University of Nevada, Reno

Cooperative Artificial Intelligence in the Euchre Card Game

A thesis submitted in partial fulfillment of the
requirement for the degree of Master of Science in
Computer Science

by

Benjamin E. Seelbinder

Dr. Bobby D. Bryant/Thesis Advisor

December, 2012

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

BENJAMIN E. SEELBINDER

entitled

Cooperative Artificial Intelligence In The Euchre Card Game

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Bobby D. Bryant, Advisor

Thomas Quint, Committee Member

Frederick Harris, Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

December, 2012

Cooperative Artificial Intelligence in the Euchre Card Game

Benjamin E. Seelbinder

University of Nevada, Reno, 2012

Supervisor: Bobby D. Bryant

Abstract

In order to provide a method for optimization in multi-agent systems using artificial intelligence (AI), a set of logic rules provides information for traversing options and maximizing expected utilities. This project explores the use of logical decision making in multi-player games, specifically in the card game Euchre. A variety of algorithms implemented herein compare decisions as quickly as possible while still trying to optimize their moves. Based on these comparisons, a better method for optimizing strategies becomes evident. This implementation contains seven AI agents with some slight variations on those AI agents. Two pairs of AI agents will work together against another pair of agents in order to maximize not only their own personal goals, but their team score. The complexity of these agents having to work on a solitary level as well as a team level takes AI to a powerful level. This thesis also presents some promising results, compared to some other Euchre programs, for how AI can cooperate as a team, whether or not the AI type is the same or different.

Contents

List of Figures

Chapter 1

Introduction

1.1 Agent Performance through Artificial Intelligence

A great heuristic in computer gaming measures the ability of the agent to make decisions which optimize gains. However, having a computer agent as an ally turns the tables and a different set of actions and intelligence need to be taken into consideration. All too often in Artificial Intelligence (AI) the "smart" agent optimizes its own gains. This works well in certain situations, but there are times when a more cooperative approach can yield higher gains. This paper takes a look at some basic AI agents in the card game Euchre and how they interact when game-winning decisions must be made. Going a bit further, implementing some cooperative agents demonstrates the true power of two agents who work with one another, maximizing the team's outcome rather than optimizing individual outcome. This idea has been defined by the Prisoner's Dilemma [26]. Another book by Millington and Funge and a book compiled by Rabin talk about AI for games (more specifically computer games) and outlines general principles of what agent-based AI is and how to go about using it [24, 27]. Card games remain a popular AI base, frequently studied because they have various qualities which are pivotal for studying and understanding [17, 20, 23]. For example, the most popular trait card games offer is imperfect information. This means that hidden information exists to each player. In

poker, an agent only sees his own cards plus any card flipped on the table or in the discard (if the agent has an ability to remember past events). The other players' cards and the cards which have yet to be dealt are a probabilistic mystery. Another great trait about card games, especially the trick-taking games, is their turn-based nature. This lays out an order of play and adds to the element of prediction. Having seen what an opponent has played can grant valuable information as to what should be played next. However, most of these games involve personal gains (such as poker, blackjack, hearts, etc.). Euchre remains a bit unconventional in that the two person teams strategize cooperatively in order to optimize their gain, but they still try to observe and counter the opposing team. This paper details various methods to Euchre in order to discover a strategy which will prevail amongst the others. From a random card-tossing agent to a fully-functional, decision-capable agent, certain strategies shine through and there is a clear winner for the agents implemented in Euchre.

1.2 Card Game - Euchre

Euchre is a trick-taking card game that has been around for more than one hundred years [15]. The deck consists of 24 cards (9, 10, J, Q, K, A of each suit). Each of the four player is dealt 5 cards, the 21st card is offered as the trump if a player right of the dealer decides to use that card as trump. The three cards not seen by the players at the beginning of the deal remain unknown and become part of what is called the kitty. If the trump card is 'ordered up', that suit immediately becomes the trump, the dealer collects that card, and must discard another card in order to again have five cards in his/her hand. The Jack of the trump suit becomes the strongest card in the game, the Jack of the off-suit (same color, different suit) becomes the second highest, and the other cards in the trump suit follow: A, K, Q, 10, and 9 (Figure 1.1 illustrates this). Once a player chooses a suit and declares it as the trump, the team which calls the suit becomes the 'makers'. The goal at this point shift into the makers having to collect at least three

tricks in order to obtain points. If the makers gather all five tricks, they receive 2 points; if they get three or four tricks, they receive 1 point; however, if the makers only get one or two tricks, the opposing team receives 2 points for 'setting' the making team.

Figure 1.1: The best 5 cards in the game when spades is trump. (image from Wikipedia)

As game play starts, the dealer chooses a card with which to start. This card determines the lead suit. In order for a play to be considered legal, other players must follow this suit if possible; otherwise, any card may be played. Once all players have played a legal card, the highest card in the trick wins (trumps beat the lead suit, the lead suit beats other non-trump off-suits, and higher card values beat lower card values). One interesting thing about Euchre is that the Jack of the trump suit and the Jack of the side suit (same color as the trump) are the two highest cards in the game, respectively. The winner of the trick then collects the trick and may commence the next trick with a card from its hand. Play continues in this fashion until all tricks have been played and collected. At this point, the total tricks taken by the team reward the number of points as mentioned above. Play continues in this fashion, rotating the dealer, until a team has 10 or more points; this team is the winning team [10, 33].

1.3 AI Implementation

Now that the game of euchre has been outlined, the ideas behind the AI can be discussed further. When considering the agent's necessary information state in order to make a

decision, the following items need to be taken into consideration:

- The basic rules
- The hand
- The cards played so far this hand
- The trump/lead
- The values of the cards

By analyzing this information, the agent has the capability to make a more informed judgement call when following the decision tree method for selecting a card to play. The basic rules restrict the players to only selecting valid cards to be played out of their current hand. The cards which have been played are important because they give an agent more complete information on which cards stay hidden in players' hands. For example, if the Jack of the trump suit has been played, an agent need not worry about the Jack of the side suit being beaten by any other card. The trump and lead are extremely important when the AI is selecting a card to play because it skews the worth of the cards. For example, if an agent has more cards of the lead suit, they are worth more now than if the previous trick had a different lead suit. The heuristic is the most important part of the AI. It encompasses most of the above information in order to evaluate which card would be the best to play. For this project, a basic table of heuristics is given by Figure 1.2.

	9	10	J	Q	K	A
♠	1	2	3	4	5	6
♥	100	200	1500	400	500	600
♣	10	20	30	40	50	60
♦	1	2	1000	4	5	6

Figure 1.2: The card values for a trick with hearts as trump and clubs as lead.

The values determined in Figure 1.2 are computed dynamically using simple multiplication and division based on what a player chose as the lead suit and which suit was

set as the side suit from this decision. These values differ slightly when the trump suit and the lead suit are the same. In order to really optimize how these heuristics work, this project uses various methods of rule induction to provide condition-action rules, decision trees, or other structures [21]. Chapter 3 further discusses how these values are used in order to make a decision on which card to play.

The following section of this thesis will introduce the reader to some key terminology necessary in order to understand this work. After that, a look into some related work will provide the reader with some information about comparative methods and other such research projects from which this thesis has been derived. Then the methods and implementation of each of the agents in this thesis will be outlined so as to draw comparisons and conclusions. Nearing the end, there is some analysis of the methods and some information about why a certain agent performed well or performed poorly. Finally, future directions for this project are discussed as final conclusions are drawn.

Chapter 2

Related Work

2.1 Games and Computing

Since at least 1950, games have been used to look into various fields' problem solving [6]. Computers implement games; effectively measuring and analyzing strategies. Games have played a large role in such fields as probability theory, problem solving, artificial intelligence (AI), etc. [18–20,25]. Also, within the realm of AI, various types of intelligence have been analyzed; such as reinforcement learning, neural networks, Markov decision processes, opponent modelling, pruning techniques, etc. [4, 13, 32]. When it comes to AI, the state space is an extremely important structure and the analysis of this structure in order to extrapolate information becomes a very detailed process. Some ideas for this thesis were inspired by the ability to improve state space evaluation as proposed by Buro *et al.* [8].

It seems that nearly every aspect of gaming has been analyzed to a certain degree. Most of the time the research topic at hand takes a deeper look at optimizing strategies or maximizing personal gains. With all of the various genres of games at hand, generalizing a strategy proves to be an extremely daunting task. Even though it may be daunting, there has been quite a bit of work done in the field of card gaming [11] which has been similar to that of computing [30].

