Optimization Of A Distribution To Maximize Casting Cost In Hearthstone

By

Ryan White

A SENIOR RESEARCH PAPER PRESENTED TO THE DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE OF STETSON UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE

STETSON UNIVERSITY

2018

0.1 Acknowledgments

I would like to acknowledge my family: mom, dad, and sister, for creating an environment where I can learn and grow. Their support and genuine interest in my life has given and continues to give me the confidence to find and excel in areas which I am passionate about. This supportive environment has enabled me to succeed and thrive in a variety of ways.

I would also like to acknowledge Dr. Erich Friedman for not only his great help in conducting my Senior Research, but also for his continued ability to push me. Dr. Friedman's classes have always been far and away the most challenging courses I have taken at Stetson, and thus those in which I find myself learning the most.

I would like to acknowledge Kathryn Sarullo for her help with my code in the second semester of this project. Without her, the code would've taken over twice as long, and I likely would not have been able to finish the amount of code that I did due to the runtime.

I would finally like to acknowledge my friends here at Stetson. College is a time of change and growth for everyone, and there isn't a group of people I would rather have done it with.

0.2 Abstract

Optimization Of A Distribution To Maximize Casting Cost In Hearthstone

By

Ryan White

May 2018

Advisor: Dr. Erich Friedman

Department: Mathematics and Computer Science

Through calculating probabilities for a predetermined number of rounds in a Hearthstone game, we seek to optimize the distribution of our 30 card deck in order to maximize the expected value of total casting cost. We assume that the casting cost, the cost required to play a card, is directly proportional to the value of that card; thus, by optimizing casting cost every round, you also optimize value.

We take two approaches to determine the optimal distribution. Our first approach calculates all combinatorial probabilities for various hands of cards, while our second ranks the best plays for each round and has the computer run through all the possibilities. In both approaches, we utilize MATLAB to calculate the best possible distribution and corresponding expected value. Specifically, we work with a total of both 4 and 5 rounds of play, and use both "the hero power" and "the coin," card game elements unique to Hearthstone, to complicate our calculations and strategies.

Contents

List of Tables

1 Preliminaries

1.1 Hearthstone

Hearthstone is a turn-based online card game played between two players. Both players begin with a total of 30 cards in their respective decks. Player 1 begins the game with 3 of his 30 cards in hand, while Player 2 starts with 4 of hers. Cards in Hearthstone can be spells, minions, or numerous other things with varied cost, health, attack, and damage. Thus, for the purpose of our research we will simplify these stats into one which we can fully analyze: casting cost. Turns and cards are marked by this "casting cost," the cost required to play a card. Both players take one turn each round, with each drawing 1 card at their turn's start. Thus, total cards drawn can be expressed as $3+n$ for Player 1 and $4+n$ for Player 2, where n is the round number. The cards played during round n, assuming the first round is denoted as round 1, cannot exceed a total casting cost of n.

For the purpose of our research, we are assuming that the worth, or value, of a card is directly proportional to its relative casting cost. Hence, 4-cost cards are 4 times as strong as 1-cost cards, and 2 times as strong as 2-cost cards. This assumption also implies that card cost is equivalent to power, regardless of how many cards it may be split across. The combination of a 2-cost card and a 1-cost card represents the same value and power that a 3-cost card does. Thus, maximizing casting cost will maximize card power, or strength, and create the best chance of winning the game.

As a slightly more in-depth explanation of the Hearthstone metagame, Hearthstone, and card games similar to it, consist of 3 types of decks: aggro, midrange, and control. These decks differ in card distribution and game plan. Aggro decks take to the board early, using low cost cards to gain an early game advantage and beat the other player quickly. Midrange decks run some low cost cards with a majority of 3, 4, and 5-cost cards to establish a strong midgame presence and steamroll the game from there. Control decks run many cards that counter opponents' early game aggression, often referred to as "removal cards." Combining

their ability to hold off aggression with expensive, usually 7-cost cards and higher, late game cards, these decks run their opponents out of cards and out value them in the late game.

These 3 decks form a relationship similar to rock, paper, scissors. While some matches are outliers, aggro generally beats midrange, midrange beats control, and control beats aggro. Our research will optimize an aggro deck as it strives to fill out its casting cost on the first several turns of the game in order to finish the game quickly. Calculations to optimize the other types of decks would be impossible as their removal cards have variable value depending on the state of the game. A 5-cost card could be better used on round 9 to maximize its value in one game, but could be game saving if played on round 5 in another.

1.2 Combinatorics

This research will consist of math almost entirely from the combinatorics field of probability, and thus, this section will be a minor refresher in this particular field of mathematics.

To begin, we will look at the probabilities regarding a particular event occurring. A classic example is the probability of drawing a card or cards from a deck. Given a standard 52 card deck, what is the probability that the card we draw is exactly the 5 of Clubs? Such a case reminds us of the basic formula for computing probabilities. The chance for an event to occur is: the number of possibilities where the event occurs divided by the total number of possibilities. We can express the probability of drawing the 5 of Clubs given a standard deck as: $P(5\clubsuit) = \frac{1}{52}$. Using additional possibilities, essentially greater than one case, requires us to make a distinction between permutations and combinations.

1.3 Permutations

Permutations are probabilities where order matters; the order in which you draw cards affects the next card drawn. For example, given that we've drawn a 5 from a standard deck,

the probabilities of drawing subsequent cards change. Previously, the probability of drawing any 5 in the deck was $\frac{4}{52}$, as there are 4 5's in a 52 card deck. However, after having drawn a 5, the probability changes to $\frac{3}{51}$, as there are now 3 5's remaining in the deck of 51 cards. Thus, the probability of drawing a 5 and then another 5 is $\frac{4}{52} \cdot \frac{3}{51}$.

1.4 Combinations

Combinations are probabilities where order does not matter. Instead of calculating the probability as drawing the 5 of clubs and then another 5, combinations calculates the probability that your first 2 cards are both the 5 of clubs and another 5. Combinations utilizes this idea of "choosing" to calculate probabilities. Such a statement is expressed as a factorial divided by a multiplication of several factorials $\frac{p_1!}{p_2!p_3!}$ such that $p_1 = p_2 + p_3$. This is expressed as $\binom{p_1}{p_2}$, as either p_2 or p_3 work when the denominator consists of only 2 factorials. We are choosing p_2 cards out of a possible p_1. Thus, one could express the probability of of drawing a 5 as $\frac{\binom{4}{1}}{\binom{52}{1}}$, which simplifies to $\frac{4}{52}$. The difference from permutations comes from multiple cards rather than simply 1 card.

1.5 Expected Value

Calculating the total expected value of a round involves first finding both the probabilities and possibilities of the given event. We can calculate the probabilities of events for our research using the aforementioned method of "combinations." We use the term "possibilities" to indicate the importance of order. In this research, we found probabilities for each individual case, then multiplied them by the amount of possibilities for each case occurring. Adding these results would give us the probability for a given number occurring on a given round. Multiplying the probabilities for each round by that round's respective value will give the expected value of the round. Finally, as expected values are additive, we can add all rounds' expected values together, yielding the expected value of the total game. The optimization of our deck will be done by maximizing the expected casting value across 4 rounds.

As an example, we can flip a weighted coin. 1 side of the coin has a value of 1, the other side a value of 2. The coin is weighted such that it has a 70% chance to land with the side valued 2 facing up, and thus a 30% chance to get 1. The expected value of such a coin flip can be found by multiplying the probabilities of each event by their respective value, thus $E(\text{weighted coin flip}) = 0.30 \cdot (1) + 0.70 \cdot (2) = 1.7$.

2 Deck Optimization for Player 1

2.1 Definitions

As previously mentioned, our research goal is to optimize a deck for some small number of rounds. However, it is trivial to optimize a Hearthstone deck for only 1, 2, or 3 rounds. We can do this by simply holding all 30 cards to a casting cost of 1. This guarantees our ability to play the maximum casting cost for each of the first 3 rounds, as all cards cost 1: we play a 1-cost on round 1, 2 1-costs on round 2, and 3 1-costs on round 3. This gives us a guaranteed maximum, as the cards played during all rounds are equivalent to the maximum possible casting cost of each turn. Thus, our first goal is to optimize a deck for the first 4 rounds, as the probabilities associated with such a case are not trivial.

We will be using a_1, a_2, a_3, and a_4 as variables to represent the amount of 1, 2, 3, and 4-cost cards, respectively, in the 30 card deck. Again, $a_1 + a_2 + a_3 + a_4 = 30$, and Player 1 has a total of 4 cards on first turn.

2.2 Round 1

We begin by calculating the probability of playing cards with a total casting cost of 0 on round 1. We have no 0-cost cards in our deck, thus, the only feasible way to have a value of 0 is to play no cards at all. Additionally, as we can only play a maximum of 1 casting cost this turn, the way we calculate the probability of having no playable cards on round 1 is to calculate the probability of having no 1-costed cards in the 4 cards that we have drawn thus far.

From Chapter 1, we calculate this probability using combinatorics. We are choosing our first 4 cards to be not 1, thus we are choosing 4 cards out of the non-1-costed cards: a_2, a_3, and a_4. We can express this as $\binom{a_2+a_3+a_4}{4}$. However, we know that probabilities are

"the chance of x happening out of the total possibilities." The total possibilities here can be expressed by the total number of cards choosing any 4, or $\binom{30}{4}$. Thus, the probability of casting a total value of 0 on round 1 is:

$$P(\text{casting 0 on round 1}) = \binom{a_2+a_3+a_4}{4}/\binom{30}{4}$$

As previously stated in Chapter 1, all probabilities for any given event must add to a total of 1. Thus, when there is one probability remaining, we are able to find it by subtracting all the previous probabilities for the event from 1. Here, we subtract $P(0)$ from 1 to get $P(1)$.

$$P(\text{casting 1 on round 1}) = 1 - P(0)$$

2.3 Round 2

We will calculate the probabilities for round 2 in a very similar way, although there are two slight differences from round 1. We will have to keep in mind both the possibility of having played cards in previous rounds and that the total number of cards has increased from 4 to 5. This will alter our previous denominator from $\binom{30}{4}$ to $\binom{30}{5}$.

While very similar to the calculation for scoring a 0 in round 1, round 2 brings in another case. The first case of being unable to play any cards is to simply draw no 1's or 2's anywhere in your first 5 cards. We can express this case as $\binom{a_3+a_4}{5}$. The second case occurs when you have 4 3's or 4's, but you have played a 1 on the first turn. This can be written as $a_3+a_4 = 4$, $a_1 = 1$. While we still have a total of 5 cards for round 2, we can account for the 1 being played in the first round by saying that, of the 5 possible places that our 1 could be, in order for it to be played on the first round, it must be in one of the first 4 places, representing the 4 cards we had access to on round 1. Thus, we include a fraction of $\frac{4}{5}$ with our probability representing $a_3+a_4 = 4$ and $a_1 = 1$. We must express this case as two probabilities multiplied together, as we need to choose 4 cards from $a_3 + a_4$ total 3 and 4-cost cards, and also need to choose 1 card from the a_1 total 1-cost cards. We write this as $\frac{4}{5}\binom{a_3+a_4}{4}\binom{a_1}{1}$. Including the previously mentioned denominator of $\binom{30}{5}$, we get:

$$P(\text{casting 0 on round 2}) = [\tbinom{a_3+a_4}{5} + \tfrac{4}{5}\tbinom{a_3+a_4}{4}\tbinom{a_1}{1}]/\tbinom{30}{5}$$

These probabilities set the theme for the rest of the calculations - almost all involve fractions and several probabilities multiplied by each other. There are only 2 cases that result in a total value of 1 being played in round 2, and both are closely related to the last case of $P(0)$ for round 2. The first case considers casting 1 for both the first and second round. The only way we would play a 1-cost card on round 2 is if we had no other 1's to accompany it, and no 2's to play instead. Thus, for this case, $a_3 + a_4 = 3$, and $a_1 = 2$. The order in which the cards are in does not matter for this specific case, as given any order of the previously mentioned 5 cards, we can always play a 1-cost on both turn 1 and turn 2. The fact that order is irrelevant for this particular case means we do not need a fraction to accompany our choose statements, or more accurately, the fraction is $\frac{5}{5} = 1$. Thus, the first case can be written as $\binom{a_3+a_4}{3}\binom{a_1}{2}$. The second case is closely related to the final case of $P(0)$ for round 2. We want 4 cards costed 3 or 4 ($a_3 + a_4 = 4$), and only 1 card costed 1 ($a_1 = 1$). However, this time we want our 1-cost card to be the last of the 5 cards, the card we draw on round 2. Such a condition means that while the 1-cost card is playable on round 2, it was not playable on round 1. By locking the first 4 cards to only 3 or 4-costed cards, we ensure that our total value will be 1 on round 2 for this case, given the last card has a casting cost of 1. Again, by needing to draw a 1-cost as the last card, we use a fraction of $\frac{1}{5}$ with our probabilities, giving us $\frac{1}{5}\binom{a_3+a_4}{4}\binom{a_1}{1}$ as the term for our second case. Adding these together and dividing by $\binom{30}{5}$, we get:

$$P(\text{casting 1 on round 2}) = [\tbinom{a_3+a_4}{3}\tbinom{a_1}{2} + \tfrac{1}{5}\tbinom{a_3+a_4}{4}\tbinom{a_1}{1}]/\tbinom{30}{5}$$

And finally, we can again express $P(n)$ for round n as the previous possibilities in the same round n subtracted from 1. Thus, for $P(2)$ for round 2, we have:

$$P(\text{casting 2 on round 2}) = 1 - P(0) - P(1)$$

2.4 Round 3

Our calculations for round 3 follow the same process and structure as those for rounds 1 and 2. Thus, for this round, we will simply list the possible hands and give the respective associated probabilities.

