suit(contract) for the game and the overall score that the team suggesting the contract is going to get. The possible contracts for *"Bazar Blot"* are:

- Clubs ♣
- Diamond ♢
- Hearts ♡
- Spades ♠
- No Trumps

The trump suit is the "strongest" suit of the round, and in case one plays it, they can take any card on the table (in case if there is no higher trump card on the table). "No Trumps" is different from the other contracts, and it means that during the game there will be no trumps and the every card will have their regular values, except the aces, which will become 19 in their value. Every player must either suggest a higher bid or:

- Pass
- Double (Coinchee or contra), if they are sure the opponents will not be able to get that many points.
- Re-Double (Re-contra), if the other team have doubled bidder's or bidder partner's contract.
- Capot, meaning that team will win the round without giving any points to the opponents.

For instance, if the final contract has been made on "Clubs 9", then the team who has suggested that contract must score 90 or more in that round to win and get $9 + (contractvalue)$ points, otherwise they will lose, and the opponent team will get $16 + 9$ points for that round.
The bidding procedure is over when one of the following happens:

- Four *passes* are announced.
- A contract is *Doubled* and (and not) *Re-Doubled* (The definitions of Double and Re-Double are given in the next section)

After the bidding process is over the actual playing part begins.

1.4 Declarations

The *declarations* are special combinations of cards that need to be announced during the first trick and be shown to the rest of the players during the second trick to get bonus points for them. The exception is *Belote/Re-Belote*, which is not being announced during the first trick, but the player needs to announce while playing either *King* or *Queen* of trump suit. If any of these rules are being broken the *declarations* are being omitted. There are 5 types of such combinations:

- Tierce - a sequence of three cards of the same suite - 20 points
- Fifty - a sequence of four cards of the same suite - 50 points
- Hundred - a sequence of five cards of the same suite - 100 points
- Belote/Re-Belote - a pair of *King* and *Queen* of the trump suit - 20 points
- Four of a Kind - all four cards of the same rank of all suits - points differ with the contract (see the table below)

Card	**Trump**	**No-Trump**
A	11	19
K	10	10
Q	10	10
J	20	10
10	10	10
9	14	0
8	0	0
7	0	0

Note that the sequences are in the "A K Q J 10 9 8 7" order of the same suit.
It is also important that **one card can only participate in at most one declaration**.

Furthermore, if more than 1 players have any of these combinations, they can either sum them up and get credits for each of them if they are teammates or if they are opponents the one that has a larger combination voids the smaller one.
Note: Tierce<50<100<Four of a Kind
In case if 2 opponents have the same combination, the one with a top rank of combination voids the other.
E.g. Tierce(A,K,Q) < Tierce(K,Q,J)
The only combination that cannot be voided is *Belote/Re-Belote*

1.5 Playing

The dealer's successor(the player to the right of the dealer) plays their card first. The first player can play any card; however, the subsequent players must follow the suit if they can. If they do not have a card of the same suit, they must play a trump card. If they cannot play trump either, they can play any suit. If the first card played is trump, the subsequent players must follow the suit as well as play a trump card that beats all the cards on the table if they can. It is called *"raising"* and applies only for trump suits. If they cannot play such a card, they put any trump card. In case they do not have any trump, they can play any suit. If a non-trump card was played first, and then trump, the subsequent player must follow the suit of the first card, and does not have to put a higher trump; however, if the player on turn does not have a card that follows the suit of the first card, they have to put a higher trump. If any of the above rules are being broken, the opponents get 162 points in addition to all *"declarations"* declared in the round regardless of who has made the declaration and current round ends. When each player has played 1 card, the player whose card was the highes wins the *"hand"* or *"trick"* (A hand or a trick is a state when each of all 4 players have placed down a single card) and collects the cards played in this trick and plays first in next hand/trick. After 8 tricks have been played in total, and no one has any card remaining, the cards won on each trick for each team are being calculated and added to their respective scores. The rules for calculating the scores is discussed in the next section.

1.6 Scoring

Every team counts the points on the tricks they have won together. The winner of the last trick adds 10 to the overall points.

Each card has a specific value, depending on the contract.

Card	Trump	Regular	No-Trumps
A	11	11	19
K	4	4	4
Q	3	3	3
J	20	2	2
10	10	10	10
9	14	0	0
8	0	0	0
7	0	0	0

If the contract for the round has been agreed to be any of the suits, then the scoring is being calculated with the *Trump* table, and for the rest of the suits by the *Regular* table. For the case, if the contract has been agreed to be *No-Trumps*, the calculations are done using the *No-Trumps* table.

The overall score is divided by ten and rounded. The rounding is done differently. If usually we take the limit of rounding to be 5, in Bazar Blot this limit is 6. If the result of division and rounding for the team that has made the final bid is greater than or equal to the bid, the bidding team sums up the points received from the round to the bid and adds to the overall score of the game, and is said to be the winner of the round. The opponent team gets $(16 - (\textit{the score of bidding team}))$ points. Otherwise, the opponent team gets 16 points in addition to the bid, and the bidding team is said to lose the round and gets no points. Special cases are *Capot*, *Contra/Re-contra*, and *No-Trumps*. If the contract has been agreed to be *No-Trumps*, the bid is being multiplied by 2 and summed to the overall points for the round for the winning team. If the team fails to get the necessary points, the opposing team gets $(16 + 2 \times \textit{bid})$ points for the round. If any of the teams has done *Capot* regardless the bid, they get $(25 + \textit{bid})$ points. If the

contract has been *Doubled(Contra)*, the winning team gets $(2 \times bid + 16)$ points. If the contract has been *Doubled(Contra)* and *Re-Doubled(Re-Contra)* the winning team gets $(4 \times bid + 16)$ points. Each team add the points retrieved from *declarations* to the overall score.