2.2 Card Games

From a generic viewpoint on card games, Jonathan Schaeffer speaks about the difficulties associated with imperfect information games. He says that "programs must deal with incomplete knowledge, multiple competing agents, risk management, opponent modeling, and deception" [30]. Although these are all very true of Euchre, not all of them will be addressed in this thesis.

In 1983, Jack Mostow reasearched a concept in which he coined a term about how to make advice 'operational'. He defines operational as the final product of converting non-operational advice (such as "don't lead a suit in which some opponent has no cards left") into an executable procedure [25]. The reason advice is non-operational in this example is because the player cannot see the opponents' cards and therefore cannot make an educated guess based on this. His research creates a set of logic-based rules which combine non-operational items into a conditional statement, thereby making them operational. His research is a great example of taking a fresh look into the logic behind AI in card gaming (whether it be gambling, trick-taking, etc.). Since this paper, various other authors have looked into the methodology of card games and how to create a strategy for optimization of some sort. In 1998, two papers were written about research which delved into the analysis of two very popular card games: bridge and poker [4, 31].

The paper on Bridge by Smith *et al.* outlines planning techniques for the championship of computer bridge [31]. The bridge agent described in this paper succeeds in furthering the significance of AI planning systems. The thorough research from this paper aided the design of this thesis by illustrating the importance of every aspect of a game and how to look at them in order to determine the best possible method of action. The bridge player uses a game tree to evaluate a turn based on a weighted average. Although this method demonstrates robust methods for determining what to play, it does not give much insight as to how to assist a partner.

Further research by Billings *et al.* investigates opponent modeling through the

popular game of poker [4]. In their research, Billings *et al.* defines opponent modeling as being able to "determine a likely probability distribution for your opponent's hidden cards" [4]. However, the paper looks into the modeling and not into the other, more in-depth, characteristics that might better the player's ability for success; such as aiding a teammate.

In 2001, two theses were written about Euchre, however, they just scratch the surface of computational programming. A thesis by Wright takes analyzes the math in Euchre including the shuffling phase and the dealing phase [35]. Although these might be helpful in creating a database for the cards a player could be holding, the redistribution and sheer magnitude of the various deals creates a computation mess. The second thesis by Holmes creates four agents for Euchre. However, all four of the agents base their decision processes on reinforcement learning [14]. This leaves little room for improvement in the way of cooperative gaming since all of the agents are looking for and noting most of the same strategies while trying to optimize personal gains.

Another thesis by Livingston [22] considers the implications of reinforcement learning in Cut-Throat Euchre and Sergent-Major. These games pose similar problems to regular Euchre and the strategies are extremely comparable. His results show that after about 200,000 games, the percent of games won increase around 2% (28%, up from 26%, wins for one type of agent and 36%, up from 34%, for the other). He discusses his results and mentions that although reinforcement learning might not be the most effective manner by which to train an agent, it still shows improvement. His ideas helped direct these methods of playing in order to find a great strategy overall rather than trying to develop one on its own. This thesis outlines a structure which takes advantage of both of these concepts to create a new kind of player with a special twist.

2.3 Existing Euchre Programs

Various types of Euchre programs exist in the world, but very few are open for comparison. The following outlines two different Euchre programs which were made available through the Internet.

2.3.1 Bid Euchre

The first program Bid Euchre was written by John Ratliff in August of the year 2005 [28]. Bid Euchre was written in C++ and has one AI agent within the game. The methods and decision processes of this agent are very similar to two of the basic agents mentioned in this thesis (HIGH and LOW). Bid Euchre uses various functions to calculate a score, determine whether or not to take a trick, and then has four functions for selecting a card: selecting high, selecting low, selecting the best lead card, and selecting the lowest trump.

2.3.2 Euchre 0.7

Another program for Euchre called Euchre 0.7 contains three different levels of AI; easy, medium, and hard. This program was written in C++ by Nathan Buckles in February 2002 [7]. This program explores a wide variety of methods and grows increasingly complex as the AI agents become more difficult. The easy agent has a very elementary thought process; win when it can, otherwise play low. This method is similar to the HIGH agent with an added case. It also does not contain nearly as many cases for processing information in order to obtain the most suitable card.

The second agent implemented in Euchre 0.7 is the medium difficulty agent. The processes of this agent are very similar to that of the easy agent; however, now the agent takes turn order into effect. This causes the agent to play a little more like the HIGH! agent rather than the HIGH agent.

The hard agent gets much more complex in its methods. On top of the methods described above, this agent looks for 'hints' from its partner for a card to play as well as

analyzes the cards which have been played and are still unknown.

2.4 Cooperative Play

Reputable agents have been made for card games; however, most card games base their heuristics on the premises of optimizing personal gains in order to win. A paper on multi-agent systems by Carmel and Markovitch takes an in-depth look into the methodology of opponent modeling in such a system [9]. However, as the name implies, the agents who are modeled in this paper are always opponents. The paper goes on to mention the problem of modeling is difficult because it relies on the strategies of the other agents.

This thesis explores methods in cooperative play (rather than oppositional) and does not create a dependency between the strategies of agents. These ideas help in creating a more independent agent who adapts to any given situation within the game of Euchre rather just being able to model an opponent or plan for itself.

A method proposed by Frank and Basin searches for an equilibrium point for the players (which they call 'exhaustive strategy minimisation') [12]. Although this proved to be a strong strategy, the 'exhaustive' quality of their model seems to be a bit too elementary for the methods mentioned in this paper. This being because the exhaustive search will not allow for quick decisions and cannot be extrapolated into games other than bridge or fields.

Chapter 3

Methodology

One of the most basic methods for good play in Euchre is to take as many tricks as possible (whenever possible) to ensure that the other team does not gather them. This strategy works really well for the most part, but it misses out on a couple of key problems that arise during the game as far as teamwork and formulation of a decision goes. This thesis discusses the amount of information available in Euchre for a given agent at any point in the game. This information reinforces the importance of how an agent thinks as well as its ultimate decision. Because this game operates in imperfect information, only certain methods are going to be effective in choosing a card to play. For example, an agent based on mathematical fact (aside from probability) is not going to be able to make a very educated choice because cards have not been played and are therefore unknown in the state space. Because of this lack of information, methods such as concentrating on personal gains, Markov Decision Processes (MDP), and logic based decisions may be used.

The rest of this chapter will talk about the various types of AI implemented in this thesis. This chapter will be starting from very basic agents who pick a card from their hand based on simple math, building through more complex methods involving turn order, and ending with complex agents who have very detailed decision processes.

3.1 Simple Rule Base - High, Low, and Random

The first and most basic of the elementary methods involves picking a card based on the hand which was dealt. Within this method, four different decision models create a solid foundation of AI agents. The first three being fairly basic and the last one being a bit more complex. The first agent contains no special strategies and just plays a random card from its hand (under the constraints that the card is considered a legal move). This agent will henceforth be referred to as RANDOM. After this agent was up and running, the models gain a more concrete set of rules to decide a card. A 'play high' and a 'play low' agent were added to the game. As one might think, the 'play high' (HIGH) agent always plays the highest valued card in its hand and the 'play low' (LOW) agent always plays the lowest valued card in its hand.

3.2 Complex Decision Processes

From here, the agents got a little more complex. The next agent is a combination of the HIGH and LOW agent which has been given the name 'high with caution' (HIGH!). The main ideas behind this agent are decided, in order, as follows:

- Look at the cards that have been played in the trick.
- If your partner has played and is winning, let it win and do not play high.
- If your partner has played and is losing, try to win and look at the rest of the cards.
- If another card is better than all of the cards in your hand, play low and dump (play a low point) a card.
- If you have a card that is higher than the two opponents' card, play high and win the trick.

Looking at the HIGH! agent, the complexity of adding the HIGH and LOW agents together and looking at the trick rather than just selecting a card from your hand creates a

much smarter and more formidable agent. Because the HIGH! agent looks at the whole trick on the table rather than just the cards in its hand, the ability of this agent to make more informed decisions increases. The HIGH! agent also implements a basic idea of looking at how well its partner is faring and tries to accommodate this. Although this cooperation is minimal, it still demonstrates the power of having an agent who will aid the team rather than an agent whose main goal is self-serving.

