ABSTRACT

MATHEMATICAL PROGRAMMING AND MAGIC: THE GATHERING®

Alexander Esche, M.S.
Department of Mathematical Sciences
Northern Illinois University, 2018
Nathan Krislock, Co-Director
Alexander Garivaltis, Co-Director

In this paper mathematical programming techniques were used to determine the optimal strategy for playing Magic: The Gathering ®. Games with the cards Lightning Bolt, Mountain, and Vexing Devil were evaluated using the minimax algorithm to determine the winner when all information about the cards is assumed known to both players. Computation time was shortened through the use of an evaluation function, a random forest algorithm that had been trained on 1000 completed games. A winning percentage was established for each pair of decks where the number of creatures was less than eight. Using linear programming, the optimal mixed strategy was then calculated. By repeating the simulations, a standard deviation for the winning percentages was estimated. Techniques from robust optimization were then used to determine the optimal strategy under different possible variations. Last, an imperfect information player was constructed that made choices based on guessing the order of the cards in its deck and the composition of the opponent's deck, playing through the perfect information games of these guesses, and making the choice that won in most of these simulations. With decks of eight or fewer creatures, this imperfect information player played below or near a player who used an aggressive heuristic. When the number of possi-ble creatures was increased to 16, the imperfect information player's performance was better than the aggressive heuristic.

NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

MAY 2018

MATHEMATICAL PROGRAMMING AND MAGIC: THE GATHERING®

BY

ALEXANDER ESCHE

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICAL SCIENCES

Thesis Co-Directors:
Nathan Krislock
Alexandar Garivaltis

ACKNOWLEDGEMENTS

I would like to thank my parents for being great, professors for being awesome, and Melissa for being the best.

DEDICATION

Dedicated to my grandparents.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Review of Literature

Two articles, by Ward and Cowling [1] and by Cowling, Ward and Powley [2], have been written on the use of artificial intelligence in Magic: The Gathering®. The first of these compares the performance of a player who makes random choices, an agent that uses a reasonable heuristic, and a Monte Carlo player that assumes a random permutation for the player and opponent's deck and plays through using the heuristic. Then the best performing choice is chosen and the process repeated for the next possible choice. Ward and Cowling only consider one possible deck and only include creatures and lands. As well, the Monte Carlo procedure was only tested for card play selection, not whether a creature attacked or blocked. Their results point to some of the difficulties with defining heuristics, as the models that chose blockers based on the heuristic performed poorer than those that chose blockers randomly. In the second paper, the Monte Carlo tree search procedure is defined and used. They again use only creatures and lands, and the Monte Carlo player performed near but not better than an expert rule-based player. A third article relating to Magic: The Gathering®, by Chatterjee and Ibsen-Jensen [3], demonstrates that when using certain cards (none of which appear in this research), determining the legal combinations of blockers is a coNP-complete problem and a P problem otherwise.

Given the lack of publications on Magic®, it is beneficial to examine the use of artificial intelligence in other imperfect-information games. Angeris and Li [4] present a player for the

game of dominoes that uses a Monte Carlo minimax approach. The Monte Carlo approach performed better than a rule-based artificial intelligence player but performed poorly when challenged to a human player. Ginsberg's player for the game of bridge is believed to be at the level of an expert human player [5].

1.2 The Rules of Magic®

In Magic: The Gathering®, both players start with a deck with a minimum of 60 cards. (A simple explanation of these rules is available online at [6].) The decks are shuffled and each player draws the top seven cards from the deck into his or her hand, which he or she can look at but is not revealed to the opponent. Each player starts with 20 life; the goal is to reduce the opponent's life to 0. A turn has five phases - Beginning, Main I, Combat, Main II, and Ending. In the Beginning phase, the player whose turn it is draws the top card of the deck into his or her hand and untaps lands and creatures. If a player needs to draw a card but has no cards left in the deck, that player loses. A player may play cards that are in his or her hand. At the end of the turn, if a player has more than seven cards in his or her hand, he or she must discard cards into the graveyard until his or her hand has seven cards in it. The player chooses which cards to discard.

One type of card is a land; the specific land used in this project is a Mountain. A player may play one land per turn, but only on his or her turn during a Main phase. Lands enter untapped but may be tapped to pay for other cards. The other cards in this project have a cost of 1, so one Mountain needs to be tapped to pay for the card; higher cost cards would require multiple lands to be tapped. A second type of card is an instant. An instant can be played in a Main or Combat phase on either player's turn. The instant in this project is Lightning Bolt, which costs 1 and deals 3 damage to a creature or a player. Damage to a

player reduces his or her life by an equivalent amount. After an instant is played, it is placed in the graveyard; the graveyard is not used further in this project.

The third type of card is a creature. Creatures have power (attack) and toughness (defense); the creature used in this project is Vexing Devil, which costs 1 and has 4 power and 3 toughness. Creatures enter untapped and remain in play unless they are destroyed. If a creature takes damage greater than or equal to its toughness, it will be destroyed and moved to the graveyard. Damage on a creature that has not been destroyed is removed at the Ending phase. Creatures can attack on their owner's turn during a Combat phase if a turn has passed since they came into play.

The order of the Combat phase is attackers are declared, blockers are declared, and damage is resolved. When a player is being attacked by a creature, if the defending player has an untapped creature, that creature can block. If a creature is not blocked, it deals damage equal to its power to the defending player and that player's life goes down equivalently. If a creature is blocked, both creatures deal damage equal to their power to the other creature at the same time; if the creature has taken more damage than its toughness, it is destroyed. A creature cannot block more than one creature, but more than one creature can block a creature. This is important because several small creatures can block a larger creature to destroy it, but a large creature can only block one of several attacking small creatures. The attacker chooses how to distribute the damage to multiple blocking creatures. Since all of the creatures in this project are equivalent, there will not be cases of multiple creatures blocking a creature; a rational attacker will always destroy one of the blockers, making multiple blockers equivalent to a single blocker. Damage dealt in excess of a creature's toughness is ignored. An ability called trample, where this is relevant, is not included in this project. The Vexing Devil also has an additional rule: when a player plays it, the opposing player may pay 4 life and immediately send the Vexing Devil to the graveyard.

The formal rules for playing cards involves a process called the stack. For the purposes of this project, a succinct explanation of the stack is that when a player casts a spell (an instant or creature), that card is put on the stack and the other player has an opportunity to put cards on the stack. If the opposing player can cast a spell, that card is then on top of the first-cast card. When neither player adds a card to the stack, the top card on the stack comes out first, is cast, and its effects are resolved. Thus, the last spell cast is resolved first.

1.3 Decision Trees and Random Forests

The game playing in this project is simplified by using a random forest algorithm, which is an algorithm that is based on the decision tree algorithm. A brief introduction to the topic is presented in James et al. [7], and some useful explanations are presented here. A decision tree is an algorithm to generate a predictive model for a variable which can either be quantitative (regression decision tree) or qualitative (classifier decision tree). A regression decision tree is evaluated based on its residual sum of squares (RSS), the square of the difference between its prediction and the actual value. The prediction for a subset of the prediction space is the mean of the response for all of the observations in this subset. The decision tree that minimizes RSS is obtained through an algorithm known as recursive binary splitting. Starting from the entire prediction space, a variable and cutpoint are chosen so that the division of the prediction space on this variable leads to the greatest reduction in RSS. This process is then repeated on each of the divided intervals with the next best variable until a maximum tree length or other stopping criteria is reached. A classification decision tree is generated through a similar algorithm, except instead of evaluating based on RSS, the classification error rate, which is the percentage of observations in a subset that are not of the most common class in that subset, can be used. In practice, the classification

error is not sensitive enough for tree growing, and the Gini index, which is the sum of the classification error rate multiplied by one minus the classification error rate for each subset, or the entropy, which is the sum of negative the classification error rate multiplied by the logarithm of the classification error rate for each subset, is used.

An example of a decision tree is provided in Figure 1.1. If the input's value of the variable A14thL2 (variable names that would intuitively start with a number were preceded by an A, as variables are not allowed to start with a number in Julia or R) is greater than or equal to 55, the left path is chosen; otherwise, the right path is chosen. The top row of the node shows the dominant conclusion for that node. The second row shows the proportion of the training data that corresponds to each conclusion. For example, after going left at the first node, 68% of the training data that had A14thL2 $\geq$55 had an outcome of player one winning. Going right at this node, 59% of the training data that also had a value of A5thL1 $<$ 26 had an outcome of player two winning. The third row in the node shows the percentage of cases in the training data that fit into this node. For example, the terminal node in the bottom right accounts for 54% of the training data.