Beginning with the probability of playing nothing on round 3, our possible hands are:
6 4-cost cards, in any order: $\binom{a_4}{6}$,
5 4-cost and 1 1-cost cards, given that the 1-cost is not the last card drawn: $\frac{5}{6}\binom{a_4}{5}\binom{a_1}{1}$
4 4-cost and 2 1-cost cards, given that both 1-cost cards are not the last card drawn: $\frac{4}{6}\binom{a_4}{4}\binom{a_1}{2}$
3 4-cost and 3 1-cost cards, given that all 1-cost cards are not the last card drawn: $\frac{3}{6}\binom{a_4}{3}\binom{a_1}{3}$
5 4-cost and 1 2-cost cards, given that the 2-cost is not the last card drawn: $\frac{5}{6}\binom{a_4}{5}\binom{a_2}{1}$
4 4-cost, 1 1-cost, and 1 2-cost cards, given that the 1-cost is in the first 4 cards and the 2-cost in the first 5. This can be expressed as $\frac{4}{6}\frac{4}{5} = \frac{8}{15}$, where $\frac{4}{6}$ is the probability for the 1-cost in the first 4, and $\frac{4}{5}$ is the probability for the 2-cost in the first 5, given the 1-cost. Thus: $\frac{8}{15}\binom{a_4}{4}\binom{a_1}{1}\binom{a_2}{1}$.
Thus, our total probability for casting 0 on round 3, given our 6 card drawn from the 30 card deck at that point, is:

$$P(\text{casting 0 on round 3}) = [\binom{a_4}{6} + \frac{5}{6}\binom{a_4}{5}\binom{a_1}{1} + \frac{4}{6}\binom{a_4}{4}\binom{a_1}{2} + \frac{3}{6}\binom{a_4}{3}\binom{a_1}{3} + \frac{5}{6}\binom{a_4}{5}\binom{a_2}{1} + \frac{8}{15}\binom{a_4}{4}\binom{a_1}{1}\binom{a_2}{1}]/\binom{30}{6}$$

The possible hands for casting 1 on round 3 are:
5 4-cost and 1 1-cost cards, given that the 1-cost is the last card drawn: $\frac{1}{6}\binom{a_4}{5}\binom{a_1}{1}$,
4 4-cost and 2 1-cost cards, given that one of the 2 1-costs is the last card drawn: $\frac{2}{6}\binom{a_4}{4}\binom{a_1}{2}$,
3 4-cost and 3 1-cost, given that one of the 3 1-costs is the last card drawn: $\frac{3}{6}\binom{a_4}{3}\binom{a_1}{3}$,
2 4-cost and 4 1-cost cards, in any order: $\binom{a_4}{2}\binom{a_1}{4}$,
4 4-cost, 1 1-cost, and 1 2-cost cards, given that the 1-cost is the last card drawn: $\frac{1}{6}\binom{a_4}{4}\binom{a_1}{1}\binom{a_2}{1}$,
3 4-cost, 2 1-cost, and 1 2-cost cards, given that a 1-cost is in the first 4, a 2-cost is in the first 5, and a 1-cost is the last card. As these cases were more complicated than most, we found the orders of the case that failed to cast 1 on round 3. We found 2 sets of orders that failed

our casting criteria: those that draw the 2-cost last, with a $\frac{1}{6}$ probability of occurring, and those that draw both 1-cost cards last, with a $\frac{2}{6}\frac{1}{5}$ probability. Summing these orders results in $\frac{1}{6} + \frac{2}{6}\frac{1}{5} = \frac{7}{30}$. Finally, subtracting from 1 gives us $1 - \frac{7}{30} = \frac{23}{30}$. Thus: $\frac{23}{30}\binom{a_4}{3}\binom{a_1}{2}\binom{a_2}{1}$.
Thus, our total probability for casting 1 on round 3 is:

$$P(\text{casting 1 on round 3}) = [\tfrac{1}{6}\tbinom{a_4}{5}\tbinom{a_1}{1} + \tfrac{2}{6}\tbinom{a_4}{4}\tbinom{a_1}{2} + \tfrac{3}{6}\tbinom{a_4}{3}\tbinom{a_1}{3} + \tbinom{a_4}{2}\tbinom{a_1}{4} + \tfrac{1}{6}\tbinom{a_4}{4}\tbinom{a_1}{1}\tbinom{a_2}{1} + \tfrac{23}{30}\tbinom{a_4}{3}\tbinom{a_1}{2}\tbinom{a_2}{1}]/\tbinom{30}{6}$$

The possible hands for casting 2 on round 3 are:
5 4-cost and 1 2-cost cards, given that the 2-cost is the last card drawn: $\frac{1}{6}\binom{a_4}{5}\binom{a_2}{1}$,
4 4-cost, 1 1-cost, and 1 2-cost cards, given that the 2-cost is the last card drawn: $\frac{1}{6}\binom{a_4}{4}\binom{a_1}{1}\binom{a_2}{1}$,
3 4-cost, 2 1-cost, and 1 2-cost cards, given that the 2-cost is the last card drawn: $\frac{1}{6}\binom{a_4}{3}\binom{a_1}{2}\binom{a_2}{1}$,
2 4-cost, 3 1-cost, and 1 2-cost cards, in any order: $\frac{1}{6}\binom{a_4}{2}\binom{a_1}{3}\binom{a_2}{1}$,
3 4-cost, 1 1-cost, and 2 2-cost cards, given that the 1-cost is in the first 4 cards: $\frac{4}{6}\binom{a_4}{3}\binom{a_1}{1}\binom{a_2}{2}$,
2 4-cost, 1 1-cost, and 3 2-cost cards, given that the 1-cost is in the first 4 cards: $\frac{1}{6}\binom{a_4}{2}\binom{a_1}{1}\binom{a_2}{3}$,
1 4-cost, 1 1-cost, and 4 2-cost cards, given that the 1-cost is in the first 4 cards: $\frac{4}{6}\binom{a_4}{1}\binom{a_1}{1}\binom{a_2}{4}$,
1 1-cost and 5 2-cost cards, given that the 1-cost is in the first 4 cards: $\frac{4}{6}\binom{a_1}{1}\binom{a_2}{5}$,
4 4-cost and 2 2-cost cards, in any order: $\binom{a_4}{4}\binom{a_2}{2}$,
3 4-cost and 3 2-cost cards, in any order: $\binom{a_4}{3}\binom{a_2}{3}$,
2 4-cost and 4 2-cost cards, in any order: $\binom{a_4}{2}\binom{a_2}{4}$,
1 4-cost and 5 2-cost cards, in any order: $\binom{a_4}{1}\binom{a_2}{5}$,
6 2-cost cards, in any order: $\binom{a_2}{6}$.
Thus, our total probability for casting 2 on round 3 is:

$$P(\text{casting 2 on round 3}) = [\tfrac{1}{6}\tbinom{a_4}{5}\tbinom{a_2}{1} + \tfrac{1}{6}\tbinom{a_4}{4}\tbinom{a_1}{1}\tbinom{a_2}{1} + \tfrac{1}{6}\tbinom{a_4}{3}\tbinom{a_1}{2}\tbinom{a_2}{1} + \tbinom{a_4}{2}\tbinom{a_1}{3}\tbinom{a_2}{1} + \tfrac{4}{6}\tbinom{a_4}{3}\tbinom{a_1}{1}\tbinom{a_2}{2} + \tfrac{4}{6}\tbinom{a_4}{2}\tbinom{a_1}{1}\tbinom{a_2}{3} + \tfrac{4}{6}\tbinom{a_4}{1}\tbinom{a_1}{1}\tbinom{a_2}{4}\tfrac{4}{6}\tbinom{a_1}{1}\tbinom{a_2}{5} + \tbinom{a_4}{4}\tbinom{a_2}{2} + \tbinom{a_4}{3}\tbinom{a_2}{3} + \tbinom{a_4}{2}\tbinom{a_2}{4} + \tbinom{a_4}{1}\tbinom{a_2}{5} + \tbinom{a_2}{6}]/\tbinom{30}{6}$$

Finally, we can express the probability for casting 3 on round 3 as the previous round probabilities all subtracted from 1:

$$P(\text{casting 3 on round 3}) = 1 - P(0) - P(1) - P(2)$$

2.5 Round 4

Our calculations for round 4 follow the same process and structure as those for the previous rounds. Round 4 probabilities proved to be quite a bit shorter than expected as the possible hands began to depend heavily on remaining hand size.

As a perfect example, there are actually no possible hands that result in casting 0 on round 4, we can play anything drawn. Thus, the probability is:

$$P(\text{casting 0 on round 4}) = 0$$

By the above reasoning concerning hand size, the only hand where a 1-cost is played on round 4 is the one with no cards in hand and drawing a 1-cost as the last card. Anything left in hand on round 4 will be playable, as our round number is now equal to our maximum casting cost value, 4. The only possible hand with no cards left in hand before drawing the last 1-cost is 6 1-cost cards: $\binom{a_1}{7}$.

Thus, as round 4 now has a total hand size of 7, the total probability for playing 1 on round 4 is:

$$P(\text{casting 1 on round 4}) = \binom{a_1}{7}/\binom{30}{7}$$

Following the exact reasoning from above, the only hand where a 2-cost is played on round is 6 1-cost and 1 2-cost cards: $\binom{a_1}{6}\binom{a_2}{1}$.

Thus, the total probability is:

$$P(\text{casting 2 on round 4}) = \binom{a_1}{6}\binom{a_2}{1}/\binom{30}{7}$$

Using the same reasoning, we can see that the possible hands for casting 3 on round 3 are:

6 1-cost and 1 3-cost cards, in any order: $\binom{a_1}{6}\binom{a_2}{1}$,

5 1-cost and 2 3-cost cards, in any order: $\binom{a_1}{5}\binom{a_2}{2}$,

7 3-cost cards, in any order: $\binom{a_3}{7}$,

6 3-cost and 1 1-cost cards, given that the 1-cost is in the first 5 cards: $\frac{5}{7}\binom{a_3}{6}\binom{a_1}{1}$,

6 3-cost and 1 2-cost cards, in any order: $\binom{a_3}{6}\binom{a_2}{1}$,

5 3-cost, 1 1-cost, and 1 2-cost cards, given that the 1-cost is in the first 5 cards: $\frac{4}{7}\binom{a_3}{6}\binom{a_1}{1}$,

5 3-cost and 2 2-cost cards, given that a 2-cost is in the first 5. The easiest way of calculating this is to find the case where this set of cards would result in 4, which is 3333322, with a probability of $\frac{2}{7}\frac{1}{6} = \frac{1}{21}$, found by defining the case as needing to have 2 2-costs at the end. Subtracting this from 1 to find the probabilities that do work gives $\frac{20}{21}$. Thus: $\frac{20}{21}\binom{a_3}{5}\binom{a_2}{2}$,

4 3-cost, 1 1-cost, and 2 2-cost cards, given that the 1-cost is in the first 4 cards and that a 2-cost is in the first 5 cards. The probability for a 1-cost in the first 4 is $\frac{4}{7}$, and the probability for a 2-cost in the first 5 afterwards is $\frac{4}{6}$, resulting in a fraction of $\frac{4}{7}\frac{4}{6} = \frac{8}{21}$. Thus: $\frac{8}{21}\binom{a_3}{4}\binom{a_1}{1}\binom{a_2}{2}$,

4 3-cost and 3 1-cost cards, given that all 3 1-costs are in the first 5: $\frac{2}{7}\binom{a_3}{4}\binom{a_1}{3}$.

Thus, the total probability is:

$$P(\text{casting 3 on round 4}) = [\binom{a_1}{6}\binom{a_3}{1}+\binom{a_1}{5}\binom{a_2}{2}+\binom{a_3}{7}+\frac{5}{7}\binom{a_3}{6}\binom{a_1}{1}+\binom{a_3}{6}\binom{a_2}{1}+\frac{4}{7}\binom{a_3}{5}\binom{a_1}{1}\binom{a_2}{1}+\frac{1}{21}\binom{a_3}{5}\binom{a_2}{2}+\frac{8}{21}\binom{a_3}{4}\binom{a_1}{1}\binom{a_2}{2}+\frac{2}{7}\binom{a_3}{4}\binom{a_1}{3}]/\binom{30}{7}$$

And we can finally express the probability for casting 4 on round as:

$$P(\text{casting 4 on round 4}) = 1 - P(0) - P(1) - P(2) - P(3)$$

2.6 Expected Value

As mentioned in 1.5, expected value is calculated by multiplying probabilities by their respective values in order to find the average result. We are looking to maximize our expected value over 4 rounds of play by optimizing the number of 1, 2, 3, and 4-cost cards in our deck. In order to maximize 4 rounds of play, we must calculate the expected value for each individual

round.

We can find the expected value of round 1 by adding the product of each probability and its value. Intuitively, this equation makes sense; the average value of round 1 should be the probability of getting a 1. Thus, the expected value looks like:

$$E(\text{round } 1) = 0 \cdot P(0) + 1 \cdot P(1)$$

The expected values of the following rounds all resemble round 1, with additional probabilities and values for each round to account for the increased possibilities and values.

$$E(\text{round } 2) = 0 \cdot P(0) + 1 \cdot P(1) + 2 \cdot P(2)$$

$$E(\text{round } 3) = 0 \cdot P(0) + 1 \cdot P(1) + 2 \cdot P(2) + 3 \cdot P(3))$$

$$E(\text{round } 4) = 0 \cdot P(0) + 1 \cdot P(1) + 2 \cdot P(2) + 3 \cdot P(3) + 4 \cdot P(4)$$

Thus, we can find the total expected value over 4 rounds by adding the expected value for each round:

$$E(\text{total}) = E(\text{round } 1) + E(\text{round } 2) + E(\text{round } 3) + E(\text{round } 4)$$

We were able to optimize this deck by coding our probabilities - see the Appendix for the code used. Our code tests every possible distribution of cards where $a_1 + a_2 + a_3 + a_4 = 30$, and finds the case where the 4 round expected value is at a maximum. We found that in order to maximize the expected value for 4 rounds, we need: $a_1 = 13, a_2 = 12, a_3 = 5, a_4 = 0$. Additionally, we found the expected values for each round:

$E(\text{round } 1) = 0.9132$

$E(\text{round } 2) = 1.9937$

$E(\text{round } 3) = 2.9984$

$E(\text{round } 4) = 3.9302$

$E(\text{total}) = 0.9132 + 1.9937 + 2.9984 + 3.9302 = 9.8355$

3 Deck Optimization for Player 2

3.1 Round 1

As Hearthstone is game played between 2 players, it would be remiss to exclude calculations and optimization for the second player. These probabilities proved to be very slightly different, as for the most part, the only changes that occurred were related to the change in hand size. Player 2 has 5 cards on round 1 as opposed to Player 1's 4, and thus 1 more card every subsequent turn as well. This alters both the denominators representing total probability and the fractions representing relevant order.

