COMP4106A Project Report

NICHOLAS CHIASSON — 100891716

Euchre

Problem Domain

Game Rules

Euchre is a trick taking, paired team based card game. There are variations of the game, but here is a brief summary of the rules as I am familiar with them. Before the game starts, four players form into two teams, seating teammates across a from each other at a table. The deck of cards used for the game has 24 cards, made from a standard playing card deck with cards ranked 2 through 8 being omitted (jokers also omitted). At the start of the game, a dealer is chosen randomly. The game immediately enters the dealing phase. During this phase, the dealer deals to each player in clockwise order beginning with the player on his left. The dealer must deal 5 cards to each player over two cycles around the table, in any way that he likes. For example, on the first round he may give two of the players players two cards and the other two players one card, then three cards to the first two players and four cards to the next two players on the next round. What matters is that each player ends up with five cards after two rounds of dealing. After the cards are dealt, the game enters a new stage in which a suit is chosen to be “trump”. The trump suit is considered the highest valued suit for the duration of the hand. To go about choosing trump, one more card is revealed from the top of the remaining deck. Each player, in clockwise sequence beginning again with the one left of the dealer, is asked if that player wants to order up the card as trump. If any of the players decide to order it up, a few things happen. Firstly, the card’s suit is named trump. Next, the player’s team who ordered up the card is labelled the “makers” team for the hand since they chose trump, and the other team is labelled the “defenders” team for the hand. These team teams score points differently since one team is supposed to have an advantage over the other, but this matter won’t be discussed in this project. Next, the player who ordered up trump is asked if he wants to play the hand alone or with his teammate’s help. This also affects how points are scored since it effectively gives one team a handicap. Finally, the dealer picks up the card, then chooses any a from his hand (including the one he just picked up) to put back underneath the deck, returning him to a five card hand. Should no player choose to order the card up as trump, each player in the same order is asked which suit they want to be trump, having the option to pass if they don’t feel confident about any suit. If all players pass, the hand is considered failed and the deck is shuffled and dealt anew, repeating the entire deal phase and trump selection phase until trump is chosen. Having trump chosen, the first round of the hand begins with the player left of the dealer placing a card from his hand onto the table face up. During a round, each player in clockwise order puts a card onto the table face up. Each player must play a card matching the suit of the one played by the first player to put down a card in the round. However, if a player does not have a card of that suit, he may put down any one of the cards in his hand. After each player has put down one card, the player who played the highest ranking card wins the round. Cards are ranked as follows for any given round:

1. Jack of trump suit (Right Bauer)
2. Jack of suit matching colour of trump suit (Left Bauer)
3. Ace of trump suit
4. King of trump suit
5. Queen of trump suit
6. Ten of trump suit
7. Nine of trump suit
8. Ace of first suit played
9. King of first suit played
10. Queen of first suit played
11. Jack of first suit played
12. Ten of first suit played
13. Nine of first suit played
14. Any card not of trump suit or first suit played

As one can tell, there are two special cards: the Right Bauer and Left Bauer. Playing the Right Bauer, which is the Jack of trump suit, guarantees a won round. The Left Bauer is the next best card that can be played, and it is the Jack with the same coloured suit as the trump suit (for example: Jack of Diamonds if trump is Hearts, or Jack of Spades if trump is Clubs, etc). The player who won the round plays the first card of the next round, and these rounds continue until all players have emptied their hands (totalling five rounds). At the end of the five rounds, one team will have won more rounds than the other and points are assigned to each team based on their label and how many rounds they won. Finally, to play a new hand, a new dealer is chosen as the player seated left of the current dealer, and the game enters back into the dealing phase.

Problem interpretation

As one might be able to tell, even without getting into the specifics of scoring points, the game is significantly complex in mechanics. For this reason, this project will not aim to implement every facet of the game, but instead target a simplified version, complete enough to make use of AI techniques to run the core parts of the game. With that being said, here is a discussion about the game mechanics intentionally omitted from this project.

The game will not implement the trump selection phase of the game. Instead, trump will be automatically ordered up from the top card of the deck after dealing and the dealer will discard his lowest ranking card. This was decided upon to simplify the implementation. This particular game mechanic does provide a bit of an AI problem, but not significant enough to be necessary for the implementation of the game. Should this feature have been included, a possible obvious brute force solution would be to have each player, when deciding on the best suit for trump, to generate all permutations of possible hands for opponent players, and then determine the value of which suit should be trump to best benefit the player himself. This kind of solution would not be very space considerate at all and would only be a draft solution.

The concept of teams in the implementation is more implicit than explicit. What that means is that the only notion of teams in the game is derived from the particular AI algorithms and heuristics, which will be discussed in further detail later in this report. All of that to say, two of the players in this implementation play as though they are on a team, while the other two play as if they are alone.

All computer players know every other player's hand. This was a main design issue and will be discussed further in the report.

Finally, points in the game are only assigned for each round in a hand. The points for each player reset at the beginning of each new hand, thus the game is much more short-lived than its real-life version. This was intended to also simplify the implementation, however, it certainly would have been an AI challenge having the computer players determine what cards to play by searching a depth which spanned more than one hand. Needless to say, that would have opened the implementation up to a huge space and efficiency problem.