A random forest is an improved method over a simple decision tree with two additional features. First, a random forest generates multiple decision trees that use random samples of the data and then averages the responses. This approach is known as bagging. However, a random forest uses a second feature where it decorrelates the trees by allowing each tree to only use one of a random sample of the available predictors at each split. A new sample of predictors is taken at each split, and [7] recommends that the sample size be the square root of the available predictors.

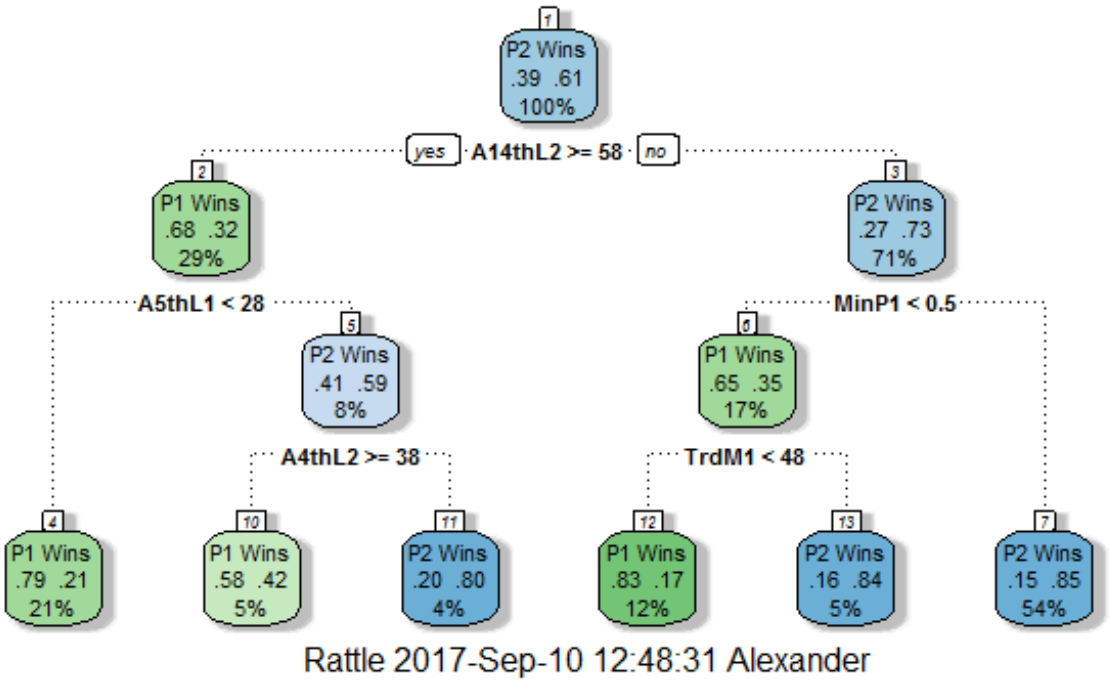

Figure 1.1: An example of a decision tree. Graphics developed by Rattle [8].

CHAPTER 2

METHODS

2.1 Perfect-Information Games

In the game considered in this project, there are two main areas where a player can make a choice. First, in what deck to choose and, second, how to play the game. After both players have chosen a deck, the decks are shuffled, leading to a combination of a random permutation of each. While in the actual game there is much information that is unknown to a player, the first stage of analysis in this project assumed that both players have complete knowledge of all of the cards in both decks and hands. The decision of what choice to make is then determined through the minimax algorithm. This algorithm states that a player will choose a choice that maximizes the value when the opponent chooses the choice that minimizes the value [9]. In these games, a choice where a player wins has value one and a choice where he or she loses has value negative one. In practice, this algorithm is run by performing a depth-first search of the possible choices. The game is played using default choices until an ending node where one of the players wins is reached. This result is returned to the previous choice and if the player making the choice sees it as a win, it moves back another choice. If the player making the choice sees it as a loss, that player will make the opposite choice. All of the choices in this phase of the project are constructed as binary choices, but ternary or more complicated choices could be easily accommodated. If a player has tried both possibilities for a choice and both return a loss, the loss is returned to the

previous choice. The result that is returned from the first choice can be interpreted as the perfect-information winner of the game.

In order to decrease the time required for calculating the winner of a game, two simplifications are made. The first is the use of an evaluation function. Shannon [10] defines an evaluation function, f(P), which can be applied to a position P and whose value determines to which category (won, lost, etc.) the position P belongs. For this project, the evaluation function is called once either player falls at or below 12 life. The function is a random forest model trained on 1000 completed games where one player started at or below 12 life. When the function returns its final winner, the input states to the evaluation function are then tested using the complete minimax, verifying that the evaluation function gave the correct result. If the evaluation function and minimax disagree, which happens in less than 5% of games, the result is not counted in the results.

Second, since a small number of games would still take a very long time to evaluate even using the evaluation function, a cap for using the evaluation function was established. If this cap is reached, the function returns the game as a draw and it is not included in the results. The determination of what cap presented the best combination of speed and accuracy was determined experimentally.

There is an immense number of possible starting permutations that can be generated for any pair of decks, but Monte Carlo simulation, where a large number of random simulations are tested and the results averaged, can be used to generate an estimate for the win percentage for a certain pair of decks. To reduce the time required to complete the evaluation of all possible combinations of decks, only decks with a number of each card that was divisible by four were tested. This led to 45 different decks. Because opposite pairs (for example, 0 creatures and 4 Lightning Bolts vs 4 creatures and 8 Lighting Bolts compared to 4 creatures and 8 Lighting Bolts vs 0 creatures and 4 Lightning Bolts) were expected to produce identical results (within expected statistical deviation), only cases where the second deck had

greater creatures than the first deck or equal creatures and greater or equal Lightning Bolts than the first were tested. This produced 1035 combinations that needed to be tested. The decks were tested with 200 games, with each player going first half of the time. The number of draws and results where the random forest did not match the minimax were recorded and not included in the win percentage.

2.2 A Brief Explanation of the Coding

The method that was used to determine the minimax winner was a depth-first search of the game tree. The base environment that the code ran in was a loop that continued as long as neither player was the winner, which moved the players through each phase of a turn and into the next turn after all of the phases had been completed. An important function was the legal moves function, which would look at the state of the players and determine which, if any, of the possible moves were available to a given player. The different moves are summarized in Table 2.1.

When a player has one or more legal moves available, the player will call the stepp function in order to perform the move. Before performing the move, copies of both players are made. The move is performed, and if neither player is defeated as a result, the same loop of phases and turns continues inside this stepp function. When a player is defeated, the number of the winning player is returned by the stepp function to the previous loop. The function looks at this returned value in relation to the player who made the legal move. If the player who made the move wins, then it is the desired outcome and it will return the value to the previous loop. If it is not the desired value, then the function will return the game to the state before the move was made and make the opposite move than it originally

Table 2.1: Legal Moves

Abbrev.	Description of Move
N	Do nothing
M	Play a Mountain
S	Play a creature
P	Pay 4 life to destroy a just-played creature
D	Destroy an untapped, not just-summoned creature with a Lightning Bolt
J	Destroy an untapped, just-summoned creature with a Lightning Bolt
T	Destroy tapped creature with a Lightning Bolt
L	Deal 3 damage to the opponent with a Lightning Bolt
A	Attack with a creature
B	Block with a creature

made (if it has not yet done so). If the stepp function returns an undesired result for both possible actions, it has exhausted its possibilities and must return the undesired result.

2.3 Linear Programming

Once winning percentages have been estimated for each combination of decks, these estimates can be constructed into a zero-sum matrix game. It is appropriate to consider this a zero-sum game because each win for a player is equivalent to a loss for the other player; a win percentage for a combination of decks can be seen as one minus the win percentage for the opposite player. The matrix C in the matrix game is constructed with the entry c_{ij} being the win percentage if player one chooses deck i and player two chooses deck j. The goal of player one is to find the strategy that maximizes the average payout; the goal of player two is to find the strategy that gives the lowest payout. Writing the strategies for

players one and two as vectors p and q, whose components add to one, each component p_i is the percentage of time player one chooses deck i and equivalently q_j is the percentage of time player two chooses deck j.

