Euchre AI

Final Project for Artificial Intelligence and Heuristic Programming

By Andrew Klenotic & Dylan Stavarz

9 December, 2014

The University of Akron, Fall 2014

**PROBLEM STATEMENT:**

In today's busy times, it is often difficult for a group of friends to get together reliably for any sort of social activity. This is especially problematic when the activity planned for requires a certain number of participants to be able to work effectively and one or more participants has to cancel at the last minute. Trying to plan to have a substitute is not always a viable option as that person will not have a very good time if everyone does in fact arrive as planned. This was the inspiration for our EuchreAI implementation. Euchre is a card game played generally by four people in two teams of two players. The game is one of taking tricks, where each player plays one card and based on the values of those cards, the player playing the highest ranking card “takes the trick”. The values of the cards change depending on which suit is named as the “trump suit” (the choosing of which is part of the strategy employed) and the “hand suit” or “trick suit” (based on the suit of the first card played in each trick). There are five tricks per hand, at which point the score is tallied based on how many tricks were won by each team and which team chose trump and then the deal switches to the next player for the next hand. Generally Euchre is played to a set number of points though alternatively it could be played to the best score after a certain number of hands.

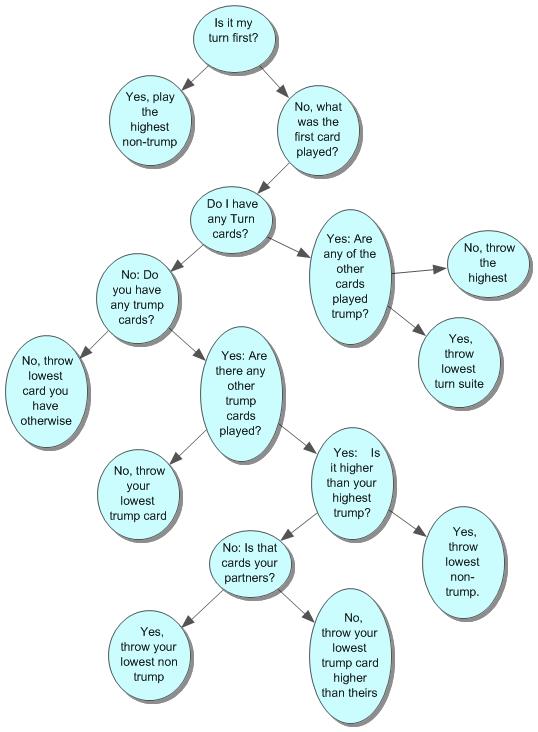
The EuchreAI project is designed to allow a group of Euchre players who are short by one player to replace the missing human player with our AI in order to preserve the social feel of the game and not turn it into just another “Euchre on the computer”. The system is designed to replicate the decision making process of a human player and allow the interaction with the system to supplement real table play and not take away from it. Additionally, implementing scorekeeping by the AI will allow the human players to pass on part of the tedious task of scorekeeping to the computer in the hopes of increasing the fun and social interaction further.

**APPROACH:**

The core of our project will be the EuchreAI software itself, developed on Linux in C++. The base AI will be implemented by way of two primary rule-based decision trees. The first decision tree handles the “Calling Round” or the round of each hand before the tricks are played where the players define the “trump suit” for the hand. Euchre is played with a deck of 24 cards (A, K, Q, J, 10, 9 of each of the four suits, Spades, Diamonds, Hearts and Clubs). Twenty of those cards are dealt to the players (five to each of the four players). Dealing is handled in two rounds, the first round each player is dealt one to four cards and in the second round, they are dealt the remainder so that each player ends with five cards. The EuchreAI program implements this by randomly generating the number of cards for the first round and requiring the human partner to handle the actual dealing of the cards. Once the cards are dealt, the remaining four cards (known as the “kitty”) are set aside. The top card of the kitty is then turned over to help decide the trump suit. Starting with the player to the left of the dealer, each player in turn decides whether to “pick up” or “order up” the top card (in which the suit of that card becomes the trump suit). The player then takes that card into his or her hand and discards one to keep their hand at five cards. If no player takes the card (passes) then a second round begins where the players (again starting with the player to the left of the dealer) can arbitrarily choose trump from the remaining three suits.