1.7 Environment Type

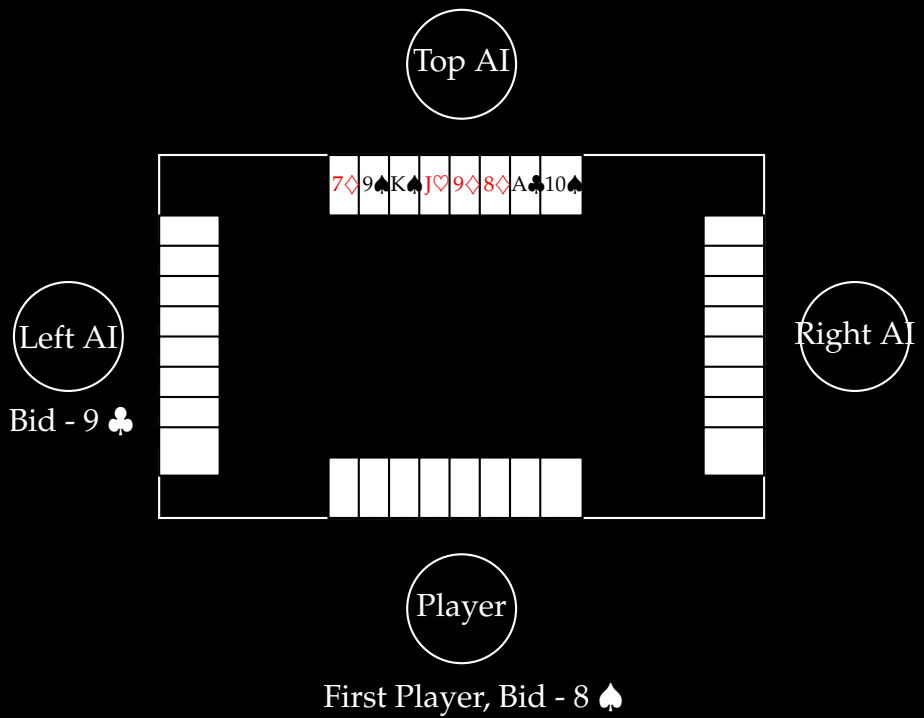
- Partially Observable (Opponents' cards are not visible)
- Multi-Agent (4 players)
- Stochastic (The deck is shuffled after each round)
- Sequential (Actions are interconnected over time)
- Static (No changes over time)
- Discrete (Finite number of actions)
- Known (Rules are fully known)

2 Methods

2.1 Hill Climbing Algorithm for Bidding

You can find a provided template for the Belote AI model. The general environment consists of a player and 3 AIs; let's call them *Left AI*, *Right AI* and *Top AI*. The first stage of the game is the implementation of the bidding part, where each player should bid a number for the hand they received and try to negotiate with their partner. We decided to use the Hill-Climbing Algorithm by using a manual and mildly complicated cost function for each player pair and each possible suit available to bid on (*Spades, Hearts, Diamonds, Clubs, No-Trumps*). The choice of this algorithm was due to dealing with a choice of 6 options, hence there is no need to make many extra moves, we only need to compare all available costs and choose the highest one. This logic seemed to be the closest to the Hill-Climbing algorithm. The Algorithm will count the cost for each suit, compare it to the opponents' bid, and take into account the partner's previous bid. The main operation will include giving a weight to the higher cost cards (Aces and Tens for the Suit "Aces", Jacks and Nines for the rest), add the partner's bid to the corresponding suit, find the maximum among the 5 possibilities, and compare it to the opponents' bid.

Scenario



If at the start of the game you recalled 8 Spades, and the opponent (Left AI), raised the bid to 9 Clubs, your partner's algorithm will look into the hand; say it has these cards (7♢, 9♠, King♠, Jack♡, 9♢, 8♢, Ace♣, 10♠). The objective value for each of the suits would be calculated by using the table of values, and 3 hyperparameters for giving weight to the Partner's bid (2), No Trumps value reduction (due to No Trumps generally having a higher value) (2), and for Pass value increase (as it is generally low) (3.5);

- $f(Suit)$ = The values associated with the suit if it is chosen as trump + $1/2 *$ The bid of the Friend of the same suit.
- $f(NoTrumps)$ = (The values associated with the suit if it is chosen as trump + $1/2 *$ The bid of the Friend of the same suit) / 2.
- $f(Pass)$ = (Previous Highest Bid) $*$ 3.5.