3.3 Markov Decisions Processes

Heading a different direction than the simplicity of just picking a card, the next agent takes a deeper look into probabilistic mathematics in order to make a decision. More specifically, this agent uses Markov Decision Processes, albeit at a very basic level, to determine what the probabilities are for its own hand being strong compared to the rest of the agents and how likely it might be that playing a card will result in taking that trick. Markov chain simulations are a way to infer an action to take based on sets of evidence and state spaces [29]. This agent (from now on MDP) keeps track of all of the cards which have been played as well as those which remain unknown. In the beginning, the MDP agent knows little of the game other than the cards (and the worth of said cards) within its own hand. The other crucial piece of information that MDP knows is the value of all of the cards in the deck, and thus the summation of these values; obtaining a total value for the deck. Using this value and the value of MDP's own hand, the agent can determine the probabilities of winning based on playing a given card from within the hand. Once the trick and cards within the hand have been analyzed and compared to the deck based on what has been played and what has yet to have been played, the agent decides to either play high or low based on the probability of winning that hand. This decision pivots on a threshold. In this instance the game has four players, the threshold has been set to 0.25; meaning that if the cards which the agent hold make up more than 25% of the points of the entire deck or what is left of it, then it would be beneficial for the agent to try and

win the trick. The formula for MDP decision is given below:

$$x_t = \sum_{i=1}^{24-n} c_i - \sum_{p=24-n+1}^{24} c_p$$

where x_t is the total number of points left over (from Figure 1.2. The number of cards played n is used to differentiate between the two sums. The points of a given card i which is still in play is given by c_i and the points of a card p which is not in play is given by c_p. If $n = 24$ the first summation is ignored and if $n = 0$ the second summation may be disregarded. The value for the number of cards in an agent's hand is as follows:

$$x_h = \sum_{i=1}^{5-m} c_i - \sum_{p=5-m+1}^{5} c_p$$

The formula compares to the previous formula, however here x_h represents the points in the agent's hand, and m represents the cards which have been played from the agent's hand. Using these two values, the following function for deciding a move is constructed based on the threshold value τ:

$$f(n) = \begin{cases} \text{play high} & \text{if } (x_h/x_t) \geq \tau \\ \text{play low} & \text{if } (x_h/x_t) < \tau \end{cases}$$

For these experiments, a τ of 25% is used. Every trick played will update the values for the cards which remain. Because of this, the MDP agent works more dynamically than the previous agents in obtaining a decision; however, the MDP agent also acts very selfishly and only makes choices based on its own probabilities and gains.

Figure 3.1 gives the general idea behind an MDP. The prior probability is what environmental evidence and probability the agent has to win at the start of the hand. The prior probability for Euchre is going to be the evaluated heuristic of the player's hand compared to the overall heuristic of the cards in the state space (all the cards in the deck). This probability is calculated from the above formula. As more and more cards are

played, the environmental evidence becomes more and more sound and the probability of what cards remain decreases and becomes a certainty; therefore, the agent's decisions at the end can be more precise when determining what card to play.

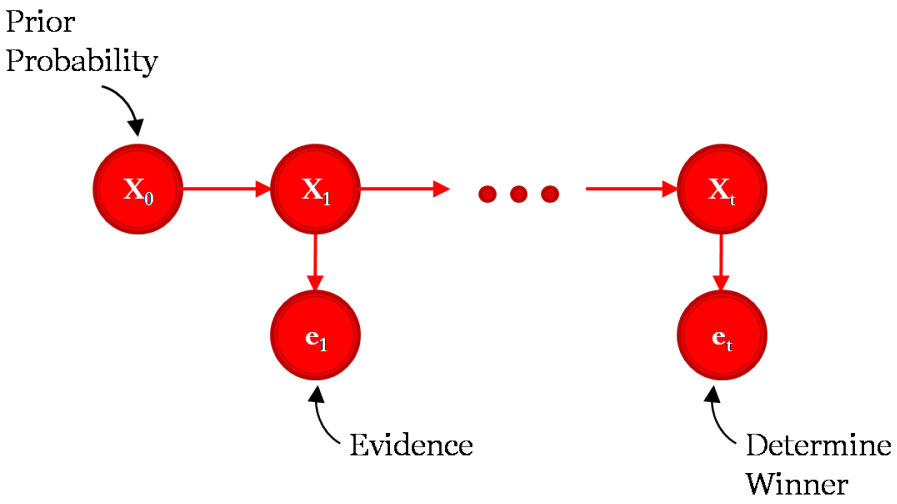

Figure 3.1: The decision process behind MDPs.

Based on this information, an agent is able to make a decision on what the best card would be to play. Specifically, the agent is able to use the comparison of his/her hand with the remaining cards to determine whether to play conservatively (a low-valued card) or aggressively(a high-valued card). In this implementation, if an agent surpasses a threshold percentage (25%), then it would be best to play a high card because their chance of winning exceeds that of the four agents at the given point in time. On the other side of the coin, if the agent's probability falls withing the sub-25% range, then it would be more beneficial to play low in order to conserve the higher valued cards until the probability of winning is greater than the threshold.

As a prime example, Figure 3.2 illustrates the first player deciding which card to play. This agent's hand has summed to more points compared to the deck's value: about 60% (which is expected since this agent has the two strongest cards for this hand: 'Jack of Clubs' and 'Jack of Spades'). Because of this reasoning, it makes sense for this player to play high because even after the 'Jack of Clubs' has been played, this player still has a 27% to win the next trick based on the values of his hand and the deck (which is still

over the threshold). The 'Self Value' and the 'Deck Value' found in Figure 3.2 are from x_h and x_t, respectively.

```
Player 0: [0]   [1]   [2]   [3]   [4]
           J♣    K♥    A♥    J♠    A♦
Pick card:Markov (determinative)
 J♣  P(0) Value 1500  Played: N
 K♥  P(0) Value 5  Played: N
 A♥  P(0) Value 6  Played: N
 J♠  P(0) Value 1000  Played: N
 A♦  P(0) Value 6  Played: N
Self Value = 2517.0          (Player 0)
Deck Value = 4249.0

Trick:
 J♣  N/A  N/A  N/A
Player 0 has played  J♣
Lead suit: ♣
Trump suit :♣
```

Figure 3.2: The first turn example using Markov decisions.

However, looking at Figure 3.3 illustrates when a player might want to play lower to be more conservative. Because the 'Jack of Clubs' is no longer a threat, the deck's value has been decreased by the heuristic of the missing card. Therefore, the player's probability based on his/her own value only gives a 22% chance of winning a trick. This settles below the threshold and warns the agent to play low and wait until he/she has a better chance at taking the trick. A flaw with the notion of method exists; which will be discussed in the analysis chapter.

Based on the implementation of this MDP, an agent exhibiting these decision options is expected to play rather similarly to the HIGH! (high with caution) agent because of the ability to differentiate between when to play a high card and when to play a low card. All of these implementations will be discussed further in the next few sections.

```
Player 1: [0]    [1]    [2]    [3]    [4]
           A♠     J♥     Q♦    10♣     Q♣
Pick card:Markov (determinative)
 A♠   P(1) Value 6   Played: N
 J♥   P(1) Value 3   Played: N
 Q♦   P(1) Value 4   Played: N
10♣   P(1) Value 200  Played: N
 Q♣   P(1) Value 400  Played: N
Self Value = 613.0          (Player 1)
Deck Value = 2749.0

Trick:
 J♣   10♣   N/A   N/A
Player 1 has played 10♣
Lead suit: ♣
Trump suit :♣
```

Figure 3.3: The second turn example using Markov decisions.

3.4 User Friendly

Although the MDP agent seemed to make very beneficial decisions, it works in a manner which creates problems for allies. These problems include such items as misleading information during a trick and disregarding the partner's probabilistic hand. In order to alleviate these issues, a new agent is introduced whose primary goal is to cooperate with its partner so the team can maximize their gains. The 'User Friendly' (UF) agent uses a data structure called the `chanceChart` to maintain the percent probability of a given player having a card at any given time.

Player 4	9	10	J	Q	K	A
♥	0.0	0.0	0.25	0.0	0.0	0.25
♠	1.0	0.333	0.0	0.0	0.25	0.0
♦	0.333	0.333	1.0	0.333	0.333	0.0
♣	0.0	0.666	0.666	0.0	0.0	0.0

Figure 3.4: A `chanceChart` after a few rounds of play.

Figure 3.4 represents the three-dimensional array of probabilities for all players and all cards in the deck. (please note that this representation is just for explanatory purposes and 'might' not be a possible chart) The first dimension of size four holds the information for all four players, including itself. The second dimension hold the suit and the third dimension determines the rank of the card. For example, if we access the array at position (2,4,5) we might be referring to the probability that the second player has the King of Clubs (assuming that 2 = second player, 4 = Clubs, and 5 = King). As a side note, this chance chart is specific to each player. This means that each player given the UF method of thinking has its own chart containing everyone's information. In general, the chart will give a 1.0 probability to a card that is guaranteed to be held by a player, a 0.0 to a card that is guaranteed **not** to be held by a player, a -1.0 for a card which a player has already used (this helps in calculating probabilities for having a suit left or not), and a number in the range of $(0.0, 1.0)$ representing the probability that a player is holding a card at a given point in the game. This chart will monitor cards played and assign those a 0.0 probability. It will also monitor and react when an agent plays something other than the lead suit and will update the probabilities of that agent having any of that suit to 0.0 (unless it is the Jack of trump or the Jack of the side suit). With the employment of this `chanceChart`, the UF agent also has a set of rules on which it runs to determine the best course of action. The UF implementation runs on these four basic precepts:

1. Monitor all card flow and maintain probabilities (card counting).
2. Choose a course of action based the position of play at the table.
3. Assist the partner at any cost.
4. Determine if opponents can be bested.