In our EuchreAI implementation, the calling round is implemented by adding the values of the cards in the hand based on a function designed to return the value of an individual card given the trump and trick suit that we designed. For example, the Jack of the trump suit (called the “Right Bower”) is the highest valued card, and our function returns a value of 24. The second highest card will be the Jack of the suit of the same color (known as the “Left Bower”) as the trump suit (if the trump suit is Spades, the Jack of Clubs is the second highest card and would have a value of 23). The remaining cards of the trump suit are valued Ace through nine, followed by the cards of the trick suit, Ace through nine. This valuation function is central to many of the decision making processes in our EuchreAI implementation. In the case of the calling round, the trick suit is ignored (as it is not set yet) and only the values of the cards based on the “hypothetical” trump suit are considered along with having that card in the computer's hand and losing the card of the lowest value. Essentially, it is simulating the value of the hand if the computer were to “order up” the top card. If the sum of the card values exceeds the current threshold (at the time of this report it is at 65 but please refer to the section on *Future Work* for more on this) the AI is programmed to pick up the card. As mentioned, should all the players fail to pick up the top card, they begin a selection process. For this, the card valuation function is called again, though this time it is called for each of the remaining suits. If any of the sums for any of the suits exceeds the 65 threshold, the AI will select the highest of those suits. Should all of the suits fail to exceed the threshold, the AI will choose to pass. If all players pass this second time, all the cards are discarded and a new deal takes place.

Once the calling round is completed and a trump suit selected, the players play five tricks. The player to the left of the dealer begins the first trick with each player playing in order. Once all players have played one card, the winner is determined and the winner of the trick plays the first card of the next trick until all five tricks have been played and the cards are dealt again. In our EuchreAI implementation, we have implemented a second and more complex rule based decision tree combined with the card valuation function (and variations of the function for different given conditions). Since the decision tree is more complex, we will not outline the process step by step as we did for the calling round, but will include it here for reference. The functions and variations used will be outlined more in the *Design and Implementation* section of this report.

****

*Decision Tree Example*

**Design and Implementation:**

As previously mentioned, the core EuchreAI program was developed in Linux using C++ 11 and currently uses the console for input and output (a GUI is planned, see *Future Work* for details). For ease of testing, a barcode reader was also used and playing cards were labeled with barcodes for easy reading into the computer and also to ensure that no human players would be required to know the computer's cards. Voice “output” was also implemented through calls to the **eSpeak** engine originally developed by Jonathan Duddington and available at [http://espeak.sourceforge.net](http://espeak.sourceforge.net/). The voice synthesis was developed to ease interaction with the computer and not require the human players to “gather around the screen” which would defeat the main premise of this project which is to preserve the human interaction. The EuchreAI program itself was written from the ground up and did not use any previously available open source euchre game as a template. This ensured that the EuchreAI would be tailored to the project and that none of the elements of the previous software would interfere with our implementation. The data structure for the cards was also designed specifically for this project. EuchreAI was designed to be used with both male and female personas. However, other than the name (John for the male and Jane for the female) and the voice used for the synthesis (which also has a silent option) the personas are functionally identical. EuchreAI also implements a log file function that can create log files to disk and/or print log file information to the screen as well. This log file was instrumental in helping to determine how the AI followed the decision tree process in the playing of the tricks at each step in the process based on the cards played by the other human players and will be used in the future to help develop a learning AI (see the section on *Future Work* for details). Here is an excerpt from the log file.