The process to solve this problem is well known and is summarized well by Fryer [11]. First, any row whose components are all smaller than another row is removed; any column whose components are all larger than another column is removed. Player one will never rationally choose one of these rows, and vice versa. This is known as removing dominated strategies. Then, consider the optimal strategy for player one, consisting of p_i for $i = 1, 2, \ldots, m$, where m is the number of decks after dominated decks have been removed. We can assume that this game has a value V, which is the average payout of the game if both players use their best strategy. The payout matrix for this project uses values between 0 and 1, with a value of 0.5 indicating no advantage for either player, and so, assuming not every entry of the payout matrix is zero, the value of the game is positive. The value of the game with this convention is the average winning percentage of player one if both players use their optimum strategies. If player one is using the optimum strategy, than the payout if player two uses deck j is greater than or equal to V because using only deck j might not be player two's optimal strategy. This is equivalent to the system

$$p_1 c_{1j} + p_2 c_{2j} + \cdots + p_m c_{mj} \geq V, \qquad j = 1, \ldots, m. \tag{2.1}$$

Letting $x_i = p_i/V$ and dividing $\sum_{i=1}^{m} p_i = 1$ by V, one obtains

$$\sum_{i=1}^{m} x_i = 1/V.$$

Then, because $V > 0$, the inequalities (2.1) become

$$x_1 c_{1j} + x_2 c_{2j} + \cdots + x_m c_{mj} \geq 1 \qquad j = 1, \ldots, m.$$

Player one wants to maximize V and thus minimize $1/V$, and the problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{m} x_i \\ \text{subject to} \quad & x_1 c_{1j} + x_2 c_{2j} + \cdots + x_m c_{mj} \geq 1, \quad j = 1, \ldots, m \\ & x_1 \geq 0, x_2 \geq 0, \cdots, x_m \geq 0 \end{aligned}$$

is a linear programming problem that can be solved by classical techniques such as the simplex method. Define $C = [c_{ij}]$ as an $m \times m$ matrix and e as the appropriately sized vector of all ones. By defining the matrix inequality

$$a \leq b, \quad a, b \in \mathbb{R}^{m \times n} \quad \Longleftrightarrow \quad a_{ij} \leq b_{ij}, \quad i = 1, \ldots, m, \; j = 1, \ldots, n$$

this problem can be expressed in matrix form as

$$C^T x \geq e.$$

A similar proof shows the dual problem, with q_i being the probability that player two chooses deck i, and $y_i = q_i/V$ can be formulated as

$$\begin{aligned} \underset{y}{\text{maximize}} \quad & \sum_{i=1}^{m} y_i \\ \text{subject to} \quad & y_1 c_{j1} + y_2 c_{j2} + \cdots + y_m c_{jm} \leq 1, \quad j = 1, \ldots m \\ & y \geq 0 \end{aligned}$$

or $Cy \leq e$.

2.4 Robust Optimization

One potential shortfall in using Monte Carlo approaches is that the values obtained may vary from the ideal nominal value. In order to get an estimate of the variability of the results, combinations of decks where the number of Lightning Bolts was divisible by eight (half of the decks) were tested against each other an additional four times each. The standard deviation of the winning percentage was calculated. Decks that were not tested had their standard deviation estimated with the mean of the decks with four fewer and four more Lightning Bolts than it.

To consider the optimum strategy under these uncertain conditions, techniques from robust optimization were used. The uncertainty set, a subset of $R^{m \times m}$, is defined as all of the values for the payout matrix that are considered possible. Robust optimization techniques determine the optimum solution that satisfies the given constraints for every payout matrix in the uncertainty set. A matrix game constructed from any of these payout matrices will have a value, so the function $V(C)$ can be defined as the value of the matrix game with the payout matrix C. If the c_{ij} decrease, $V(C)$ can decrease. A robust optimization technique will ensure that the strategy that is chosen will guarantee a winning percentage $V_{\min}$, the minimum of $V(C)$ over the uncertainty set. A strategy that is ideal when $V(C)$ is greater than $V_{\min}$ may result in payout that is less than $V_{\min}$ if the parameters change.

Two common cases of possible variation, known as uncertainty sets, can be easily handled. The first is that the parameters follow interval uncertainty, where each $c_{ij} \in [\hat{c}_{ij} - R\delta_{ij}, \hat{c}_{ij} + R\delta_{ij}]$; $\hat{c}_{ij}$ is a mean of c_{ij} over several observations, δ_{ij} is a standard deviation of c_{ij}, and R is a fixed number. We let $\hat{C}$ and Δ be the matrices of $\hat{c}_{ij}$ and δ_{ij} respectively. The Hadamard

product of ζ and a, $\zeta \circ a$, where $\zeta, a \in \mathbb{R}^{m \times n}$, is the $m \times n$ matrix where each component is the product of the respective components of ζ and a. That is, $[\zeta \circ a]_{ij} = \zeta_{ij} a_{ij}$. The uncertainty set for interval uncertainty can be written as

$$U_I = \{\hat{C} + \zeta \circ \Delta \mid |\zeta_{ij}| \leq R, \ \forall i, j\}.$$

The constraint $C^T x \geq e$ holds for all $C \in U_I$ if and only if

$$\hat{C}^T x + (\zeta^T \circ \Delta^T) x \geq e$$

holds for all ζ such that $|\zeta_{ij}| \leq R$ for all i, j, and this holds if and only if

$$\min_{|\zeta_{ij}| \leq R} \{[(\zeta^T \circ \Delta^T) x]_j\} \geq [e - \hat{C}^T x]_j, \qquad j = 1, \ldots, m.$$

Since $\Delta \geq 0$ and $x \geq 0$, an optimal ζ occurs when $\zeta_{ij} = -R$, $\forall i, j$. Thus, the robust linear programming problem can be written as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & e^T x \\ \text{subject to} \quad & (\hat{C} - R\Delta)^T x \geq e \\ & x \geq 0. \end{aligned}$$

A second possibility is ellipsoid variation. In this case, it can be assumed that $\beta \in \mathbb{R}^m$ is a vector whose Euclidean norm is less than some given radius R. The Euclidean norm, $\|\beta\|_2$, is defined as $\sqrt{\beta_1^2 + \cdots + \beta_m^2}$. For ease of notation, the columns of matrix C will be

denoted with c_j, with $C = [c_1 \cdots c_m]$. The components of each c_i are denoted by the row and columns of the original C matrix. For example,

$$c_1 = \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{m1} \end{bmatrix}.$$

Similar notation is used for Δ. The matrix B is the $m \times m$ matrix with columns β_j, $j = 1, \ldots, m$, and the individual components of B are numbered to match the numbering of C and Δ. The uncertainty set for ellipsoid variation in a single constraint can be written as

$$U_E = \{\hat{C} + B \circ \Delta \mid \|\beta_j\|_2 \leq R,\ j = 1, \ldots, m\}.$$

As above, $C^T x \geq e$ holds for all $C \in U_E$ if and only if

$$\min_{\|\beta_j\|_2 \leq R, j=1,\ldots,m} \{[(B \circ \Delta)^T x]_i\} \geq [e - \hat{C}^T x]_i, \qquad i = 1, \ldots, m,$$

holds. A solution to this problem is

$$B = [\hat{\beta}_1 \cdots \hat{\beta}_m],$$

where

$$\hat{\beta}_j = \arg\min_{\|\beta_j\|_2 \leq R} \{(\beta_j \circ \delta_j)^T x\},\ j = 1, \ldots, m.$$

We show in Appendix A that the optimum value for $\hat{\beta}_j$ is

$$\hat{\beta}_{ji} = \frac{-R\delta_{ji}x_j}{\sqrt{(\delta_{1i}x_1)^2 + \cdots + (\delta_{mi}x_m)^2}},\ i = 1, \ldots, m.$$

Thus,

$$\beta_j \circ \delta_j = \left[\frac{-R\delta_{1j}^2 x_1}{\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2}}, \cdots, \frac{-R\delta_{mj}^2 x_m}{\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2}} \right]^T.$$

It follows that

$$(\beta_j \circ \delta_j)^T x = \frac{-R\delta_{1j}^2 x_1^2}{\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2}} + \cdots + \frac{-R\delta_{mj}^2 x_m^2}{\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2}},$$

which implies

$$(\beta_j \circ \delta_j)^T x = -R\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2}.$$

Thus the robust optimization problem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & e^T x \\ \text{subject to} \quad & \hat{c}_j^T x - R\sqrt{(\delta_{1j}x_1)^2 + \cdots + (\delta_{mj}x_m)^2} \geq 1, j = 1, \ldots, m \\ & x \geq 0. \end{aligned}$$

A more general solution is presented by Ben-Tal and Nemirovski [12]. The proof requires use of conic duality, which is also presented in [12]. These results on conic duality apply to two cones. The first is the Lorentz cone, which is the subset of $\mathbb{R}^m$ where the Euclidean norm of the first $m-1$ components is less than or equal to the last component. The second is the nonnegative orthant, which is the subset of $\mathbb{R}^m$ where each component is nonnegative. The required results are that for a primal conic optimization problem,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & A_i x - b_i \in K_i, \quad i = 1, \ldots, n, \end{aligned}$$

where K_i is a Lorentz cone or nonnegative orthant, the equivalent dual problem is

$$\begin{aligned} \underset{\eta}{\text{maximize}} \quad & \sum_{i=1}^{n} b_i^T \eta_i \\ \text{subject to} \quad & \sum_{i=1}^{n} A_i^T \eta_i = c \\ & \eta_i \in K_i, \quad i = 1, \ldots, m. \end{aligned}$$

Weak duality states that the optimal value of the primal is greater than or equal to the optimal value of the dual. If the primal is strictly feasible and is bounded below, then the dual problem has an optimal solution and the optimal value of this solution equals the optimal value of the primal problem.