Thus, the total probability casting 0 on round 1 is:

$$P(\text{casting 0 on round 1}) = \binom{a_2+a_3+a_4}{5}/\binom{30}{5}$$

And the following probability for casting 1 on round 1 remains:

$$P(\text{casting 1 on round 1}) = 1 - P(0)$$

3.2 Round 2

The changes in probabilities for round 2 are derived from the same explanations above, thus the probabilities for casting 0, 1, and 2 on round 2 are:

$$P(\text{casting 0 on round 2}) = [\binom{a_3+a_4}{6} + \frac{5}{6}\binom{a_3+a_4}{5}\binom{a_1}{1}]/\binom{30}{6}$$

$$P(\text{casting 1 on round 2}) = [\binom{a_3+a_4}{4}\binom{a_1}{2} + \frac{1}{6}\binom{a_3+a_4}{5}\binom{a_1}{1}]/\binom{30}{6}$$

$$P(\text{casting 2 on round 2}) = 1 - P(0) - P(1)$$

3.3 Round 3

Round 3 requires similar changes to the previous rounds, however, the probability of casting 2 on round 3 has 2 additional possible hands due to the additional card that player 2 has. Thus, the probabilities are:

$P(\text{casting 0 on round 3}) = [\binom{a_4}{7} + \frac{6}{7}\binom{a_4}{6}\binom{a_1}{1} + \frac{5}{7}\binom{a_4}{5}\binom{a_1}{2} + \frac{4}{7}\binom{a_4}{4}\binom{a_1}{3} + \frac{6}{7}\binom{a_4}{6}\binom{a_2}{1} + \frac{25}{42}\binom{a_4}{5}\binom{a_1}{1}\binom{a_2}{1}]/\binom{30}{7}$

The final case for casting 0 on round 3 was a slightly more involved calculation. The 5 4-cost cards are obviously uncastable on round 3. The requirements for this case to have nothing to cast on round 3 are casting 1 the first round, $\frac{5}{7}$, and 2 the second round, $\frac{5}{6}$. Thus, we have $\frac{5}{7}\frac{5}{6} = \frac{25}{42}$.

$P(\text{casting 1 on round 3}) = [\frac{1}{7}\binom{a_4}{6}\binom{a_1}{1} + \frac{2}{7}\binom{a_4}{5}\binom{a_1}{2} + \frac{3}{7}\binom{a_4}{4}\binom{a_1}{3} + \binom{a_4}{3}\binom{a_1}{4} + \frac{1}{7}\binom{a_4}{5}\binom{a_1}{1}\binom{a_2}{1} + \frac{17}{21}\binom{a_4}{4}\binom{a_1}{2}\binom{a_2}{1}]/\binom{30}{7}$

The probability of casting 1 on round 3 given the case of 4 4-cost, 2 2-cost, and 1 1-cost proved difficult. Much like its Player 1 counterpart, we found orders that failed to cast 1. Again, one set required the 2-cost to be the last card drawn, $\frac{1}{7}$, and one set required both 1-cost cards to be the last 2 cards drawn, $\frac{2}{7}\frac{1}{7}$. Summing these orders gives us $\frac{1}{7} + \frac{2}{7}\frac{1}{7} = \frac{8}{42} = \frac{4}{21}$. Subtracting from 1 to properly account for the probabilities that will cast 1 rather than those that will not, we have: $1 - \frac{4}{21} = \frac{17}{21}$.

$P(\text{casting 2 on round 3}) = [\frac{1}{7}\binom{a_4}{6}\binom{a_2}{1} + \frac{1}{7}\binom{a_4}{5}\binom{a_1}{1}\binom{a_2}{1} + \frac{1}{7}\binom{a_4}{4}\binom{a_1}{2}\binom{a_2}{1} + \binom{a_4}{3}\binom{a_1}{3}\binom{a_2}{1} + \frac{5}{7}\binom{a_4}{4}\binom{a_1}{1}\binom{a_2}{2} + \frac{5}{7}\binom{a_4}{3}\binom{a_1}{1}\binom{a_2}{3} + \frac{5}{7}\binom{a_4}{2}\binom{a_1}{1}\binom{a_2}{4} + \frac{5}{7}\binom{a_4}{1}\binom{a_1}{1}\binom{a_2}{5} + \frac{5}{7}\binom{a_1}{1}\binom{a_2}{6} + \binom{a_4}{5}\binom{a_2}{2} + \binom{a_4}{4}\binom{a_2}{3} + \binom{a_4}{3}\binom{a_2}{4} + \binom{a_4}{2}\binom{a_2}{5} + \binom{a_4}{1}\binom{a_2}{6} + \binom{a_2}{7}]/\binom{30}{7}$

$P(\text{casting 3 on round 3}) = 1 - P(0) - P(1) - P(2)$

3.4 Round 4

The probabilities for round 4 require similar changes. Casting 1 is now impossible given the extra card Player 2 has in addition to the card drawn; the sum of their values must be greater than 1. By that same reasoning, casting 2 for Player 2 is now the same as casting 1 for Player 1. Casting 3 remains similar for both players, with some altered cases due to hand size.

$P(\text{casting 0 on round 4}) = 0$

$P(\text{casting 1 on round 4}) = 0$

$P(\text{casting 2 on round 4}) = \binom{a_1}{8}/\binom{30}{8}$

Several of the following probabilities are much more complex than the rest. $\frac{15}{28}$ is found from analyzing the case where the ordering cannot end with exactly 2 cards, each a non-3-cost: $\frac{6}{8}\frac{5}{7} = \frac{15}{28}$. $\frac{27}{28}$ is calculated by finding the 1 order where the specific case wouldn't work, and subtracting from 1: $1 - \frac{2}{8}\frac{1}{7} = \frac{27}{28}$. Finally, $\frac{135}{224}$ comes from multiplying the previous probability by $\frac{5}{8}$, to include the case where a 1-cost is drawn in the first 5 as well: $\frac{5}{8}\frac{27}{28} = \frac{135}{224}$.

$P(\text{casting 3 on round 4}) = [\binom{a_1}{7}\binom{a_2}{1} + \frac{6}{8}\binom{a_1}{1}\binom{a_3}{7} + \frac{15}{28}\binom{a_1}{2}\binom{a_3}{6} + \frac{15}{28}\binom{a_1}{3}\binom{a_3}{5} + \binom{a_3}{8} + \frac{6}{8}\binom{a_2}{1}\binom{a3}{7} + \frac{27}{28})\binom{a_2}{2}\binom{a_3}{6} + \frac{5}{8}\binom{a_1}{1}\binom{a_2}{1}\binom{a_3}{6} + \frac{135}{224}\binom{a_1}{1}\binom{a_2}{2}\binom{a_3}{5}]/\binom{30}{8}$

$P(\text{casting 4 on round 4}) = 1 - P(0) - P(1) - P(2) - P(3)$

3.5 Expected Value

The expected value calculations for Player 2 differ only in probabilities, not in method. Our equation remains:

$E(\text{total}) = E(\text{round 1}) + E(\text{round 2}) + E(\text{round 3}) + E(\text{round 4})$

with all round-based expected values defined as they were previously. Using code similar to

that found in the Appendix, with adjusted probabilities for Player 2's probabilities, we are able to find that the optimal distribution of cards was: $a_1 = 15, a_2 = 10, a_3 = 5, a_4 = 0$. Thus, the new expected values are:

$E(\text{round } 1) = 0.9789$

$E(\text{round } 2) = 1.9991$

$E(\text{round } 3) = 2.9988$

$E(\text{round } 4) = 3.9867$

$E(\text{total}) = 0.9789 + 1.9991 + 2.9988 + 3.9867 = 9.9635$

4 Hero Power

Our research up to this point has neglected a unique element of Hearthstone, the hero power. The hero power is a 2-cost ability given to both players that has a minor effect on the game when used. It may only be used once per turn, and does not depend on any cards in your hand. For our purposes, we shall assume the hero power provides the same value as any 2-cost card.

The probabilities for both Player 1 and Player 2 with the hero power end revolve entirely around prioritizing the use of the hero power, thus saving a card to use later. Essentially, we can state that using the hero power is always better than casting some cards with a total casting cost of 2. The hero power does not inhibit strategies for later rounds as it is reusable every round, while casting cards may affect later play as cards can only be cast once per game. This strategy simplifies the probabilities greatly.

4.1 Player 1

The probabilities for round 1 with the hero power are the same as round 1 without, as the hero power costs 2.

$P(\text{casting 0 on round 1}) = \binom{a_2+a_3+a_4}{4}/\binom{30}{4}$

$P(\text{casting 1 on round 1}) = 1 - P(0)$

Using the hero power is always superior to playing cards, as you save those cards for possible later use. Thus, for round 2,

$P(\text{casting 0 on round 2}) = 0$

$P(\text{casting 1 on round 2}) = 0$

$P(\text{casting 2 on round 2}) = 1$

The probabilities for round 3 are only nontrivial when calculating the likelihood of casting 2. Any 1-cost or 3-cost cards would result in casting 3; recall that a 1-cost with the hero power will cast 3. We then look for 2-cost and 4-cost cards, or cards played on round 1, before a hero power could be used. Thus,

$P(\text{casting 0 on round 3}) = 0$

$P(\text{casting 1 on round 3}) = 0$

$P(\text{casting 2 on round 3}) = [\binom{a_2+a_4}{6} + \frac{4}{6}\binom{a_2+a_4}{5}\binom{a_1}{1}]/\binom{30}{6}$

$P(\text{casting 3 on round 3}) = 1 - P(2)$

The probabilities for round 4 are straightforward. You will never cast 2, as any 1-cost or 2-cost card drawn on that round will be playable with the hero power for respective casting cost of 3 or 4, and any 3-cost or 4-cost card drawn will simply be playable for its casting cost.

$P(\text{casting 0 on round 4}) = 0$

$P(\text{casting 1 on round 4}) = 0$

$P(\text{casting 2 on round 4}) = 0$

$P(\text{casting 3 on round 4}) = [\binom{a_3}{7} + \frac{4}{7}\binom{a_3}{6}\binom{a_1}{1}]/\binom{30}{7}$

$P(\text{casting 4 on round 4}) = 1 - P(3)$

4.2 Player 2

The probabilities for Player 2 with a hero power are almost exactly the same as Player 1, with slightly altered fractions and probabilities to account for the change in hand size.

$P(\text{casting 0 on round 1}) = \binom{a_2+a_3+a_4}{4}/\binom{30}{5}$

$P(\text{casting 1 on round 1}) = 1 - P(0)$

$P(\text{casting 0 on round 2}) = 0$

$P(\text{casting 1 on round 2}) = 0$

$P(\text{casting 2 on round 2}) = 1$

$P(\text{casting 0 on round 3}) = 0$

$P(\text{casting 1 on round 3}) = 0$

$P(\text{casting 2 on round 3}) = [\binom{a_2+a_4}{7} + \frac{4}{7}\binom{a_2+a_4}{6}\binom{a_1}{1}]/\binom{30}{6}$

$P(\text{casting 3 on round 3}) = 1 - P(2)$

$P(\text{casting 0 on round 4}) = 0$

$P(\text{casting 1 on round 4}) = 0$

$P(\text{casting 2 on round 4}) = 0$

$P(\text{casting 3 on round 4}) = [\binom{a_3}{8} + \frac{5}{8}\binom{a_3}{7}\binom{a_1}{1}]/\binom{30}{7}$

$P(\text{casting 4 on round 4}) = 1 - P(3)$

4.3 Work with Probabilities

The hero power calculations are trivial through 5 rounds of play with a deck of 30 1-cost cards. We intended to calculate the probabilities for 6 rounds with a hero power for both players. However, the hero power creates non-trivial decisions based on both deck and game variance. A quick example to indicate this: you have played all your cards by round 5, and draw a 4. You are presented with 2 distinct options. The hero power costs 2, which leaves you unable to play both the 4-cost card and use the hero power. Therefore, do you play the

4-cost and miss out on the hero power, or do you use the hero power and save the 4-cost card for later turns to get the maximum possible value out of the hero power? We can divide these options into 2 different paths. The "value" oriented path focuses on the maximum amount of value, or total casting cost, across the entire game. In a scenario without the hero power, this changes nothing, as we would still want to maximize our casting cost in each round of play. However, with the hero power, maximizing the total casting cost across all rounds now suggests that we should use the hero power every round, as this generates a casting cost of 2 without having to use our finite resource, cards. Thus, our value-oriented path would calculate probabilities given that the hero power is used every round possible - every round after the first. The "tempo" oriented path focuses on the other option in our example - maximizing each round individually.

Tempo and value are 2 concepts frequently discussed in Hearthstone, and are not unique to the hero power. As with the different paths for the hero power, value promotes maximizing each card's potential during the entirety of a game, while tempo essentially focuses on each turn individually. Tempo sacrifices value to gain immediate power in the game. Applying these concepts to our research forces us to create 2 different sets of probabilities, as determining which path to follow would be based on current hand, cards played, and cards remaining, in addition to deck distribution. We would need to know the distribution of cards in the deck to calculate the probabilities of finding a better "value" play in later rounds, or whether it would be best to make a "tempo" play in the current round. However, needing to know deck distribution makes the decision essentially impossible to generalize. Such a necessity certainly pushes it outside the scope of this research project; thus, the need for 2 approaches.

4.4 Value: Round 1

We begin by attempting to calculate 6 rounds of value-oriented play using the hero power for Player 1. As we are calculating 6 rounds of play, we extend the card options to include 5 and 6-cost cards. The probabilities for rounds 1 and 2 are exactly the same as the hero

power calculations above when not using the value-oriented approach, aside from including the 5's and 6's as unplayable. The hero power cannot be used in round 1, thus the following is trivial.

$P(\text{casting 0 on round 1}) = \binom{a_2+a_3+a_4+a_5+a_6}{4} / \binom{30}{4}$

$P(\text{casting 1 on round 1}) = 1 - P(0)$

4.5 Value: Round 2

Round 2 is also trivial, as using the hero power is always better than casting any combination of cards. The use of the hero power guarantees additional saved cards for increased flexibility in later rounds of play.

$P(\text{casting 0 on round 2}) = 0$

$P(\text{casting 1 on round 2}) = 0$

$P(\text{casting 2 on round 2}) = 1$

4.6 Value: Round 3

The value-oriented approach to round 3 is very similar to the original hero power calculations. Examining the first probability in $P(2)$, the difference lies in which cards we can choose from to create an unplayable hand. While 2, 4, 5, and 6-cost cards are obviously unplayable, we are also prohibited from playing a 3-cost card. In the value-oriented approach, we use the hero power every round that we can. Thus, we automatically cast 2 out of the total possible casting cost of 3 this round. The only card that would force us to cast a different value than 2 on round 3 is a 1-cost card, as we would be able to cast it along with the hero power for a total casting cost of 3. Thus, the probabilities for casting 2 on round 2 are those without access to a 1 on the third round.

