Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

ANALYSIS OF GAMEPLAY STRATEGIES IN HEARTHSTONE: A DATA SCIENCE APPROACH

by
Connor W. Watson

In recent years, games have been a popular test bed for AI research, and the presence of Collectible Card Games (CCGs) in that space is still increasing. One such CCG for both competitive/casual play and AI research is Hearthstone, a two-player adversarial game where players seeks to implement one of several gameplay strategies to defeat their opponent and decrease all of their Health points to zero. Although some open source simulators exist, some of their methodologies for simulated agents create opponents with a relatively low skill level. Using evolutionary algorithms, this thesis seeks to evolve agents with a higher skill level than those implemented in one such simulator, SabberStone. New benchmarks are propsed using supervised learning techniques to predict gameplay strategies from game data, and using unsupervised learning techniques to discover and visualize patterns that may be used in player modeling to differentiate gameplay strategies.

ANALYSIS OF GAMEPLAY STRATEGIES IN HEARTHSTONE: A DATA SCIENCE APPROACH

by
Connor W. Watson

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Data Science

Department of Computer Science

May 2020

APPROVAL PAGE

ANALYSIS OF GAMEPLAY STRATEGIES IN HEARTHSTONE: A DATA SCIENCE APPROACH

Connor W. Watson

Dr. Amy K. Hoover, Thesis Advisor — Date
Assistant Professor, New Jersey Institute of Technology

Dr. Usman W. Roshan, Thesis Co-Advisor — Date
Associate Professor, New Jersey Institute of Technology

Dr. Senjuti Basu Roy, Committee Member — Date
Assistant Professor, New Jersey Institute of Technology

BIOGRAPHICAL SKETCH

Author: Connor W. Watson

Degree: Master of Science

Date: May 2020

Undergraduate and Graduate Education:

- Master of Science in Data Science,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Bachelor of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2018

Major: Data Science

Presentations and Publications:

A. Bhatt, S. Lee, F. de Mesentier Silva, C. W. Watson, J. Togelius and A. K. Hoover, "Exploring the Hearthstone Deck Space," *Proceedings of the Foundations of Digital Games*, 2018.

Games have benchmarked AI methods since the inception of the field, with classic board games such as Chess and Go recently leaving room for video games with related yet different sets of challenges. The set of AI problems associated with video games has in recent decades expanded from simply playing games to win, to playing games in particular styles, generating game content, modeling players, etc. Different games pose very different challenges for AI systems, and several different AI challenges can typically be posed by the same game... Collectible card games are relatively understudied in the AI community, despite their popularity and the interesting challenges they pose.
While the successful tactics of a Hearthstone player are at least partly determined by the deck, for many decks there are several different playstyles possible, and individual players will often prefer one playstyle over another. Can we create AI agents that can learn and recreate these playing styles, not only playing to win but doing so in the style of a particular player?

Amy K. Hoover

ACKNOWLEDGMENT

I would like to thank my advisor Dr. Amy K. Hoover for working with me every step of the way. My initial work with her and the team in "Exploring the Hearthstone Deck Space" helped guide me towards my interest in research. If it wasn't for that experience, I wouldn't be writing this paper. Dr. Hoover, this was truly an opportunity neither one of us will forget; I am incredibly grateful for my time as your first Master's student at NJIT. Additional thanks to my Co-Advisor Dr. Usman Roshan. His classes were essential for me navigating through my program, and if it weren't for his availability and answering e-mails at odd times of the day/night, there would have been many concepts (and memes) that I would have missed out on. I would like to thank the members of my Committee for their support and witnessing my defense. Thank you to IST for the equipment for my defense.

Additional credits to the SabberStone developers for maintaining the core software used to support this research, and Fernando de Mesentier Silva for helping with code as well. Another special thank you to Dr. Barry Cohen, Dr. Abdul-Rahman Itani, Dr. Senjuti Basu Roy, and Dr. Baruch Schieber for allowing me to work as a Teaching Fellow while completing my degree. To all of my Instructors throughout the years who have guided me along in my education, I thank you as well. Special shoutouts to all of the faculty that have supported me throughout my time at NJIT.

Big thank you to Nick Young for helping me through my first internship; I've gained very valuable industry experience working with you and the team, learning about data engineering and visualization. Thank you to Eileen Nguyen for your dedication, love and support every day. Thank you to Purav Patel, Chidanand Khode, and all my dear friends for continuing a supportive learning environment throughout the years. Thank you to my family for your continued support as well. And to everyone else close to me, thank you (you know who you are).

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF ALGORITHMS

Chapter **Page**

LIST OF DEFINITIONS

Agent	An AI player using a gameplay strategy and deck of cards in Hearthstone.
Aggro	An aggressive playstyle which seeks to finish the game quickly by swiftly attacking the opponent.
Cartesian Space	The Cartesian coordinate system which specifies points uniquely in a plane via a set of numerical coordinated.
Control	A defensive playstyle which seeks to play a longer drawn out game by maintaining control of the board.
Dimensionality Reduction	The process of reducing the number of columns of a data set via feature selection or feature extraction.
Evolutionary Algorithm	Population based optimization algorithm inspired by biological evolution.
Feature Vector	An observation of a structured data set with a number of columns (features).
Gameplay Strategy	The types of decisions that a player will use each turn.
Game State	A snapshot of a game (Hearthstone) which consists of the cards in hand, on the board, health for each player, etc.
Game Tree	A tree of possible game states which can be reached from the current game state.
Mana	The primary resource in Hearthstone which players must use to play cards.

Mana Cost	The amount of mana a card (or Hero Power) costs.
Mana Curve	The distribution of the number of cards that have a particular mana cost.
Matchup	A game or set of games between two players.
Metagame	Popular cards and decks at a given point in time.
Mirror Match	A game or set of games between two players using the same deck and/or scoring function.
Quality Diversity	Population based stochastic algorithm which generates a diverse collection of quality solutions to a problem.
Scoring Function	Functions used to evaluate a game state and determine the best decision based on visible game features.
Supervised Learning	Learning a function which maps input to output based on labeled data.
Unsupervised Learning	Learning a function which draws inferences on unlabeled data.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CCG	Collectible Card Game
CMA-ES	Covariance Matrix Adaptation Evolutionary Strategy
CMA-ME	Covariance Matrix Adaptation MAP-Elites
HS	Hearthstone
MAP-Elites	Multi-dimensional Archive of Phenotypic Elites
MCTS	Monte Carlo Tree Search
ML	Machine Learning
PCA	Principal Components Analysis
QD	Quality Diversity

CHAPTER 1

INTRODUCTION

While board games like Chess, Checkers, and Go are classic benchmarks in AI, they share many properties. For instance, they are each two-player, turn-taking, adversarial board games where both players have perfect information about the state of the board and moves available to themselves and the opponents. Moves are deterministic rather than stochastic, meaning that the next state of the board is determined based on the current state, and whichever move the player chooses to make next [45]. The games are also zero-sum, meaning that the loss of one player is balanced by gain for the other. While Checkers was one of the first studied for its relative simplicity compared to Chess and Go [46], a primary differentiating characteristic is the total number of possible states, where the state space of even Checkers has approximately 500 billion different states or possible positions (i.e., 5×10^{20}) [48]. Proposed as an alternative benchmark is the card game Hearthstone [7] (described in more detail in Section 2.2), which differs from these board games in its non-determinism, the size of the state space, and the exponential branching factor of the game tree [32].