The method for finding the optimal solution to a robust optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & e^T x \\ \text{subject to} \quad & C^T x \geq e \quad \forall C \in U \end{aligned} \tag{2.2}$$

is as follows.

First, we define the function Vec from $\mathbb{R}^{m \times m}$ to $\mathbb{R}^{m^2}$, which maps a matrix ζ into the vector

$$\text{Vec}(\zeta) = \begin{bmatrix} \zeta_1 \\ \vdots \\ \zeta_m \end{bmatrix}.$$

Next, we define the function Diag from $\mathbb{R}^m$ to $\mathbb{R}^{m\times m}$, which maps a vector x to the diagonal matrix

$$\mathrm{Diag}(x) = \begin{bmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_m \end{bmatrix}.$$

Consider the uncertainty set

$$U = \{\hat{C} + \zeta \circ \Delta \mid P\mathrm{Vec}(\zeta) + r \in K\},$$

where K is the Lorentz cone or the nonnegative orthant. In addition, we assume strict feasibility holds; that is, there exists $\bar{\zeta}$ such that $P\mathrm{Vec}(\bar{\zeta}) + r \in \mathrm{int}(K)$, where $\mathrm{int}(K)$ is the interior of K. The constraint $C^T x \geq e$ holds for all $C \in U$ if and only if $(\hat{c}_i + \zeta_i \circ \delta_i)^T x \geq 1$, $i = 1, \ldots, m$, holds for all ζ such that $P\mathrm{Vec}(\zeta) + r \in K$. We rewrite $\zeta_i \circ \delta_i + \hat{c}_i$ as $\alpha_i(\zeta)$, where

$$\alpha_i(\zeta) = A_i \mathrm{Vec}(\zeta) + \hat{c}_i,$$

and

$$A_i = \begin{bmatrix} 0_{m\times m(i-1)} & \mathrm{Diag}(\delta_i) & 0_{m\times m(m-i)} \end{bmatrix},$$

where $0_{p\times q}$ represents a $p \times q$ matrix of zeros. An x is thus robust feasible if and only if

$$\alpha_i^T(\zeta)x \geq 1,\ i = 1, \ldots, m, \quad \forall \zeta \text{ such that } P\mathrm{Vec}(\zeta) + r \in K.$$

It is also robust feasible if and only if the solution to

$$\min_{\tau,\zeta}\{\tau \mid \tau \geq \alpha_i^T(\zeta)x - 1, P\mathrm{Vec}(\zeta) + r \in K\} \tag{2.3}$$

is nonnegative for all i.

Consider one particular i. As stated above, if the problem (2.3) has a nonnegative optimal value and is strictly feasible, then its dual problem has an optimal solution (λ, η) with the same nonnegative optimal value. We define $\sigma = \mathrm{Vec}(\zeta)$. The constraints of problem (2.3) are

$$\tau - (A_i^T x)^T \sigma \geq \hat{c}_i^T x - 1, \quad P\sigma + r \in K$$

and the objective function is

$$\begin{bmatrix} 1 \\ 0_{m^2} \end{bmatrix}^T \begin{bmatrix} \tau \\ \sigma \end{bmatrix}.$$

Therefore, the dual problem is

$$\begin{aligned} \underset{\lambda,\eta}{\text{maximize}} \quad & (\hat{c_i}^T x - 1)\lambda - r^T \eta \\ \text{subject to} \quad & \lambda = 1 \\ & (-A_i^T x)\lambda + P^T \eta = 0_{m^2} \\ & \lambda \geq 0 \\ & \eta \in K. \end{aligned}$$

Thus, x is robust feasible if and only if there exist η, such that

$$\begin{aligned} \hat{c_i}^T x - 1 - r^T \eta &\geq 0 \\ -A_i^T x + P^T \eta &= 0_{m^2} \\ \eta &\in K. \end{aligned}$$

Thus, problem (2.2) is solved by finding the vector x, which minimizes $e^T x$ for x that satisfies the above dual system for each column of C. That is, the robust optimal solution is found by solving

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & e^T x \\ \text{subject to} \quad & (\hat{c_i}^T x - 1) - r^T \eta_i \geq 0 \\ & -A_i^T x + P^T \eta_i = 0,\ i = 1, \ldots, m \\ & \eta_i \in K,\ i = 1, \ldots, m. \end{aligned}$$

2.5 Uniqueness of Solutions

The methods in the previous two sections find optimal solutions to the given problem but do not guarantee that the results are unique. Additional methods can be used to determine if this is the case. For linear programming problems, a method by Mangasarian [13] is computationally tractable. For a primal linear programming problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & C^T x - b \geq 0 \end{aligned}$$

with optimal solution $\bar{x}$ and dual optimal solution $\bar{u}$, the sets K and L are defined as $K = \{i \mid c_i^T x = b_i, u_i > 0\}$ and $L = \{i \mid c_i^T x = b_i, u_i = 0\}$. Defining C_K and C_L as the submatrices formed from the columns of C indexed by K and L, respectively, $\bar{x}$ is the unique optimal solution if and only if there exists no solution to the system

$$\begin{aligned} C_K^T x &= 0 \\ C_L^T x &\geq 0 \end{aligned}$$

except the trivial solution $x = 0$. It should be noted that this result does not assume that x is nonnegative, and so this must be introduced as the constraint $Ix \geq 0$, where I is the identity matrix.

For the more complex robust optimization cases, the uniqueness of solutions is not as easy to verify. Since the objective function is convex, and it is being minimized over a convex set, the set of optimal solutions is convex.

2.6 Imperfect-Information Games

While finding the perfect-information minimax solution is an interesting problem, the players in the game of Magic: The Gathering® do not have access to this information. Thus, a player who does not require perfect-information was designed. This player, whenever it has multiple possible moves, generates a random deck for itself and the opponent. A player knows what deck he or she is using but does not know the deck the opponent is using. However, a player knows that the opponent has at least as many Lightning Bolts and creatures as the opponent has already played. Once the random decks are generated, the player plays through the minimax solution after each possible move. This procedure is repeated a certain number of times. The move that returns the most wins for the acting player is chosen. Unlike the perfect-information game, once a choice is made, the player is not able to change it based on future results.

One of the other main differences between this and the perfect-information games is that in the perfect-information game, when a player went below 12 life, the result from the random forest was used. Because the perfect-information player was used to train the random forest, it cannot be assumed to be accurate for the imperfect-information player. As well, since the player does not need to consider past choices once a choice is made, there

is less computational need to use a simplifying model. Thus, the game continues once a player falls below 12 life. However, the imperfect-information player continues to generate perfect-information games from the conditions of the game in order to make choices. If one of the players is below 12 life when the perfect-information simulation starts, an unmodified model would immediately output the result of the random forest algorithm. This might not be the best predictor, as the game might be dissimilar to any that were in the training set, since the training set was generated from games as soon as a player went below 12 life. When one player is near 12 life, it is likely that there is a similar game in the training set. However, it would be impossible for a game where a player was at 4 life to be in the original training set. To remedy this, similar training sets were generated for when a player went below 8 life and when a player went below 4 life. Then, when performing the perfect-information simulations, the player would use the random forest trained on games where the player's life was below the player's current life. The random forest trained on when a player went below 4 life was used from when a player went below 8 life until the end of the game.

To test the effectiveness of this player, two players that play by consistent heuristics were designed. The first, an aggressive player, plays its Lightning Bolts against the opponent as quickly as possible. It attacks whenever possible, never blocks, and unless it has fewer than 4 life, will pay the life to keep the opponent's creature from coming into play. The second, a defensive player, always destroys an opponent's creature with a Lightning Bolt if possible. It won't target the opponent itself with a Lightning Bolt unless it has enough in its hand to win the game. It attacks when it has more creatures than the opponent and blocks whenever possible. It will pay the 4 life to prevent an opponent's creature from coming into play if it doesn't have a Lightning Bolt in its hand.

CHAPTER 3

RESULTS

3.1 Perfect-Information Games

The first phase of the project was to create a player who plays the game with perfect information, that is, knowing the positions of all the cards in the deck and what cards are in the opponent's hand. The program was written in the Julia language [14]. During simulation, it was found that games where both players have relatively large numbers of creatures took longer to complete. Experiments were first performed with the players started at 12 life; the number of creatures was decreased until 1000 games could be completed in under 24 hours on one node of the Gaea computing cluster [15], an HP SL380s G7 equipped with two Intel 2.66 GHz 6-core processors and 72 GB RAM. It was found that eight or fewer creatures fulfilled this requirement.

