American University of Armenia

College of Science and Engineering

Artificial Intelligence Project

Solving Fixed-Trump Belote Découverte

Team:
Sona Bezirganyan
Hripsime Ghazaryan
Barsegh Atanyan

Fall 2021

Abstract

Different adversarial search algorithms are applied to the trick-taking game of Belote Découverte, a two-player variation of Belote. Several simplifications are made, so that the problem is reduced to a two-player zero-sum perfect information game. An optimal strategy is found for a game with a full deck, using alpha–beta search, with the assumption of full observability. Additionally, the performance is improved by applying a forward pruning technique to the search algorithms.

Keywords: Belote Découverte, Minimax, Alpha–beta, Trick-taking Card Games

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Setting and Description

1.1.1 The Game of Belote Découverte

Belote is the national card game of France, also popular in Armenia, Bulgaria, Croatia, Cyprus, Greece, Luxembourg, Moldova, and other European countries [Par79]. There are many variations of Belote, played by four, three, or two players. It is a trick-taking card game, played with the top eight cards $(7, 8, 9, 10, J, Q, K, A)$ from each suit (clubs: ♣, diamonds: ♢, hearts: ♡, spades: ♠) which results in overall 32 cards. Belote Découverte is one of the variations of two-player Belote [Par79].

Typically, one of the suits is declared to be a trump at the beginning of each game. Cards with non-trump suits are distributed with the following ordering and values: Ace (11 points), 10 (10 points), King (4 points), Queen (3 points), Jack (2 points), and 9, 8, 7 with zero points each. A subtle modification of the ordering comes in hand in the case of trumps — Jack and 9 of trumps become the top cards with corresponding values of 20 and 14.

Besides the mentioned regular points, the player can receive points by declaring melds they hold before playing the trick. There are four different types of melds - quarter, hundred, fifty, and twenty, each having the corresponding points [Par79].

If a player holds a King and a Queen of the trump suit, he scores 20 points by announcing "Belote" before playing one of them and then "Rebelote" by playing the other one.

Additionally, the player who won the last trick gets additional 10 points.

Similar to other trick-taking card games, the deck is equally distributed among the players in Belote Découverte, as well. However, in contrast with standard Belote, each player should have 16 cards: eight face-down and eight face-up, and each of the face-down cards is covered by a face-up card.

There is also a process of choosing trump for the game called bidding, which occurs during the card dealing. After dealing the first four face-up cards to the opponent player, one can either pick a trump suit or pass it till the next set of face-up cards are dealt. However, if the opponent decides not to choose a trump, the decision goes to the other player. There is also a particular case when no suit is declared as a trump. In such games, the ordering of the cards is the same as the ordering for non-trump cards, with only one difference: Ace has 19 points instead of 11 (to maintain the consistency of the overall points being 162).

Similar to other trick-taking games, there is a notion of lead for each trick. That is the player who plays the first in a trick. For the first trick, the lead is the opponent of the dealer. For other tricks, the lead is the winner of the previous trick.

At each trick, the lead of the trick may play any of his face-up cards. The other player must play according to the following rules:

- if possible, suit must be followed;
    - in case of trump suit, higher trump must be played, if possible;
- if not, trump card must be played, if possible;
- otherwise, any card can be played.

The winner of the trick is determined as follows:

- if both cards are of trump suit, the player who played a higher trump wins;
- if there is only one trump, the player who played the trump wins;
- if there are no trumps,
    - if both cards have the same suit, the player whose card has a higher order wins;
    - otherwise, the lead player wins.

The sum of the values of cards is added to the points of the winner of the trick.

After the trick, both players flip the face-down cards below the face-up cards (if any) which they played. Note that several variations exist in terms of the moment to open a face-down card. One of them is to open just after playing the card on it, and the other is to open after the trick (when both players have played) [Par79].