$P(\text{casting 0 on round 3}) = 0$

$P(\text{casting 1 on round 3}) = 0$

$P(\text{casting 2 on round 3}) = [\binom{a_2+a_3+a_4+a_5+a_6}{6} + \frac{4}{6}\binom{a_2+a_3+a_4+a_5+a_6}{5}\binom{a_1}{1}] / \binom{30}{6}$

$P(\text{casting 3 on round 3}) = 1 - P(2)$

4.7 Value: Round 4

Probabilities for round 4 follow similar reasoning to those from round 3. We discuss in detail the most challenging calculation, the $\frac{2}{3}$ from the third probability in $P(2)$. Observe first, this is for a hand of 2 1-cost cards and 5 cards not costed 1 or 2. To calculate this fraction, we know that either of the 2 1-cost cards cannot be the last card drawn, as a 1-cost card would be playable with the hero power to cast a value greater than 2. Thus one of the other 5 cards must be drawn last, resulting in a fraction of $\frac{5}{7}$. However, given that we draw one of the non 1-cost cards on round 4, there is still one case where we will be forced to cast more than 2. If our previous 2 cards drawn are both 1-cost, we will not be able to use them both by round 4. Examining this in detail, on round 2, we will always hero power. On round 3, we hero power and use a 1-cost card, thus having 1 leftover for round 4, guaranteeing that we will cast at least 3 on round 4. We can express this total with $\frac{5}{7}(1 - \frac{2}{6}\frac{1}{5})$. This simplifies to $\frac{2}{3}$.

$P(\text{casting 0 on round 4}) = 0$

$P(\text{casting 1 on round 4}) = 0$

$P(\text{casting 2 on round 4}) = [\binom{a_3+a_4+a_5+a_6}{7} + \frac{4}{7}\binom{a_3+a_4+a_5+a_6}{6}\binom{a_1}{1} + \frac{2}{3}\binom{a_3+a_4+a_5+a_6}{5}\binom{a_1}{2}]/\binom{30}{7}$

$P(\text{casting 3 on round 4}) = [\frac{1}{7}\binom{a_3+a_4+a_5+a_6}{6}\binom{a_1}{1} + \frac{2}{7}\binom{a_3+a_4+a_5+a_6}{5}\binom{a_1}{2} + \frac{4}{7}\binom{a_3+a_4+a_5+a_6}{4}\binom{a_1}{3}]/\binom{30}{7}$

$P(\text{casting 4 on round 4}) = 1 - P(3) - P(2)$

4.8 Value: Round 5

Probabilities for round 5, both trivial and not, are similar to those calculated previously.

$P(\text{casting 0 on round 5}) = 0$

$P(\text{casting 1 on round 5}) = 0$

$P(\text{casting 2 on round 5}) = [\binom{a_4+a_5+a_6}{8} + \frac{7}{8}\binom{a_4+a_5+a_6}{7}\binom{a_1}{1} + \frac{6}{8}\binom{a_4+a_5+a_6}{6}\binom{a_1}{2} + \frac{5}{8}\binom{a_4+a_5+a_6}{5}\binom{a_1}{3} +$

$\frac{4}{8}\binom{a_4+a_5+a_6}{4}\binom{a_1}{4} + \frac{7}{8}\binom{a_4+a_5+a_6}{7}\binom{a_2}{1} + \frac{9}{14}\binom{a_4+a_5+a_6}{6}\binom{a_1}{1}\binom{a_2}{1} + \frac{21}{56}\binom{a_4+a_5+a_6}{5}\binom{a_1}{2}\binom{a_2}{1}]/\binom{30}{8}$

$P(\text{casting 3 on round 5}) = [\frac{1}{8}\binom{a_4+a_5+a_6}{7}\binom{a_1}{8} + \frac{2}{8}\binom{a_4+a_5+a_6}{6}\binom{a_1}{2} + \frac{3}{8}\binom{a_4+a_5+a_6}{5}\binom{a_1}{3} + \frac{4}{8}\binom{a_4+a_5+a_6}{4}\binom{a_1}{4} +$
$\binom{a_4+a_5+a_6}{3}\binom{a_1}{5} + \frac{5}{8}\binom{a_4+a_5+a_6}{6}\binom{a_1}{1}\binom{a_2}{1} + \frac{7}{8}\binom{a_4+a_5+a_6}{5}\binom{a_1}{2}\binom{a_2}{1} + \frac{7}{8}\binom{a_4+a_5+a_6}{4}\binom{a_1}{3}\binom{a_2}{1}]/\binom{30}{8}$

$P(\text{casting 4 on round 5}) = [\binom{a_4+a_5+a_6}{2}\binom{a_1}{6} + \frac{1}{8}\binom{a_4+a_5+a_6}{7}\binom{a_2}{1} + \binom{a_4+a_5+a_6}{6}\binom{a_2}{2} + \binom{a_4+a_5+a_6}{5}\binom{a_2}{3} +$
$\binom{a_4+a_5+a_6}{4}\binom{a_2}{4} + \binom{a_4+a_5+a_6}{3}\binom{a_2}{5} + \binom{a_4+a_5+a_6}{2}\binom{a_2}{6} + \binom{a_4+a_5+a_6}{1}\binom{a_2}{7} + \binom{a_2}{8} + \frac{1}{8}\binom{a_4+a_5+a_6}{6}\binom{a_1}{1}\binom{a_2}{1} +$
$\frac{4}{8}\binom{a_4+a_5+a_6}{4}\binom{a_1}{3}\binom{a_2}{1} + \binom{a_4+a_5+a_6}{3}\binom{a_1}{4}\binom{a_2}{1}]/\binom{30}{8}$

$P(\text{casting 5 on round 5}) = 1 - P(4) - P(3) - P(2)$

4.9 Round 6

We found the probabilities in round 6 to be much more difficult and in depth than those in the previous rounds. We worked through the probability of casting 4 on round 6 before we determined it would be more worthwhile to generalize code before proceeding further with non-computer calculated probabilities.

5 Optimization Code

We use MATLAB to code our work. MATLAB is a numerical computing language designed for ease of work regarding matrices. Each of our 4 programs is between 250 and 350 lines long, and the explanation of each can be found later in this chapter, while the analysis and comparisons of their respective runtimes are found in Chapter 6. All programs can be found in the appendix.

5.1 Code: Player 1, 4 Rounds

Our first goal was to create code that would find the distribution which maximized the average value across the first 4 rounds of play for player 1. Our initial attempt ran through 7 "for" loops to create an array that would model all possible hands, and another 3 "for" loops designed to loop through all possible distributions of cards. Kathryn Sarullo helped us in cutting down the amount of "for" loops used in my code. As we found out, 10 nested loops makes the code take much longer than it should. Kathryn was able to find open source code by Jos van der Geest that created a matrix containing all possible hands [5]. By removing the 7 "for" loops that altered an array $4^7 = 16384$ times for each distribution and instead creating a matrix with 16384 rows, the runtime of our initial code decreased dramatically, from 3.5 hours to less than 1.5 hours.

We then use 3 "for" loops to test for all possible distributions of cards, and use 1 more to loop through the matrix created by the open source code. Another "for" loop used is for the probability of any 1 hand actually occurring given the particular distribution of 1, 2, 3, and 4-cost cards. This loop keeps track of the likelihood of each card occurring in that order for each row of the matrix. After calculating the value of the game for that specific matrix row, we multiply the probability of that hand occurring given the current distribution by that game value.

The code for the first round of play is elementary. We begin by defining a variable $r1$ to

be the first 4 elements of our particular matrix row. This allows us to inspect the first 4 cards of any given total hand, which are the cards that player 1 has access to on his first turn. Our one conditional simply checks for the existence of a 1-cost card in $r1$. If it is found, we replace the position of that 1-cost card with a 0, as to accurately reflect the card's non-impact on the coming rounds. We then increase *gametotal*, the variable tracking our total casting cost for that particular hand, by 1 to indicate that we have cast a total of 1 on the first round. Should a 1-cost card not be found in $r1$, nothing will happen, and the code for round 2 will begin.

Round 2, and the rounds following, all perform essentially the same function, with later rounds having slightly more complicated code. We begin by checking for the best possible play on that round. For round 2, this would be casting a 2-cost card, done by checking for the existence of a 2 in $r2$, defined as the first 5 elements of our matrix row. This represents player 1 having access to 5 total cards on the second round of play. Additionally, we define $ta2r1$ to be a matrix of 0's and 1's, with 1's in $ta2r1$ representing where in our ar matrix row the element is a 1. This is to check for the existence of multiple of a particular card, and will be used in all subsequent rounds as well to represent a variety of card costs. Utilizing this, we can now check for the next best play on round 2, should we lack a 2-cost, which is casting a total of 2 by casting 2 1-cost cards. We look for the sum of $ta2r1$ to be greater than or equal to 2. If this occurs, we find the first 2 elements of $ta2r1$ that are 1. We then set those both to 0 to represent both 1-cost cards being cast, and increase *gametotal* by 2. The next best option is simply checking for the existence of a 1 and incrementing *gametotal* by 1, and finally admitting that nothing is playable should all of those options fail.

Round 3 contains far more conditionals, but nothing new.

Round 4 contains similar code yet again, but a non-trivial choice for the ordering of the conditionals. In the previous rounds, there was always a clear frontrunner, with casting a 2-cost on round 2 being clearly better than casting 2 1-costs, and casting a 3-cost on round 3 being clearly better than casting a 2-cost and a 1-cost, and both being better than casting 3 1-costs. However, while casting a 4-cost on round 4 is obviously the best option available, it is

not immediately clear whether it is better to cast a 3-cost and a 1-cost or 2 2-costs. Although such a decision doesn't actually affect the result of this code at all, as it only computes up to 4 rounds of play, and both of these options cast 4, this choice will affect what cards we will have left in the following rounds, should both options be available to us. Determining which option is superior relies on us knowing the current distribution of cards and what cards have been played and drawn thus far, as we are looking to best optimize for what cards to come. Again, this choice is far from trivial. Thus, for the time being, we simply chose 3-cost and 1-cost to be checked for first, then 2 2-costs. It is worth addressing that there are other options for casting 4 on round 4. Those options, while better than casting below 4, are clearly worse than the previously discussed ones. Casting a 2-cost and 2 1-costs is clearly worse than 2 2-costs, and 4 1-costs worse than both of those. The reasoning, as stated earlier in the paper, is due to the loss in flexibility. Low cost cards can be more easily combined with other cards later in the game, while high cost cards struggle due to their innate inflexibility.

Finally, we calculate the sum of all of the products of each game total and the probability of it occurring given a distribution. We track the distribution with this maximum expected value and output both at the end of our code. This "best" distribution and respective expected value are our results.

5.2 Code: Player 2, 4 Rounds

There were very few differences between code for player 1 and code for player 2 in 4 rounds of play. When using the open source code to create our matrix, we had to extend the number of elements in each row from 7 to 8, to account for the extra card player 2 received every game. This increased the number of rows in the matrix from $4^7 = 16384$ to $4^8 = 65536$. Additionally, we had to adjust the code calculating the probability to account for the increase in cards. We also had to adjust $r1, r2, r3$, and $r4$ to each include 1 additional card, compensating again for the player 2's having access to 5 cards as opposed to 4 on the first round, and 1 additional every subsequent round as well. However, the actual calculations

for determining the optimal distribution remain the same throughout this code.

5.3 Code: Player 1, 5 Rounds

The differences in this code are very similar to those presented for player 2, 4 rounds. Again, we increased the matrix to account for access to that additional card. However, as we entered 5 rounds of play, we also introduced the 5-cost card. Thus, the number of rows in our ar matrix increased from 4^7 to $5^8 = 390625$. This also caused us to make similar changes to the probability section of the code. However, this time we had to add another conditional accounting for the possibility that $a5$ could equal 0. The round variables match those present in the player 1 code for 4 rounds, as player 1 does not have access to an additional card, simply an additional round of play is introduced.

We did have to add a section to the main portion of code calculating the casting cost for round 5. As with round 4, we ran into similar problems in the evaluation of which way to cast certain values is actually better. Again, determining which method of casting a certain value is better usually depends on flexibility. Casting a 5-cost card is always better than casting 5 1-cost cards, as those 1-cost cards are much more versatile and thus more easily combined with other cards in the following rounds. However, evaluating a 4-cost and a 1-cost against a 3-cost and a 2-cost is certainly non-trivial. This would again depend on our knowledge of not only the current distribution but also what cards have been played so far, and thus what cards we are more likely to draw and have a better chance of combining with these lower cost cards to create better future rounds. Such analysis is beyond the scope of this project.

5.4 Code: Player 2, 4 Rounds, Coin

The coin is future predictive, meaning that accurately evaluating the coin and determining which turn is best to cast it would require us to not only know the distribution of cards and what cards have been cast thus far, but also what cards we will draw. We can only truly

assess the coin after viewing our total hand over the course of the game, and retroactively examining and determining on which turn it would have best been cast. Again, while the coin only increases the maximum casting cost of a turn by 1, it can increase the total value of a turn, and by result, the game, by more than 1. Thus, immediately casting the coin when a possible increase in casting cost of 1 is detected would neglect future turns' possible increase in casting value of more than 1. This logic extends to casting 2, as perhaps a turn passes after we have used the coin, and we have only 4-cost cards on round 3, and thus miss a total casting cost of 3 on that round.

As a result of the coin's nature to require future values, we have chosen a "dumb" algorithm instead. While this fix will not perfectly optimize the code, it attempts to evaluate and rank the usage of the coin compared to the other possible card combinations on any given round. We begin by declaring $coin = 1$, a variable that will indicate the existence of the coin throughout each game. Whenever the code uses the coin, it will set the value of *coin* to 0, indicating to future rounds that the coin has been used. By coding to not only check for the existence of certain cards in a given round but also the coin, we can evaluate the proper round usage for this particular "dumb" algorithm.