Algorithms designed to beat these games largely rely on building and searching a game tree of possible positions for the game. An example of a game tree is shown for Tic-Tac-Toe in Figure 1.2 with positions defined in Figure 1.1. Player X has placed an X in the upper right hand corner, and Player O is searching for possible locations to place its O. This partial tree shows three positions that Player O could place its O, but note that there are a total of eight possible positions considering that one is taken by the X. Complete game trees would show the states after all possible moves by both players, until the final possible moves are achieved. Two popular human

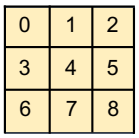

Figure 1.1 Tic-Tac-Toe is played on boards of size 3 × 3. Each positioned is numbered for reference.

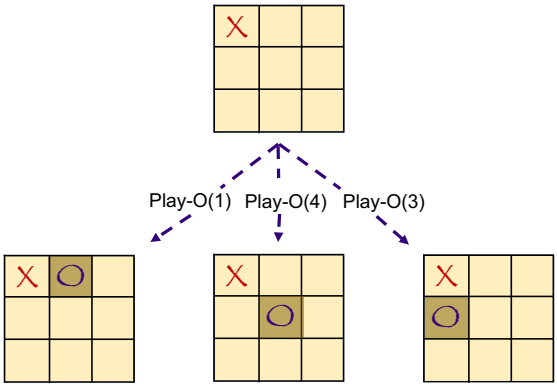

Figure 1.2 The initial state for Player O is shown at the top and all possible moves played next to the X played by Player X shown in the next turn. Note that there are five additional moves available to Player 0, but only three of eight are shown.

competitive algorithms Deep Blue [10] and AlphaGo [52] both rely on building and pruning such trees to decide the next moves to play.

Inherent to picking the best actions in a game based on a game tree is determining values of particular game states. While Monte Carlo Tree Search (MCTS) approaches state value assignment based on how likely they are to result in a win for the player, the complexity of Hearthstone often requires making narrow assumptions about the players' choices to perform well due to the stochastic nature of the game [53]. An alternative to MCTS is developing scoring functions to determine the values of states. While heuristics developed by hand can be effective [20], their success relies

on expert knowledge and often perform poorly when played with a different set of initial conditions.

Experiments in this thesis test the performance of two popular heuristics provided by the Hearthstone simulator SabberStone [1] that are designed to emulate a general form of basic gameplay strategies in Hearthstone, and compare their performance relative to two learned with a technique called Covariance Matrix Adapation MAP-Elites [23]. Beyond comparing their win rates, through supervised and unsupervised approaches, additional experiments aim to uncover whether the particular scoring function can be determined from game state information available at each turn. The hope is that such an approach can form a basis for new approaches to build effective algorithms for agents to play and win games of Hearthstone.

1.1 Hypotheses

By analyzing the performance of different simulated Hearthstone agents, the data should show a clear difference in how two opposing strategies approach the game, with the added expectation that evolved heuristics via CMA-ME should perform better than the curated ones included with SabberStone. In addition, by comparing supervised learning models trained on player data, a model should confidentally predict which of two heuristics an agent is using based on game data. Finally, by using unsupervised dimensionality reduction techniques, the data should result in clear geometric separation in Cartesian space of the different types of agents.

[1] https://github.com/HearthSim/SabberStone

CHAPTER 2

BACKGROUND

This chapter details collectible card games in general and Hearthstone in particular. In addition, this chapter also includes a discussion of previous AI approaches to Hearthstone, and briefly introduces a recently proposed algorithm included in the experiments of this thesis.

2.1 Collectible Card Games

Although different types of card games have existed since the 1300s [21], the first collectible card game called Magic: the Gathering was created in 1993 by Richard Garfield [55]. CCGs are significantly different because players cannot own all of the cards at once by making a simple purchase. Instead, players need to buy booster packs, which are packets containing a small randomized subset from the entire set of cards. This could have been an incentive for players to continue to buy more packs to find more powerful cards, or cards that have synergy with their current collection of cards. Not only that, but it has become a fun way to compete with friends; players construct their own decks of 60 cards out of the over 9000 cards in the set of all cards, and with more cards being added each year, the space of decks is large enough that it is quite uncommon for two players to share the exact same deck of cards [58]. Since Magic: the Gathering, many other card games have been created with similar yet unique rule sets such as Yu-Gi-Oh!, Pokémon, and Cardfight!! Vanguard. These games see increased tournament level play, with players competing professionally all over the world. Although these games have physical cards, the popularity of electronic video games have also created digital versions of the games. One advantage of online versions of card games is that with the right software, one can capture the turn by turn plays that happen within a particular game.

Figure 2.1 A game state in the first turn of a game of Hearthstone, showing some of the different pieces for the player and opponent.

2.2 Hearthstone

To gain an understanding of the test bed for this research, this section introduces the game, how it is played, and what the different pieces of the game mean in reference to each player. Because Hearthstone is not open-source, one developer-friendly simulator called SabberStone is discussed as well as the gameplay AI used for the experiments in this thesis. This section briefly introduces other research areas related to AI methods for Hearthstone, as well as outlining the interconnected research challenges of generating decks and gameplay strategies [32].

Figure 2.2 A game state in the first turn of a game of Hearthstone, showing some of the different pieces for the player and opponent. This includes a card on the current player's side of the board, as well as a timer (the burning rope animation) indicating time is running out for their turn.

2.2.1 Introduction to Hearthstone: The Basics

Hearthstone is another collectible card game, created by Blizzard Entertainment in 2014, designed as a two-player adversarial CCG for solely online digital play between two players on unique devices (mobile or desktop). Shown in Figure 2.1, the hero controlled by the player on the current device (surrounded by the light blue square), is at the bottom of the *board*, shown as the portrait of one of the *hero characters* in the game named Gul'dan. The board contains both players' hero characters, and the common space between the two players to play cards. Before starting a game, players must create their own *decks* of cards which can be assigned to one of ten *hero classes* available in the game, each with their own name and image [1]. A player's turn ends after 75 seconds or when the player presses the *End Turn* button highlighted in green on the right-hand side of the board shown in Figure 2.1. As shown in Figure 2.2, the current player is running out of time, so they are given a text warning, as well as a rope with a burning animation from left to right, horizontally in the middle of the screen, until it reaches the End Turn button. A *round* in Hearthstone concludes after both players have completed a turn. Games can last a maximum of 44 rounds and a turn, or 89 turns in total. In Hearthstone, there are also two types of formats that a player can use to build a deck and play games with other players. In the *Wild* format, all cards from the beginning of the game's creation are allowed for play. In the *Standard* format, players are only allowed to play with all Basic and Classic cards, as well as a subset of cards from roughly the past two years of packs from a point in time, which are changed in a rotating format throughout each year as new packs are released. In this thesis, the cards and decks used represent a subset of the Standard format active at the time when the "Rise of Shadows" booster pack was released.

2.2.2 Resource Management: Mana

One aspect of Hearthstone which is unique from other games is resource management. Turns are partially controlled by the resources owned by each player called *Mana Crystals*. Most player actions in the game are initiated by cards, which have a given *mana cost* to play it from the hand. As shown using circles in Figure 2.1, both the player and the opponent have visible numbers showing their Mana Crystal summaries; on the left is the player's *available mana*, or simply *mana*, and on the right is the player's *permanent Mana Crystals*, or simply *maximum mana*. At the start of the game, players start with one permanent Mana Crystal on their first turn, and gain an additional Mana Crystal added to their *maximum mana* on each of their successive turns until they reach the maximum of ten. Thus, in the first turn, players can only play cards which cost one mana, and by turn ten players can play cards that cost ten mana. At the start of each player's turn, the *turn player's* available mana available is refilled to the maximum amount available for that player's turn.