The game ends when both players play all their cards, and each player calculates total points gained from tricks and melds. The total points gained from the trick are 162. When there are no melds and belote-rebelote, making a contract means winning 80 or more points (with melds and belote-rebelote, the corresponding points are added to or subtracted from the threshold value 80 depending on the player who owns them). If the player who declared a trump makes the contract, both he and the opponent get their respective points. Otherwise, the opponent gets all the points. Points are recorded and a sequence of such games is played until one of the players reaches some milestone.

1.1.2 Defining Belote Découverte as a Game Problem

To be able to solve the game of Belote Découverte several simplifications are made to be able to obtain tractability. Firstly, just a single instance of a game is considered, not a sequence. Additionally, only the card play of that particular game is considered meaning that dealing and bidding are done, there is a fixed trump already, and the declarer is assumed to be the MAX player. Next, we assume to have perfect information, meaning that face-down cards are also known. Also, melds are excluded to keep the property of Belote being a zero-sum game. Furthermore, different sizes of decks are considered, based on the number of suits of cards included in the game, ranging from 1 to 4.

To represent Belote Découverte as a game problem, we define several aspects of the game. Before going into details, for convenience, we will call the player who plays first in the game MAX, and the other player will be called MIN. First of all, we need to formally represent a game state.

To that end, for each player p, we denote a set h_p as a set[1] of pairs

$$h_p = (cs_1, cs_2, ..., cs_{2n}),$$

where n is the number of suits, and the first element c_t of the pair

$$cs_i = (c_t, c_b)$$

is the top card of the corresponding card stack, and the second element c_b is the bottom card of the same card stack. Each card c is a pair consisting of its suit and type

$$c_k = (suit, type).$$

If there is no bottom card or no card at all at the given card stack cs_i, a special value `None` is used to indicate that.

A state also includes the card trick, which shows the lead player, the card played by each player, the trick number, and the trick score. The trick score is the sum of the values of the cards in the trick

$$trick = (lead, c_{MAX}, c_{MIN}, trickScore, trickNumber).$$

Additionally, each state stores the current score. Note that the current score of a state is the sum of scores of all tricks that were won by the MAX player. At each state `s`, `TO-MOVE(s)` indicates the player that is to make an action. An action is playing one of the top cards c_t from a card stack cs_i such that $c_t \neq None$ and it follows the rules described above. As a result, the card is added to the trick, and the corresponding changes are made in cs_i:

- if there is a card below, then c_t becomes c_b and c_b becomes `None`
- if not, cs_i is removed from h_p

If as a result of an action there are two cards in the trick, the trick winner is decided based on the game rules. Consequently, the score of the game is updated, and the winner becomes the lead.

A state is considered to be terminal if there are no cards left to play for each player p, that is, $h_p = \emptyset$.

For a terminal state `s`, `Utility(s)` is defined to be `s.score`

[1] For the ease of implementation, a list is used instead of a set in the code

Note that this representation is for the perfect information Belote Découverte, where only a single game is considered, the trump is fixed, and there are no melds. Depending on the progress and results, some simplifications may be omitted, and corresponding modifications will be provided for the problem definition.

1.2 The Structure of the Project Paper

The next chapter is devoted to the literature review and related work. Afterward, several possible approaches and algorithms are represented to solve the problem, and the results are compared. Then, a summary of implemented solutions is discussed.

Chapter 2

Literature Review

Different trick-taking card games have been of interest to various researchers throughout recent decades. Although there are no remarkable papers related to Belote, extensive research has been done for variations of similar trick-taking card games such as Skat, Klaverjas, Whist.

Before tackling and solving a certain problem, it is of crucial importance to understand its complexity in order to derive appropriate algorithms. In their paper *On the Complexity of Trick Taking Card Games*, Bonnet, Jamain, Saffidine [BJS13] focus on the complexity of perfect information games. They discuss several aspects of the game that put constraints on the problem, such as the number of players in each team, suits, and cards in each suit. Also, a notion of symmetry is introduced, meaning that all players have the same number of cards from each suit. Thus, the problem is divided into various fragments depending on the above-described constraints, and different fragments are observed. In general, the authors propose that with an unbounded number of teams, suits and cards, the problem is in PSPACE. Moreover, reducing the Generalized Geography problem, the authors prove that the above-described fragment is PSPACE-complete.