The player would use a random forest algorithm to predict the winner when either player fell below 12 life. Training data was generated by starting 1000 games with random combinations of decks (with eight or fewer creatures), playing until one player had 12 or fewer life, recording the values of the predictors, and then playing the game to completion and recording the winner. Since 60 predictors were being used, seven, approximately the square root of 60, random features were selected for each step of the tree creation, as is recommended by [7]. The random forest functions of the package DecisionTree.jl [16] were used to generate the trees. To determine the number of trees to use in each forest, the out-of-sample accuracy and time for evaluation of 200 new random combinations were evaluated. The time

for completion was calculated on the author's personal computer, a Toshiba Satellite with a 2.13 GHz Intel processor and 4 GB of RAM. The results can be seen in Figures 3.1 and 3.2.

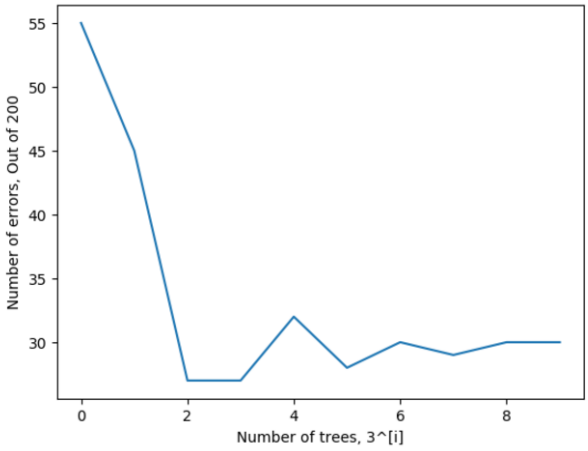

Figure 3.1: The accuracy of the random forest with increasing number of trees.

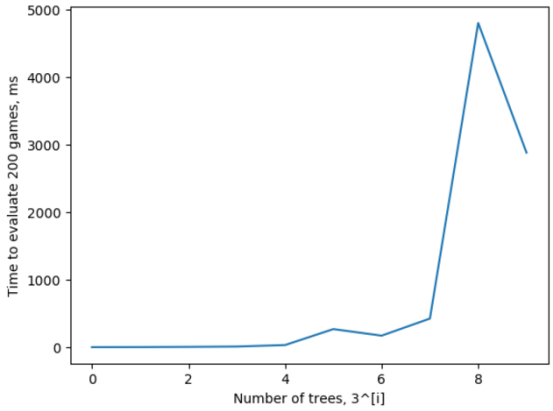

Figure 3.2: The time to evaluate the random forest with increasing number of trees.

An accuracy of 85% and a short evaluation time suggested that the number of trees should be between 9 and 27. To determine which to choose, the hypotheses that increasing the number of trees from 9 to 27 increased the accuracy and that increasing the number of trees from 9 to 27 increased the amount of time needed to evaluate the random forest were tested. For 20 different sets of 200 randomly generated games whose results had been computed with the minimax algorithm, the number of incorrect predictions the random forest algorithm made and the time it took to complete the evaluation were measured. The results were then tested for statistical significance.

To test whether the accuracy improved between 9 and 27 trees, the null and alternative hypotheses were $H_0 : \mu_9 = \mu_{27}$ and $H_A : \mu_9 < \mu_{27}$. The test statistic

$$t = \frac{(\bar{X} - \bar{Y}) - (\mu_X - \mu_Y)}{\frac{S_X^2}{n} + \frac{S_Y^2}{m}}$$

$$= \frac{37.8 - 34.25}{17.43/20 + 30.69/20} \approx -1.475$$

had

$$r = \frac{(\frac{S_X^2}{n} + \frac{S_Y^2}{m})^2}{\frac{(S_X^2/n)^2}{n-1} + \frac{(S_Y^2/m)^2}{m-1}}$$

$$= \frac{(17.43/20 + 30.69/20)^2}{\frac{(17.43/20)^2}{19} + \frac{(30.69/20)^2}{19}} \approx 35.32,$$

i.e., 35 degrees of freedom. This probability

$$p(t < -1.475 \mid \mu_9 = \mu_{27}) = .075$$

is greater than $\alpha = 0.05$, and thus the null hypothesis that the accuracies were equal was not rejected.

To test whether the evaluation time increased between 9 and 27 trees, the null and alternative hypotheses were $H_0 : \mu_9 = \mu_{27}$ and $H_A : \mu_9 < \mu_{27}$. The test statistic

$$t = \frac{4.77 - 11.55}{.037/20 + 1.079/20} \approx -121.51$$

had

$$r = \frac{(.037/20 + 11.55/20)^2}{\frac{(.037/20)^2}{19} + \frac{(1.079/20)^2}{19}} \approx 2188.48$$

i.e., 2188 degrees of freedom. This probability

$$p(t < -121.51 \mid \mu_9 = \mu_{27}) = 1.53 \times 10^{-975}$$

is less than $< \alpha = 0.05$, and thus the null hypothesis was rejected. Using 27 trees instead of 9 failed to provide a significant increase in accuracy but resulted in a significant increase in evaluation time, and so nine trees were used. The summary statistics are provided in Table 3.1.

Table 3.1: Results with Different Number of Trees in Random Forest

	Num. Incorrect (Out of 200)	Time (ms)
9 Trees Sample Mean	37.8	4.77
9 Trees Sample Variance	17.43	.037
27 Trees Sample Mean	34.25	11.55
27 Trees Sample Variance	30.69	1.079
Number of Samples	20	20

To improve the accuracy of the simulation, after a game was completed, the conditions that were input into the final random forest were tested using minimax evaluation. Even when only considering decks with eight or fewer creatures, the occasional game would take several hours to complete. To prevent this, the number of times that the random forest was

used was recorded during the simulation. If the number of evaluations exceeded a certain cap, the program returned a draw. The effect of different caps on the number of draws returned and evaluation time can be seen in Table 3.2; 500,000 evaluations was chosen as a cap due to its moderate time to completion and small number of draws.

Table 3.2: Performance with Different Caps on Number of Games Evaluated

Cap of Evaluations	Number of Draws (Out of 100)	Time to Complete 100 games (s)
5,000	20	152
50,000	10	209
500,000	3	366
5,000,000	0	519

The different combinations were then tested, for over a week, using a median of six nodes on the Gaea computing cluster. The result for each combination was saved as a .csv file after it was completed. Some combinations would not complete the required 200 games within the 24 hours a node of Gaea could be requisitioned for; these nodes are recorded in Table 3.3. As a result of this, games where the number of Lightning Bolts and creatures summed to 12 or less were not included in the subsequent analysis. It was also found that games which had no lands resulted in only draws (although these decks had no possible way to win), so these decks were not considered in the analysis.

The winning percentages for each combination were read from each CSV and combined into a payoff matrix, which can be seen in Appendix B. By solving this matrix game, first by a program of the author's construction and then by the JuMP package [17], it was found that the optimal play was to use no creatures and 40 Lightning Bolts 20% of the time, eight creatures and 28 Lightning Bolts 40% of the time, and eight creatures and 32 Lightning Bolts 40% of the time.

Table 3.3: Games That Did Not Finish

Combinations
0C8Lv4C4L
0C8Lv8C0L
0C8Lv8C4L
4C8Lv8C0L
4C8Lv8C4L
8C0Lv8C0L
8C0Lv8C4L
8C0Lv8C12L
8C4Lv8C4L

After the subsequent analysis was completed, the conditions used to train the random forest algorithm were relaxed, allowing for 100 games to finish in 24 hours on a Gaea node instead of 1000. This allowed for a model to be trained on games with up to 16 creatures. Only decks with more than 12 non-land cards (of which there were 55) were considered, but even among this subset there were some combinations that would not complete 200 games within 24 hours. It was found that the verification that the random forest gave, the correct result through minimax analysis was the bottleneck step; thus, for these 23 combinations (out of 3025) that did not finish, the winning percentage was calculated by completing 200 games without verifying the random forest. The results of the linear programming are presented in Table 3.4.

Table 3.4: Optimum Strategies with 16 Creatures

Deck	Percentage in Optimum Strategy
0C36L	16%
4C32L	8%
8C32L	5%
12C20L	14%
12C24L	7%
12C28L	17%
16C16L	6%
16C20L	16%
16C24L	10%

3.2 Robust Optimization

The mean standard deviation of the results was 2.3%, with a maximum of 7.8%. The first case tested was the interval variation. Considering as a reference the normal distribution, deviations of two to three standard deviations could be considered to have an estimated likelihood of 1-5% and thus might be reasonable cases to consider for robust optimization. Thus, interval variation where $|\zeta_{ij}| \leq 3$ and $|\zeta_{ij}| \leq 2$ was considered. The results can be seen in Table 3.5. However, these deviations are not necessarily realistic, as the extreme cases in the uncertainty set allow for all of the deviations to occur at their minimum simultaneously, which would be unlikely if the deviations were independent.