For our case, here are the 6 options we are left with after applying the cost function for each suit and pass

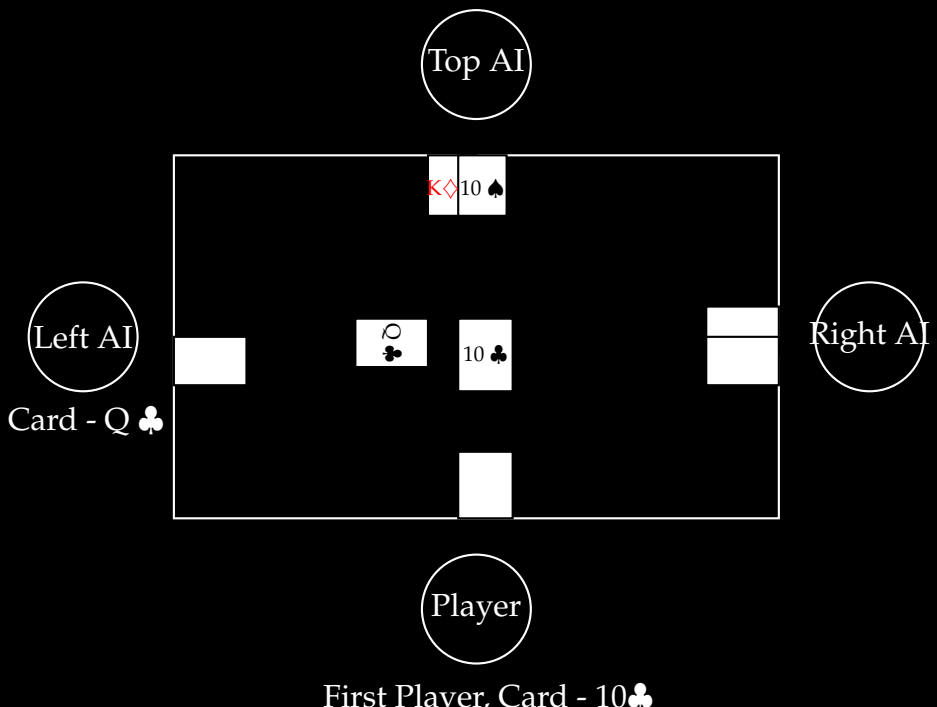
- Diamonds: 14 = 0 (8♢) + 14 (9♢) + 0 (7♢),
- Spades: 32 = 4 (King♠) + 14 (9♠) + 10 (1♠0) + 8 (Partner's Bid) / 2,
- Hearts: 20 (Jack ♡),
- Clubs: 11 (Ace♣),
- No Trumps: 12.5 = (4 (King♠) + 19 (Ace♣) + 2 (Jack♡)) / 2,
- Pass: 31.5 = 3.5 * 9

Here it is obvious that the AI is going to call its partner's bid (Spades, as it has the highest objective value), the question is: How much? There is another logic imported that would decide if there is a need for the AI to increase the bid by 1, 2 or more, and the main logic is going to include the bade suit cost divided by 20 (as each 10 objective equals to 1 point, and divided with another 2 for emergency) and rounded and will be added to the point count, in this case we will have an objective difference of 32/20 = 1.6, rounded to 2. The point count would include adding 2 points to the partner's

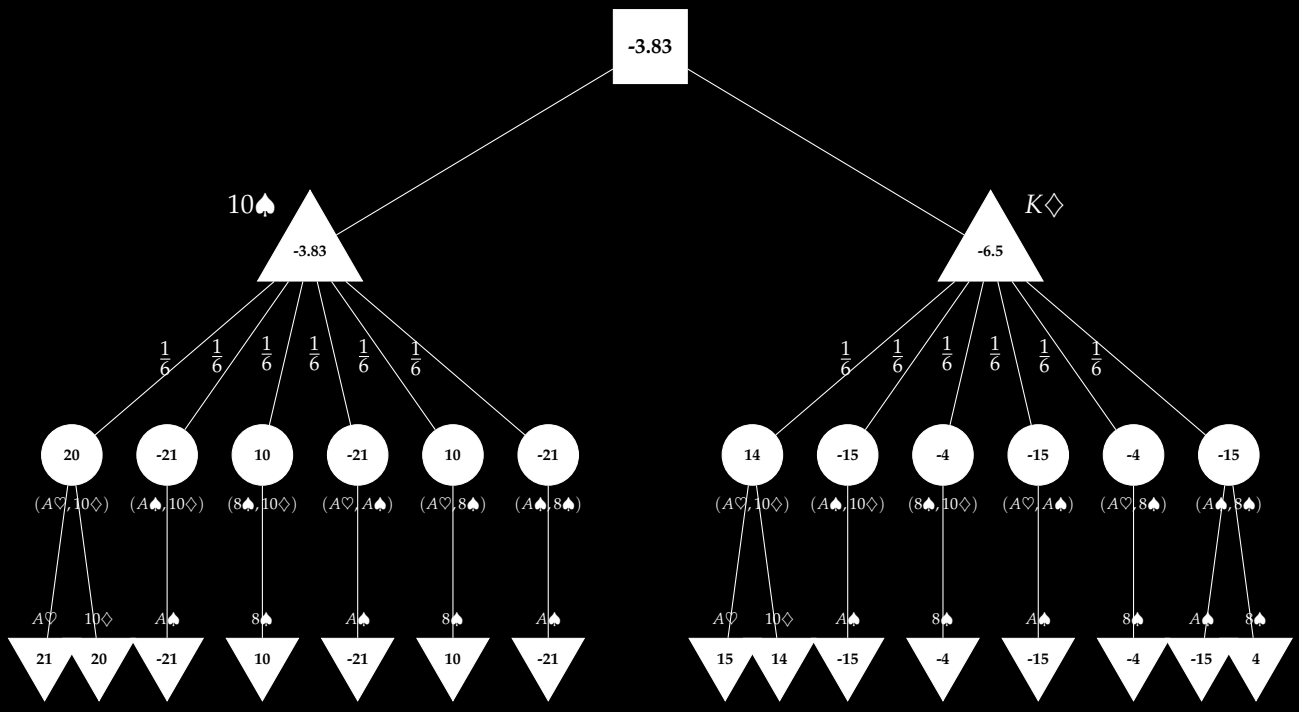
bid, hence the algorithm will return 10 Spades. The mode is not going to include the special combinations mentioned in the game description, such as ”Tierce”, ”Fifty”, etc, as well as there are going to be no options for ”Coinchee” and ”Cabot”. Hence, only the potential value of the cards in the hand will be considered when making a bid. There is going to be set a minimum value for the initial bid and the raises can have any value, as it is allowed in the game. However, the agent will not be able to reduce the bid.