We position the check for the coin in round 1 after the possibility of being able to cast 1, and before the possibility of having nothing to cast. We check for the value of the coin to still be 1 as to maintain structure used throughout the rest of the code with the coin, and there to be at least 2 2-cost cards in $r1$. Our method essentially says, "You may cast the coin if you cannot cast a 1, but you must also have the ability to cast a 2 both this round and the next." We check for the existence of another 2-cost card as to not waste the coin. If we were to simply attempt to find 1 2-cost card, we may have nothing to cast the following round. In this particular situation, we would end up casting the same amount of value using the coin - round 1: coin + 2, round 2: 0, as not using the coin - round 1: 0, round 2: 2. This situation clearly wastes the coin's potential to be used in a subsequent round. We do not check for combinations of cards creating a total casting cost 2, as the only combination of cards would be 2 1-costs, and we have already determined this particular coin usage "worse" than that

of 1 1-cost. Should we use the coin, we will have not found a 1-cost, thus we will not have found 2-costs by the same token. Using the coin here would increase *gametotal* by 2, and guarantee our ability to cast 2 on the following round as well.

In round 2, we position the check for the coin after all possible ways to cast 2. We attempt to find a similar hand here, looking in $r2$ for the existence of 2 ways to cast 3, either 2 3-cost cards or 3-cost, 2-cost, 1-cost. We do not check for 1 2 and 1 2 or 3 and 1 1 1 as those possibilities are already covered previously in the round by the code checking the 2 1-cost cards in order to cast 2 on round 2. We also check for the value of the coin to still be 1, as to ascertain that it has not been already used in round 1. Should these conditions all be true, we would use the coin and increase the game total by 3, knowing that we would also be able to cast 3 on the following round. This process is repeated in the same position for rounds 3 and 4, with a slight modification in the round 4 code. As we are not yet using 5-cost cards for our 4 round game, we instead check for a 3-cost and a 2-cost card on only round 4. No other combination of cards exists with our checking of the various ways to cast 4 on round 4. Our game ends at 4 rounds, so looking for multiple card combinations summing to 5 is unnecessary.

Finally, we check at the end of the set of conditionals to ensure that the coin indeed has been used. If it has not, we increment *gametotal* by 1 to account for our not using it and it having innately a value of 1. Thus, the expected value of the coin will have a maximum of 11 rather than 10. Our code reflects this change as well.

6 Results

6.1 Player 1, 4 Rounds

The results from the code for player 1 with 4 rounds of play are:

$maxetotal = 9.7417$

$max1 = 15$

$max2 = 7$

$max3 = 6$

$max4 = 2$

For a complete summarization of results, see Table 1 in section 6.6.

This indicates that for player 1, solely examining play through 4 rounds, the optimal distribution of a deck of 30 cards for such a player would be 15 1-costs, 7 2-costs, 6 3-costs, and 2 4-cost cards, resulting in an average game total of 9.7417 out of a possible 10. Similar to our work last semester, the distribution found has a much larger amount of 1-cost cards than any other cost. This is due to the respective brevity of this specific game. 4 rounds of play is very few for a game of Hearthstone. Thus, the impact a 1-cost card has on the game is far greater than it would be were we to attempt to calculate the same variables for a game with 10 rounds of play. By a similar token, as there are only 4 rounds, the maximum value for any 1 game is 10. Failing to maximize your casting cost by even 1-cost on any round already yields an *etotal*, the expected value of a particular distribution, of only 9. Thus, failing to have at least 1 1-cost card in your hand on the first round already drops your *etotal* to 9. We assume that this large amount of 1-cost cards in the optimal distribution is mainly due to its great effect on ensuring our maximization of total casting cost on the first round. However, as mentioned earlier in this paper, too many 1-costs approaches the trivial solution of 30 1-cost cards, which only allows for an *etotal* of 7, far below our goal of 10. 15 1-cost cards defines that necessary balance between ensuring total casting cost is maximized on round 1 and not running out of cards to play by round 4. The amount of 2, 3, and 4-cost cards decreasing

in that order is also to be expected. The innate flexibility of lower cost cards means that having respectively more of them should have a greater chance of maximizing your *etotal* every game.

Our calculated *maxetotal* of 9.7417 again indicates that this distribution we found is the "best" at maximizing our total casting cost. This "best" was found by taking the product of the value casted for a particular hand and the probability of that hand occurring, then taking the sum of those products for 1 distribution, and finally finding the distribution that had the largest sum. The 9.7417 suggests that player 1, with a deck of 30 cards with the above distribution, should usually get a game total of 10. Proportionally, we find $10 - 9.7417 = .2583$, yielding $\frac{.2583}{10} = .02583$, or 2.583%, indicating that this distribution is good, but not exceptional.

6.2 Player 2, 4 Rounds

The results from the code for player 2 with 4 rounds of play are:

$maxetotal = 9.9527$

$max1 = 16$

$max2 = 8$

$max3 = 5$

$max4 = 1$

For player 2, the optimal distribution of a 30 card deck is 16 1-costs, 8 2-costs, 5 3-costs, and 1 4-cost, resulting in an average game total of 9.9527. Recall that the only difference between player 1 and player 2 is the turn order of Hearthstone, and the game's attempt to compensate. To account for player 2 taking the second turn, she also receives an extra card in her hand on round 1, and by extension every round. Player 1 has, or had, access to a total of 4 cards on round 1, 5 on round 2, 6 on round 3, etc. Player 2, by contrast, had 5 cards on round 1, 6 on round 2, 7 on round 3, etc. This access to an extra card each round is the only difference between these 2 calculations. We attempt to emphasize this point because while

the changes in the distribution may not be extreme, the increase in *maxetotal* is astonishing.

Player 1's *maxetotal* of 9.7417 was already very close to the perfect game of 10. 10 as an average should be impossible to achieve unless the solution is trivial and thus the same hand is always drawn. Thus, our *maxetotal* of 9.9527 for player 2 showcases just how good this distribution is at maximizing casting cost every round. By only shifting 2 cards, a 3 and 4-cost to a 1 and 2-cost, and increasing the hand size by 1, the average game result for our distribution improved by .211, a 2.166% increase on player 1's 9.7417. It is worth noting that again, for a small amount of rounds, lower cost cards are superior to higher cost cards. Lower cost cards improve flexibility and the ability to cast 1 on the first turn, but also increase the likelihood of your running out of cards down the road. Such a situation is obviously preferable in games with fewer rounds. Calculating proportionally again, we find $10 - 9.9527 = .0473$, yielding $\frac{.0473}{10} = .00473$, or .473%. This distribution clearly is much better than that for player 1 with 4 rounds, as $.473 < 2.583$.

6.3 Player 1, 5 Rounds

The results from the code for player 1 with 5 rounds of play are:

$maxetotal = 14.3027$

$max1 = 9$

$max2 = 9$

$max3 = 7$

$max4 = 5$

$max5 = 0$

For player 1 and 5 rounds of play, the optimal distribution of a 30 card deck is 9 1-costs, 9 2-costs, 7 3-costs, 5 4-costs, and 0 5-costs, resulting in an average game total of 14.3027. A fifth round obviously changes the variables drastically. With the inclusion of this fifth round, the new optimal distribution must attempt to balance our previous concerns, having a 1-cost card on the first round and flexibility to cast later, against new concerns, running out of cards

to actually maximize casting cost on a later round. This new concern is the primary reason for a much more even distribution across 1, 2, 3, and 4-cost cards. The exclusion of 5-cost cards from this optimal distribution indicates that while accounting for an additional round certainly requires changes, the inability of the 5-cost card to be cast on any other round outweighs its guarantee of maximizing that fifth round. The 5-cost card for a game going up to 5 rounds of play is easily compared to a 4-cost card for a game going up to only 4 rounds of play. We can see by comparison that the optimal distributions for an n-cost card, where n is the highest round calculated for that game, are going to tend towards 0. We can expect this trend to continue should we increase n further. Again, this is due to our inability to cast an n-cost card on any round but the final one. This restriction appears severe enough to limit our n-cost cards distinctly from the rest of the distribution.

The new *maxetotal* value of 14.3027 is proportionally much worse than either of our previous calculations. Recall that our best possible game will now have a total casting cost of 15 rather than 10. Proportionally, we can then see that $15 - 14.3027 = .6973$, and that $\frac{.6973}{15} = .04649$, or 4.649%. Comparing results from the previous games, we can see $4.649 > 2.583 > .473$. Our new distribution's *maxetotal* is much further away from the best possible game than either of the two previous distributions with their respective games.

6.4 Player 2, 4 Rounds, Coin

The results from the code for player 2 with 4 rounds of play and the coin are:

$maxetotal = 10.9857$

$max1 = 14$

$max2 = 8$

$max3 = 6$

$max4 = 2$

For player 2 and 4 rounds of play with the coin, the optimal distribution of a 30 card deck is 14 1-costs, 8 2-costs, 6 3-costs, and 2 4-costs, resulting in an average game total of

10.9857. These results are very similar to those yielded by both player 1 and player 2 in 4 rounds without the coin. The small differences between games can be explained with the introduction of the coin. Note the decrease in 1-cost cards to 14, and recall the change in the code on round 1. After searching for and failing to find a 1-cost card in our opening hand of 5 cards, without the coin we would note a total value cast of 0 on round 1. However, with the coin, after searching for and failing to find that 1-cost card on round 1, we have the option to search for 2 2-cost cards. Take this optimal distribution for example, and note the given that our hand of 5 cards on round 1 has no 1-cost cards. Thus, our 5 card hand is made up of some combination of 2, 3, and 4-cost cards. In order to still cast value on round 1, and guarantee value on round 2, we must have 2 2-cost cards in this 5 card hand. Calculating this probability gives us $[\binom{8}{2}\binom{8}{3}]/\binom{16}{5} = \frac{14}{39} = .3590 = 35.90\%$. Given this distribution, and no 1-cost cards in the hand on round 1, there is still a 35.90% chance that we are able to cast 2 on round 1, and 2 again on round 2, thus not failing to cast value on either turn, where without the coin we would have missed at least 1 casting cost on the first round. Further, the probability that we fail to get at least 1 1-cost card on the first round is $[\binom{16}{5}]/\binom{30}{5} = \frac{8}{261} = .0307 = 3.07$ percent. Additionally, 35.90% of the time that we do indeed fail to cast a 1-cost card on round 1, we are able to utilize the coin to make up for it. Thus, the probability that we are unable to cast 1 on round 1 or use the coin to cast 2 with the guarantee of also being able to cast 2 on the second round is $\frac{25}{39}\frac{8}{261} = \frac{200}{10179} = .0196 = 1.96\%$. With this distribution, we are only unable to cast anything on round 1 1.96% of the time. This calculation showcases why our *maxetotal* for this round is so close to the perfect game value of 11. The coin, in combination with a starting hand of 5 rather than that of 4, appears to create a stalwart and consistent deck that produces perfect games far more often than not.

Examining the *maxetotal* value of 10.9857, we see that $11 - 10.9857 = .0143$, and that $\frac{.0143}{11} = .0013$, or .13%. Comparing this result with our previous coding proportions, we see that $.13 < .473 < 2.583 < 4.649$. The distribution found for player 2 and 4 rounds with the coin improves upon our best code measured proportionally. This distribution gets even closer to always resulting in the optimal game of 11 than the previous best one did to 10. This further indicates how effective the coin is at improving upon a player's "curve," meaning how

well they can maximize their casting cost each turn throughout the game.

6.5 Code Runtimes

The runtimes for the 4 programs are as follows:

Player 1, 4 Rounds: 4676.876 seconds

Player 2, 4 Rounds: 18754.203 seconds

Player 1, 5 Rounds: 198000 seconds

Player 2, 4 Rounds, Coin: 22533.664 seconds

The most notable point to make regarding our code is just how much faster the final iteration is than the first attempted iteration. As discussed earlier, our first attempt at code for player 1 and 4 rounds did not use the open source code, and instead ran through an additional 7 "for" loops to create the array ar which represented the hand for the particular game. This code for player 1 and 4 rounds took approximately 3.5 hours to run, well over twice as long as the current 78 minutes. This massive reduction in runtime affected not only the player 1 code, but the code for all different games run.

The difference in runtimes is entirely expected and numerically calculable. Examining the difference between the $p1$ and $p2$ code, we can see that $p1$ takes 4676.876 seconds to run, while $p2$ takes 18754.203 seconds. We also notice that $\frac{18754.203}{4676.876} = 4.010$. That is, the code for player 2 takes almost exactly 4 times as long to run as it does for p1. This increase in runtime makes perfect sense; recall the difference between the $p1$ and $p2$ code. The only disparity between them is the extra card received for player 2 throughout the game. This card increases each row of the matrix created by the source code from a length of 7 to a length of 8. Recalling again that in both instances, 1, 2, 3, and 4-cost cards are the only available cards in a player's deck, we note that the number of unique hands has increased from 4^7 to 4^8, an increase by a factor of 4, almost perfectly matching the increase in runtime by a factor of 4.01.

Comparing $p2$ and $p2coin$, we can see an increase by a factor of $\frac{22533.664}{18754.203} = 1.202$, or roughly a 20% increase. It is certainly more difficult to pinpoint exactly where the increase in runtime came from. Observe the number of conditional statements in our $p2$ code: 25. Now observe the number of conditionals added to that code to get the $p2coin$ code: 5, for a total of 30. By calculating the proportional increase to be $\frac{30}{25} = 1.200$. This value is almost exactly equal to the proportional increase in runtime of 1.202; there is only a difference of .002. Thus, we can assume the increase in runtime came directly from the increase in conditionals.

Finally, we made a mistake calculating the runtime for player 1 and 5 rounds, $p1r5$, and thus only have an approximate of roughly 55 hours, or around 198000 seconds. This code is easiest to compare to the player 1, 4 round code. There are 2 noticeable differences, both of those previously mentioned. As we have increased to 5 rounds, we see large change in the matrix. We now have access to 8 cards, thus an increase in length of each row of the matrix from 7 to 8. Additionally, we are now able to use 5-cost cards, thus each element has 5 possibilities for what it may be. Thus, we see a proportional increase of $\frac{5^8}{4^7} = 23.842$. A matrix nearly 24 times the size of our previous one for $p1$ should have an effect similar to increase the runtime by around 24 times. We have also increased the number of conditionals in our code by 18, from 25 to 43. This increase, represented proportionally, looks like $\frac{43}{25} = 1.720$. Taking our original runtime of $p1$ of 4676.876 seconds, we can attempt to find the approximate value for the new runtime of $p1r5$ by multiplying this value by our calculated proportional increases. We are able to find that $4676.876 \cdot 23.842 \cdot 1.720 = 191790.453$ seconds, approximately. Converting to hours, our calculation yields 53.275 hours, very close to our original approximate. Thus, we can conclude that the increase in runtime is most likely directly related to the increase in both matrix size and number of conditional statements.