What refers to perfect information Belote Découverte, there is only one player in each team, both the number of suits and cards in each suit are bounded. For this configuration/fragment, there is yet no proof of hardness. However, Wästlund classified the two-player single suit Whist as a problem in PTIME. In case of tractability issues in our problem with 4 suits, we might consider further simplifying it by reducing the number of suits. Another important note is that often trick-taking games include a trump suit that results in different values for cards and hence for tricks. Though, Bonnet et al. view this particular problem setting as a generalization of the ones described above and propose it is still bounded [BJS13].

One of the biggest challenges in card games is imperfect information. The paper *GIB: Imperfect Information in a Computationally Challenging game* by Matthew L. Ginsberg [Gin01] investigates several problems that may arise while solving imperfect information games — more specifically, contract bridge. It describes what techniques are used in GIB (at that time, strongest computer bridge player) to tackle those issues. The first of them is partition search which partitions game states into sets that have the same game value. Ginsberg states that one of the main reasons that partition search might succeed is that in games like chess, tic-tac-toe, "the reason for the victory is generally a local one." In other words, several aspects of the game state might be irrelevant at some point, and the decision is based on only some local aspects. The states that share those specific aspects may end up in the same set in partition search. However, this phenomenon would hardly be applicable to trick-taking card games since the winner is defined to be the player with the maximum number of points or tricks won, so all aspects of the game are taken into account [Gin01].

As we mentioned above, we will discuss different approaches applied to some trick-taking card games similar to Belote. The first game we consider is Skat. Just like Belote Découverte, Skat is a trick-taking game and is played with 32 cards. The number of players is three, and at the beginning, each of them is dealt 10 cards. The remaining 2 cards are dealt face-down. There are 2 main phases of the game: the bidding phase and the card play phase. In the bidding phase, the aim of each player is to become the soloist of the hand. After the bidding phase, the game can be considered a two-player one since the two players losing the bid become partners against the soloist. The card play phase consists of 10 tricks. There are 120 available card points, and the goal of the soloist is to take at least 61 of them. The paper discusses techniques for state value approximation. This is a crucial aspect of the search if the state space is large and a depth limit needs to be fixed for the game tree. The main approach that is discussed is to apply learning. In case of the availability of imperfect information game data, supervised learning algorithms can be used to evaluate the parameters. The rest of the techniques described in the paper mainly refer to the bidding phase and game state inference. The former is not of much interest to us in the scope of this paper. The latter is not relevant in the case of perfect information assumption [Bur+09].

In their paper “A Skat Player Based on Monte Carlo Simulation,” Sebastian Kupferschmid and Malte Helmert [KH06] discuss solving the game of Skat using Monte Carlo simulation and alpha–beta search. Alpha–beta search is an improvement over the minimax search, which is an algorithm aimed at finding an optimal move for a given player at the given state. Details regarding the way these algorithms work can be found in *Artificial Intelligence: A Modern Approach, 4th US ed.* by Stuart Russell and Peter Norvig [RN21]. Kupferschmid and Helmert [KH06] discuss different techniques and enhancements to determine the game-theoretical value of a Skat hand having perfect information. Monte Carlo search is applied to handle the problem of uncertainty in an imperfect information game. The idea of this approach is the following: every time an action is to be made by the computer player, a set of deals consistent with how the game has developed so far are generated. This reduces the problem to a perfect information game so that algorithms like alpha–beta search can be applied to solve it. Then, the results of these deals are used to determine the action to make in the original, imperfect information game.

The architecture of this Skat player consists of a bidding engine and a card play engine. The bidding engine relies on a learning approach. The features include the number of jacks and the length of each suit, and a Least Mean Squares algorithm is used to estimate the probability of winning, given the hand. As already stated above, we are not going to include a bidding engine in our project; thus, we will not go into more details in this part.