These ground rules create a 'smart' methodology for thinking. Not only does the agent consider all possibilities for cards, but it also places its partner on a pedestal and plays in any way possible to assist it. The UF agent may be analyzed at one of four playing points as outlined below:

1. First to play
    (a) If your partner is out of a suit, play low in that suit, hoping that the partner can trump.
    (b) Otherwise, see if your partner has a high chance of winning in a suit (based on the probabilities).
    (c) If he has a good chance, play that suit.
    (d) Otherwise, play a high card in the hopes that you can win the trick.
2. Second to play
    (a) Compare the strength of the previous player's card with the probabilities of the other agents.
    (b) If the partner is strong in the lead suit, play low and let it win.
    (c) If the partner does not have that suit:
        i. If the suit is the trump, play high to win.
        ii. Otherwise, play low and allow your partner the chance to trump.
    (d) If you can beat the first player, play high and attempt to win.
    (e) Otherwise, play low
3. Third to play
    (a) Compare the strength of the previous players' cards with the probabilities of the other agents.
    (b) Since one is your partner, check to see if your partner is winning.
    (c) If your partner is winning, check the final player to see if he might have a card stronger than your partner.
        i. If he does, play high and try to beat the last player.
        ii. Otherwise, play low and let your partner win.

(d) If your partner is losing, check probabilities and play high.

(e) Otherwise, play low and dump a card.

4. Last (fourth) to play

(a) Compare the strength of the previous player's card with the probabilities of the other agents.

(b) If your partner is winning, let it win.

(c) Otherwise, check and see if you can win:

i. If you can win, play high.

ii. Otherwise, play low.

This method takes a huge step in creating an agent who performs more intelligently. This advanced strategy hones in on the stronger aspects of a trick-taking game, but also respects the bond created by having a partner. However, the problem with having an agent who acts too politely results in more aggressive players tending to trample over the cautious agent (this will be discussed further in the results section).

3.5 Hybrid User Friendly

The problem of having an overly cautious agent becomes evident when playing an aggressive agent. The aggressive agent tends to dominate play, especially if this agent has a particularly strong hand. This issue may be fixed by creating an agent who is both aggressive and cautious. The newly created 'User Friendly 1.5' (UF.5) agent combines the aggressive behavior of the HIGH! agent and the cautious play of the UF agent to create a more well-rounded agent who adapts given any situation. The UF.5 agent also retains the `chanceChart` from the UF agent. When it comes time to play, the UF.5 agent utilizes the threshold idea from the MDP agent, calculating its probability of doing well. If UF.5 can perform well, then it will choose to be more aggressive and if UF.5 does not have a good

hand, then it will play according to its cautious rules. Ideally this will create an even more dynamic agent who can adapt to any situation based on the available information presented in the state space.

Chapter 4

Implementation and Data Collection

For each figure in this section, a comparison will be shown between two methods, and the numbers used for comparison are the average number of points earned by the team in the game and the number of games won in a 31 game competition. Unless otherwise noted, the teams contain parters of the same strategy. The teams are divided into their respective columns in each figure and the average points and the number of games won is taken over the 31 different trial runs. **Note**: A game of Euchre can be won with 11 points if the winning team gains 2 points on the last hand with a score of 9; at times this increases the average points earned, but overall the scores normalize.

4.1 Simple Rule Base

The simple rule base data compares the three possible combinations of an AI pitted against a different AI. Figures 4.1, 4.2, and 4.3 show the comparisons for each of these. As expected, the LOW player did not fare too well against the HIGH and RANDOM players. For the competitions against LOW, the HIGH agent was able to a bit better than RANDOM as far as the number of games won, but the difference in average points from the HIGH/LOW game (9.742 to 5.516) was more than the points from the RANDOM/LOW game (8.903 to 7.129). As these results show, the scores against the HIGH player fared

much better than the RANDOM agent. However, the interesting result came from the game with RANDOM agent against the HIGH agent (as seen in Figure 4.1). Because HIGH has more knowledge and is a circumstantially more aggressive in taking the tricks in order to win, this AI agent was expected to perform better than the RANDOM agent. The comparison of the number of games won being off by only one with RANDOM being greater than HIGH was rather interesting. The reasoning behind this is analyzed in section 5.1.

	Random	Highest
Average Points	8.226	7.452
Games Won	16	15

Figure 4.1: A comparison of RANDOM vs. HIGH.

	Highest	Lowest
Average Points	9.742	5.516
Games Won	25	6

Figure 4.2: A comparison of HIGH vs. LOW.

	Lowest	Random
Average Points	7.129	8.903
Games Won	9	22

Figure 4.3: A comparison of LOW vs. RANDOM.

4.2 Adding Complex and Markov Decision Processes

After adding the two complex methods of decision making (the HIGH! and MARKOV processes), there became quite a few more comparisons to analyze. The first set of data collected (Figures 4.4, 4.5, and 4.6) was a comparison of HIGH! against the rest of the other agents. The result here were very impressive. The HIGH! agent was able to defeat the other types of AI in more than 80% of the matches for all types. One of the most

impressive games was HIGH! vs. RANDOM. Although the RANDOM agent was able to do really well against the HIGH and LOW agents, the combined knowledge of these two in HIGH! was able to beat RANDOM 27-4 with an average number of points per game 3.871 points in favor of the high with caution agent.

	High!	High
Average Points	9.323	5.355
Games Won	26	5

Figure 4.4: A comparison of HIGH! vs. HIGH.

	High!	Random
Average Points	10.000	6.129
Games Won	27	4

Figure 4.5: A comparison of HIGH! vs. RANDOM.

	High!	Low
Average Points	9.806	4.484
Games Won	29	2

Figure 4.6: A comparison of HIGH! vs. LOW.

The results found in Figures 4.7, 4.8, and 4.9 match up the MARKOV agent with the basic three other AI implementations. These were not nearly as strong as the results from the HIGH! agent. In fact, both RANDOM and LOW agents were able to beat the MARKOV agent; counter-intuitively, the MARKOV agent was able to beat the HIGH agent. The games wherein the MARKOV agent was against the RANDOM and LOW agents were extremely close, both ending in a 15-16 score for the number of games won. However, one promising piece of information about the MARKOV method is the average number of points. The numbers for this agent compared to each of the three opponents was always in the range of 8 to 9, which is very good for losing a game of Euchre and shows just how close the games really were.

	Markov	High
Average Points	8.903	6.548
Games Won	22	9

Figure 4.7: A comparison of MARKOV vs. HIGH.

	Markov	Random
Average Points	8.065	8.065
Games Won	15	16

Figure 4.8: A comparison of MARKOV vs. RANDOM.

	Markov	Low
Average Points	8.032	8.484
Games Won	15	16

Figure 4.9: A comparison of MARKOV vs. LOW.

The final comparison for these slightly more intelligent agents is between the HIGH! and MARKOV agents (Figure 4.10). With an average number of points of 9.226 and a number of games won of 23 out of 31, the HIGH! agent is clearly much stronger than the MARKOV agent, being able to keep the MDP agent at a low average number of points of just 5.968 per game.

	High!	Markov
Average Points	9.226	5.968
Games Won	23	8

Figure 4.10: A comparison of HIGH! vs. MARKOV.

4.3 User Friendly Agent

After implementing the UF agent, a comparison was made between a team of two UF agents and a team of two of some other type of agent. The results here were very promising. First, the UF agent competed against the 3 basic agents (RANDOM, HIGH, LOW) to see how it would fair against them. Figures 4.11, 4.12, and 4.13 demonstrate how

the UF agent was able to succeed against each of these agents with a win percentage of 67.74%, 74.19%, and 70.97%, respectively.

	User Friendly	Random
Average Points	8.806	7.516
Games Won	21	10

Figure 4.11: A comparison of UF vs. RANDOM.

	User Friendly	Highest
Average Points	9.452	7.032
Games Won	23	8

Figure 4.12: A comparison of UF vs. HIGH.

	User Friendly	Lowest
Average Points	9.355	5.968
Games Won	22	9

Figure 4.13: A comparison of UF vs. LOW.