We can also estimate the runtime for player 2 given 5 rounds of play. Take the estimated runtime for $p1r5$, 191790. Note that the only difference between player 1 and player 2 is the additional card given to the second player. Thus, the expansion of the matrix from 5^8 to 5^9 should result in a correlated increase in runtime by a factor of 5: $191790 \cdot 5 = 958950$ seconds, or 266.375 hours.

6.6 Summary of Results

Our computer results are summarized in the table below.

Table 1: Summary of Results

Rounds	Player 1	Player 2
4	E(total)=9.7417 a_1=15 a_2=7 a_3=6 a_4=2 runtime=1.30 hours	E(total)=9.9527 a_1=16 a_2=8 a_3=5 a_4=1 runtime=5.21 hours
5	E(total)=14.3027 a_1=9 a_2=9 a_3=7 a_4=5 a_5=0 runtime=55 hours	Computationally Infeasible
4 + Coin	Player 1 does not get the coin	E(total)=10.9857 a_1=14 a_2=8 a_3=6 a_4=2 runtime=6.26 hours

7 Future Work

7.1 Hero Power

While we attempted to calculate probabilities by hand for the hero power, and did so successfully through 5 rounds of play, such a solution is still trivial. Creating algorithms for both the "tempo" and "value" hero power lines of play and determining which is better for the situations we have currently coded is certainly an option for future work.

7.2 Optimization of Probability

At the beginning of this research, we choose to optimize the expected value across 4 rounds of play. The other option we were considering was finding the distribution of cards such that it maximized the probability of playing perfectly over those 4 rounds. While the probabilities for each playable case would remain the same, we would be looking at cases that enable each player to cast "on curve" every single turn. Playing on curve is a phrase coined from numerous games similar to Hearthstone, and represents casting a total of n on round n. While obviously playing on curve would be the optimal result in the research we conducted this semester, finding the distribution to maximize the probability that a player can cast perfectly every turn would likely have focus on different areas, mainly the first and final rounds, as those are the ones with the greatest chance of missing a casting cost.

8 Appendix

8.1 Code: Player 1, 4 rounds

```
clear;
clc;
close;

%create matrix
v = [1 2 3 4];
v1 = {v, v, v, v, v, v, v};
v2 = allcomb(v1{:});

%declare values
probdenom = (30*29*28*27*26*25*24);
e = 0;
c=30;
maxetotal=0;
etotal=0;
max1=0;
max2=0;
max3=0;
max4=0;

%test for all possible distributions of cards
for a1=0:c
    for a2=0:c-a1
        for a3=0:c-a1-a2
            a4=c-a1-a2-a3;
%loop through the matrix
for i = 1 : size(v2, 1)
    ar = v2(i, :);

%set temp values
gametotal = 0;
prob = 1/probdenom;
tempa1=a1;
tempa2=a2;
tempa3=a3;
tempa4=a4;
%calculate probability for each hand of cards existing
for j=1:7
    if ar(j)==1
        if a1==0
            prob=0;
        else
            prob=prob*a1;
            a1=a1-1;
        end
    elseif ar(j)==2
        if a2==0
            prob=0;
        else
            prob=prob*a2;
            a2=a2-1;
        end
    elseif ar(j)==3
        if a3==0
            prob=0;
        else
            prob=prob*a3;
            a3=a3-1;
        end
    elseif ar(j)==4
        if a4==0
            prob=0;
        else
            prob=prob*a4;
            a4=a4-1;
```

```
        end
    end
end
r1 = ar(1:4);
%round 1
if (ismember(1,r1))
    a1r1 = find(r1==1,1);
    ar(a1r1) = 0;
    %disp('Round 1: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 1: Could not play anything :(')
end
r2 = ar(1:5);
%round 2
ta2r1 = (r2==1);
if (ismember(2,r2))
    a2r2 = find(r2==2,1);
    ar(a2r2) = 0;
    %disp('Round 2: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta2r1)>=2)
    a2r11 = find(ta2r1==1,2);
    ar(a2r11) = 0;
    %disp('Round 2: Played 2 1s!');
    gametotal = gametotal+2;
elseif(sum(ta2r1)==1)
    a2r1 = find(ta2r1==1,1);
    ar(a2r1) = 0;
    %disp('Round 2: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 2: Could not play anything :(')
end
r3 = ar(1:6);
%round 3
ta3r1 = (r3==1);
if (ismember(3,r3))
    a3r3 = find(r3==3,1);
    ar(a3r3) = 0;
    %disp('Round 3: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r3) && ismember(1,r3))
    a3r2 = find(r3==2,1);
    a3r1 = find(r3==1,1);
    ar(a3r2)=0;
    ar(a3r1)=0;
    %disp('Round 3: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta3r1)>=3)
    a3r111 = find(ta3r1==1,3);
    ar(a3r111) = 0;
    %disp('Round 3: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r3))
    a3r2 = find(r3==2,1);
    ar(a3r2)=0;
    %disp('Round 3: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==2)
    a3r11 = find(ta3r1==1,2);
    ar(a3r11) = 0;
    %disp('Round 3: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==1)
    a3r1 = find(ta3r1==1,1);
    ar(a3r1) = 0;
    %disp('Round 3: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 3: Could not play anything :(');
end
r4 = ar(1:7);
```

```
%round 4
ta4r2 = (r4==2);
ta4r1 = (r4==1);
if (ismember(4,r4))
    a4r4 = find(r4==4,1);
    ar(a4r4) = 0;
    %disp('Round 4: Played a 4!');
    gametotal = gametotal+4;
elseif (ismember(3,r4) && ismember(1,r4))
    a4r3 = find(r4==3,1);
    a4r1 = find(r4==1,1);
    ar(a4r3)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 3 and a 1!');
    gametotal = gametotal+4;
elseif (sum(ta4r2)>=4)
    a4r22 = find(ta4r2==1,2);
    ar(a4r22) = 0;
    %disp('Round 4: Played 2 2s!');
    gametotal = gametotal+4;
elseif (ismember(2,r4) && sum(ta4r1)>=2)
    a4r2 = find(r4==2,1);
    a4r11 = find(ta4r1==1,2);
    ar(a4r2)=0;
    ar(a4r11)=0;
    %disp('Round 4: Played a 2 and 2 1s!');
    gametotal = gametotal+4;
elseif (sum(ta4r1)>=4)
    a4r1111 = find(ta4r1==1,4);
    ar(a4r1111) = 0;
    %disp('Round 4: Played 4 1s!');
    gametotal = gametotal+4;
elseif (ismember(3,r4))
    a4r3 = find(r4==3,1);
    ar(a4r3)=0;
    %disp('Round 4: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r4) && ismember(1,r4))
    a4r2 = find(r4==2,1);
    a4r1 = find(r4==1,1);
    ar(a4r2)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta4r1)==3)
    a4r111 = find(ta4r1==1,3);
    ar(a4r111) = 0;
    %disp('Round 4: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r4))
    a4r2 = find(r4==2,1);
    ar(a4r2)=0;
    %disp('Round 4: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==2)
    a4r11 = find(ta4r1==1,2);
    ar(a4r11) = 0;
    %disp('Round 4: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==1)
    a4r1 = find(ta4r1==1,1);
    ar(a4r1) = 0;
    %disp('Round 4: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 4: Could not play anything :(');
end
tempetotal=gametotal*prob;
etotal=etotal+tempetotal;

a1=tempa1;
a2=tempa2;
a3=tempa3;
```

```
a4=tempa4;

end
if etotal>maxetotal
    maxetotal=etotal;
    max1=tempa1;
    max2=tempa2;
    max3=tempa3;
    max4=tempa4;

end
tempetotal=0;
etotal=0;


        end
    end
end

maxetotal
max1
max2
max3
max4
```

8.2 Code: Player 2, 4 Rounds

```
clear;
clc;
close;

%create matrix
v = [1 2 3 4];
v1 = {v, v, v, v, v, v, v, v};
v2 = allcomb(v1{:});

%declare values
probdenom = (30*29*28*27*26*25*24*23);
c=30;
maxetotal=0;
etotal=0;
max1=0;
max2=0;
max3=0;
max4=0;

%test for all possible distributions of cards
for a1=0:c
    for a2=0:c-a1
        for a3=0:c-a1-a2
            a4=c-a1-a2-a3;
%loop through the matrix
for i = 1 : size(v2, 1)
    ar = v2(i, :);

%set temp values
gametotal = 0;
prob = 1/probdenom;
tempa1=a1;
tempa2=a2;
tempa3=a3;
tempa4=a4;
for j=1:8
    if ar(j)==1
        if a1==0
            prob=0;
        else
            prob=prob*a1;
```

```
            a1=a1-1;
        end
    elseif ar(j)==2
        if a2==0
            prob=0;
        else
            prob=prob*a2;
            a2=a2-1;
        end
    elseif ar(j)==3
        if a3==0
            prob=0;
        else
            prob=prob*a3;
            a3=a3-1;
        end
    elseif ar(j)==4
        if a4==0
            prob=0;
        else
            prob=prob*a4;
            a4=a4-1;
        end
    end
end
r1 = ar(1:5);
%disp(ar);
%round 1
if (ismember(1,r1))
    a1r1 = find(r1==1,1);
    ar(a1r1) = 0;
    %disp('Round 1: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 1: Could not play anything :(')
end
r2 = ar(1:6);
%round 2
ta2r1 = (r2==1);
if (ismember(2,r2))
    a2r2 = find(r2==2,1);
    ar(a2r2) = 0;
    %disp('Round 2: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta2r1)>=2)
    a2r11 = find(ta2r1==1,2);
    ar(a2r11) = 0;
    %disp('Round 2: Played 2 1s!');
    gametotal = gametotal+2;
elseif(sum(ta2r1)==1)
    a2r1 = find(ta2r1==1,1);
    ar(a2r1) = 0;
    %disp('Round 2: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 2: Could not play anything :(')
end
r3 = ar(1:7);
%round 3
ta3r1 = (r3==1);
if (ismember(3,r3))
    a3r3 = find(r3==3,1);
    ar(a3r3) = 0;
    %disp('Round 3: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r3) && ismember(1,r3))
    a3r2 = find(r3==2,1);
    a3r1 = find(r3==1,1);
    ar(a3r2)=0;
    ar(a3r1)=0;
    %disp('Round 3: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta3r1)>=3)
```

```
    a3r111 = find(ta3r1==1,3);
    ar(a3r111) = 0;
    %disp('Round 3: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r3))
    a3r2 = find(r3==2,1);
    ar(a3r2)=0;
    %disp('Round 3: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==2)
    a3r11 = find(ta3r1==1,2);
    ar(a3r11) = 0;
    %disp('Round 3: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==1)
    a3r1 = find(ta3r1==1,1);
    ar(a3r1) = 0;
    %disp('Round 3: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 3: Could not play anything :(');
end
r4 = ar(1:8);
%round 4
ta4r2 = (r4==2);
ta4r1 = (r4==1);
if (ismember(4,r4))
    a4r4 = find(r4==4,1);
    ar(a4r4) = 0;
    %disp('Round 4: Played a 4!');
    gametotal = gametotal+4;
%question 1: would you rather played 3,1 or 2,2?
elseif (ismember(3,r4) && ismember(1,r4))
    a4r3 = find(r4==3,1);
    a4r1 = find(r4==1,1);
    ar(a4r3)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 3 and a 1!');
    gametotal = gametotal+4;
elseif (sum(ta4r2)>=4)
    a4r22 = find(ta4r2==1,2);
    ar(a4r22) = 0;
    %disp('Round 4: Played 2 2s!');
    gametotal = gametotal+4;
elseif (ismember(2,r4) && sum(ta4r1)>=2)
    a4r2 = find(r4==2,1);
    a4r11 = find(ta4r1==1,2);
    ar(a4r2)=0;
    ar(a4r11)=0;
    %disp('Round 4: Played a 2 and 2 1s!');
    gametotal = gametotal+4;
elseif (sum(ta4r1)>=4)
    a4r1111 = find(ta4r1==1,4);
    ar(a4r1111) = 0;
    %disp('Round 4: Played 4 1s!');
    gametotal = gametotal+4;
elseif (ismember(3,r4))
    a4r3 = find(r4==3,1);
    ar(a4r3)=0;
    %disp('Round 4: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r4) && ismember(1,r4))
    a4r2 = find(r4==2,1);
    a4r1 = find(r4==1,1);
    ar(a4r2)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta4r1)==3)
    a4r111 = find(ta4r1==1,3);
    ar(a4r111) = 0;
    %disp('Round 4: Played 3 1s!');
    gametotal = gametotal+3;
```

```
elseif (ismember(2,r4))
    a4r2 = find(r4==2,1);
    ar(a4r2)=0;
    %disp('Round 4: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==2)
    a4r11 = find(ta4r1==1,2);
    ar(a4r11) = 0;
    %disp('Round 4: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==1)
    a4r1 = find(ta4r1==1,1);
    ar(a4r1) = 0;
    %disp('Round 4: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 4: Could not play anything :(');
end
tempetotal=gametotal*prob;
etotal=etotal+tempetotal;

a1=tempa1;
a2=tempa2;
a3=tempa3;
a4=tempa4;

end
if etotal>maxetotal
    maxetotal=etotal;
    max1=tempa1;
    max2=tempa2;
    max3=tempa3;
    max4=tempa4;

end
tempetotal=0;
etotal=0;


        end
    end
end

maxetotal
max1
max2
max3
max4
```