Additionally, several enhancements are described to achieve some significant speed-ups. One of them is the order in which moves are considered since this can have a significant impact on the number of nodes pruned during the alpha–beta search. The technique described here relies on the idea that if a certain move results in a cut-off in one subgame, then it is meaningful to consider this move first when the subgame is reconsidered but with different search bounds. The order in which the remaining moves are considered is determined in a way that will allow reducing branching, and this is done using some game-specific knowledge. The experiments show that compared with the initial version with arbitrary move ordering, this version reduces the average search time.

Another improvement is achieved through a forward pruning technique. The idea is to extend the approaches we know for single-agent search problems to adversarial search. As already described above, if we have an upper bound U and a lower bound L for the score in a given subgame in the transposition table, and if M is the current score, then if $M + L \geq 61$ or $M + U \leq 60$, there is no need to examine that subgame further. The question is how to calculate those bounds. The idea is to relax the problem by increasing the set of applicable actions for the MAX player and/or reducing them for the MIN player. This way allows us to obtain an upper bound, and a lower bound can be determined in a symmetric way. It is important that the bounds are as narrow as possible and, at the same time, easy to calculate [KH06].

Forward pruning can be applied to Belote Découverte. If the game value is limited to Win or Lose, the player who takes more than half of points is declared to be the winner. So, pruning might be applied for a portion of the game tree when the score is above 80 (MAX will definitely win) or the sum of the score and value of the remaining cards is below 82 (MIN will definitely win). In case the game value is from 0 to 162, pruning can still be used for the latter case because according to the rules, if the player who declared the trump wins less than half of the points, the opponent is awarded all the 162 points.

The other similar game we discuss is Klaverjas, an example of Jack-Nine card games, which are a form of trick-taking games, where the 9 and the Jack of the trump suit are the highest-ranking trumps. This example of Jack-Nine games is played by four players in two teams. The goal of Klaverjas is similar to the goal of Belote Découverte: each team should aim to get as high a score as possible. Both games have similar rules and rankings of the cards.

The objective of the paper “Computing and Predicting Winning Hands in the Trick-Taking Game of Klaverjas” by Rijn, Takes, and Vis [KH06] is to discuss and determine whether an instance of the game is winning for the starting player. To that end, both exact and machine learning approaches are discussed. Additionally, there are several simplifications in the problem: there is complete information of the team's cards, there is no bidding process, only allowing the first player to decide the trump suit for the instance of the game.

Minimax and alpha–beta algorithms are used to find the exact game value. However, to avoid traversing the whole game tree of games with a huge number of states, some heuristic evaluation functions can be applied. To find an appropriate evaluation function, classical machine learning supervised techniques are used on a dataset of approximately 1M game instances with handcrafted features [RTV20].