A second uncertainty set to consider is where the Euclidean norm of the deviations is less than a certain radius. Thus, the Euclidean norm of the vectorization of ζ was considered. Again considering that two to three standard deviations is a reasonable range to consider, one obtains the results seen in Table 3.5.

Table 3.5: Optimum Strategies For Eight Creatures (in Percent)

Deck	No Variation	$\|\zeta_{ij}\| \leq 2$	$\|\zeta_{ij}\| \leq 3$	$\|\zeta\|_2 \leq 2$	$\|\zeta\|_2 \leq 3$
0C28L	0	0	0	2	3
0C32L	0	0	0	5	5
0C36L	0	0	0	10	5
0C40L	20	51	46	18	15
0C44L	0	0	0	0	2
4C24L	0	0	7	0	3
4C28L	0	0	0	6	8
4C32L	0	5	12	6	11
4C36L	0	0	0	5	6
4C40L	0	15	14	1	3
8C20L	0	0	0	1	4
8C24L	0	0	0	8	8
8C28L	40	17	17	13	11
8C32L	40	12	3	19	12
8C36L	0	0	0	5	5

3.3 Uniqueness of Solutions

For the first linear programming result, the optimal solution was tested to see which components of the vector $C^T x$ equaled one. This occurred for three components, those where x_i was nonzero. The constraint that x_i was greater than zero provided an additional 30 columns where x_i equaled 0. In all of these cases, the corresponding dual variables were positive, and so these were the columns of C_K. C_K was thus a 33×33 matrix, and since its rank was 33, the rows were independent and the only solution to the system $C_K^T x = 0$ is the trivial solution. Thus, the solution is unique.

When examining the solution under robust conditions, the two players are no longer considering the same conditions, and so it is unlikely that they will have the same solution, as was seen in the original case. For the case where $|\zeta_{ij}| \leq 2$, there were six columns where $C^T x = 1$ and 27 rows where $x = 0$, and the corresponding dual variables were again all positive. C_K again had rank 33, and so the solution is unique. The case where $|\zeta_{ij}| \leq 3$

provides similar results. Again, there were six rows where $C^T x = 1$ and 27 rows where $x = 0$, with all of the signs of the dual variables positive. In using a numerical solver, a variable is rarely set to exactly 0; a value of 10^{-10} can be reasonably assumed to be zero. The smallest dual variable was approximately 5×10^{-5}. If this dual solution is considered zero, the solution is no longer unique, but there are five orders of magnitude between this value and those considered 0.

3.4 Imperfect-Information Games

The effectiveness of the imperfect-information game was measured by playing it against the two heuristic players in 200 games, with each player going first half of the time. The decks were chosen randomly. This process was repeated 20 times to establish a consistent average. Against the aggressive heuristic, the imperfect-information player won an average of 48% of the time with a standard deviation of 4.3%. Against the defensive heuristic, the imperfect-information player won an average of 62% of the time with a standard deviation of 5.8%. When faced against each other, the defensive heuristic beat the aggressive heuristic an average of 34% of the time with a standard deviation of 7.3%. The result that the imperfect-information player was an even match with the aggressive heuristic player was not desired; the imperfect-information player uses much more computational power to make its decisions. Cases where up to 16 creatures were allowed in the deck were tested to see if the imperfect-information player performed relatively better as the number of creatures improved. Under these conditions, the imperfect-information player won 59% of the time against the aggressive heuristic, with a standard deviation of 8.9%.

CHAPTER 4

CONCLUSION

In this work, advances have been made in understanding optimal play for Magic: The Gathering®. Optimal play for players when both have perfect information of the game can be determined by the minimax algorithm, which was proved possible for some of the possible deck pairings. Computation time was reduced by the use of a random forest algorithm. By considering the deck choice of each player as a strategy in a matrix game, with a payout of the winning percentage of the first player for the pair of decks, linear programming techniques could be used to determine the optimal deck choice for a player. Robust optimization techniques provided a solution when the player considered the standard deviation of the winning percentage and chose a strategy that would guarantee the highest minimum winning percentage. Finally, an imperfect-information player was constructed that made choices by simulating perfect information games from the current state of the game.

4.1 Future Work

This work may introduce the game of Magic® to more researchers and thus broaden its presence in mathematical research. The work as presented suggests a pair of hypotheses that could be tested. The first is whether the results of the linear and robust programming that were obtained remain consistent when every pair of decks (not just those divisible by 4) is considered. This would take a quadratic increase in computation time. The second is finding a method to train the evaluation function for more cases (with more creatures than 16). Since the computation of the minimax result (in order to train the random forest) has reached its computational limit, a new evaluation function must be discovered in order to make calculations of winning percentages feasible. A possible solution is the introduction of an unsupervised learning algorithm. Unlike a supervised learning algorithm, an unsupervised learning algorithm is not given set cases and variables to make its predictive models. Unsupervised learning has recently been used to create the most advanced player of the board game Go (Silver et al. [18]). Further analysis will consider work on a larger variety of cards. It should be noted that in the current rules of Magic®, players are restricted to having a maximum of four copies of any non-land card. However, the most restrictive tournament format of Magic®, Standard, contains over 1000 cards. There are approximately 10^{100} combinations of choosing 60 different cards, which is a subset of the possible legal decks which could be constructed. Thus, the methods developed in this work would probably be best suited for finding the relative performance of decks already known to be competitive rather than identifying a global optimum.

REFERENCES

[1] C.D. Ward and P.I. Cowling. *Monte Carlo Search Applied to Card Selection in Magic: The Gathering*. Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on

[2] Peter I. Cowling, Colin D. Ward, and Edward J. Powley. *Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering*. IEEE Transactions on Computational Intelligence and AI in Games (Volume: 4, Issue: 4, Dec. 2012)

[3] Krishnendu Chatterjee and Rasmus Ibsen-Jensen. *The Complexity of Deciding Legality of a Single Step of Magic: the Gathering*. ECAI. 2016

[4] Guillermo Angeris and Lucy Li. *CS221 Project Final: DominAI*. `https://web.stanford.edu/ guillean/dominai.pdf`

[5] Matthew L. Ginsberg. *GIB: Imperfect Information in a Computationally Challenging Game.*. Journal of Artificial Intelligence Research 14 (2001): 303-358.

[6] Magic: The Gathering Quick Start Guide, `media.wizards.com/2014/docs/EN_M15_QckStrtBklt_LR_Crop.pdf`

[7] Gareth James et al. *An Introduction to Statistical Learning*. Vol. 112. New York: Springer, 2013.

[8] Graham J. Williams. *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*. Springer, 2011.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Egnlewood Cliffs 25 (1995): 27.

[10] Claude Shannon. *Programming a Computer for Playing Chess*. Philosophical magazine 41 (1950): 256-275.

[11] Michael John Fryer. *An Introduction to Linear Programming and Matrix Game Theory*. John Wiley and Sons Inc, 1978.

[12] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization*. SIAM, 2001.

[13] O. L. Mangasarian *Uniqueness of Solution in Linear Programming*. Linear algebra and its applications 25 (1979), 151-162.

[14] Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah. *Julia: A Fresh Approach to Numerical Computing*. SIAM Review 59 (2017): 6598.

[15] Computing Facilities - Gaea,
`http://niu.edu/hpc/facilities/gaea.shtml`

[16] Decision Tree Classifier and Regressor,
`https://github.com/bensadeghi/DecisionTree.jl`

[17] Iain Dunning, Joey Huchette, and Miles Lubin. *JuMP: A Modeling Language for Mathematical Optimization*. SIAM Review 59 (2017): 295-320.

[18] David Silver et al. *Mastering the Game of Go Without Human Knowledge*. Nature 550.7676 (2017): 354-359.