Resource management in Hearthstone is a partial contributor to a player's turn by turn gameplay. The progression of available mana per turn dictates the types of cards that a type of player can use. For example, on turn one, a player is only allowed to play one card that costs one mana. On turn two, a player is allowed to play one card that costs two mana, or two cards that cost one mana each. This excludes the presence of cards which cost zero mana, since this will not decrease the player's amount of available mana. These possibilities continue to grow with each turn as the amount of available mana for the turn player increases each turn.

2.2.3 Classes, Heroes

There are ten classes that a player can choose from to construct their deck. As shown in Figure 2.3, each class has a set of hero characters that a player can choose from to construct their decks [1]. While some heroes are gained through purchases or through

events, each class has one available hero by default for the player to access. Each of the heroes have a name, and a unique *Hero Power* which they can use once per turn at the cost of two mana. Each Hero Power provides a unique effect shown in Table 2.1. The classes of cards and heroes together form an expanded lore based on the game World of Warcraft, also created by Blizzard.

Table 2.1 Hero Powers[a]

Class	*Hero Power Name*	*Hero Power Text*
Demon Hunter	Demon Claws	+1 Attack this turn.
Druid	Shapeshift	+1 Attack this turn. +1 Armor.
Hunter	Steady Shot	Deal 2 damage to the enemy hero.
Mage	Fireblast	Deal 1 damage.
Paladin	Reinforce	Summon a 1/1 minion.
Priest	Lesser Heal	Restore 2 Health.
Rogue	Dagger Mastery	Equip a 1/2 Dagger.
Shaman	Totemic Call	Summon a random Totem.
Warlock	Life Tap	Draw a card. Take 2 damage.
Warrior	Armor Up!	Gain 2 armor.

[a]Hero Powers are unique to each class, but all cost two mana to play. Some may naturally fit a particular gameplay strategy like that of the Hunter class, which when played does two damage to the enemy hero. However, others like the Warlock's Lifetap (i.e., costs 2 mana to draw an additional card from the player's deck) may fit other types of strategies different than those which use the Hunter's Hero Power.

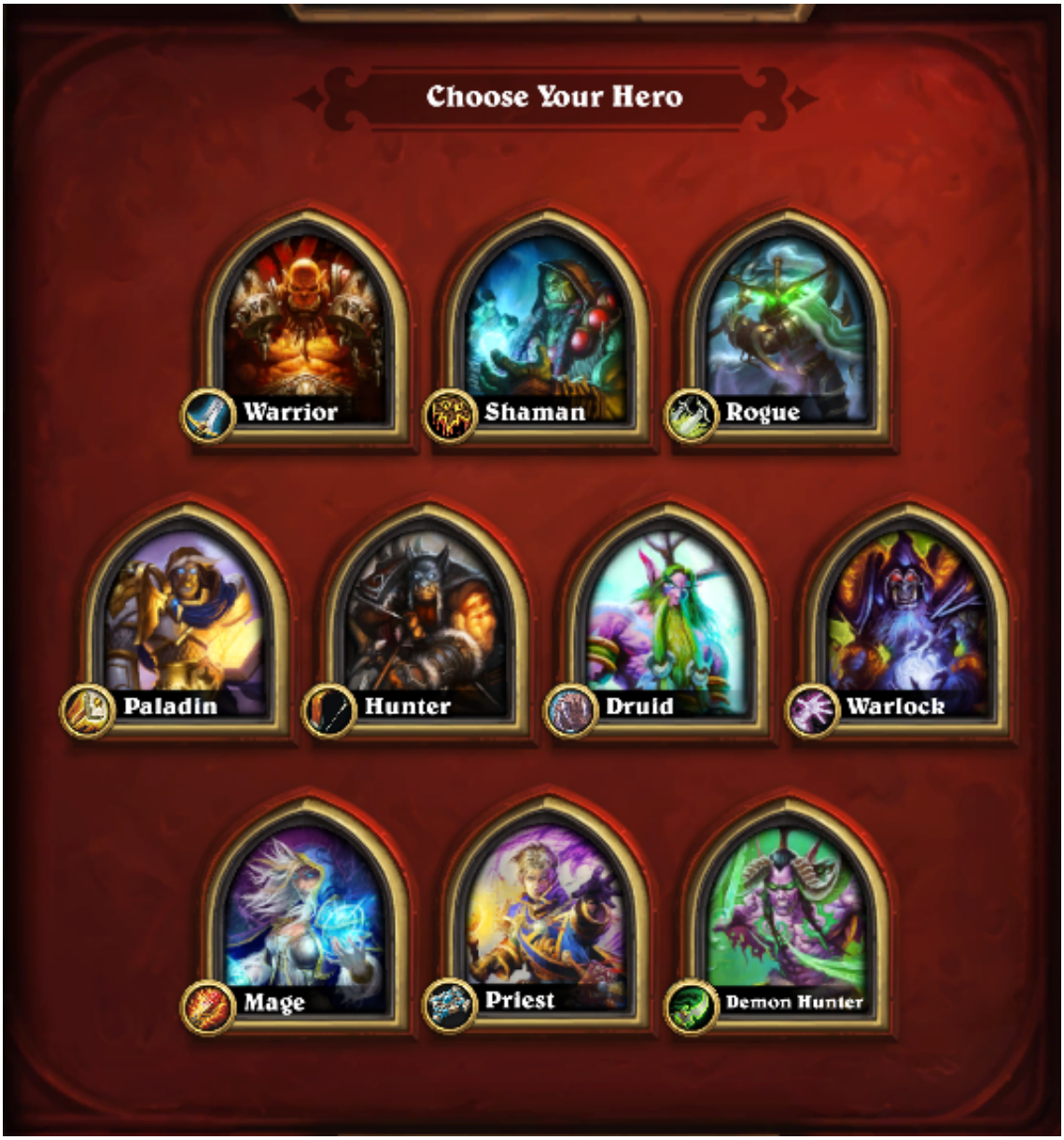

Figure 2.3 The ten hero classes available for players to choose from, and their default hero characters.

2.2.4 Decks of Cards

Players may register for free accounts and can retrieve cards completely free of charge, but they also have the option to purchase additional booster packs for the chance to get more cards faster (hence *collectible* card game). Upon successful registration, players have access to a set of 143 *Basic* cards to customize and build their decks of cards, which rarely changes unless game designers determine the set should be changed. In addition to the Basic cards, each class has a set of class specific cards; each class has ten collectible cards in the Basic set, and even more in the additional booster packs. Players have access to all ten of the classes, and when creating a deck must select one as the basis of the deck. This then limits the cards they can put into the deck to the class-agnostic Basic cards, and any class-specific cards the player owns. Once a class is selected, a player can utilize any combination of those to create a deck of 30 cards, with the exception that generally, a player can only put a maximum of two copies of a card in a deck.

Each card has a *rarity*, and the rarity influences the relative likelihood of receiving that card, and the amount of those types of cards a player can include in a deck. Players can include two copies of a card if its rarity is free, common, rare, or epic. However, a player can only include one copy of each legendary card in the deck. Although the increasing rarity decreases the likelihood of receiving the card, rarity is not necessarily indicative of the usefulness of a card. When building a deck, players should keep in mind the cards themselves, and how they play together. Details on the cards selected in this thesis are found in Appendix E, and the amount of common cards between the decks are found in Appendix F.

2.2.5 Types of Cards

There are four categories of cards in Hearthstone, each of which shown in Figure 2.4. Common to all playable cards is the *mana cost*, which is displayed in the top left-hand

Figure 2.4 Four types of playable cards in Hearthstone shown left to right as follows: minion, spell, weapon, and hero.