2.2 Expectimax

In the second part of the game, one of the available options is Expectimax. This algorithm consists of Min, Max and Chance nodes; the problem lies in the chance nodes which have a lot of nodes coming out of them. For example, consider the first hand in the game when the bot has to make a move. Let's say the bot is a max node. The bot can make 8 moves, so there are 8 branches coming out of the root. Next we need to have chance nodes coming out of each of the 8 nodes. The chance nodes each have $\binom{24}{8}$ nodes since there are 32 cards, and we know 8 of them we will have to take account of each possible hand the opponent can have which is $\binom{24}{8}$. This by itself already renders the Expectimax algorithm not practical since it is not feasible to check that many nodes. For example, consider the case when there are only 2 cards left for each player, and the first two moves for the play are made.



The algorithm calculated that the remaining cards in the game (besides its own and the ones played) are: A♠, A♡, 10♢ and 8♠. The trumps are spades. Here we have a situation, when we have 2 eligible moves, and we can risk our higher value 10♠ to gain the Q♠, but we also know that there is a risk that the Right AI has A♠, which will beat everyone, hence by using the 10♠ card, we will lose more points than if we use the K♢. For making the best decision, our algorithm will use Expectimax algorithm, consider all possible combination of 4 remaining cards that our opponent can possibly have, and return the best choice according to it. Here is the general process of the algorithm for this specific scenario.



Hence, according to the algorithm, the AI will pick 10♠ for this play. This algorithm is going to have issues if it is not intertwined with other plays as well, as it may use strong cards whenever it can, without trying to save them for a better occasion, and vice versa. However, it can be modified into a specific algorithm, that would use many aspects of stochastic minimax and have additional conditions for a better cost.