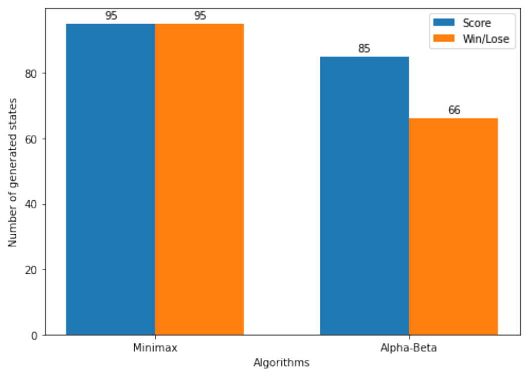
Chapter 3

Solving Belote Découverte

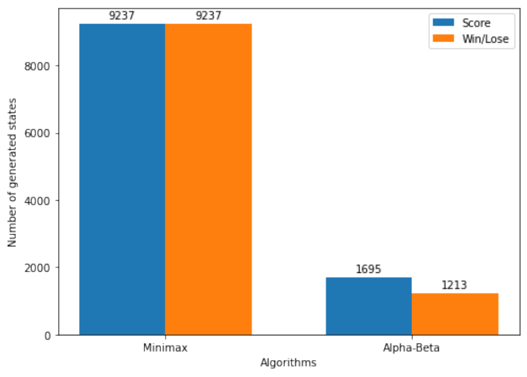
3.1 Methods and Results

After applying the simplifications described above, the problem is reduced to a two-player zero-sum perfect information game. The first algorithm we tried to use is minimax search. However, it is worth mentioning that there is a subtle modification to the standard minimax when it comes to Belote and trick-taking card games in general. Since the lead player is determined by the result of the previous trick, turns of MIN and MAX do not always follow each other. So two adjacent levels of the game tree may represent the same player. Quite naturally, for the case of a 4 suit game, it is not tractable in terms of memory. Although the depth of the game tree is 32, the branching factor plays a crucial role. Experiments are done with different numbers of suits (1-4). The 3 suit (that is 24 cards) variation generates a strategy for each state. Furthermore, alpha–beta search is used to solve the general problem of 4 suits. Although the number of generated states decreases significantly for the problem instances with up to 3 suits, the case with 4 suits still remains intractable.

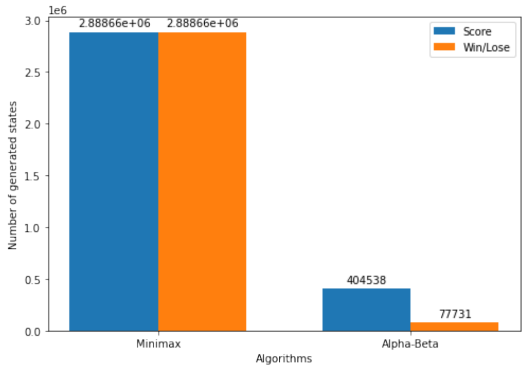
As described in the introduction, a sequence of Belote games is usually played to determine the winner. However, in the scope of this work, only a single instance of the game is considered. Thus, there is no need to maximize the score of the MAX player. Rather, it is enough to ensure a win. So, a modification of the game problem is introduced, with the difference from the initial one being only the utility. In this modified version, the utility of a terminal state is 1 if MAX makes the contract and 0 otherwise.



(a) 1 suit



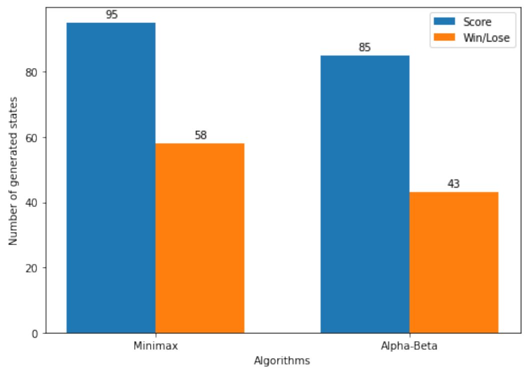
(b) 2 suits



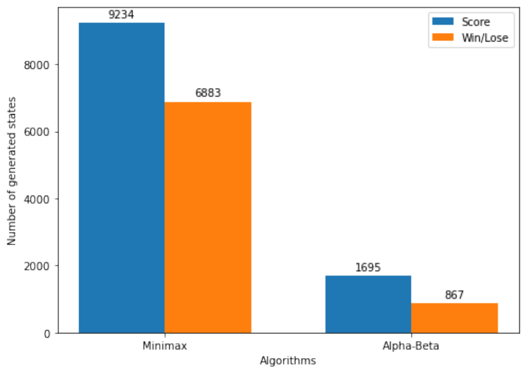
(c) 3 suits

Figure 3.1: A comparison of numbers of the generated states with different numbers of suits