APPENDIX A

PROOF OF THE ROBUST OPTIMAL VARIATIONS FOR ELLIPTICAL VARIATION OF A SINGLE VARIABLE

The problem that needs to be solved is

$$\begin{aligned} \underset{\beta_i}{\text{minimize}} \quad & (\beta_i \circ \delta_i)^T x \\ \text{subject to} \quad & \|\beta_i\|_2 \leq R. \end{aligned}$$

Expanding the objective and constraints, the problem is

$$\begin{aligned} \underset{\beta_i}{\text{minimize}} \quad & \beta_{1i}\delta_{1i}x_1 + \cdots + \beta_{mi}\delta_{mi}x_m \\ \text{subject to} \quad & \sqrt{\beta_{1i}^2 + \cdots + \beta_{mi}^2} \leq R. \end{aligned}$$

This minimization problem on β_i is a convex optimization problem. Slater's condition, that there exists a strictly feasible point, which in this case is equivalent to there existing β_i such that $\|\beta_i\|_2 < R$, can be shown to hold by the point $\beta_i = 0$. Thus there is a solution to the KKT conditions, which for this system are

$$\begin{aligned} & \lambda \geq 0, \\ & \lambda(\|\beta_i\|_2 - R) = 0, \\ & \delta_{ji}x_j + \lambda\frac{\beta_{ji}}{\|\beta_i\|_2} = 0, \ j = 1, \ldots, m, \\ & \|\beta_i\|_2 \leq R. \end{aligned}$$

If $\delta_{ji}x_j = 0$ for $j = 1, \ldots, m$, then any feasible β_i will be optimal. By inspection, if any $\delta_{ji}x_j \neq 0$, then λ cannot equal 0, which means $\|\beta_i\|_2 = R$. Thus

$$\delta_{ji}x_j + \lambda\frac{\beta_{ji}}{R} = 0, \qquad j = 1, \ldots, m.$$

Solving for β_{ji},

$$\beta_{ji} = \frac{-\delta_{ji}x_j R}{\lambda}, \qquad j = 1, \ldots, m.$$

Substituting these values into an expression for the norm of β_i,

$$\|\beta_i\|_2 = R = \sqrt{\beta_{1i}^2 + \beta_{2i}^2 + \cdots + \beta_{mi}^2}.$$

Thus,

$$R = \sqrt{\frac{(\delta_{1i}x_1)^2R^2}{\lambda^2} + \cdots + \frac{(\delta_{mi}x_m)^2R^2}{\lambda^2}}.$$

Factoring out common terms, we get

$$R = \frac{R}{\lambda}\sqrt{(\delta_{1i}x_1)^2 + \cdots + (\delta_{mi}x_m)^2}.$$

Then, after canceling an R from both sides, we obtain

$$1 = \frac{1}{\lambda}\sqrt{(\delta_{1i}x_1)^2 + \cdots + (\delta_{mi}x_m)^2},$$

and thus

$$\lambda = \sqrt{(\delta_{1i}x_1)^2 + \cdots + (\delta_{mi}x_m)^2}.$$

Thus, the expression for β_{ji} is

$$\beta_{ji} = \frac{-R\delta_{ji}x_j}{\lambda} = \frac{-R\delta_{ji}x_j}{\sqrt{(\delta_{1i}x_1)^2 + \cdots + (\delta_{mi}x_m)^2}}.$$

APPENDIX B

PAYOUT MATRIX FOR PERFECT-INFORMATION GAMES (WINNING PERCENTAGE FOR ROW PLAYER)

	0C16L	0C20L	0C24L	0C28L	0C32L	0C36L	0C40L	0C44L	0C48L	0C52L	0C56L
0C16L	0.5	0.4596	0.42	0.4	0.42	0.4171	0.4192	0.4343	0.4632	0.4595	0.4671
0C20L	0.5404	0.5	0.4545	0.48	0.4623	0.4422	0.4444	0.4623	0.4764	0.4809	0.5819
0C24L	0.58	0.5455	0.5	0.485	0.48	0.475	0.4619	0.4848	0.4869	0.5304	0.6167
0C28L	0.6	0.52	0.515	0.5	0.495	0.5	0.4924	0.4923	0.5231	0.5294	0.6878
0C32L	0.58	0.5377	0.52	0.505	0.5	0.495	0.495	0.5026	0.4974	0.544	0.6524
0C36L	0.5829	0.5578	0.525	0.5	0.505	0.5	0.5101	0.5051	0.5404	0.5408	0.6633
0C40L	0.5808	0.5556	0.5381	0.5076	0.505	0.4899	0.5	0.5127	0.5327	0.5663	0.705
0C44L	0.5657	0.5377	0.5152	0.5077	0.4974	0.4949	0.4873	0.5	0.51	0.5381	0.6289
0C48L	0.5368	0.5236	0.5131	0.4769	0.5026	0.4596	0.4673	0.49	0.5	0.5228	0.6387
0C52L	0.5405	0.5191	0.4696	0.4706	0.456	0.4592	0.4337	0.4619	0.4772	0.5	0.5469
0C56L	0.5329	0.4181	0.3833	0.3122	0.3476	0.3367	0.295	0.3711	0.3613	0.4531	0.5
4C12L	0.4555	0.4301	0.3436	0.3655	0.3081	0.365	0.3737	0.3788	0.4643	0.4922	0.5593
4C16L	0.5181	0.4619	0.5	0.4545	0.4091	0.435	0.4171	0.4694	0.4819	0.5137	0.6
4C20L	0.5897	0.5101	0.4848	0.48	0.4774	0.445	0.4899	0.4772	0.4974	0.5269	0.6455
4C24L	0.5707	0.5327	0.5176	0.5025	0.4899	0.495	0.4924	0.5025	0.5202	0.5469	0.6888
4C28L	0.593	0.545	0.5226	0.5025	0.5025	0.5	0.495	0.5	0.5077	0.5436	0.6754
4C32L	0.585	0.5431	0.51	0.5051	0.4975	0.4874	0.505	0.5051	0.5153	0.599	0.6935
4C36L	0.5556	0.5561	0.5226	0.5101	0.5128	0.495	0.4975	0.5025	0.5101	0.5431	0.6649
4C40L	0.55	0.5381	0.5236	0.513	0.5	0.4949	0.4925	0.5075	0.51	0.5226	0.6923
4C44L	0.5464	0.4974	0.5104	0.5053	0.5103	0.4948	0.5	0.4974	0.5	0.5104	0.634
4C48L	0.511	0.5193	0.5	0.4844	0.4301	0.433	0.4667	0.4573	0.4824	0.5103	0.5798
4C52L	0.4717	0.4096	0.3957	0.2818	0.3125	0.298	0.3368	0.3622	0.385	0.4333	0.5027
8C8L	0.4865	0.4404	0.3231	0.3523	0.3005	0.3568	0.3608	0.4133	0.449	0.4897	0.5914
8C12L	0.5365	0.5103	0.4615	0.4422	0.39	0.4221	0.4365	0.4694	0.4764	0.5	0.644
8C16L	0.5412	0.5077	0.4874	0.4847	0.4673	0.46	0.495	0.4724	0.4845	0.5368	0.658
8C20L	0.5657	0.5126	0.5101	0.4975	0.4824	0.4949	0.5	0.4798	0.5051	0.5241	0.6753
8C24L	0.593	0.55	0.51	0.5	0.5	0.4949	0.5051	0.4949	0.5127	0.5455	0.6954
8C28L	0.5879	0.5377	0.5176	0.5075	0.5	0.5	0.505	0.5101	0.5152	0.5736	0.6735
8C32L	0.5685	0.5327	0.505	0.5025	0.505	0.5075	0.495	0.5101	0.5176	0.5829	0.69
8C36L	0.5436	0.5309	0.5127	0.5026	0.5025	0.5127	0.5	0.5	0.5153	0.5606	0.6513
8C40L	0.5368	0.5211	0.5132	0.5	0.4924	0.4724	0.4798	0.4874	0.4975	0.5279	0.6205
8C44L	0.5519	0.5169	0.4863	0.4656	0.4866	0.4322	0.4343	0.4541	0.4697	0.5233	0.5825
8C48L	0.4702	0.4483	0.3478	0.2947	0.2857	0.3472	0.3351	0.3608	0.349	0.4216	0.5054