8.3 Code: Player 1, 5 Rounds

```
% clear;
clc;
close;

%create matrix
v = [1 2 3 4 5];
v1 = {v, v, v, v, v, v, v, v};
v2 = allcomb(v1{:});

%declare values
probdenom = (30*29*28*27*26*25*24*23);
c=30;
maxetotal=0;
etotal=0;
max1=0;
max2=0;
max3=0;
```

```
max4=0;
max5=0;

%test for all possible distributions of cards
for a1=0:c
    for a2=0:c-a1
        for a3=0:c-a1-a2
            for a4=0:c-a1-a2-a3
                a5=c-a1-a2-a3-a4;
%loop through the matrix
for i = 1 : size(v2, 1)
    ar = v2(i, :);

%set temp values
gametotal = 0;
prob = 1/probdenom;
tempa1=a1;
tempa2=a2;
tempa3=a3;
tempa4=a4;
tempa5=a5;
for j=1:8
    if ar(j)==1
        if a1==0
            prob=0;
        else
            prob=prob*a1;
            a1=a1-1;
        end
    elseif ar(j)==2
        if a2==0
            prob=0;
        else
            prob=prob*a2;
            a2=a2-1;
        end
    elseif ar(j)==3
        if a3==0
            prob=0;
        else
            prob=prob*a3;
            a3=a3-1;
        end
    elseif ar(j)==4
        if a4==0
            prob=0;
        else
            prob=prob*a4;
            a4=a4-1;
        end
    elseif ar(j)==5
        if a5==0
            prob=0;
        else
            prob=prob*a5;
            a5=a5-1;
        end
    end
end
r1 = ar(1:4);
%disp(ar);
%round 1
if (ismember(1,r1))
    a1r1 = find(r1==1,1);
    ar(a1r1) = 0;
    %disp('Round 1: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 1: Could not play anything :(')
end
r2 = ar(1:5);
%round 2
ta2r1 = (r2==1);
```

```
if (ismember(2,r2))
    a2r2 = find(r2==2,1);
    ar(a2r2) = 0;
    %disp('Round 2: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta2r1)>=2)
    a2r11 = find(ta2r1==1,2);
    ar(a2r11) = 0;
    %disp('Round 2: Played 2 1s!');
    gametotal = gametotal+2;
elseif(sum(ta2r1)==1)
    a2r1 = find(ta2r1==1,1);
    ar(a2r1) = 0;
    %disp('Round 2: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 2: Could not play anything :(')
end
r3 = ar(1:6);
%round 3
ta3r1 = (r3==1);
if (ismember(3,r3))
    a3r3 = find(r3==3,1);
    ar(a3r3) = 0;
    %disp('Round 3: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r3) && ismember(1,r3))
    a3r2 = find(r3==2,1);
    a3r1 = find(r3==1,1);
    ar(a3r2)=0;
    ar(a3r1)=0;
    %disp('Round 3: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta3r1)>=3)
    a3r111 = find(ta3r1==1,3);
    ar(a3r111) = 0;
    %disp('Round 3: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r3))
    a3r2 = find(r3==2,1);
    ar(a3r2)=0;
    %disp('Round 3: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==2)
    a3r11 = find(ta3r1==1,2);
    ar(a3r11) = 0;
    %disp('Round 3: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==1)
    a3r1 = find(ta3r1==1,1);
    ar(a3r1) = 0;
    %disp('Round 3: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 3: Could not play anything :(');
end
r4 = ar(1:7);
%round 4
ta4r2 = (r4==2);
ta4r1 = (r4==1);
if (ismember(4,r4))
    a4r4 = find(r4==4,1);
    ar(a4r4) = 0;
    %disp('Round 4: Played a 4!');
    gametotal = gametotal+4;
%question 1: would you rather played 3,1 or 2,2?
elseif (ismember(3,r4) && ismember(1,r4))
    a4r3 = find(r4==3,1);
    a4r1 = find(r4==1,1);
    ar(a4r3)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 3 and a 1!');
    gametotal = gametotal+4;
```

```
elseif (sum(ta4r2)>=4)
    a4r22 = find(ta4r2==1,2);
    ar(a4r22) = 0;
    %disp('Round 4: Played 2 2s!');
    gametotal = gametotal+4;
elseif (ismember(2,r4) && sum(ta4r1)>=2)
    a4r2 = find(r4==2,1);
    a4r11 = find(ta4r1==1,2);
    ar(a4r2)=0;
    ar(a4r11)=0;
    %disp('Round 4: Played a 2 and 2 1s!');
    gametotal = gametotal+4;
elseif (sum(ta4r1)>=4)
    a4r1111 = find(ta4r1==1,4);
    ar(a4r1111) = 0;
    %disp('Round 4: Played 4 1s!');
    gametotal = gametotal+4;
elseif (ismember(3,r4))
    a4r3 = find(r4==3,1);
    ar(a4r3)=0;
    %disp('Round 4: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r4) && ismember(1,r4))
    a4r2 = find(r4==2,1);
    a4r1 = find(r4==1,1);
    ar(a4r2)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta4r1)==3)
    a4r111 = find(ta4r1==1,3);
    ar(a4r111) = 0;
    %disp('Round 4: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r4))
    a4r2 = find(r4==2,1);
    ar(a4r2)=0;
    %disp('Round 4: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==2)
    a4r11 = find(ta4r1==1,2);
    ar(a4r11) = 0;
    %disp('Round 4: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==1)
    a4r1 = find(ta4r1==1,1);
    ar(a4r1) = 0;
    %disp('Round 4: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 4: Could not play anything :(');
end
r5=ar(1:8);
%round 5
ta5r2 = (r5==2);
ta5r1 = (r5==1);
if (ismember(5,r5))
    a5r5 = find(r5==5,1);
    ar(a5r5) = 0;
    %disp('Round 5: Played a 5!');
    gametotal = gametotal+5;
%question: which kind of 5 is second best?
elseif (ismember(4,r5) && ismember(1,r5))
    a5r4 = find(r5==4,1);
    a5r1 = find(r5==1,1);
    ar(a5r4)=0;
    ar(a5r1)=0;
    %disp('Round 5: Played a 4 and a 1!');
    gametotal = gametotal+5;
elseif (ismember(3,r5) && ismember(2,r5))
    a5r3 = find(r5==3,1);
    a5r2 = find(r5==2,1);
    ar(a5r3)=0;
```

```
    ar(a5r2)=0;
    %disp('Round 5: Played a 3 and a 2!');
    gametotal = gametotal+5;
elseif (ismember(3,r5) && sum(ta5r1)>=2)
    a5r3 = find(r5==3,1);
    a5r11 = find(ta5r1==1,2);
    ar(a5r3)=0;
    ar(a5r11)=0;
    %disp('Round 5: Played a 3 and 2 1's!');
    gametotal = gametotal+5;
elseif (sum(ta5r2)>=2 && ismember(1,r5))
    a5r22 = find(ta5r2==1,2);
    a5r1 = find(r5==1,1);
    ar(a5r2)=0;
    ar(a5r1)=0;
    %disp('Round 5: Played 2 2's and a 1!');
    gametotal = gametotal+5;
elseif (ismember(2,r5) && sum(ta5r1)>=3)
    a5r2 = find(r5==2,1);
    a5r111 = find(ta5r1==1,3);
    ar(a5r2)=0;
    ar(a5r111)=0;
    %disp('Round 5: Played a 2 and 3 1's!');
    gametotal = gametotal+5;
elseif (sum(ta5r1)>=5)
    a5r11111 = find(ta5r1==1,5);
    ar(a5r11111)=0;
    %disp('Round 5: Played 5 1's!');
    gametotal = gametotal+5;
elseif (ismember(4,r5))
    a5r4 = find(r5==4,1);
    ar(a5r4)=0;
    %disp('Round 5: Played a 4!');
    gametotal = gametotal+4;
elseif (ismember(3,r5) && ismember(1,r5))
    a5r3 = find(r5==3,1);
    a5r1 = find(r5==1,1);
    ar(a5r3)=0;
    ar(a5r1)=0;
    %disp('Round 5: Played a 3 and a 1!');
    gametotal = gametotal+4;
elseif (sum(ta5r2)>=2)
    a5r22 = find(ta5r2==1,2);
    ar(a5r22)=0;
    %disp('Round 5: Played 2 2's!');
    gametotal = gametotal+4;
elseif (sum(ta5r1)>=4)
    a5r1111 = find(ta5r1==1,4);
    ar(a5r1111)=0;
    %disp('Round 5: Played 4 1's!');
    gametotal = gametotal+4;
elseif (ismember(3,r5))
    a5r3 = find(r5==3,1);
    ar(a5r3)=0;
    %disp('Round 5: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r5) && ismember(1,r5))
    a5r2 = find(r5==2,1);
    a5r1 = find(r5==1,1);
    ar(a5r2)=0;
    ar(a5r1)=0;
    %disp('Round 5: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta5r1)>=3)
    a5r111 = find(ta5r1==1,3);
    ar(a5r111)=0;
    %disp('Round 5: Played 3 1's!');
    gametotal = gametotal+3;
elseif (ismember(2,r5))
    a5r2 = find(r5==2,1);
    ar(a5r2)=0;
    %disp('Round 5: Played a 2!');
    gametotal = gametotal+2;
```

```
elseif (sum(ta5r1)>=2)
    a5r11 = find(ta5r1==1,2);
    ar(a5r11)=0;
    %disp('Round 5: Played 2 1's!')
    gametotal = gametotal+2;
elseif (ismember(1,r5))
    a5r1 = find(r5==1,1);
    ar(a5r1)=0;
    %disp('Round 5: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 5: Could not play anything :(');
end
tempetotal=gametotal*prob;
etotal=etotal+tempetotal;

a1=tempa1;
a2=tempa2;
a3=tempa3;
a4=tempa4;
a5=tempa5;

end
if etotal>maxetotal
    maxetotal=etotal;
    max1=tempa1;
    max2=tempa2;
    max3=tempa3;
    max4=tempa4;
    max5=tempa5;

end
tempetotal=0;
etotal=0;

            end
        end
    end
end

maxetotal
max1
max2
max3
max4
max5
```

8.4 Code: Player 2, 4 Rounds, Coin

```
clear;
clc;
close;

%create matrix
v = [1 2 3 4];
v1 = {v, v, v, v, v, v, v, v};
v2 = allcomb(v1{:});

%declare values
probdenom = (30*29*28*27*26*25*24*23);
c=30;
coinround=0;
maxetotal=0;
etotal=0;
max1=0;
max2=0;
max3=0;
max4=0;
```

```
maxcoin=0;

%test for all possible distributions of cards
for a1=0:c
    for a2=0:c-a1
        for a3=0:c-a1-a2
            a4=c-a1-a2-a3;
%loop through the matrix
for i = 1 : size(v2, 1)
    ar = v2(i, :);

%set temp values
gametotal = 0;
coin=1;
prob = 1/probdenom;
tempa1=a1;
tempa2=a2;
tempa3=a3;
tempa4=a4;
for j=1:8
    if ar(j)==1
        if a1==0
            prob=0;
        else
            prob=prob*a1;
            a1=a1-1;
        end
    elseif ar(j)==2
        if a2==0
            prob=0;
        else
            prob=prob*a2;
            a2=a2-1;
        end
    elseif ar(j)==3
        if a3==0
            prob=0;
        else
            prob=prob*a3;
            a3=a3-1;
        end
    elseif ar(j)==4
        if a4==0
            prob=0;
        else
            prob=prob*a4;
            a4=a4-1;
        end
    end
end
r1 = ar(1:5);
%round 1
ta1r2 = (r1==2);
if (ismember(1,r1))
    a1r1 = find(r1==1,1);
    ar(a1r1) = 0;
    %disp('Round 1: Played a 1!');
    gametotal = gametotal+1;
elseif (coin==1 && sum(ta1r2)>=2)
    a1r2 = find(ta1r2==1,1);
    ar(a1r2)=0;
    %disp('Round 1: Used the coin and played a 2!');
    gametotal = gametotal+2;
    coin = 0;
    coinround=1;
else
    %disp('Round 1: Could not play anything :(')
end
r2 = ar(1:6);
%round 2
ta2r1 = (r2==1);
ta2r3 = (r2==3);
if (ismember(2,r2))
```

```
    a2r2 = find(r2==2,1);
    ar(a2r2) = 0;
    %disp('Round 2: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta2r1)>=2)
    a2r11 = find(ta2r1==1,2);
    ar(a2r11) = 0;
    %disp('Round 2: Played 2 1s!');
    gametotal = gametotal+2;
elseif (coin==1 && ((sum(ta2r3)>=2) || (ismember(3,r2) && ismember(2,r2) && ismember(1,r2))))
    a2r3 = find(ta2r3==1,1);
    ar(a2r3) = 0;
    %disp('Round 2: Used the coin and played a 3!');
    gametotal = gametotal + 3;
    coin = 0;
    coinround=2;
elseif(sum(ta2r1)==1)
    a2r1 = find(ta2r1==1,1);
    ar(a2r1) = 0;
    %disp('Round 2: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 2: Could not play anything :(')
end
r3 = ar(1:7);
%round 3
ta3r1 = (r3==1);
ta3r2 = (r3==2);
ta3r4 = (r3==4);
if (ismember(3,r3))
    a3r3 = find(r3==3,1);
    ar(a3r3) = 0;
    %disp('Round 3: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r3) && ismember(1,r3))
    a3r2 = find(r3==2,1);
    a3r1 = find(r3==1,1);
    ar(a3r2)=0;
    ar(a3r1)=0;
    %disp('Round 3: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta3r1)>=3)
    a3r111 = find(ta3r1==1,3);
    ar(a3r111) = 0;
    %disp('Round 3: Played 3 1s!');
    gametotal = gametotal+3;
elseif (coin==1 && ((sum(ta3r4)>=2) || (ismember(4,r3) && (sum(ta3r2)>=2)) || (sum(ta3r2)>=4)))
    if (ismember(4,r3))
        a3r4 = find(ta3r4==1,1);
        ar(a3r4)=0;
        %disp('Round 3: Used the coin and played a 4!');
        gametotal = gametotal+4;
        coin = 0;
        coinround=3;
    else
        a3r22 = find(ta3r2==1,2);
        ar(a3r22)=0;
        %disp('Round 3: Used the coin and played 2 2's!');
        gametotal = gametotal+4;
        coin = 0;
        coinround=3;
    end
elseif (ismember(2,r3))
    a3r2 = find(r3==2,1);
    ar(a3r2)=0;
    %disp('Round 3: Played a 2!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==2)
    a3r11 = find(ta3r1==1,2);
    ar(a3r11) = 0;
    %disp('Round 3: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta3r1)==1)
```

```
    a3r1 = find(ta3r1==1,1);
    ar(a3r1) = 0;
    %disp('Round 3: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 3: Could not play anything :(');
end
r4 = ar(1:8);
%r5 = ar(1:9);
%round 4
ta4r1 = (r4==1);
ta4r2 = (r4==2);
if (ismember(4,r4))
    a4r4 = find(r4==4,1);
    ar(a4r4) = 0;
    %disp('Round 4: Played a 4!');
    gametotal = gametotal+4;
%question 1: would you rather played 3,1 or 2,2?
elseif (ismember(3,r4) && ismember(1,r4))
    a4r3 = find(r4==3,1);
    a4r1 = find(r4==1,1);
    ar(a4r3)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 3 and a 1!');
    gametotal = gametotal+4;
elseif (sum(ta4r2)>=4)
    a4r22 = find(ta4r2==1,2);
    ar(a4r22) = 0;
    %disp('Round 4: Played 2 2s!');
    gametotal = gametotal+4;
elseif (ismember(2,r4) && sum(ta4r1)>=2)
    a4r2 = find(r4==2,1);
    a4r11 = find(ta4r1==1,2);
    ar(a4r2)=0;
    ar(a4r11)=0;
    %disp('Round 4: Played a 2 and 2 1s!');
    gametotal = gametotal+4;
elseif (sum(ta4r1)>=4)
    a4r1111 = find(ta4r1==1,4);
    ar(a4r1111) = 0;
    %disp('Round 4: Played 4 1s!');
    gametotal = gametotal+4;
elseif (coin==1 && (ismember(3,r4) && ismember(2,r4)))
    a4r3 = find(r4==3,1);
    a4r2 = find(r4==2,1);
    ar(a4r3)=0;
    ar(a4r2)=0;
    %disp('Round 4: Used the coin and played a 3 and a 2!');
    gametotal = gametotal+5;
    coin = 0;
    coinround=4;
elseif (ismember(3,r4))
    a4r3 = find(r4==3,1);
    ar(a4r3)=0;
    %disp('Round 4: Played a 3!');
    gametotal = gametotal+3;
elseif (ismember(2,r4) && ismember(1,r4))
    a4r2 = find(r4==2,1);
    a4r1 = find(r4==1,1);
    ar(a4r2)=0;
    ar(a4r1)=0;
    %disp('Round 4: Played a 2 and a 1!');
    gametotal = gametotal+3;
elseif (sum(ta4r1)==3)
    a4r111 = find(ta4r1==1,3);
    ar(a4r111) = 0;
    %disp('Round 4: Played 3 1s!');
    gametotal = gametotal+3;
elseif (ismember(2,r4))
    a4r2 = find(r4==2,1);
    ar(a4r2)=0;
    %disp('Round 4: Played a 2!');
    gametotal = gametotal+2;
```

```
elseif (sum(ta4r1)==2)
    a4r11 = find(ta4r1==1,2);
    ar(a4r11) = 0;
    %disp('Round 4: Played 2 1s!');
    gametotal = gametotal+2;
elseif (sum(ta4r1)==1)
    a4r1 = find(ta4r1==1,1);
    ar(a4r1) = 0;
    %disp('Round 4: Played a 1!');
    gametotal = gametotal+1;
else
    %disp('Round 4: Could not play anything :(');
end
if coin==1
    gametotal = gametotal+1;
    coin=0;
    coinround=5;
end
tempetotal=gametotal*prob;
etotal=etotal+tempetotal;

a1=tempa1;
a2=tempa2;
a3=tempa3;
a4=tempa4;
end
if etotal>maxetotal
    maxetotal=etotal;
    max1=tempa1;
    max2=tempa2;
    max3=tempa3;
    max4=tempa4;
    maxcoin=coinround;

end
tempetotal=0;
etotal=0;


        end
    end
end

maxetotal
max1
max2
max3
max4
maxcoin
```