Figure 2.5 The minion named "Voidwalker" is a one-mana cost card (shown in the top left). It's attack power is one (shown in the bottom right), and its total Health is three (shown in the bottom right).

Figure 2.6 The spell named "Soulfire" is a one-mana cost card (shown in the top left). It has an effect which when played, allows the player to deal four damage to a target on the field, but also forces the player to discard a random card.

Figure 2.7 The hero card named "Bloodreaver Gul'dan" allows the player to upgrade their chosen hero when played from the hand. It's a ten-mana cost card (shown in the top left). When played, the player's chosen hero gains five additional armor (shown in the bottom right), and activates the Battlecry effect shown in the lower region of the card. The player's Hero Power is also changed, but this is not depicted as part of the card.

Figure 2.8 The weapon card named "Overlord's Whip" is a three-mana cost card (shown in the top left) which gives the player's hero character two additional attack points (shown in the bottom left). It also has four durability (shown in the bottom right), and when decreased to zero, the weapon will be destroyed. This weapon also has an additional effect, and as long as it's equipped to the hero, when the player plays a minion, it is automatically damaged by one point.

corner next to the card's portrait. Some cards even have a mana cost of zero. Cards can be referred to as an *N-mana cost card*, where N is an integer that refers to the card's mana cost.

Minions are cards that once played, become movable units that can do damage to other minions and to the opposing hero. An example minion is shown in Figure 2.5. The amount of damage a minion can do is shown in the bottom left-hand corner of the card and is called its *attack power*. Each minion can sustain a finite amount of damage defined by its *Health* (or *health points*) displayed in the bottom right-hand corner. Some minions have *effects* which are shown in the middle of the card underneath the card's name. There are different types of effects like *Battlecry* which only activates once when the minion is played from a player's hand, *Deathrattle* which activates once when the minion has died, or *Taunt* which is a continuously active effect which forces the opponent attacks to target this minion for attacks before any choosing other targets for attacks.

Spells are another type of card which do not have attack power or health points. An example spell is shown in Figure 2.6. Spells do cost a number of mana, but are a type of card which when activated, activates its effect, and then disappears. Two exceptions exist with spells, as some can have the *Secret* ability or the *Quest* ability. Secret spells are spells that stay hidden on the board until a condition is met, and then the effect is activated. Quest spells have a continuous condition which allows the spell to stay in effect until the player that owns the spell has completed the condition, and then they gain an additional benefit. Neither Secret or Quest spells are used in this thesis.

In addition, a player's hero character can be upgraded during gameplay with one of few *hero cards*. These cards belong to a particular hero class and change the current hero to a new hero character, set the change the current Health value, add an amount of *armor* (extra Health not included in the total Health value), and a changed Hero Power. An example hero card is shown in Figure 2.7. Although hero cards are not explicitly used, some decks in this thesis feature a minion card which changes the player's hero character.

Lastly, *weapons* are a unique type of card which have a mana cost, attack power, and durability (Health), but are not placed on the board like minions. Instead, once activated, the weapon is equipped to the player and transforms the player's hero into a minion-like unit that can also place attacks, just like minions. Each attack that the player's hero character makes while equipped with a weapon decreases the weapon's durability by one, and once it reaches zero, the weapon breaks and leaves the game. An example weapon is shown in Figure 2.8. Weapons are not used in this thesis.

Note that over time some of these generalizations blur, especially given that there are many different types of effects applicable to each card. For instance, some minions can't be damaged while on the board, others can't deal damage, and the list goes on. There are even some cards that belong to a particular class which give decks

that use those cards a unique advantage over other classes. For example, the Druid class has access to cards which manually increase the maximum available mana for that player by one (but never above 10).

2.2.6 Gameplay: Turn Sequencing

In a game of Hearthstone, both players start by *drawing* three cards, always from the top of the deck (which is face down so that cards in the deck are hidden to both players). Before the first turn, both players have the option to perform *mulligan* once by optionally deciding to put back some of the cards from their hand into the deck, shuffling the deck, then re-drawing the same amount of cards from the top of their deck. Each player has 30 Health to start, and the primary goal of defeating the opponent is achieved by decreasing the opponent's Health from 30 to zero. The decisions made per turn are somewhat limited by the amount of mana the turn player has. The player who goes first has the advantage of being the first person to have potential cards played. But, the player who goes second has the advantage of receiving an additional card added to their hand called "The Coin." This card is not a part of any deck, and it comes from outside of the game. It costs zero mana to play and has the effect of giving the player who activates it an extra mana to spend during the turn they play the card. This means that on Player Two's first turn, they can even play a card that costs two mana by playing "The Coin".

When starting a turn, the turn player's maximum mana gets one additional Mana Crystal, and is fully replenished so that their available mana is equal to their maximum available mana. They also draw the top card of their deck, adding one card into their hand (unless their hand is at maximum capacity of 10 cards, in which the card drawn is removed from the game). This then opens up their turn so that they can execute tasks. For example, players can play minions, spells, or weapons from their hand, they can use their Hero Power, or they can choose to attack with any

minions they already have on board. Players can execute any combination of these tasks, so long as they have available mana for each task.

2.2.7 Decision Making

Players have a large variety of decisions that they can make per turn, and these can vary based on the goal(s) of the deck they are playing, the deck the opponent is playing, the cards available in their hand or on the board, etc. If the turn player has at least one minion on their side of the field, they can choose to attack with those minions. If the player has a weapon, they can choose to attack with their hero as well. Because some portions of the game are not visible to players, they may also need to make quick assumptions about the game in order to make decisions.

2.3 Human Player Strategies

Two major gameplay aspects of Hearthstone include building a deck, and executing a gameplay strategy. Both of these elements are intertwined through a reliance on one another, and present their own areas of interesting research questions [32].

2.3.1 Building a Deck: Mana Curve

Because each card costs a certain amount of mana, decks tend to develop different types of balance for cards that have different mana costs so that they can be played at various turns in the game [6]. For example, if a deck has only one-mana cost cards, then the player would have the guaranteed ability to play cards in the early turns, but may find themselves quickly running out of cards in their hand. On the other hand, if a deck has only five-mana cost cards, while the cards generally have more attack power and Health, the player would have to wait until at least turn five to play their cards. By balancing the amount of cards in a deck at each level of mana cost, a player can have a better chance to get cards in their hand with appropriate mana costs at different points in the game. In doing this, players will see their cards

form a *mana curve* which is a histogram that maps the number of cards in the deck at each mana cost. Mana curves for the decks used in this thesis can be referred to via Appendix D.

2.3.2 Building a Deck: Metagame

In addition to having a balanced mana curve, players can also consider building their cards around a *metagame*, which refers to the cards and types of cards that are popular at a given point in time. The metagame is an always changing list of cards which can be impacted by the format players design decks around, the booster packs available at the time, and even the decks used by professional players. One issue which can arise is when a metagame becomes over-saturated with particular cards or particular decks, which can be difficult to play in if a player does not have cards to defeat those in the metagame. Although both players and Hearthstone game designers play an active effort in creating more balance in the meta, there may be more ways to balance the metagame with the use of multi-objective optimization and evolutionary algorithms [51].