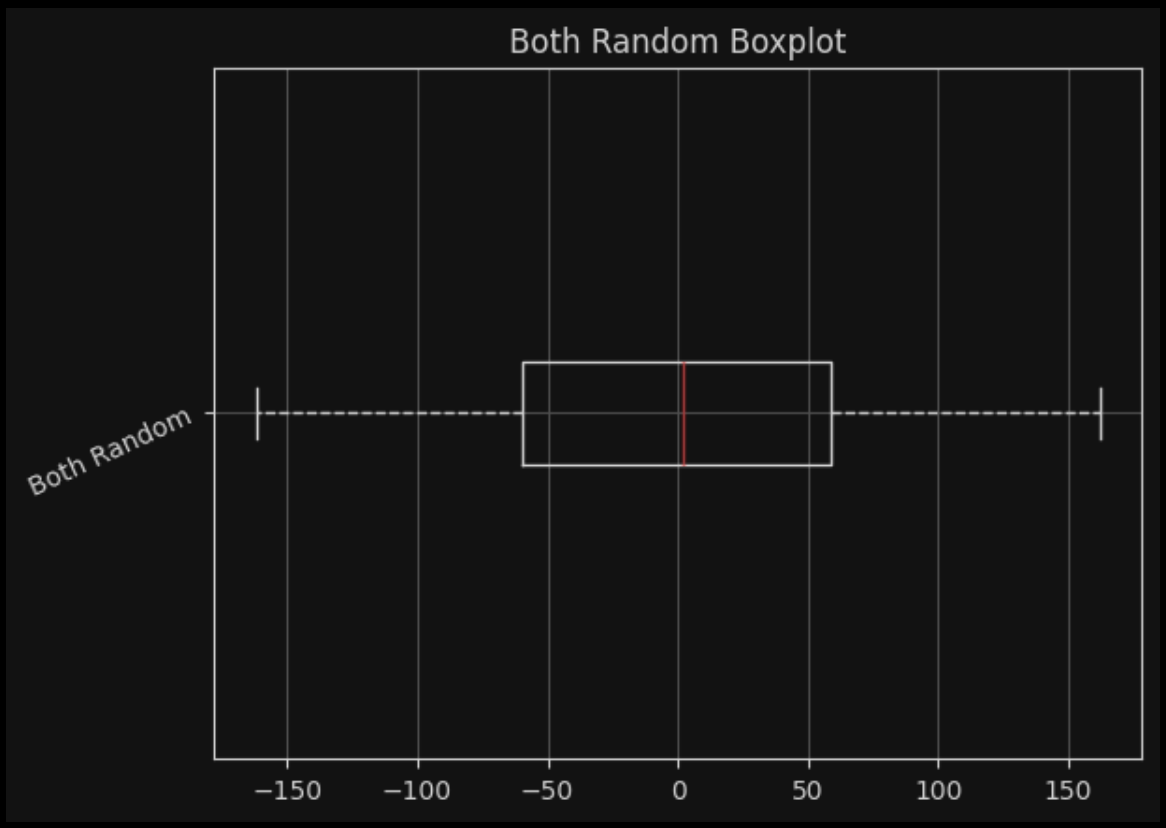
2.3 Cheating Depth-Limited Minimax

One solution to our issue is to provide the opponent cards to the bot, which changes our environment type from Partially Observable to fully Observable and allows us to use standard MiniMax. However even with standard MiniMax we will have an upper bound of $(8!)^4 \approx 2 * 10^{18}$ possible terminal states since we have maximum of 8 moves that each player can make per trick and 4 cards in a trick. This as well is not computable in our testing even with alpha-beta pruning hence we decided to limit the depth to 4 and 12 and for non terminal nodes use the (Current Score) / 10 as a cost. The current score refers to how many points was taken by a team up to that point. This gave us two algorithms which we tested against a random bot which chose cards randomly from available valid moves.

3 Simulation Results

3.1 Random Bot vs Random Bot (Control)

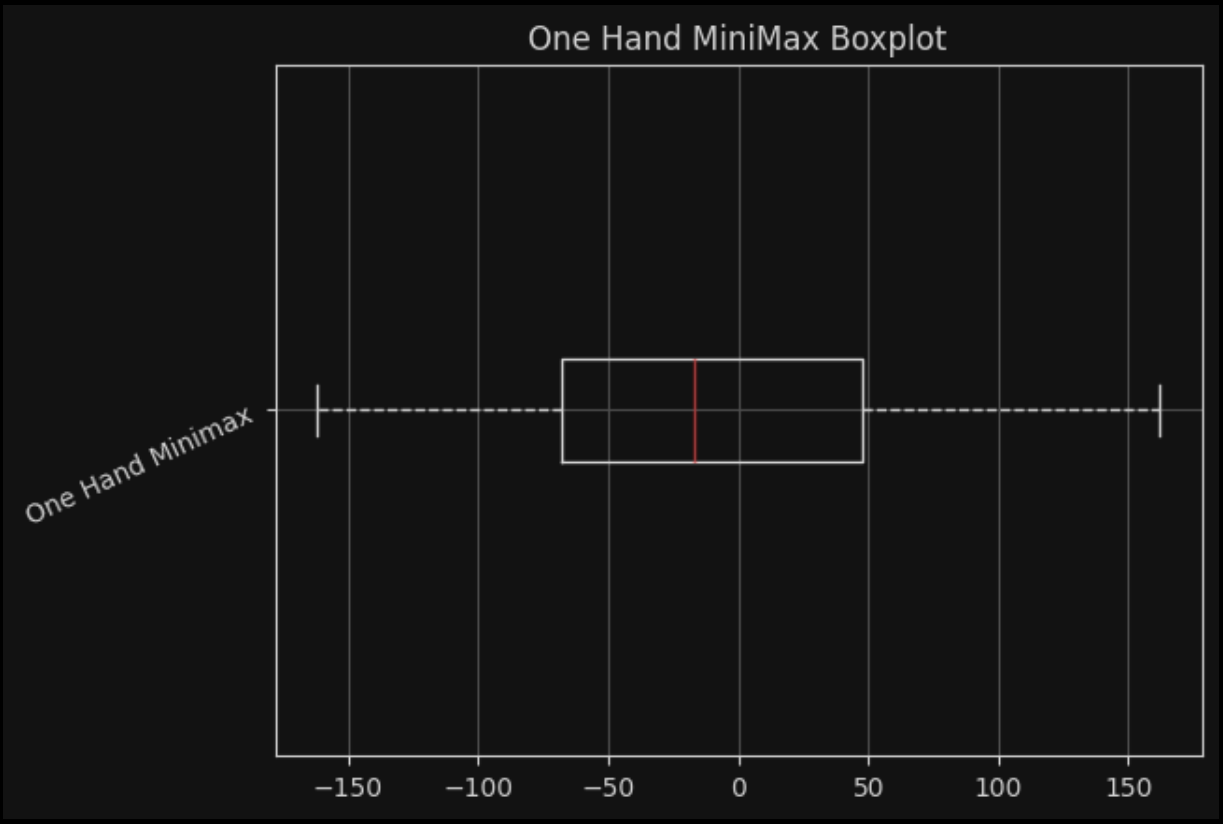
In order to get meaningful results we need to ensure that our sample size is sufficient enough. The game has very large variance and is very dependent on luck so only with big enough sample size we can properly asses the performance of our agents. For all of the simulations we used a sample size of approximately 500 full rounds.