The outcome here was to be expected. The most basic AI agents are not going to fare too well against a more complex and thoughtful agent. The three basic agents were extremely comparable against the UF agent in terms of how close the other agents' scores were while the UF agent was able to do rather well. The next two comparisons with the UF agent are against the HIGH! agent and the MDP agent. The MDP agent turned out to be similar to that of the basic agents. The HIGH! agent completely dominated the UF agent wherein the UF agent had a win percentage of 35.48%.

	User Friendly	HIGH!
Average Points	7.419	8.774
Games Won	11	20

Figure 4.14: A comparison of UF vs. HIGH!.

The low win percentage from the MDP process in Figure 4.15 required some re-working to the agents in order to create an agent that will fair well against all agents rather than just most of them. The thought process here was to figure out why the HIGH! agent

	User Friendly	MDP
Average Points	8.968	6.839
Games Won	21	10

Figure 4.15: A comparison of UF vs. MDP.

was doing so well and to incorporate that into how the UF agent was playing as these are the two strongest of the available agents. Since the major difference between these two agents was that one was extremely aggressive and the other was very cautious, the idea was to create an agent that held both of these traits, but was able to exploit when to use one over the other. This was the basis for the creation of the UF.5 agent. However, before looking at the numbers for the UF.5 agent, this paper will compare some of the 'Mixed-Agent' teams (since the UF.5 agent was born of these comparisons).

4.4 Mixed-Agent Teams

In order to truly look at the power of the agents working as a team, the final type of teams are those wherein the agents on a single team do not necessarily need to be the same type. As a quick first comparison, Figure 4.16 illustrates a team with agents HIGH and LOW with the opposing team as two RANDOM agents, since random was able to prevail out of the three basic types of AI. Surprisingly, the mixed team of HIGH/LOW was able to beat the RANDOM agents 18-13 although their average points per game were not too far off. The reasoning behind this most likely because the HIGH agent claims the tricks in the beginning of the hand by dumping the higher point cards first; whereas the LOW agent has a more conservative approach and tries to hold on to the more powerful cards until the end. This overall strategy was able to perform substantially better than the team of HIGH agents or the team of LOW agents against a team of RANDOM agents. The idea behind these two agents working in a sort of symbiosis helped in developing the strategy of the UF.5 agent based on the cooperative efforts of the HIGH and the LOW agents.

Because of the surprising results from Figure 4.16, the next comparison was to take

	Highest	Random
	Lowest	Random
Average Points	8.581	8.000
Games Won	18	13

Figure 4.16: A comparison of HIGH/LOW vs. RANDOM/RANDOM.

the teamwork of the complex MARKOV and HIGH! players and to match them against the HIGH/LOW combination. As expected, the more complex team was able to beat the basic AI team in about 65% of the matches and the average points per game was much more impressive at 9.000 for the more developed AI team.

	Markov	Highest
	High!	Lowest
Average Points	9.000	6.903
Games Won	20	11

Figure 4.17: A comparison of MARKOV/HIGH! vs. HIGH/LOW.

To complete the three types of tests, the two teams which have not combatted yet (MARKOV/HIGH! with the RANDOMs) are the last two teams to be examined. Because of the evidence from the previous two games, it is not too surprising that Figure 4.18 shows how much stronger the complex team operated against the basic team of random; again their average points are 9.000 per game.

	Markov	Random
	High!	Random
Average Points	9.000	6.452
Games Won	23	8

Figure 4.18: A comparison of MARKOV/HIGH! vs. RANDOM/RANDOM.

The results from this data might seem like a miscalculation in most cases. For example, the HIGH! was not foreseen to do nearly as well as expected and the MARKOV implementation was predicted to do a little better than anticipated. After looking at these basic mixed teams, more information was be extrapolated to see how a UF agent would

fair with the surprisingly strong HIGH! agent. The next two comparisons are exactly that; a combination of UF and HIGH! with other agents to see how strong they are together. Tests were run against all agents, but the the key teams here are against a team of two HIGH! agents and against a team of two UF agents. Figures 4.19 and 4.20 illustrate these teams against one another.

	User Friendly	HIGH!
	HIGH!	HIGH!
Average Points	6.613	8.516
Games Won	10	21

Figure 4.19: A comparison of UF/HIGH! vs. HIGH!/HIGH!.

	User Friendly	UF
	HIGH!	UF
Average Points	8.452	7.710
Games Won	19	12

Figure 4.20: A comparison of UF/HIGH! vs. UF/UF.

The team combating the HIGH! agents seemed to do about the same as a UF vs HIGH! game, but the team against the UF agents showed the power of being aggressive from time to time. Though, when is it important to be aggressive? When is it good to be cooperative? This idea brought about the idea for an agent that can be either. Thus, the UF.5 agent was created. The final piece in this section demonstrates the power of the combination agent.

4.5 Hybrid User Friendly

The hybrid agent created for testing purposes is, as previously mentioned, named UF.5 and combines the aggression of the HIGH! agent with the cooperation of the UF agent. The outcome of which was rather astounding. The following figures (4.21 - 4.26) illustrate two UF.5 agents playing against pairs of all of the other agents.

	Hybrid UF	Random
Average Points	8.871	6.065
Games Won	21	10

Figure 4.21: A comparison of UF.5/UF.5 vs. RANDOM/RANDOM.

	Hybrid UF	Highest
Average Points	9.613	5.935
Games Won	26	5

Figure 4.22: A comparison of UF.5/UF.5 vs. HIGH/HIGH.

	Hybrid UF	Lowest
Average Points	10.032	4.258
Games Won	29	2

Figure 4.23: A comparison of UF.5/UF.5 vs. LOW/LOW.

	Hybrid UF	HIGH!
Average Points	8.419	7.774
Games Won	18	13

Figure 4.24: A comparison of UF.5/UF.5 vs. HIGH!/HIGH!.

	Hybrid UF	MDP
Average Points	9.516	5.935
Games Won	25	6

Figure 4.25: A comparison of UF.5/UF.5 vs. MDP/MDP.

	Hybrid UF	UF
Average Points	8.419	7.742
Games Won	17	14

Figure 4.26: A comparison of UF.5/UF.5 vs. UF/UF.

The first three games are nothing too special since they are against the three basic methods of thinking for this game. The UF.5 is finally able to best the HIGH! agent by approximately 16 percentage points (58% wins by the UF.5 agent compared to 42% wins by the HIGH! agent) and the UF.5 agent is able to perform better against the regular UF agent as well. These numbers surpass the average UF's performance (more discussion in Section 5.1). However, these statistics in cooperative and aggressive AI demonstrate why

data should be collected and how a certain AI should never be under- or overestimated. This information provides quite a bit of incentive to do future work which may be found in Section 6.1.

4.6 Existing Program Comparison

The preexisting Euchre programs were very helpful in evaluating the ability of the various agents in this thesis. First, the Bid Euchre program was tested. The UF agent was able to very well against Bid Euchre, even though the point spread was rather close (as seen in Figure 4.27). Next, Bid Euchre played against the UF.5 implementation for 31 games; the results may be found in Figure 4.28.

	UF	Bid Euchre
Average Points	7.986	7.621
Games Won	20	11

Figure 4.27: A comparison of UF vs. Bid Euchre.

	Hybrid UF	Bid Euchre
Average Points	8.315	7.463
Games Won	20	11

Figure 4.28: A comparison of UF.5 vs. Bid Euchre.

The UF.5 implementation was able to beat the Bid Euchre agent in 64.52% of the games. Taking an analytical approach as to why, it seems that the methods for calculating the values of cards and the narrow search were probably the reason. The cards value in Bid Euchre range from 100-600 (from 9 to Ace) with a slight varied amount for the Jack of Trump (800 points) and the Jack of the off suit (700 points). After this, a flat 1000 points was added for any card from the trump suit. This means that all of the cards have a rather high value. This causes little to no emphasis on the cards that are not in the trump or even in the lead. This subtle difference causes large differences in the worth of cards. The other item was the small space of available moves and information utilized by

the agent. Looking at the value of a card is a good strategy, but it only goes so far when trying to determine the moves of the opponents and the moves of the partner.

After testing the Bid Euchre program, the Euchre 0.7 program was analyzed. Figure 4.29 and Figure 4.30 illustrate the results from a 31 game competition.

	UF	Euchre 0.7 E
Average Points	8.973	7.241
Games Won	22	9

Figure 4.29: A comparison of UF vs. Euchre 0.7 on easy.

	Hybrid UF	Euchre 0.7 E
Average Points	9.157	6.583
Games Won	23	8

Figure 4.30: A comparison of UF.5 vs. Euchre 0.7 on easy.