|  |
| --- |
| PLAY A TRICK  ============  Drew is going first in this trick.  Human Drew plays King of Spades  Trick Suit is -=> Spades  Human Jennifer plays Ace of Spades  Human Kevin plays Queen of Diamonds  My turn to go - Trump=Clubs - Trick=Spades  MAIN AI ROUTINE BEGINS  ======================  Current Display of Cards.  1 [0] C13 cv(21) / 2 [1] C11 cv(24) / 3 [2] C14 cv(22) / 4 [3] C9 cv(18) / 5 [4] C10 cv(19)  My lowest card is - 3  My highest card is - 1  Given Trump:C Trick:S.  Current Display of Trick Cards.  [0] J0 cv(0) / [1] S13 cv(16) / [2] S14 cv(17) / [3] D12 cv(4)  The lowest card is - 3  The highest card is - 2  Given Trump:C Trick:S.  Checking for Trick cards - Spades  :.I do not have Trick cards. Do I have any Trump - Clubs  :.I do have Trump, are any other Trump cards played?  :.There are no other Trump cards, so play my lowest Trump card. (FTW!)  I'm playing card number 4 the Nine of Clubs  Jane takes the trick! |

In this example, Jane, the AI player has all clubs as outlined in the display of cards. The cv() is the value of the card valuation function that as was mentioned previously, is at the core of some of the AI decisions. The “highest” and “lowest” cards are listed by the array index of the card [0]-[4]. The J0 represents a “joker” or in our implementation, a blank card or one that has not been played yet. After displaying the initial state, the AI begins to cycle through the decision tree. In this case, it only needs to check three steps before choosing the fourth card (in this case, the card is identified by how the humans would see it, 1-5 instead of the array index 0-4). As indicated in the log file, Jane (our AI) took this trick by playing the lowest card of the trump suit (as there were no other trump cards played, playing a higher one was not necessary, saving that card for later, which is good strategy). In the event that a card of the trump suit had been played, the AI would have chosen to play the lowest card that was greater than the highest trump card played. This is where our custom functions factor in to the decision tree. This is a listing.

|  |
| --- |
| **char oppositecolor();** // Returns the suit that is the opposite suit of the given suit's same color.  **bool insuit();** // Returns whether the given card is in a certain suit.  **bool notinsuit();** // Returns whether a card is not in a given suit.  **bool gotany();** // Returns whether any cards of a specific suit are in a hand.  **int cardvalue();**  // Returns the value of a card based on trick and trump suits.  **int lowestcard();** // Returns the array index of the lowest value card of a hand.  **int lowestcardgt();** // Returns the array index of the lowest card, greater than a card value.  **int highestcard();** // Returns the array index of the highest card in a hand.  **int lowestcardof();** // Returns the array index of the lowest card of a certain suit.  **int lowestcardofgt();** // Returns the array index of the lowest card of a suit higher than a card value.  **int highestcardof();**  // Returns the array index of the highest card of a given suit.  **int lowestcardnotof();**  // Returns the array index of the lowest card not of a certain suit.  **int highestcardnotof();** // Returns the array index of the highest card not of a certain suit. |

The functions above have all been implemented to take into account the “Left Bower” as was mentioned above, which is the second highest ranking card. It is considered to be a member of the trump suit, and not a member of its actual suit for the duration of that hand, so all functions that return a suit or factor in whether a card is or is not of a certain suit has to take this condition into account. It should be noted that the Jacks of the opposite color (the red Jacks given a black trump suit, for example) have no special value or function and are ranked normally. There is an option in Euchre where a player can choose to “go it alone” if they feel their hand is strong enough. In this case, the partner of the player “going it alone” does not participate in that hand and the player takes on the other team alone. We chose not to implement this option at this time because first, should the AI's hand be strong enough, chances are the outcome will not change and second, having the AI choose to ignore the partner goes against the premise of the implementation which is to allow the humans to play. In additon to the “stretch goal” of implementing the AI being able to “speak” we also were able to implement the goal of having the AI keep score, though as this does not actually have an Artificial Intelligence component, the specifics of the scorekeeping will not be covered in this report. We also chose not to implement any hardware of our own at this time (for example, the ability to physically deal cards or have a physical card holder and indicator for the AI's cards). This was not found to be necessary to realize the basic concept and would work better as a future evolution of the project.