2.3.3 Gameplay Strategies

Like other games, Hearthstone has a variety of styles of play, also known as *gameplay strategies* that players can execute during a game [51]. The deck that a player builds belongs to one of several *archetypes* which closely relates to the gameplay strategy used. Despite having many types of strategies, the two discussed in this thesis are *aggro* and *control*. Generally, aggro players attempt to win the game quickly by filling their deck with lower cost cards, and tend to focus on aggressively dealing direct damage to the opponent while ignoring the opponent's minions. On the other hand, control players attempt to play a longer game and seek to win in later turns after controlling the board for the duration of the game; using removal spells and defensive minions help the control player last long enough to execute their strategy

effectively. Because aggro players seek to win quickly, they generally have more lower cost cards in their decks, which causes their mana curve to be right skewed, as in Figures D.6-D.10. Meanwhile, because control players seek to win in later turns, their mana curves tend to either be more uniform or ragged with no clear distribution, as in Figures D.1-D.5.

2.4 Artificial Intelligence in Games

Although AI has always proven to be useful when applied to games, in recent years, research has seen a growing spike in the number and variety of studies revolved around games. Especially since the defeat of Garry Kasparov by Deep Blue, researchers have an increased interest in games such as Go [52], StarCraft [56], and Hearthstone [59].

2.4.1 Game Tree Search for Playing Games

Early on, games proved to be useful for research in AI methods. For example, machine learning began with the idea that a computer can be programmed so that it will learn to play a better game of Checkers than can be played by the person who wrote the program [46]. This early application started to drive further interest in general purpose learning machines, but also confirmed that machines can learn to play games at human level. Not only that, this study utilized the notion of searching through a tree of possible moves with a look-ahead to determine the next move. Another major groundbreaking moment for AI recognized the defeat of Garry Kasparov by the Deep Blue computer Chess system developed by IBM [10]. This massively parallel system was designed for calculating searches through Chess game trees with an automated evaluation function analysis. Complete game trees will show all possible moves and states in a game, but compared to Checkers, games like Chess are shown to have upwards of 10^{43} possible game states [49]. One pass through the game tree from the root node to a leaf is considered a solution to the game; successful strategies for playing games based on these trees should decide which parts of the tree to explore.

Since then, other games have also been used for the application of search algorithms through game trees, especially Monte-Carlo Tree Search [12]. This tree search algorithm combines multiple parts including a selection function which recursively finds a leaf node (some ending state of a series of game states), expands and simulates more branches after that game state, then backpropogates to the current state to determine a decision based on the simulated outcomes. Thus, the values for a state is based on how likely a player is to win, should they make that decision. This tree search algorithm proved effective in board games like Kriegspiel [13], AlphaGo's success in Go [25, 52], and even multiplayer Poker [9]. Some approaches have seen success in combining MCTS with deep learning in Atari [27], and others use reinforcement learning techniques, such as the OpenAI approach for Dota 2 [40]. But the application of MCTS to card games like Magic: the Gathering [58] helps to highlight previous gameplay research in Hearthstone [19, 35, 47, 54].

2.4.2 Previous AI Approaches to Hearthstone

The main research areas of Hearthstone mirror the major elements of the game, including deck building and gameplay strategy. For players, deck building may be challenging due to the expansive amount of cards, and not knowing how they work together, or not knowing how to counter popular cards. Using vector embeddings and dimensionality reduction algorithms like t-SNE [37], it's possible to visualize decks by class [36], which may help to recommend decks for players. The use of dimensionality reduction for visualization is also utilized in this thesis. Through the use of genetic algorithms, one study explored evolving competitive decks for the nine original classes a player can choose from [24]. Another utilized evolutionary algorithms focused on three hero classes which are typically used for aggressive playstyles to develop twice evolved decks whose agents could beat agents that played using once evolved decks

[6]. Using their performance in games, one can even use neural networks to predict the win rate of decks as well [34].

In terms of gameplay research in Hearthstone, some progress has been made to help AI play Hearthstone by optimizing the decision making process using feature engineering and supervised learning [43]. Given its previous successes in algorithms like AlphaGo, MCTS has been the main approach for simulated Hearthstone players [47, 35, 19, 54], but the game's complexity creates issues for MCTS alone to succeed. For example, the branching factor of Hearthstone often requires players to make narrow assumptions about the game and the succeeding turns. Although a similar decision making process can be found in other games like Chess, the difference in Hearthstone is that the board and pieces are not fully observable to both players, which may lead to ill-informed guesses and presumptions about the cards in the opponent's hand, and in both players' decks. Still, there may be other algorithms which can develop agents with high success rates.

To contrast the experiments previously explored via MCTS, the experiments in this thesis utilize the newly proposed quality diversity algorithm called CMA-ME [23] to evolve several ANNs which are used to produce enhanced functions for evaluating game states. Because of the challenges of MCTS applied to Hearthstone, this thesis compares the win rates of agents using hand curated evaluatoin functions found in SabberStone, with those using ANNs for evaluating game states. By analyzing the set of games these agents play, the data may show patterns which not only indicate differences in gameplay strategy, but may also help move AI towards conquering Hearthstone and other AI problems. The use of CMA-ME should also lead towards agents whose decision making process is better (thus producing a higher win rate) when compared to the agents included in SabberStone [17]. In addition, other experiments provide new benchmarks for predicting and visualizing gameplay strategies in Hearthstone. By utilizing Principal Component Analysis [44, 50], the

player strategies can be projected from a higher dimension onto a smaller feature space [15, 26]. Inspired by research in other fields [3, 42], supervised learning models are used to predict the gameplay strategies of simulated players in both the original and principal component feature spaces. Through the use of simulated agents and their verbose logging via SabberStone, extensive feature engineering similar to [43] is used to coerce the AI logs into table based data sets which can be used for the analysis process and model training.

2.5 A Hearthstone Simulator: SabberStone

Because Hearthstone is not an open-source game, one group of community developers named HearthSim have developed a simulator named SabberStone [17], mostly maintained by darkfriend77. SabberStone is developed as a .NET Core application modeling the Hearthstone ruleset, gameplay, and interactions of different pieces in the game to work as a console application. SabberStone also has a test project which uses an AI with predefined gameplay strategies to run AI simulations. The AI agents utilize *heuristic scoring functions* which are used to evaluate the quality of a game state in the game tree. Although the AI can mimic aggro and control strategies to a degree, this thesis utilizes the AI to capture differences between the gameplay strategies during the simulation, and determines which of these are better when compared against each other.

2.6 Covariance Matrix Adaptation MAP-Elites: CMA-ME

Some algorithms focus on generating a single artificial neural network, but a new algorithm [23] searches to optimize the parameters of an ANN's weights by combining Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [16] with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [28, 29, 30, 33]. Instead of generating one best-performing network, the goal is to optimize a variety of networks known as *candidate solutions* along any of the desired characteristics for

the given problem. The newly developed Covariance Matrix Adaptation MAP-Elites (CMA-ME) does not converge towards one singular good solution, instead it searches for a variety of high-quality solutions by utilizing a process similar to natural evolution; this can be referred to as a *quality diversity algorithm.*

2.6.1 Evolving Scoring Functions by Combining CMA-ES and MAP-Elites

Evolutionary strategies are algorithms that use *generations* of sampling solutions to a problem and move the overall population towards areas of desired goals, or *fitness*, similar to the process of evolution in nature. CMA-ES via Figure 2.9 is a type of evolutionary strategy which is a derivative-free optimizer for optimizing single-objective functions in continuous domains [31]. CMA-ES utilizes an *evolutionary path* of changes over each generation, and with each generation a selection of μ most fit solutions, which update the covariance matrix C of the successive generation. CMA-ES also uses a restart rule if a good solution is not found in the current evolution, by generating a new mean and covariance matrix from the current best candidate solution. Meanwhile, MAP-Elites maintains a *behavior space* of solutions called *elites* in a Cartesian grid, shown in Figure 2.10. This algorithm not only tries to maximize the amount of cells filled in the grid, but also maximize the quality of the solutions used to fill each grid [39].