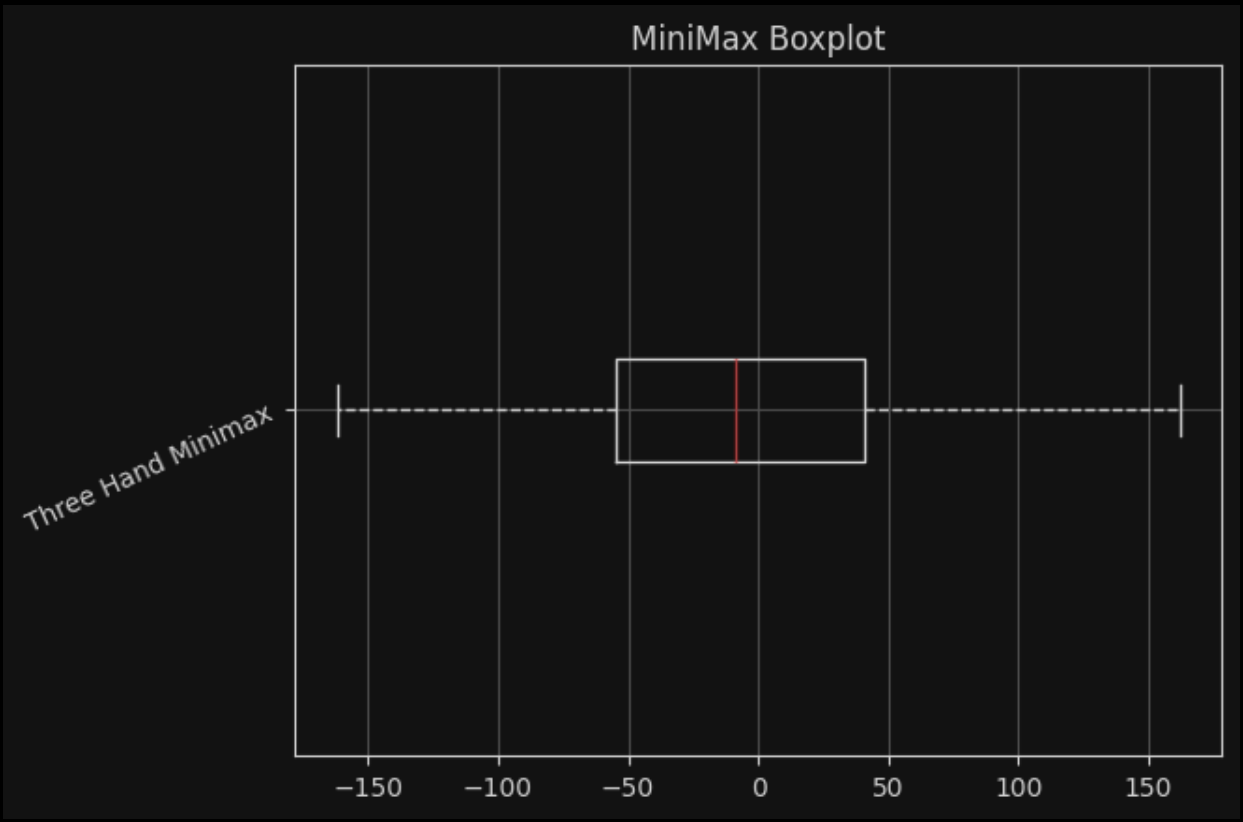
Here we can clearly see that our control simulation with RandomvsRandom has mean of approximately 0. Meaning that our sample size was sufficient enough to judge both agents as having the same performance.

3.2 Random Bot vs One Hand and Three Hand MiniMax

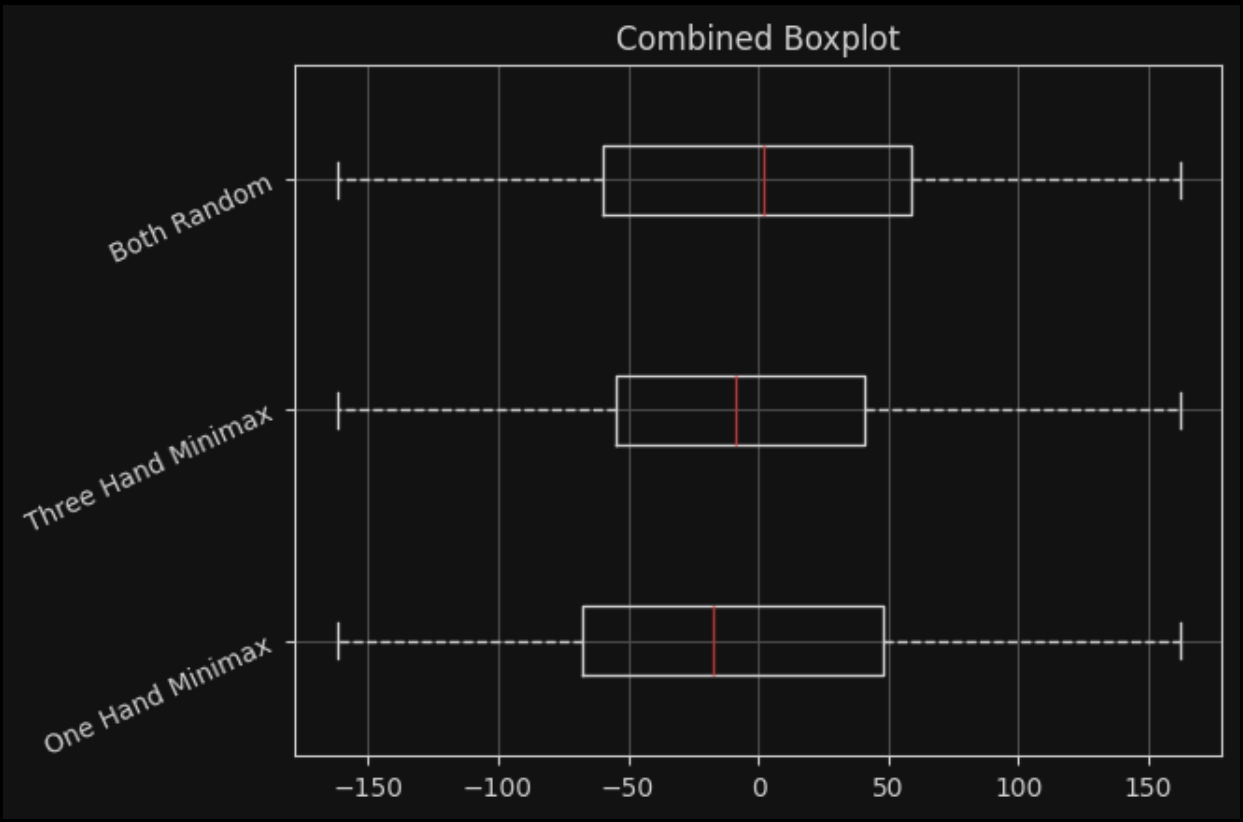
One Hand MiniMax refers to MiniMax with Depth-Limit of 4, effectively making it maximize it's current score for one hand. The games were simulated wth 2 friendly players utilizing the MiniMax agent and the other 2 utilizing Random Agent.