	4C12L	4C16L	4C20L	4C24L	4C28L	4C32L	4C36L	4C40L	4C44L	4C48L	4C52L
0C16L	0.5445	0.4819	0.4103	0.4293	0.407	0.415	0.4444	0.45	0.4536	0.489	0.5283
0C20L	0.5699	0.5381	0.4899	0.4673	0.455	0.4569	0.4439	0.4619	0.5026	0.4807	0.5904
0C24L	0.6564	0.5	0.5152	0.4824	0.4774	0.49	0.4774	0.4764	0.4896	0.5	0.6043
0C28L	0.6345	0.5455	0.52	0.4975	0.4975	0.4949	0.4899	0.487	0.4947	0.5156	0.7182
0C32L	0.6919	0.5909	0.5226	0.5101	0.4975	0.5025	0.4872	0.5	0.4897	0.5699	0.6875
0C36L	0.635	0.565	0.555	0.505	0.5	0.5126	0.505	0.5051	0.5052	0.567	0.702
0C40L	0.6263	0.5829	0.5101	0.5076	0.505	0.495	0.5025	0.5075	0.5	0.5333	0.6632
0C44L	0.6212	0.5306	0.5228	0.4975	0.5	0.4949	0.4975	0.4925	0.5026	0.5427	0.6378
0C48L	0.5357	0.5181	0.5026	0.4798	0.4923	0.4847	0.4899	0.49	0.5	0.5176	0.615
0C52L	0.5078	0.4863	0.4731	0.4531	0.4564	0.401	0.4569	0.4774	0.4896	0.4897	0.5667
0C56L	0.4407	0.4	0.3545	0.3112	0.3246	0.3065	0.3351	0.3077	0.366	0.4202	0.4973
4C12L	0.5	0.4715	0.4337	0.3434	0.3687	0.3655	0.402	0.4213	0.4583	0.5128	0.5902
4C16L	0.5285	0.5	0.4472	0.4422	0.4121	0.4495	0.4623	0.4394	0.4847	0.5	0.6409
4C20L	0.5663	0.5528	0.5	0.5	0.4824	0.44	0.48	0.5051	0.4949	0.526	0.6398
4C24L	0.6566	0.5578	0.5	0.5	0.4925	0.4925	0.5	0.4899	0.5101	0.5492	0.7031
4C28L	0.6313	0.5879	0.5176	0.5075	0.5	0.5	0.505	0.505	0.5127	0.5699	0.6447
4C32L	0.6345	0.5505	0.56	0.5075	0.5	0.5	0.5025	0.5152	0.5152	0.5765	0.7296
4C36L	0.598	0.5377	0.52	0.5	0.495	0.4975	0.5	0.4925	0.5152	0.5758	0.7437
4C40L	0.5787	0.5606	0.4949	0.5101	0.495	0.4848	0.5075	0.5	0.5178	0.5427	0.6495
4C44L	0.5417	0.5153	0.5051	0.4899	0.4873	0.4848	0.4848	0.4822	0.5	0.5155	0.6042
4C48L	0.4872	0.5	0.474	0.4508	0.4301	0.4235	0.4242	0.4573	0.4845	0.5	0.5784
4C52L	0.4098	0.3591	0.3602	0.2969	0.3553	0.2704	0.2563	0.3505	0.3958	0.4216	0.5
8C8L	0.5027	0.4583	0.4301	0.3782	0.3298	0.3731	0.3846	0.3979	0.4663	0.4919	0.5519
8C12L	0.5206	0.49	0.4712	0.4121	0.4061	0.4422	0.4495	0.4639	0.4747	0.4948	0.5902
8C16L	0.5477	0.5077	0.5	0.4822	0.4596	0.4495	0.47	0.4848	0.4897	0.5436	0.6753
8C20L	0.5939	0.5202	0.5153	0.505	0.4925	0.4925	0.4772	0.4848	0.5152	0.57	0.6579
8C24L	0.601	0.5427	0.5075	0.5	0.4975	0.5075	0.4975	0.5075	0.5255	0.602	0.6884
8C28L	0.6396	0.5431	0.52	0.5176	0.5101	0.5	0.4975	0.4975	0.5255	0.5729	0.702
8C32L	0.6289	0.5202	0.5176	0.5127	0.5102	0.5075	0.5051	0.5152	0.5226	0.5729	0.7085
8C36L	0.5677	0.5231	0.5152	0.5051	0.4949	0.4975	0.495	0.4975	0.5204	0.5596	0.6443
8C40L	0.5183	0.5288	0.4949	0.4694	0.4596	0.4899	0.4874	0.4874	0.495	0.5226	0.6492
8C44L	0.5109	0.5163	0.4579	0.4521	0.4249	0.4093	0.402	0.4619	0.4592	0.5128	0.6203
8C48L	0.4124	0.3904	0.3622	0.3161	0.2917	0.2887	0.3179	0.3158	0.4346	0.4706	0.5087

	8C8L	8C12L	8C16L	8C20L	8C24L	8C28L	8C32L	8C36L	8C40L	8C44L	8C48L
0C16L	0.5135	0.4635	0.4588	0.4343	0.407	0.4121	0.4315	0.4564	0.4632	0.4481	0.5298
0C20L	0.5596	0.4897	0.4923	0.4874	0.45	0.4623	0.4673	0.4691	0.4789	0.4831	0.5517
0C24L	0.6769	0.5385	0.5126	0.4899	0.49	0.4824	0.495	0.4873	0.4868	0.5137	0.6522
0C28L	0.6477	0.5578	0.5153	0.5025	0.5	0.4925	0.4975	0.4974	0.5	0.5344	0.7053
0C32L	0.6995	0.61	0.5327	0.5176	0.5	0.5	0.495	0.4975	0.5076	0.5134	0.7143
0C36L	0.6432	0.5779	0.54	0.5051	0.5051	0.5	0.4925	0.4873	0.5276	0.5678	0.6528
0C40L	0.6392	0.5635	0.505	0.5	0.4949	0.495	0.505	0.5	0.5202	0.5657	0.6649
0C44L	0.5867	0.5306	0.5276	0.5202	0.5051	0.4899	0.4899	0.5	0.5126	0.5459	0.6392
0C48L	0.551	0.5236	0.5155	0.4949	0.4873	0.4848	0.4824	0.4847	0.5025	0.5303	0.651
0C52L	0.5103	0.5	0.4632	0.4759	0.4545	0.4264	0.4171	0.4394	0.4721	0.4767	0.5784
0C56L	0.4086	0.356	0.342	0.3247	0.3046	0.3265	0.31	0.3487	0.3795	0.4175	0.4946
4C12L	0.4973	0.4794	0.4523	0.4061	0.399	0.3604	0.3711	0.4323	0.4817	0.4891	0.5876
4C16L	0.5417	0.51	0.4923	0.4798	0.4573	0.4569	0.4798	0.4769	0.4712	0.4837	0.6096
4C20L	0.5699	0.5288	0.5	0.4847	0.4925	0.48	0.4824	0.4848	0.5051	0.5421	0.6378
4C24L	0.6218	0.5879	0.5178	0.495	0.5	0.4824	0.4873	0.4949	0.5306	0.5479	0.6839
4C28L	0.6702	0.5939	0.5404	0.5075	0.5025	0.4899	0.4898	0.5051	0.5404	0.5751	0.7083
4C32L	0.6269	0.5578	0.5505	0.5075	0.4925	0.5	0.4925	0.5025	0.5101	0.5907	0.7113
4C36L	0.6154	0.5505	0.53	0.5228	0.5025	0.5025	0.4949	0.505	0.5126	0.598	0.6821
4C40L	0.6021	0.5361	0.5152	0.5152	0.4925	0.5025	0.4848	0.5025	0.5126	0.5381	0.6842
4C44L	0.5337	0.5253	0.5103	0.4848	0.4745	0.4745	0.4774	0.4796	0.505	0.5408	0.5654
4C48L	0.5081	0.5052	0.4564	0.43	0.398	0.4271	0.4271	0.4404	0.4774	0.4872	0.5294
4C52L	0.4481	0.4098	0.3247	0.3421	0.3116	0.298	0.2915	0.3557	0.3508	0.3797	0.4913
8C8L	0.5	0.4769	0.4235	0.3545	0.3949	0.3351	0.3807	0.3476	0.4359	0.5026	0.5668
8C12L	0.5231	0.5	0.4796	0.4518	0.4322	0.4205	0.4596	0.4433	0.4646	0.5312	0.617
8C16L	0.5765	0.5204	0.5	0.4824	0.4724	0.4772	0.4848	0.4798	0.5128	0.5	0.6237
8C20L	0.6455	0.5482	0.5176	0.5	0.5	0.49	0.4975	0.505	0.5152	0.5745	0.6753
8C24L	0.6051	0.5678	0.5276	0.5	0.5	0.4899	0.5	0.5025	0.5076	0.5729	0.7121
8C28L	0.6649	0.5795	0.5228	0.51	0.5101	0.5	0.4975	0.5075	0.5303	0.533	0.72
8C32L	0.6193	0.5404	0.5152	0.5025	0.5	0.5025	0.5	0.5075	0.5178	0.5533	0.6923
8C36L	0.6524	0.5567	0.5202	0.495	0.4975	0.4925	0.4925	0.5	0.5153	0.5431	0.6345
8C40L	0.5641	0.5354	0.4872	0.4848	0.4924	0.4697	0.4822	0.4847	0.5	0.5408	0.623
8C44L	0.4974	0.4688	0.5	0.4255	0.4271	0.467	0.4467	0.4569	0.4592	0.5	0.5683
8C48L	0.4332	0.383	0.3763	0.3247	0.2879	0.28	0.3077	0.3655	0.377	0.4317	0.5