The downfall of the easy agent was how limited its decisions were. The UF agent was able to beat Bid Euchre 70.96% of the time. The simple decision process of this agent continued to show with the 74.19% win percentage for the UF.5 agent. Following the games against the easy agent, the medium agent was also tested in a bout of 31 games. The results in Figure 4.31 and Figure 4.32 demonstrate this by the fact that the win percentage for the UF and UF.5 agent had dropped to 54.84% and 61.29%, respectively. Although this win percentage is still good (and quite strong for the UF.5 agent), it is clear that adding complex decision processes to an agent benefit the outcome greatly.

	UF	Euchre 0.7 M
Average Points	8.127	7.635
Games Won	17	14

Figure 4.31: A comparison of UF vs. Euchre 0.7 on medium.

As the agents from Euchre 0.7 get progressively better, the percentages reflect the change. The final test from the Euchre 0.7 program was against the hard agent. This agent greatly surpasses the skill of the HIGH! agent and is extremely comparable to the UF or UF.5 agent. Figure 4.33 illustrates that the UF agent's relaxed attitude in playing

	Hybrid UF	Euchre 0.7 M
Average Points	8.422	7.168
Games Won	19	12

Figure 4.32: A comparison of UF.5 vs. Euchre 0.7 on medium.

was beaten by the hard agent in Euchre 0.7 most of the time. These newly implemented actions for the Euchre 0.7 hard agent cause the UF.5 agent to only win about half of the time. Figure 4.34 illustrates their similarities based on the win percentage of 51.61% for the UF.5 agent.

	UF	Euchre 0.7 H
Average Points	6.427	9.328
Games Won	13	18

Figure 4.33: A comparison of UF vs. Euchre 0.7 on hard.

	Hybrid UF	Euchre 0.7 H
Average Points	8.254	8.675
Games Won	16	15

Figure 4.34: A comparison of UF.5 vs. Euchre 0.7 on hard.

This final competition of 31 games proved to be quite a daunting task for the UF.5 agent. Even though a lot of their methodology was the same, there were still some slight differences in the details and what each agent observed and used to its advantage.

Overall, the UF.5 agent was able to beat all of the other agents with which it was competing for this thesis. The methods of these other open-source projects was very similar to some of the agents described herein; however, the minute details of what one method valued over the other were different. These details demonstrate the strength of one aspect of the game when compared to another. Based on these results, there is not a clear aspect which is the most important. The results do show that the strength of the agent can be determined by the utilization of the details found within the game.

Chapter 5

Analysis

5.1 Overview

From a generic standpoint, Euchre is a game about trick-taking. It differs from some other trick-taking games in that the number of tricks bid does not matter; the main goal being that you take more than two of the tricks in order to receive points. Based on this simple idea, one would believe that a strong method to playing this game would be to just play the highest card in order to secure as many tricks as possible. However, based on the data gathered here, it would seem that just playing the highest card (HIGH) is not a viable option in this game. The reason behind this being the paired play. In order to ensure that a team performs well, it makes no sense to just play the higest valued card because it might be detrimental to the team by forcing the partner to play its highest card as well. The best way to split the difference in this case will be to observe the partner's play and to try to adhere to the cards that one another is playing and has played. The simple instance of the HIGH! agent benefits from looking at its partner and making more intelligent decisions. However, there are still some downfalls to this method. Building upon this, the UF agent and the UF.5 agent fix these problems and prevail against all of the different AI agents implemented in this paper. The UF.5 agent bests the other agents by at least 8 percentage points and claims victories in the range of 58%-94% of

games against certain agents. These percentage points are calculated by subtracting the UF.5's win percentage from the other agent's win percentage. The rest of this section will analyze why this agent was able to inch itself above the rest. This section will also derive some useful information about the other agents.

5.2 Strength of Methods

The games played proved to be very surprising; however, their results mean a bit more than just their numbers by themselves. By comparing all of these types of AI against each other, some very positive information is provided as far as this implementation goes. First, although the LOW agent is not that strong of an agent by itself, the mixed strategy results demonstrate how strong of an adversary it can be as a conservative agent when paired with a very aggressive agent. The same results proved to be true for the MARKOV agent; therefore, it appears that the MARKOV agent acts more conservatively as the LOW agent whereas the HIGH! agent acts a bit more aggressive like the HIGH agent.

Although the basic AIs were seemingly intelligent by themselves, the forte of the complex AIs proved that when extra time is taken to work out the more minute details behind decision making, or through using the probabilistic models such as the MDP, the agents perform better in accumulating more points per game (on average) than just the basic models for AI.

Finally, the comparison of mixed team strategies really proves to be an effective tool for determining how agents should act. As previously mentioned, the LOW agent appeared to be very feeble by itself; however, paired with the HIGH agent, these two were able to complement one another and beat random more often than just the LOW player by itself. Although the MARKOV model was relatively close to tying the RANDOM agent in a one-on-one face off, when combined with the HIGH! agent, the odds were evened out a little bit and the MDP agent balanced out the team and help the cautious agent in

order to beat the randomly modeled player.

Building the UF agent meant extracting all of these strengths and combining them into one agent that overcomes the weaknesses of the individual agents. The UF agent was able to play as a fairly conservative player whose main purpose was to assist its partner in playing well, putting the partner's needs above its own. UF was beating most of the agents, but the HIGH! agent proved to be too aggressive and was walking all over the polite, delicate UF agent(s). In order to get UF back into the game, the extreme aggression of the HIGH! agent was harnessed and put into the implementation of UF. This is how UF.5 came to be. In the end, UF.5 was able to adapt to a given situation based on the cards in its own hand, the progressive play of the other agents, and the methods of its own partner. The cards in its hand told the agent exactly how much it should invest in the strength of the hand, causing the agent to decide on a more aggressive or a more passive approach. The progression of play let the UF.5 agent understand how the other agents were getting rid of cards, how many of a suit still existed, and when others could play off of the lead suit in order to trump and claim points. This was an extremely useful tactic because the UF.5 agent was able to determine when to be aggressive and take tricks as well as when to be timid and sink into the background. The UF.5 agent observed its partner and was able to assist it by either playing higher or lower to suit the needs of the team.

Two good ways to look at how well an agent (or a team of two agents) performs in the game of Euchre is to look at the number of tricks taken and the number of overall games won. Figure 5.1 illustrates the percentages of tricks taken over the 31 game test. Each set of 31 games contains about 1,500 tricks. Figure 5.1 illustrates that the percentages of tricks taken by the UF.5 agent were always above 50% of the total number of tricks. The improvement from UF (the bar on the left) to UF.5 (the bar on the right) is by about 5 percentage points for each different style of AI.

Figure 5.2 shows the percentage of wins for the UF agent and the UF.5 agent against the other agents. In this graph, the UF agent is extremely strong against most

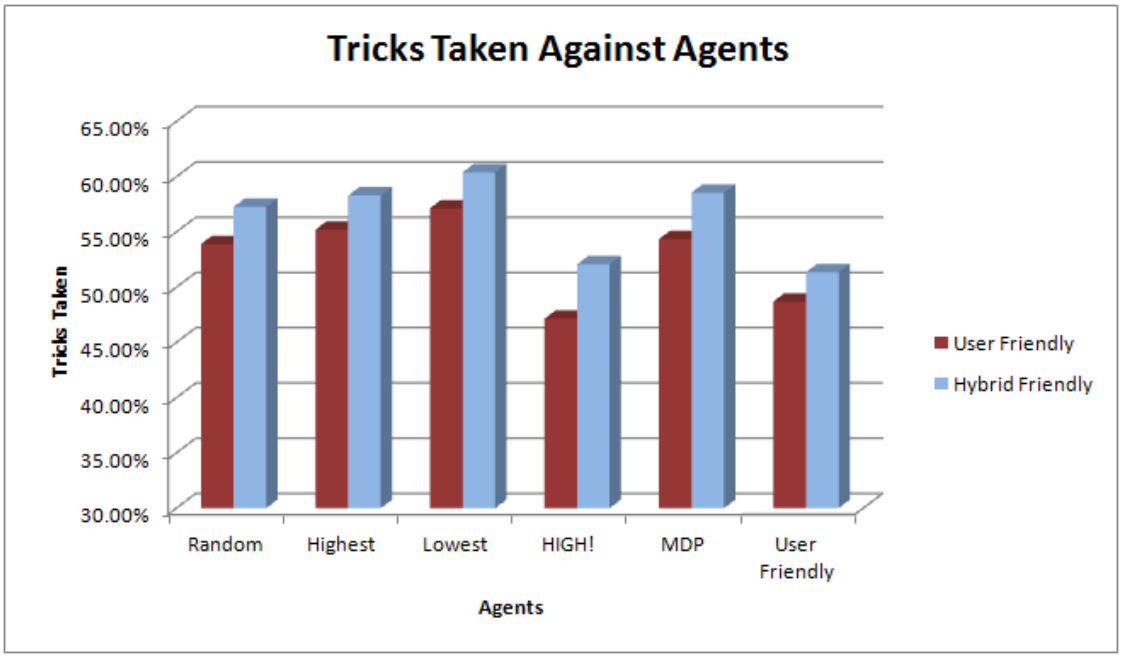

Figure 5.1: A comparison of the tricks taken by UF and the UF.5 agent against other agents.

of the agents (60-70%); however, the aggression of the HIGH! player stands out causing a mere 35.48% win rate for the UF agent. The change from UF to UF.5 takes this win percent up to 58.06% all while increasing the wins for against the other agents. Some games were as high as 93.55% (against the LOW agent). The final column is the UF agent against itself (the left column) and then the UF agent against the UF.5 agent (the right column). The comparison with the UF.5 agent is expected to be around 50% since it is playing a version of itself.