**Results:**

As of the writing of this report, we have developed a competent AI capable of sufficiently complimenting a human player. Generally speaking, the computer acted as predicted. In fact, in some cases the computer appeared to make a wrong play, but on analysis it did in fact make the correct play (with the discrepancy being the result of our failing to take a factor into consideration). We have found a condition late in the testing process where if the “trick suit” and “trump suit” are the same, there are times where the AI could have won but instead chose to play a lower card. This will be corrected as the project progresses. One other minor “bug” was discovered late in the testing process where if the AI plays the “Left Bower” first in any given trick, it considers the trick suit to be the actual suit of the “Left Bower” card when in fact it should be the “trump suit” as we explained earlier in this report. However, given that the only card greater is the “Right Bower” and that the human players are capable of noting and correcting for this, it would not have had any impact on prior testing or results. That said, the EuchreAI does seem to perform within expectations.

With regards to the social aspect of the game, the implementation seems to be successful in keeping the spirit and flow of an actual card game of Euchre. Especially with the barcode scanner, the game seems to flow quickly and with the voice integration, the players can effectively play the game without having to pay much attention to the screen in general. While there are still improvements and enhancements that could help perfect this system, the basic concept itself has proven to be successful and workable.

**Conclusions and Future Work:**

As we have shown basic proof of concept, at this point we have to consider the future of this project. While we played a few dozen hands during the testing and each of those hands played five tricks (which called the main AI routine five times), there are 5,100,480 possible variations of just one single five-card hand. Considering each player has a five cards, that is a huge combination of possible states to be tested. Additionally, the EuchreAI needs to develop more of a spirit of teamwork. At this point, it does not really consider when to let its partner try for the win in all cases. As part of the future of this project it would be good to implement an ID3 algorithm using C4.5. To accomplish accumulating enough data to allow this to work, a website has been considered where users can download EuchreAI for their own use and send the log files back through the website (or alternatively sent directly by EuchreAI itself) for analysis. For this to be practical, in addition to developing the website, some enhancements to the program should be implemented.

First, to appeal to the greatest number of potential testers, the program should be ported to a Windows environment as well as Mac and tablet/phone platforms. Additionally, a GUI should be designed to make the interaction even easier. Optical character recognition could be perfected to replace the card reader provided the conversion process can be carried out quickly enough (so as not to slow down the game). Finally, speech recognition (most likely CMU Sphinx) should be implemented to allow players to bypass any keyboard interaction. As Euchre does not allow for partner communication it would not need to be implemented much further than that.

Additionally to enhance the social experience, it would be good to put a face to the EuchreAI. Having a male and female model different facial expressions to indicate status (happy if the AI's team is ahead or sad/angry if they lose a hand or are behind) along with appropriately spoken phrases of encouragement or discouragement could help enhance the social experience. As a further evolution of this project, a fully contained unit, much like a table top robot would be possible, which would contain the ability to deal and read its own cards with minimal human assistance.

Overall, we feel that this project has good future potential. With the basic enhancements to the interface and additional platforms, it should be very easy to encourage others to use the software and provide additional testing data. Development of the project was enjoyable and the testing was fun so it should stand to reason that it would likely appeal to a wide enough base that we should be able to get a good sampling of data to perfect the AI routines.

**Remarks and Credits:**

Idea proposed by: Dylan Stavarz

Trick Decision Tree Design & Graphic: Dylan Stavarz

Calling Round Decision Tree Design: Andrew Klenotic & Dylan Stavarz

Programming Implementation: Andrew Klenotic

Testing: Andrew Klenotic

Initial Presentation: Andrew Klenotic (edited by Dylan Stavarz)

Progress Update Presentation: Andrew Klenotic (edited by Dylan Stavarz)

Final Presentation: Andrew Klenotic

Final Report: Andrew Klenotic