Motivation

When deciding on a project topic, I thought that this game would make for a great learning experience in AI because of how many factors of decision making it consists of. I had taken a liking to it because it would allow me to implement multiplayer game playing algorithms with teams, which I thought would be a challenge at the time. The game itself is actually quite enjoyable once you get an understanding of the rules, but it's the complexity of the rules that pose challenges in design and to new players. For these reasons, I figured it would, if anything, be a great chance to flex my software development skills to try creating an artificially intelligent opponent to practise the game against.

Design

Architecture

The application was designed with a model/view/controller design architecture pattern in mind. The controller directs the flow of the game, setting up all of the players, card information, and performing player turns. The view creates the UI and sets up the interface between the UI and the controller. Finally the model contains classes for data in the game, such as cards, players, and game state.

There is the concept of a game loop, but not in the same sense that normal games understand it. Normally, a game loop runs at a high frequency, 30-60 hertz, but in this game, the game loop is used to pass the turns of each round of the game. As such, the loop is implemented with an event queue, posting the next turn every second, with the exception of the user player’s turn. For the user player’s turn, the game waits for input before enacting the turn and posting the next turn.

The game can be run in spectator mode or in play mode. If run in spectator mode, all of the players will be computer controlled. If run in play mode, the user must control Player 1 and the rest of the players will be computer controlled.

Addressing the fact that all computer players know all players’ hands, this issue was carefully considered. When a computer player takes its turn, that player must search through all possible future turns up to some depth to determine what play would be the best in the current state of the game. The original approach to the problem was for the player to generate all possible combinations of other players’ hands and then conduct the search based on each of these possibilities. This would be possible, but not space or time efficient. The issue is that it treats the search as many searches from thousands of separate root nodes. The search at depth 12 can take seconds already, so using combinatorics to determine possible hands of the other players would not be such an easy thing to keep under good time. Therefore, to remove the issue, trivial as its implementation is, it was decided to have computer players simply know all other players’ hands, that way, the breadth of the search could be vastly reduced.

AI Techniques

This project implements multiplayer game playing search algorithms. These AI algorithms are used when a computer controlled player needs to decide which card in its hand to play during any given turn. The algorithms implemented include Min-Max search with Alpha-Beta pruning, and Paranoid Min-Max search with Alpha-Beta pruning. It is worth noting that the regular Min-Max with Alpha-Beta pruning was used for considering teammates, since it maximized every other search depth level, which happens in this game to be where a teammate sits relative to any player.

The searches are conducted with a depth of 12 since the breadth of each search iteration becomes smaller with depth.

The state of the game at any given moment consists of the cards in each players’ hands, each player’s number of points, the notion of which cards are valid for the each player to play, the current player, the trump suit, and the suit that lead the round. Searching through states like this can be very fast and allowed for a search depth of 12 within reasonable time due to the fact that not all cards in each players’ hands are valid to be played at any given time. This means that many possible states can only be expanded into very few adjacent states. The first player to play a card in a card after cards are dealt will take the longest time to determine which card to play, since all of his cards are valid to be played, so there is a larger breadth to search. However, each consecutive player will take less time since valid cards depend on the leading card of a round. This, coupled with the fact that each player’s hand size decreases with each passing turn means that searching can be done very fast later in the game.

The heuristics used for the search algorithms were as simple as calculating player points. The team heuristic would add both teammates points and subtract all opponent points, while the paranoid heuristic function would add the current player points and subtract all other player points.

In this implementation of the game, it was decided to make players 2 and 4 (left and right) play paranoid, even though they should be on a team together, and make players 1 and 3 (bottom and top) play considering the benefit of their teammates (each other).

Results

The implementation, though very simplified compared to the actual game, seems to work very well and players always appear to make smart decisions. As expected with this kind of game, the players working for their team seem to win more often, but not always. This makes sense because maximizing two players provides more possibility for the team to win rather than only looking to maximize one player. The game seems to pass the difficulty test as well, since a human player cannot win with ease. In other words, the AI appears to match or even exceed human skill level.

Expanding on the project, it would have been nice to implement more multiplayer game playing algorithms. Perhaps it would have been interesting to see how a Best-Reply search algorithm would have performed in this game, being a context in which Best-Reply is really not suitable.

One thing that would almost certainly improve the efficiency of the searching done in this context would be to implement move ordering. Such a feature would likely allow for a much greater search depth.

References

Rules

https://cardgames.io/euchre/
https://en.wikipedia.org/wiki/Euchre

AI Algorithms

http://people.scs.carleton.ca/~oommen/Courses/COMP4106Winter17/AICh04IntelGamePlaying.pdf

Appendix

Running the Code

Instructions also noted in README.txt

- Install Python 3.5 or higher
- Open terminal
- Change directory into project src directory
- Enter into command line: **python3 main.py**
- By default, game is run with four computer players
- To run with user player (to actually play the game yourself), pass command line argument “true”:
  - Run **python3 main.py true**