Conclusively, the improvement on the UF agent to the UF.5 agent was substantial. Throughout the testing of these agents, it became rather clear that having a simple or single-tracked mind for a strategy just was not going to work. As the agents decision processes changed, their play styles differed and this was the factor that was able to create an agent that harnessed all of the strengths of the agents. Combining these into one agent yielded the dynamic UF.5 agent, which held an overall win percentage of 73.11% across all agents.

Another set of data for analysis in the game Euchre is the percentage of points won

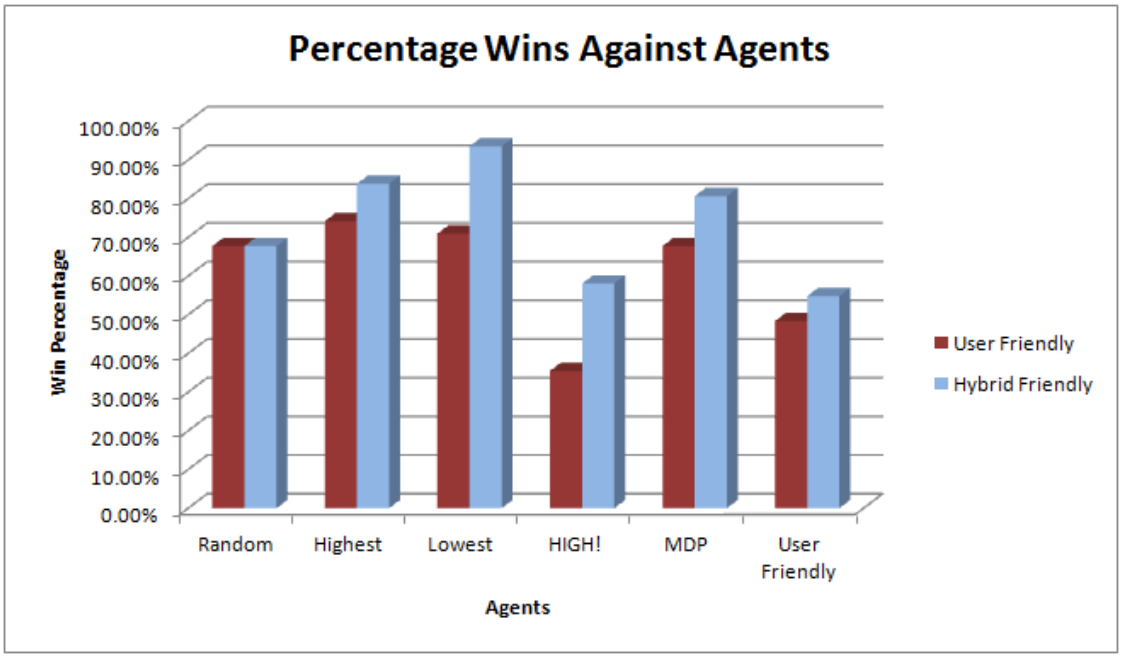

Figure 5.2: A comparison of the game wins UF and the UF.5 agent against other agents.

in order to win the entire game. Having a close score does not always mean that an agent performed better or worse in the game as a whole although there is a slight correlation. This is because the spread of the points shows how much one team dominated the game play and the amount of hands won. Figure 5.3 illustrates how a team might have only taken about 56% of the points yet still won about 71% of the games looking at the UF agent against the HIGH agent in Figure 5.3 and Figure 5.2.

5.3 Pitfalls

Although most of the results from this experiment were positive, the data points to some setbacks which arose in the midst of the positive results. One of the most interesting concepts is the strength of the RANDOM player. This is attributed to the small state space of game. When an agent only has a hand of 5 cards, a subset of which are legally playable, the RANDOM agent has a rather high probability of playing the same cards as the LOW, HIGH, HIGH!, or MARKOV agents; and therefore can play against these agents just as well.

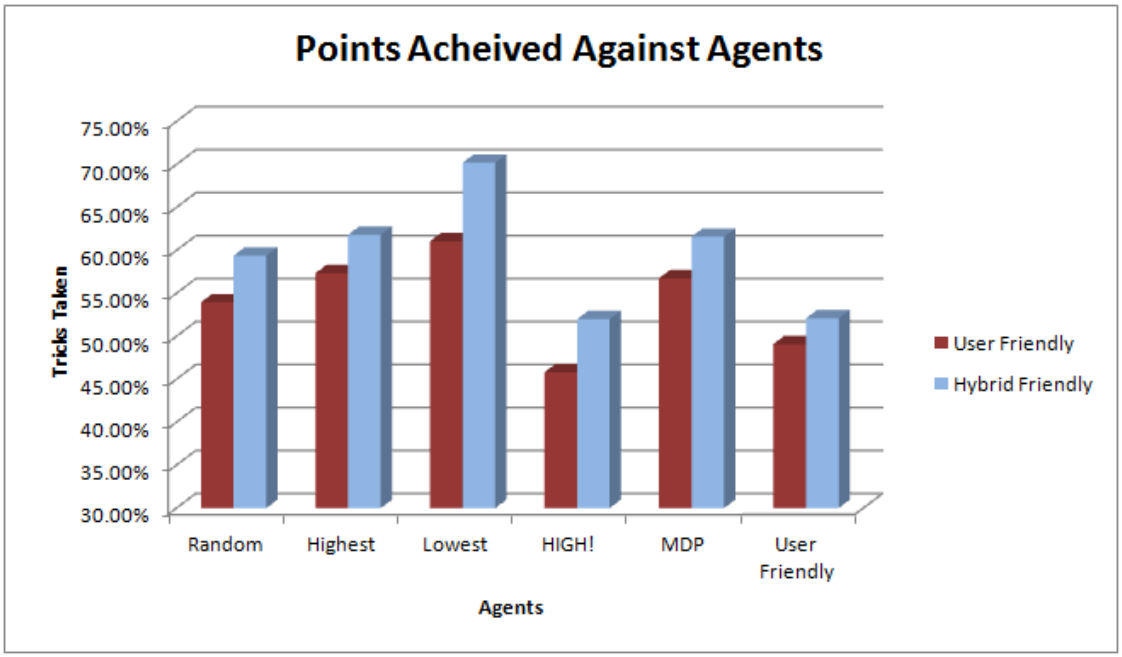

Figure 5.3: A comparison of the number of points taken by the UF and the UF.5 agent against other agents.

An interesting point of information for these AI decision trees was how quickly they were searched. Originally, pruning techniques [1, 5, 16, 32, 34] were researched in order to try and speed up the process of choosing a card to play, but in the end the state space of Euchre was easy to analyze and extremely efficient in the way of time complexity. Therefore the researched methods of searching trees in AI for imperfect information, along with their pruning methods, did not provide any speed up.

Another problem with the agents' decision making process might be in the heuristic values. The problem here is that they might be too varied from trump, to lead, to off suit. This means that because the values used to determine the card played differs in orders of magnitude, maybe it is not the best idea to simply look for the highest, lowest, or a sum of the value of the cards in order to choose how to play.

One issue is for the implementation of the MARKOV values. In this case, after a card has been played the deck value would be better off with that card's value staying in the trick's evaluation until the trick is completely over. For example, in Figure 3.3 after the Jack of Clubs has been played, the subtraction of the 1,500 points misconstrues the

estimation for the next player and he/she would be better off leaving this value in until the trick is over. At this point, the subtraction of all of the cards played leads to a more precise calculation. The method used for the results is the former.