This graph shows the BoxPlot of points that each player gained after each round. Note that the point is not referring to the score that get's calculated based on gained points, but rather the actual added values of cards calculated after each hand. We can see that One Hand MiniMax preformed worse than the Random Bot which was expected since in the game maximizing hands is not a good strategy. When a hand is maximized all of the trump cards are thrown in the beginning disregarding the potential to get more points by keeping them.



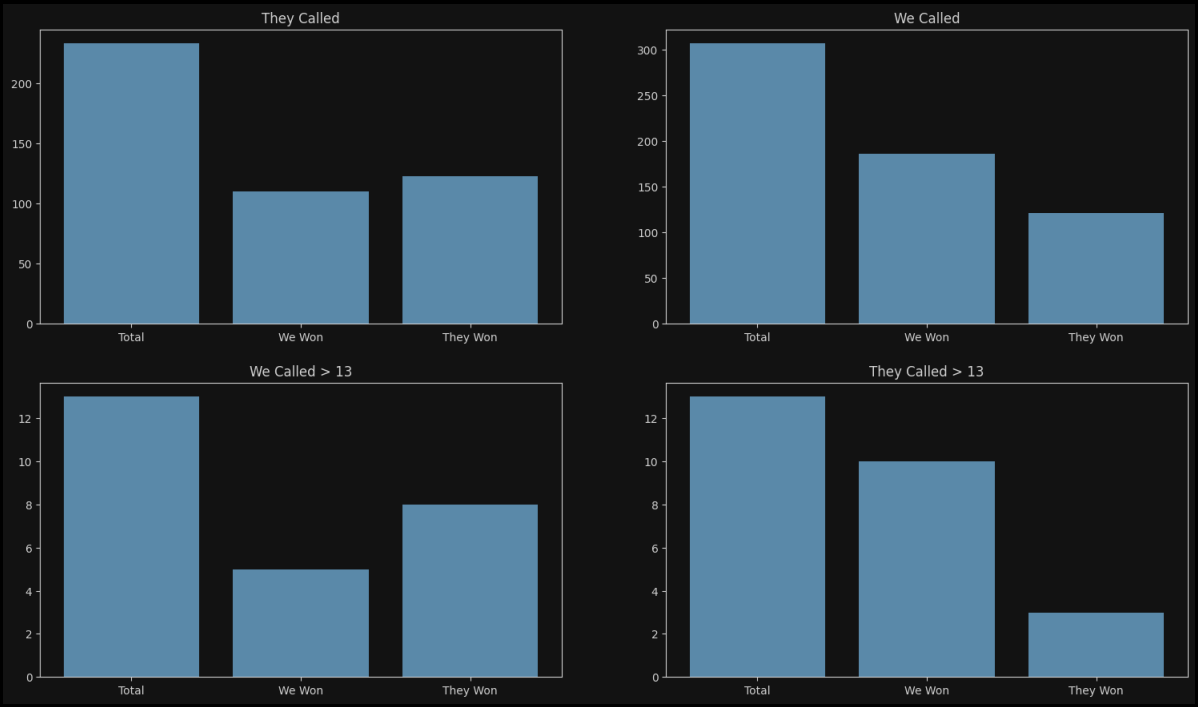
In the Three hand version we can see that it is acting slightly better since in the last three hands it plays optimally since with the depth of 12 it can actually reach the terminal states. However as was the case in the One Hand version it still lacks the ability to save trump cards in the beginning which over many games makes it on average worse than the Random Bot.



Here is the combined plot to better illustrate what we've discussed.