CMA-ME is a quality diversity algorithm which seeks to combine some of the elements of both CMA-ES and MAP-Elites. This algorithm keeps the map and archive technique of MAP-Elites, and alters CMA-ES by using *emitters*, a population of modified searches. The solutions from the emitters are kept in the grid of elites, which can be better than those found by just CMA-ES or MAP-Elites alone. The end goal is to optimize a wide variety of artificial neural networks along any desired characteristics of the generated candidate solutions (i.e. candidate ANNs).

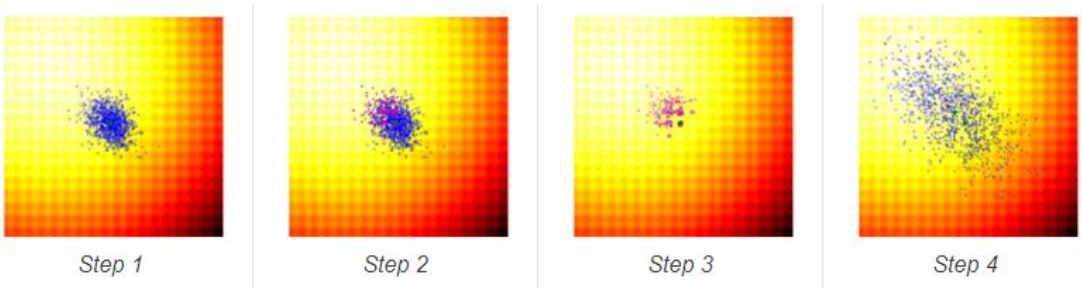

Figure 2.9 CMA-ES features four general steps. First, calculate the fitness score of each candidate solution in the current generation. Then, select top 25% of the population (purple). Using those only, calculate the covariance matrix of the next generation. And finally, sample a new set of candidate solutions.

Source: [28]

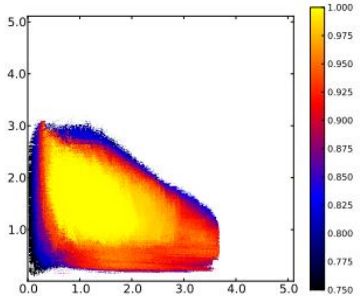

Figure 2.10 Quality diversity algorithms such as MAP-Elites produce a wide variety of high-performing solutions for the problem space. This example grid shows each candidate solution mapped in Cartesian space, and the fitness of the solution in normalized performance.

Source: [39]

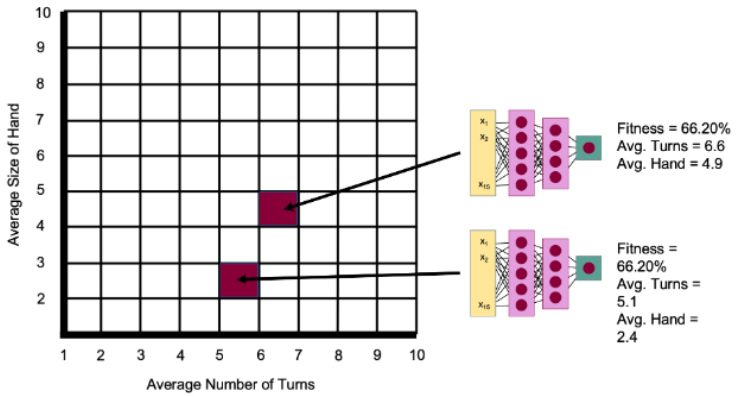

Figure 2.11 CMA-ME generated scoring functions which are represented as fully connected ANNs with fixed network topology, described in Figure 2.12.

The artificial neural networks are then used instead of the SabberStone scoring functions used to evaluate game states. Each network uses 15 observable Hearthstone game features via Table 2.2 and combines the results into a scalar value, similar to the AggroScore or ControlScore functions. This algorithm uses a fully connected feed-forward network of 26 nodes to transform each of the observable features into a score which is used to evaluate the state, the topology shown in Figure 2.12. With each network, a pre-determined N number of games are played, and the map of elites uses three quantities called *behavior characteristics* to determine how it is saved on the grid: the horizontal axis is the average number of turns per game, the vertical axis is the average number of cards in the agent's hand, and the weight of the solution on the grid is its average win rate across the N games. Only one network can occupy each cell, so networks with higher win rates replace those with lower win rates; with each generation during evolution, networks with higher win rates are saved on the grid. Not only does this algorithm finds networks which score game states that lead to a high win rate, but also a wider variety of solutions compared to CMA-ES or MAP-Elites alone.

Table 2.2 ANN Input Layer[a]

Game Feature	*Description*
HeroHp	Total health points for the current player.
OpHeroHp	Total health points for the current opponent.
HeroAtk	Attack points for the current player.
OpHeroAtk	Attack points for the current opponent.
HandTotCost	Total mana cost of cards in hand for the current player.
HandCnt	Amount of cards in hand for the current player.
OpHandCnt	Amount of cards in hand for the current opponent.
DeckCnt	Amount of cards left in the deck for the current player.
OpDeckCnt	Amount of cards left in the deck for the current opponent.
MinionTotAtk	Total attack points of current player's minions on board.
OpMinionTotAtk	Total attack points of current opponent's minions on board.
MinionTotHealth	Total health points of current player's minions on board.
OpMinionTotHealth	Total health points of current opponent's minions on board.
MinionTotHealthTaunt	Total health points of current player's Taunt minions on board.
OpMinionTotHealthTaunt	Total health points of current opponent's Taunt minions on board.

[a]15 observable game features that are used as input to the ANN.