8.5 Code: Optimization of Average Game Total for Player 1

The following MATLAB code optimizes the distribution of 1, 2, 3, and 4-cost cards in the deck. This code was written during the first semester of work on the project, and thus includes hand calculated probabilities talked about at length in Sections 2 and 3 of this paper. Reliance on probabilities creates robust code, but becomes incredibly inefficient anywhere past 4 rounds of play. Hence, the other code included in this project is much different than that included here.

```
pc1=4;
pc2=pc1+1;
c=30;
a1max=0;
a2max=0;
a3max=0;
```

```
a4max=0;
e=0;
for a1=0:c
    for a2=0:c-a1
        for a3=0:c-a1-a2
            a4=c-a1-a2-a3;
            %round 1
            %initial choose statement variable for round 1
            a2a3a4c4=0;
            %if statement for a2+a3+a4 choose for round 1
            if (a2+a3+a4)<4
                a2a3a4c4=0;
            else
                a2a3a4c4=nchoosek(a2+a3+a4,4);
            end
            p10=a2a3a4c4/nchoosek(c,4);
            p11=1-p10;
            e1=0*p10+1*p11; %expected value for round 1
            %round 2
            %initial choose statement variables for round 2
            a3a4c5=0;
            a3a4c4=0;
            a3a4c3=0;
            a1c2=0;
            a1c1=0;
            %if statements for a3+a4 chooses for round 2
            if (a3+a4)==4
                a3a4c5=0;
                a3a4c4=1;
                a3a4c3=nchoosek(a3+a4,3);
            elseif (a3+a4)==3
                a3a4c5=0;
                a3a4c4=0;
                a3a4c3=1;
            elseif (a3+a4)<3
                a3a4c5=0;
                a3a4c4=0;
                a3a4c3=0;
            else
                a3a4c5=nchoosek(a3+a4,5);
                a3a4c4=nchoosek(a3+a4,4);
                a3a4c3=nchoosek(a3+a4,3);
            end
            %if statements for a1 chooses for round 2
            if a1==1
                a1c2=0;
                a1c1=1;
            elseif a1==0
                a1c2=0;
                a1c1=0;
            else
                a1c2=nchoosek(a1,2);
                a1c1=nchoosek(a1,1);
            end
            p20=(a3a4c5+(4/5)*a3a4c4*a1c1)/nchoosek(c,5);
            p21=(a3a4c3*a1c2+(1/5)*a3a4c4*a1c1)/nchoosek(c,5);
            p22=1-p20-p21;
            e2=0*p20+1*p21+2*p22; %expected value for round 2
            %round 3
            %initial choose statement variables for round 3
            a4c6=0;
            a4c5=0;
            a4c4=0;
            a4c3=0;
            a4c2=0;
            a4c1=0;
            a2c6=0;
            a2c5=0;
            a2c4=0;
            a2c3=0;
            a2c2=0;
            a2c1=0;
            a1c4=0;
```

```
a1c3=0;
a1c2=0;
a1c1=0;
%if statements for a4 chooses for round 3
if a4==5
    a4c6=0;
    a4c5=1;
    a4c4=nchoosek(a4,4);
    a4c3=nchoosek(a4,3);
    a4c2=nchoosek(a4,2);
    a4c1=nchoosek(a4,1);
elseif a4==4
    a4c6=0;
    a4c5=0;
    a4c4=1;
    a4c3=nchoosek(a4,3);
    a4c2=nchoosek(a4,2);
    a4c1=nchoosek(a4,1);
elseif a4==3
    a4c6=0;
    a4c5=0;
    a4c4=0;
    a4c3=1;
    a4c2=nchoosek(a4,2);
    a4c1=nchoosek(a4,1);
elseif a4==2
    a4c6=0;
    a4c5=0;
    a4c4=0;
    a4c3=0;
    a4c2=1;
    a4c1=nchoosek(a4,1);
elseif a4==1
    a4c6=0;
    a4c5=0;
    a4c4=0;
    a4c3=0;
    a4c2=0;
    a4c1=1;
elseif a4==0
    a4c6=0;
    a4c5=0;
    a4c4=0;
    a4c3=0;
    a4c2=0;
    a4c1=0;
else
    a4c6=nchoosek(a4,6);
    a4c5=nchoosek(a4,5);
    a4c4=nchoosek(a4,4);
    a4c3=nchoosek(a4,3);
    a4c2=nchoosek(a4,2);
    a4c1=nchoosek(a4,1);
end
%if statements for a2 chooses for round 3
if a2==5
    a2c6=0;
    a2c5=1;
    a2c4=nchoosek(a2,4);
    a2c3=nchoosek(a2,3);
    a2c2=nchoosek(a2,2);
    a2c1=nchoosek(a2,1);
elseif a2==4
    a2c6=0;
    a2c5=0;
    a2c4=1;
    a2c3=nchoosek(a2,3);
    a2c2=nchoosek(a2,2);
    a2c1=nchoosek(a2,1);
elseif a2==3
    a2c6=0;
    a2c5=0;
    a2c4=0;
```

```
    a2c3=1;
    a2c2=nchoosek(a2,2);
    a2c1=nchoosek(a2,1);
elseif a2==2
    a2c6=0;
    a2c5=0;
    a2c4=0;
    a2c3=0;
    a2c2=1;
    a2c1=nchoosek(a2,1);
elseif a2==1
    a2c6=0;
    a2c5=0;
    a2c4=0;
    a2c3=0;
    a2c2=0;
    a2c1=1;
elseif a2==0
    a2c6=0;
    a2c5=0;
    a2c4=0;
    a2c3=0;
    a2c2=0;
    a2c1=0;
else
    a2c6=nchoosek(a2,6);
    a2c5=nchoosek(a2,5);
    a2c4=nchoosek(a2,4);
    a2c3=nchoosek(a2,3);
    a2c2=nchoosek(a2,2);
    a2c1=nchoosek(a2,1);
end
%if statements for a1 chooses for round 3
if a1==3
    a1c4=0;
    a1c3=1;
    a1c2=nchoosek(a1,2);
    a1c1=nchoosek(a1,1);
elseif a1==2
    a1c4=0;
    a1c3=0;
    a1c2=1;
    a1c1=nchoosek(a1,1);
elseif a1==1
    a1c4=0;
    a1c3=0;
    a1c2=0;
    a1c1=1;
elseif a1==0
    a1c4=0;
    a1c3=0;
    a1c2=0;
    a1c1=0;
else
    a1c4=nchoosek(a1,4);
    a1c3=nchoosek(a1,3);
    a1c2=nchoosek(a1,2);
    a1c1=nchoosek(a1,1);
end
p30=(a4c6+(5/6)*a4c5*a1c1+(4/6)*a4c4*a1c2+(3/6)*a4c3*a1c3+(5/6)*a4c5*a2c1+(4/5)*(4/6)*a4c4*a1c1
p31=((1/6)*a4c5*a1c1+(2/6)*a4c4*a1c2+(3/6)*a4c3*a1c3+a4c2*a1c4+(1/6)*a4c4*a2c1*a1c1+(23/30)*a4c
p32=((1/6)*a4c5*a2c1+(1/6)*a4c4*a1c1*a2c1+(1/6)*a4c3*a1c2*a2c1+a4c2*a1c3*a2c1+(4/6)*a4c3*a1c1*a
p33=1-p30-p31-p32;
e3=0*p30+1*p31+2*p32+3*p33; %expected value for round 3
%round 4
%initial choose statement variables for round 4
a3c7=0;
a3c6=0;
a3c5=0;
a3c4=0;
a3c1=0;
%a2c2=0; already defined
%a2c1=0; already defined
```

```
a1c7=0;
a1c6=0;
a1c5=0;
%a1c3=0; already defined
%a1c1=0; already defined
%if statements for a3 chooses for round 4
if a3==6
    a3c7=0;
    a3c6=1;
    a3c5=nchoosek(a3,5);
    a3c4=nchoosek(a3,4);
    a3c1=nchoosek(a3,1);
elseif a3==5
    a3c7=0;
    a3c6=0;
    a3c5=1;
    a3c4=nchoosek(a3,4);
    a3c1=nchoosek(a3,1);
elseif a3==4
    a3c7=0;
    a3c6=0;
    a3c5=0;
    a3c4=1;
    a3c1=nchoosek(a3,1);
elseif a3==3 || a3==2 || a3==1
    a3c7=0;
    a3c6=0;
    a3c5=0;
    a3c4=0;
    a3c1=nchoosek(a3,1);
elseif a3==0
    a3c7=0;
    a3c6=0;
    a3c5=0;
    a3c4=0;
    a3c1=0;
else
    a3c7=nchoosek(a3,7);
    a3c6=nchoosek(a3,6);
    a3c5=nchoosek(a3,5);
    a3c4=nchoosek(a3,4);
    a3c1=nchoosek(a3,1);
end
%if statements for a1 chooses for round 4
if a1==6
    a1c7=0;
    a1c6=1;
    a1c5=nchoosek(a1,4);
elseif a1==5
    a1c7=0;
    a1c6=0;
    a1c5=1;
elseif a1<5
    a1c7=0;
    a1c6=0;
    a1c5=0;
else
    a1c7=nchoosek(a1,7);
    a1c6=nchoosek(a1,6);
    a1c5=nchoosek(a1,5);
end
p40=0;
p41=a1c7/nchoosek(30,7);
p42=(a1c6*a2c1)/nchoosek(30,7);
p43=(a1c6*a3c1+a1c5*a2c2+a3c7+(5/7)*a1c1*a3c6+(4/7)*a1c1*a2c2*a3c5+(1/21)*a2c2*a3c5+(8/21)*a1c1
p44=1-p40-p41-p42-p43;
e4=0*p40+1*p41+2*p42+3*p43+4*p44; %expected value for round 4
etemp=e1+e2+e3+e4;
if etemp>e
    e=etemp;
    a1max=a1;
    a2max=a2;
    a3max=a3;
```

```
                    a4max=a4;
                end

            end
        end
end
e
a1max
a2max
a3max
a4max
```

8.6 Code Runtimes

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
p1	1	4676.876 s	1359.136 s	
ismember	423860272	3317.732 s	944.788 s	
ismember>ismemberR2012a	423860272	2372.944 s	1177.665 s	
ismember>ismemberBuiltinTypes	423860272	1195.280 s	1195.280 s	

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
p2	1	18754.203 s	5670.234 s	
ismember	1612679024	13083.890 s	3681.910 s	
ismember>ismemberR2012a	1612679024	9401.980 s	4618.339 s	
ismember>ismemberBuiltinTypes	1612679024	4783.641 s	4783.641 s	

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
p2coin	1	22533.664 s	7128.819 s	
ismember	1612684480	15404.822 s	4378.598 s	
ismember>ismemberR2012a	1612684480	11026.224 s	5369.639 s	
ismember>ismemberBuiltinTypes	1612684480	5656.585 s	5656.585 s	

References

[1] Blizzard Entertainment. https://playhearthstone.com/en-us/game-guide/

[2] Gamepedia. https://hearthstone.gamepedia.com/Mana_curve

[3] Gamepedia. https://hearthstone.gamepedia.com/Meta

[4] John E. Freund. *John E. Freund's Mathematical Statistics, Sixth Edition.* Chapters 1, 2, 3, and 4

[5] Jos van der Geest. https://www.mathworks.com/matlabcentral/fileexchange/10064-allcomb-varargin-

[6] Wizards of the Coast. https://magic.wizards.com/en/gameplay/how-to-play