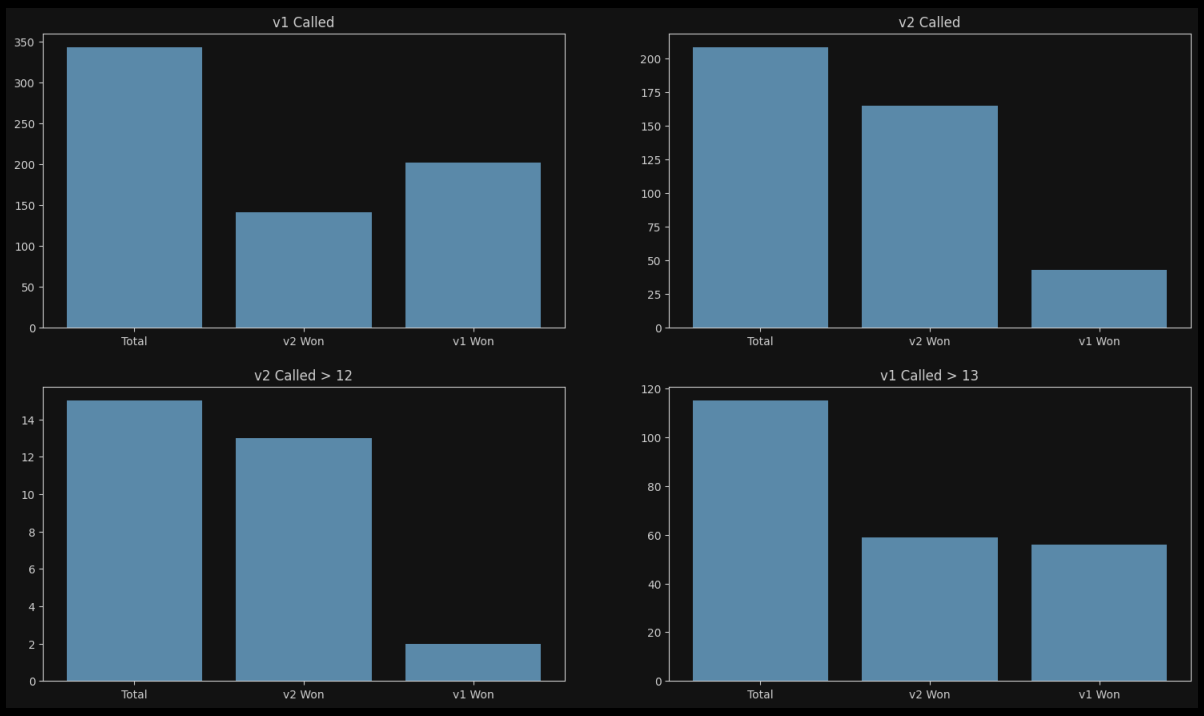
3.3 v1 Bidding Bot vs v1 Bidding Bot (Control)

In order to understand the weakness of the first version of the Hill Climbing Agent(v1) we simulate about 500 games where the playing bots are all random and the bidding bots are both v1.



The top two graphs represent the rounds in which one side made the final bid and whether or not they won that round. From the graphs we can see that on average whoever calls wins more often than loses, but the problem that they overestimate how many points they can get is apparent. This is even more apparent in the bottom two graphs which are similar to the top two but are filtered to only shows when the bots made a big final bid. In most of the cases when v1 makes a big final bid it loses the round which means we need to adjust the values to make it bid more realistically.

3.4 v1 Bidding Bot vs v2 Bidding Bot



For our v2 of the Hill Climbing Algorithm with it's adjusted values we were able fix the problem of the v1 overestimating how many points it can get. As we can see from the top two graphs when v1 makes the final bet it mostly wins, but with still many loses. In contrast when v2 makes the final bid it wins drastically more often. This is also apparent in the graphs where v2 makes a big final bid. By adjusting the confidence of the v1 we were able to make it judge it's capacity to take points more accurately than before.

4 Conclusion

In this paper, we discussed the complexities of the game Bazar Blot and suggested possible implementations of AI agents for this game.

For the auction part, our suggestion was to use Hill Climbing algorithm, which works sufficiently fine; however, with further adjustments of cost values of bids can result in having a much better bidding agent. Furthermore, there many variations of Hill Climbing that can potentially be implemented and give a more consistent outcome.

For the playing agents, we have discussed some variations of Expectimax algorithm. At the earliest stages of the game there are a lot of possible arrangements of cards; hence the width of the Expectimax tree at the first depth becomes $\binom{24}{8}$, which cannot be computed in real time, so we decided to use a Cheating MinMax algorithm to overcome this problem. The Cheating MinMax was able to perform better than the normal MinMax; however, in this case, the problem was with the depth of the tree. The agent required so much time to come up with a good option for that trick that the game lost its dynamism, which is a valuable part of this game. To resolve this issue, the decision was made to use a Depth-Limited Cheating MinMax algorith. The performance of this agent was below the expectations; however, it performed the best among all the implemented agents. The reason for that is that the Bazar Blot game itself does not rely on good performance in the early stages, rather than at the last stages. Playing optimal throughout the whole game is the only key to winning. Furthermore, in the majority of cases, there are such few eligible moves that even playing randomly gives a sufficiently good result, that is why in our analysis, the Depth-Limited Cheating MinMax agent performed almost at the same level as the Random agent in the version without an auction.

To get a better result, one can further adjust the costs of the Hill Climbing algorithm, use learning agents, dive dipper into the MinMax algorithm, and find some tools to reduce the width of each level of the MinMax tree. There are some versions of this game where reinforcement learning is used, and those implementations give an incomparable better result than any other existing agent.