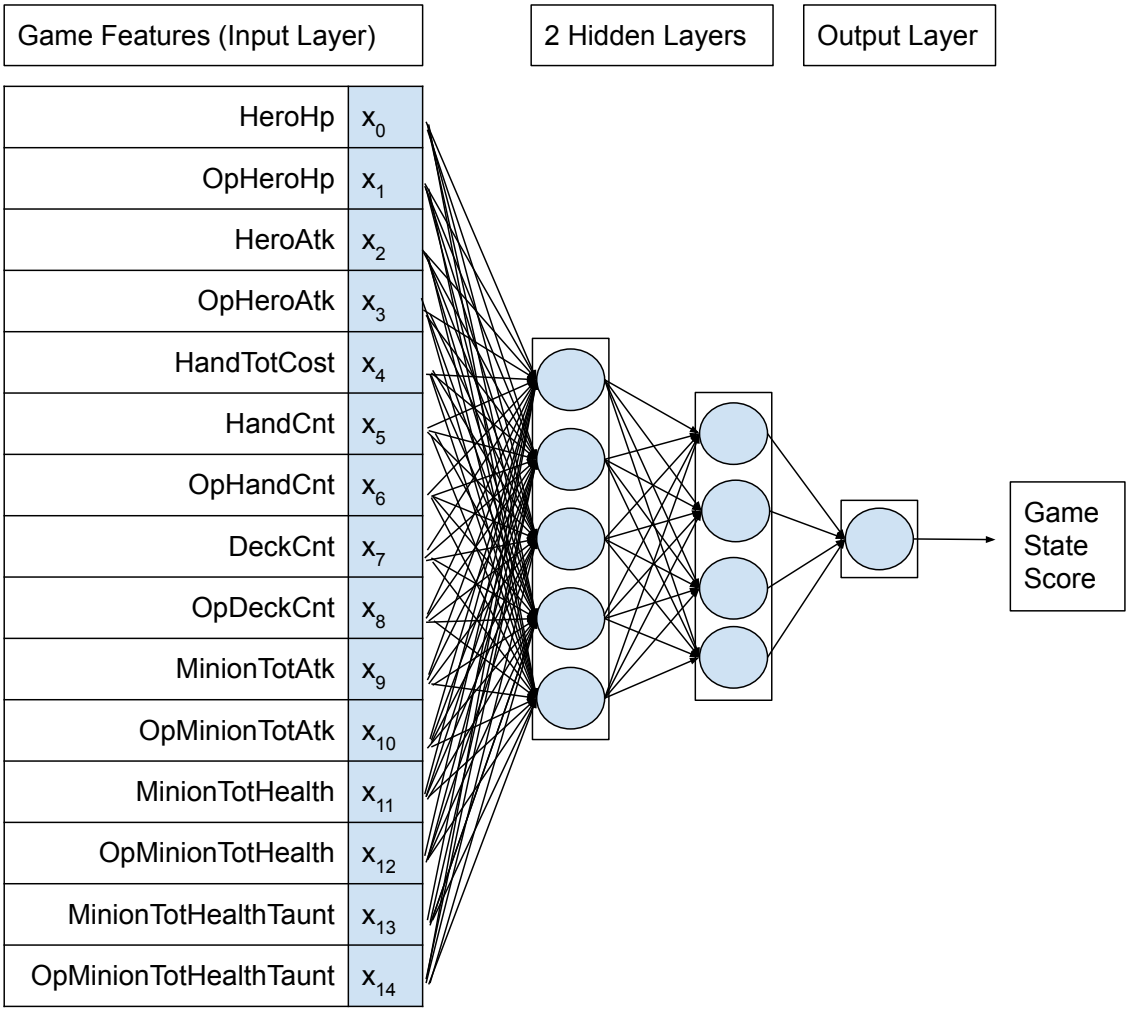

Figure 2.12 CMA-ME evolves a set of ANNs used to evaluate game states as a substitute for the heuristic scoring functions provided with SabberStone.

CHAPTER 3

METHODOLOGY

This section describes the methods for both generating and testing scoring functions in the Hearthstone simulator SabberStone, and methods for an unsupervised approach to data exploration, including a supervised classification of game states based on strategy. The code for this paper including feature engineering, experiments and results, can be found on GitHub [1].

To explore the robustness of CMA-ME, ten different Warlock deck lists are used to run the experiments in this thesis. CMA-ME is also run to evolve several different types of ANNs such that they can be used in matchups against agents using AggroScore and ControlScore as the game state evaluators.

By comparing the differences in strategy, the data should confirm the basis that learning how to play a particular strategy requires a different approach than learning how to play other strategies.

Although it is possible that a subset of the population of human Hearthstone players could have been considered, given the challenge of surveying Hearthstone players, the SabberStone code base is utilized in a High Performance Computing environment to run a large amount of games quickly. To aid the analysis process and allow for increased accessibility of data, this thesis utilizes simulated players to retrieve samples of games played between different strategy matchups. Although this method may be faster, these simulated players are not truly playing at human level, which means that by nature the types of moves detected may be different than those that could be detected for human players. The experiments and results in this thesis may be able to be used as a basis for player modeling.

[1] https://github.com/cww5/Sabber_Work_2019F

3.1 Methods for Generating and Testing Scoring Functions

SabberStone simulates games through a turn-local game tree search illustrated in Figure 3.1. Starting from the game state at the beginning of the turn (i.e., the root node), these agents then build a partial game tree by determining all of the available actions and game states reachable from that node. From the initial game state, this player can either end the turn or play the card "Murloc Raider." The AI tries each available action, and the resulting game state is evaluated and added to the next level of the tree. Ending the turn would result in a score of 0 while playing the card is worth 1002. While there are only two available actions from the game state at the root node of this tree, if the number of possible resulting game states exceeds the maximum width parameter of the AI, only the highest rated are kept. The decks and scoring functions examined in this thesis are designed to replicate two different styles of play called aggro and control (see Section 2.3.3 for more detail).

Figure 3.2 shows the beginning of the first turn for two warlock heroes named Gul'dan where the bottom, friendly player goes first. While Gul'dan (Player 1) has four cards in total, only one is playable with one mana crystal. However, the End Turn button can be pressed at any time regardless of whether the mana is spent.

Shown in Algorithm 3.1 the AggroScore prioritizes placing minions on the board before the opponent (lines 10 and 11) and maximizing their total attack power (line 16). When the opponent begins to build a defense with *Taunt* minions, which are required to be killed before the opponent can be attacked, the game state is penalized (lines 14 and 15). An aggro playstyle favors aggressively attacking the opponent hero (line 19). Scores are saved in the result variable declared on line 9 and returned on line 19. AggroScore is the function determining the scores for the game states in Figure 3.1. The right node representing the game state after playing the card is calculated as 1002 because the current player has one minion on the board while the opponent has zero (+1000 from line 11). On line 16 the attack power of this minion

Figure 3.1 On the left is a game tree search available to this player at the beginning of the first turn, indicated by the number of available mana crystals (right, top). While the player has four cards in hand to play (right, bottom), the two actions available for one mana ending the turn (free) or playing the card "Murloc Raider" for one mana and highlighted in green. Nodes are scored by a function to determine the value of each selection. Ending the turn does not increase the score while playing the card results in a score of 1002.

(a) Before Playing Card: Murloc Raider

(b) After Playing Card: Murloc Raider

Figure 3.2 Corresponding to the game tree shown in Figure 3.1, this player starts with only one card it can afford to play in a (left), and plays it in b (right). The End Turn button is highlighted in green when it is the only available option, but it could have been selected instead of playing the card.

is added to the result for a total of 1002. Figure 3.2a shows the initial state of the game at the root node of the game tree in Figure 3.1, while Figure 3.2b shows the rightmost leaf node after playing the card "Murloc Raider."

Algorithm 3.1 The AggroScore Evaluation Heuristic

```
1 public class AggroScore : Score
2 {
3         public override int Rate()
4         {
5                 if (OpHeroHp < 1)
6                         return int.MaxValue;
7                 if (HeroHp < 1)
8                         return int.MinValue;
9                 int result = 0;
```

```
10                  if (OpBoardZone.Count == 0 &&
                        BoardZone.Count > 0)
11                      result += 1000;
12                  //Difference lines 13-14
13                  if (OpMinionTotHealthTaunt > 0)
14                      result += OpMinionTotHealthTaunt *
                            -1000;
15
16                  result += MinionTotAtk;
17                  //Diffence line 18
18                  result += (HeroHp - OpHeroHp) * 1000;
19                  return result;
20          }
21  }
```

While ControlScore in Algorithm 3.2 considers many of the same game features, a key difference is the emphasis on aggressively attacking the opponent's hero. Instead of weighting the difference in hero health by 1000 (line 19 in Algorithm 3.1), in Algorithm 3.2 on line 18 it is only weighted by 10. Rather than the difference in hero health, control strategies in Hearthstone encourage reducing the power of the opponent by first destroying the minions currently on the board. Lines 13 and 14 both reward the game state when the friendly player is able to have more minions than the opponent, and reward the state when the friendly player has Taunt minions with more attack power than the opponent.