The major pitfall for the UF agent was the lack of aggression when playing. It seems that this agent behaves too politely and wants to help its partner so much, that it forgets about winning the game sometimes against the more aggressive agents. When putting two of these UF agents on the same team, the polite nature of the two agents conflicts and they do not know how to react to one another. This problem led to the second implementation (the UF.5 agent).

Chapter 6

Discussion

6.1 Future Work

Quite a bit of work has been done trying to create a more intelligent system for Euchre using complex methods; however, there is still a lot to be done for this type of work. One simple task would be to engage these AI types against a human player to see how well AI compares to natural intelligence [2].

In order to create a more robust game of Euchre, AI which would determine how or when to bid for a trump or order up a trump would greatly improve the overall AI. The methods behind this would not be too terribly difficult and involve assuming a suit as a trump to see which would be the best for an agent to try and play. Although reinforcement learning has been shown to be inconclusive from time to time in the field of imperfect information card games [22], it would be worth the investigation to see if, based on these improved results, there would be a way to introduce a variety of reinforcement learning models. A possible method for investigation might be derived from Matsuno's work [23].

Another extension of this work would be to delve deeper into the AI agents' monitoring of the other players. A paper by D. Billings looks at the challenge of poker not only from a probabilistic stand point, but also from a deceptive stand point. He describes these considerations and the architecture behind a poker program named 'Poki' [3]. An

ability to infer how an agent might play in order to change a best response based on the current play style of an agent would be a really interesting extension of this experiment. One example of this would be to try and 'guess' when an opponent or teammate is out of a certain suit and to react to this by playing another suit to hinder or playing cards to assist the other agents.

An extension of the MDP into a more complex system of filtering, smoothing, and prediction would be interesting grounds to see how well probability plays into a trick-taking game or whether it really is just a set of rules or conditions which make an agent's decision the 'best response' to another agent's actions. Based on the probability calculated by the deck and hand values, there could be more than one option for the player to use. For example, when the value is above 50% always play high, when the percent is below 25% always play low, but if the percent is within these values, calculate a more complex method to see what might be done.

A final method for improving the values of the cards would be to implement a style of machine learning. Through genetic algorithms or neural networks, the value of each card could be determined and this would provide a better understanding of what cards should be considered better than another card.

6.2 Conclusion

Starting from scratch in order to create a complete euchre game with some basic and complex AI proved to be a daunting task, but the end result demonstrated great results and a direction for the future. Euchre as a test bed illustrated great characteristics for inspection such as the imperfect information with perfect recall and the probability behind the card dealing with the hidden cards in the kitty. The game was able to be programmed and visualized so that the data could be collected to provide conclusive results. A general heuristic for the values of each cards was created in order to determine how an agent should played based on the type of AI being implemented. The heuristic values calculated optimal

values in a complex rule base by the HIGH, LOW, and HIGH! players in order to logically determine which card would be the best response. These decisions helped illustrate how agents can elicit varying strategies (aggressive, conservative, neutral). Another, more analytical, approach was to use a probabilistic Markov decision process to make a decision for the agents. This encompassed taking in the knowledge of the entire state space of cards and comparing this to an agent's own hand. Based on this information, the agent could calculate a probability of winning and would act differently. All of these agents were modeled and compared through rigorous data of the average number of points won per game and the number of games won over a 31 game set. The analysis of this information provided some insight as to why and when an agent might be stronger than another agent. Future work for this project in the methods of more meticulous decisions processes (either rule-based or Markov) may be improved upon and demonstrates that although this game's AI is rather detailed, there is still a lot of work which could be done in order to create even more intelligent agents.

Bibliography

[1] B. W. Ballard. The *-Minimax Search Procedure for Trees Containing Chance Nodes. *Artificial Intelligence*, 21(3):327–350, 1983.

[2] A. Barzel. The Perplexing Conclusion: The Essential Difference between Natural and Artificial Intelligence is Human Beings' Ability to Deceive. *Journal of applied philosophy*, 15(2):165–178, 1998.

[3] D. Billings and A. Davidson. The challenge of poker. *Artificial Intelligence*, 134(1):201–240, 2002.

[4] D. Billings and J. Schaeffer. Opponent Modeling in Poker. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 493–499. JOHN WILEY & SONS LTD, 1998.

[5] J. R. S. Blair, D. Mutchler, and C. Liu. Games with Imperfect Information. In *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning, AAAI Press Technical Report FS93-02, Menlo Park CA*, pages 59–67, 1993.

[6] G. W. Brown and J. von Neumann. Solutions of Games by Differential Equations. Technical report, DTIC Document, 1950.

[7] N. Buckles. Euchre 0.7. Open source Euchre program, February 2002. Accessed through linux.softpedia.com/progDownload/Euchre-Download-19696.html, December 2012.

[8] M. Buro and J. Long. Improving State Evaluation, Inference, and Search in Trick-Based Card Games. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI2009)*, pages 1407–1413, 2009.

[9] D. Carmel and S. Markovitch. Opponent Modeling in Multi-agent Systems. *Adaption and Learning in Multi-Agent Systems*, pages 40–52, 1996.

[10] W. B. Dick. *The Modern Pocket Hoyle; Containing all the Games of Skill and Chance.* Dick and Fitzgerald, 1868.

[11] M. Dresher. Games of Strategy. *Mathematics Magazine*, 25(2):93–99, 1951.

[12] I. Frank and D. Basin. Search in games with incomplete information: a case study using Bridge card play. *Artificial Intelligence*, 100(1):87–123, 1998.

[13] H. Fujita and S. Ishii. A reinforcement learning scheme for a multi-agent card game. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 5, pages 4071–4078. IEEE, 2003.

[14] M. J. Holmes. Machine Learning in Euchre: a Comparison of Techniques. Master's thesis, University of Northern Iowa, May 2001.

[15] Hoyle. *Hoyle: The Official Name in Gaming*, chapter Rules for Euchre. Encore, Inc., http://www.hoylegaming.com/c-16-rules.aspx#euchre, November 2010.

[16] P. Huang and K. Sycara. Multi-agent Learning in Extensive Games with Complete Information. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 701–708. ACM, July 2003.

[17] S. Ishii and H. Fujita. A Reinforcement Learning Scheme for a Partially-Observable Multi-Agent Game. *Machine Learning*, 59(1):31–54, 2005.

[18] S. J. Johansson. On using Multi-agent Systems in Playing Board Games. *International Conference on Autonomous Agents: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 8(12):569–576, 2006.

[19] R. W. Johnson. Using Games to Teach Markov Chains. *Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 13(4):337–348, 2003.

[20] G. Kendall and C. Smith. The Evolution of Blackjack Strategies. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2474–2481. IEEE, 2003.

[21] P. Langley and H. A. Simon. Applications of Machine Learning & Rule Induction. *Communications of the ACM*, 38(11):54–64, 1995.

[22] J. R. Livingston. Transfer Of Learnt Knowledge With Card Games. Honors Thesis, November 2005. The University of Tasmania.

[23] Y. Matsuno, T. Ymazaki, and S. Ishii. A Multi-Agent Reinforcement Learning Method for a Partially-Observable Competitive Game. In *Proceedings of the fifth international conference on Autonomous agents*, pages 39–40. ACM, 2001.

[24] I. Millington and J. Funge. *Artificial Intelligence for Games*. Morgan Kaufmann, 2 edition, 2009.

[25] J. Mostow. A Problem-Solver for Making Advice Operational. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-83)*, pages 279–283, 1983.

[26] W. Poundstone. *Prisoner's Dilemma*, chapter 3, pages 37–64. Anchor, 1 edition, 1993.

[27] S. Rabin. *AI Game Programming Wisdom*, chapter 2, pages 71–75. Charles River Media, 1 edition, 2002.

[28] J. Ratliff. Bid euchre. Open source Euchre program, August 2005. Accessed through bideuchre.sourceforge.net, December 2012.

[29] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 17, pages 631–640. Pearson Education, 2 edition, 2003.

[30] J. Schaeffer and H. J. van den Herik. Games, computers, and artificial intelligence. *Artificial Intelligence*, 134(1-2):1–8, 2002.

[31] S. Smith and D. Nau. Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 1079–1086. JOHN WILEY & SONS LTD, 1998.

[32] M. R. Sturtevant and R. E. Korf. On Pruning Techniques for Multi-Player Games. In *Proceedings of the National Conference on Artificial Intelligence*, pages 201–208. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2000.

[33] The Diagram Group. *The Ultimate Book of Card Games*. Sterling Publishing Co., Inc., 2004.

[34] M. van Lent and D. Mutchler. A Pruning Algorithm for Imperfect Information Games. In *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning*, 1993.

[35] J. J. Wright. The Mathematics Behind Euchre. Honors Thesis, May 2001. Ball State University.