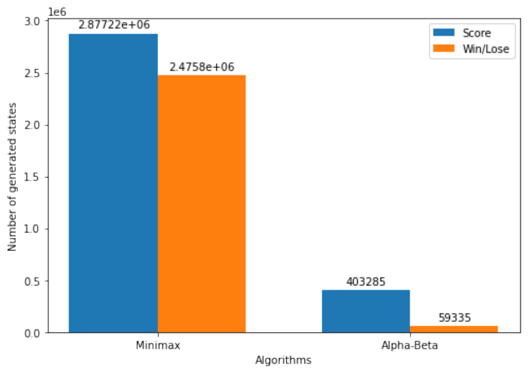
The results of running the minimax and alpha–beta search algorithms with the utility being the score of the MAX player and the 0/1 utility described above are compared by calculating the number of generated states. Figure 3.1a shows the number of generated states when running the algorithms using a 1 suit deck. As was expected, there is no difference between the 2 utilities for minimax search since no pruning is applied here. However, in the case of alpha–beta, the number of generated states is decreased when the game is treated as a win/lose one. This is because having only 0-s and 1-s as utilities allows more nodes to be pruned away. While in the case of having just 1 suit, the difference between the number of generated states is small, the improvement becomes more visible when the number of suits is increased. Figures 3.1b and 3.1c show the results for 2 suit and 3 suit games, respectively. Not only the number of generated states decreases significantly in case of a 3 suit game, but also the 4 suit case becomes tractable with alpha–beta search using the 0/1 utility. The number of generated states by running alpha–beta search on a 4 suit game is 4066474.



(a) 1 suit



(b) 2 suits



(c) 3 suits

Figure 3.2: A comparison of numbers of the generated states with different numbers of suits, applying forward pruning

To improve the performance further, we applied the forward pruning technique described in Chapter 3. To do so, we generalise the idea of making a contract for a small-sized deck, as well. So, for 4 suits, the threshold value is 80, that is, if MAX wins more than or equal to 80 points, he makes the contract. For the 3 suits case, we have taken the threshold value to be 75, which is the total points in the game, 132, divided by 2 and subtracted 1 from it. Using the same procedure, a threshold value is obtained for 2 suits, being 50. Finally, for the 1 suit case, the threshold value is 35. In general, applying forward pruning to the case where the utility is the score of MAX does not improve the performance as much as it does with the 0/1 utility. This was expected, because in the first case we are able to prune a node only if the player who declared the trump wins less points than the threshold, while in the second case, additional pruning is done when MAX wins more than the threshold value. Figures

3.2a, 3.2b, and 3.2c depict the results of applying forward pruning to the search algorithms with 1 suit, 2 suits, and 3 suits cases respectively. In case of the 4 suit case, we again have an improvement over the version without forward pruning, with the number of generated states being 3669142 instead of 4066474.

Chapter 4

Conclusion

4.1 Conclusion

In this paper, we tried to solve the game of Belote Découverte, with the assumptions of having fixed trump, full observability, and the exclusion of melds and belote-rebelote. Different adversarial search algorithms were applied, using 2 different utilities. While treating the game as a Win/Lose one and thus using the 0/1 utility allowed us to generate an optimal strategy for a game with a full deck, the best results were obtained by combining the 0/1 utility with a forward pruning technique.

4.2 Future work

In the future, we plan to obtain further improvements in the performance by using some move ordering heuristics, which can have a significant effect on the number of nodes pruned during the alpha–beta search. Furthermore, state evaluation may be done using some learning techniques or handcrafted evaluation functions. Additionally, we would like to omit some of the simplifications. Specifically, we want to consider the partially observable case with melds and belote-rebelote.

Bibliography

[Par79] David Parlett. *The Penguin Book Of Card Games*. 1979.

[Gin01] Matthew L. Ginsberg. *GIB: Imperfect Information in a Computationally Challenging Game*. 2001.

[KH06] Sebastian Kupferschmid and Malte Helmert. *A Skat Player Based on Monte Carlo Simulation*. 2006.

[Bur+09] Michael Buro et al. *Improving State Evaluation, Inference, and Search in Trick-Based Card Games*. 2009.

[BJS13] Edouard Bonnet, Florian Jamain, and Abdallah Saffidine. *On the Complexity of Trick-Taking Card Games*. 2013.

[RTV20] J.N. van Rijn, F.W. Takes, and J.K. Vis. *Computing and Predicting Winning Hands in the Trick-Taking Game of Klaverjas*. 2020.

[RN21] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 4th US ed.* 2021.