Algorithm 3.2 The ControlScore Evaluation Heuristic

```
1  public class ControlScore : Score
2  {
```

```
3           public override int Rate()
4           {
5                   if (OpHeroHp < 1)
6                           return int.MaxValue;
7                   if (HeroHp < 1)
8                           return int.MinValue;
9                   int result = 0;
10                  if (OpBoardZone.Count == 0 &&
                        BoardZone.Count > 0)
11                          result += 1000;
12                  //Difference lines 13-14
13                  result += (BoardZone.Count -
                        OpBoardZone.Count) * 50;
14                  result += (MinionTotHealthTaunt -
                        OpMinionTotHealthTaunt) * 25;
15
16                  result += MinionTotAtk;
17                  //Diffence line 18
18                  result += (HeroHp - OpHeroHp) * 10;
19                  return result;
20          }
21  }
```

Figure 3.3 shows the beginning of the first turn for the second player of a Hearthstone game. To offset the advantage the first player receives, the second player is given an additional card from their deck, and a card called "The Coin." If the player starts the turn by playing "The Coin," he will have two mana to spend. Since "Kobold

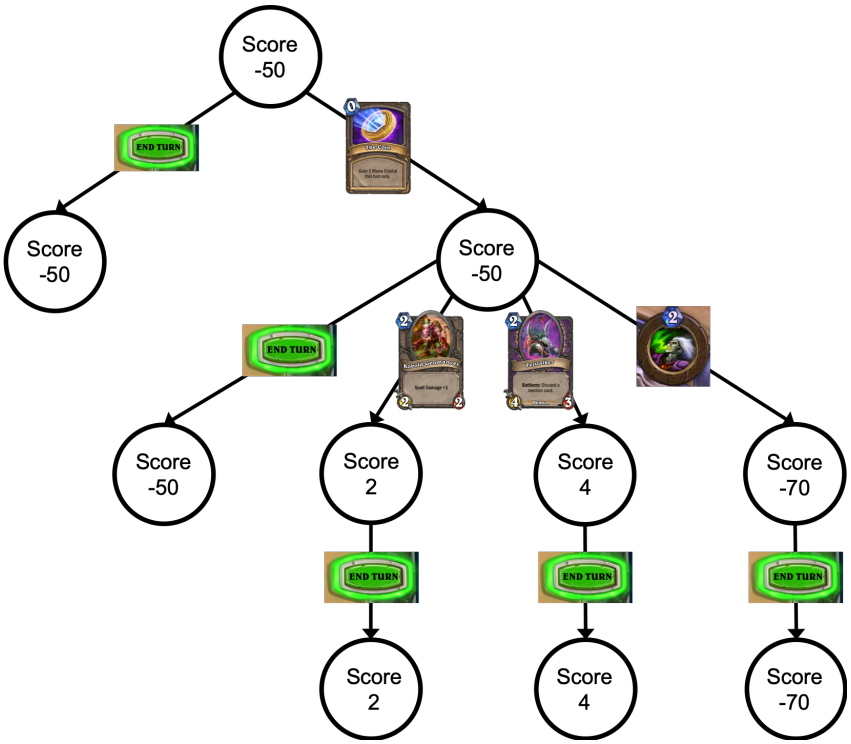

Figure 3.3 The second player begins the first turn at a disadvantage as the first player has placed a minion on the board. The ControlScore heuristic described in Algorithm 3.2 calculates the value of the different options available to the player. To offset this imbalance, second players are given an extra card from their deck and a special card called "The Coin," which gives the player an additional Mana Crystal. If "The Coin" is played, it is possible to play the "Kobold Geomancer", and "Felstalker" cards or play the "Lifetap" hero power. Because the Felstalker results in the highest reward, it is chosen by the player following "The Coin."

```
Player1: PLAYING / Player2: PLAYING - ROUND 1 - Roffle
Hero[P1]: 30 / Hero[P2]: 30

* Calculating solutions *** Player 1 ***
- Player 1 - <Roffle> --------------------------
PlayCardTask => [Roffle] play 'Mecharoo[28]'(MINION) to Pos[0] Option -1
Time: 10/6/2019 10:48:35 PM, Level: INFO, Location: Game, Blocktype: PLAY, TextPlayCardTask => [Roffle] play 'Mecharoo[28]'(MINION) to Pos[0] Option -1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Mecharoo[28]' set data TAG_LAST_KNOWN_COST_IN_HAND to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data RESOURCES_USED to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data NUM_RESOURCES_SPENT_THIS_GAME to 1
Time: 10/6/2019 10:48:35 PM, Level: INFO, Location: PayPhase, Blocktype: ACTION, TextPaying 'Mecharoo[28]' for 1 Mana, remaining mana is 0.
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Mecharoo[29]' set data ZONE_POSITION to 3
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Argent Squire[26]' set data ZONE_POSITION to 2
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Scarab Egg[36]' set data ZONE_POSITION to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data NUM_CARDS_PLAYED_THIS_TURN to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data LAST_CARD_PLAYED to 28
Time: 10/6/2019 10:48:35 PM, Level: INFO, Location: PlayMinion, Blocktype: ACTION, TextRoffle plays Minion 'Mecharoo[28]' to board position 0.
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data NUM_MINIONS_PLAYED_THIS_TURN to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Mecharoo[28]' set data ZONE to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Zone, Blocktype: PLAY, TextEntity ''Mecharoo[28]' (MINION)' has been added to zone 'PLAY' in position '0'.
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Mecharoo[28]' set data ZONE_POSITION to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Mecharoo[28]' set data EXHAUSTED to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data NUM_OPTIONS_PLAYED_THIS_TURN to 1
Time: 10/6/2019 10:48:35 PM, Level: DEBUG, Location: Entity, Blocktype: TRIGGER, Text'Player[2]' set data COMBO_ACTIVE to 1
```

Figure 3.4 Verbose logs of the SabberStone AI agents during runtime of the simulation.

Geomancer" and "Felstalker" both cost two mana, they then become possible to play. The solution is selected by choosing the the actions that lead to the highest value leaf node, which is scored at four. The solution is then playing the card "The Coin" and then "Felstalker."

These scoring functions are designed to replicate aggro and control strategies playable by most decks and heroes. For instance when choosing between attacking a minion or the opponent hero directly, many human players may choose to attack the hero. However, players trying to control the board may choose the former, but each decision depends on the power of the particular minion and the value of the opponent hero's health.

3.2 Methods for Collecting and Analyzing Data

To prepare the data for analysis, first the SabberStone code was altered to output logs as shown in Figures 3.4 - 3.7. In order to run a large amount of games quickly, the repository was also configured to run in a High Performance Computing cluster environment. Python scripts were used to mine the logs for the relevant statistics, and cleaned into pandas based DataFrames [38]; this includes the end of turn statistics for each player during run-time of the games. This proved to be useful, especially when comparing the win rates across N games of particular matchups.

```
Player1: PLAYING / Player2: PLAYING - turn no 1
Hero[P1]: 30 / Hero[P2]: 30
>>>>>>>>>TASK TYPE CHECK (is EOT?): True
>>>>>>>>>TASK TYPE: END_TURN
ALL TURN TASKS:
PlayCardTask => [Roffle] play 'Mecharoo[28]'(MINION) to Pos[0] Option -1
EndTurnTask => [Roffle]
COMPLETE ALL TURN TASKS
CURRENT PLAYER: P1 Roffle
AMOUNTHEALEDTHISTURN 0
HEROPOWERACTIVATIONSTHIS TURN 0
NUMATTACKSTHISTURN 0
NUMCARDSDRAWNTHISTURN 1
NUMCARDSPLAYEDTHISTURN 1
NUMCARDSTODRAW 0
NUMELEMENTALSPLAYEDLASTTURN 0
NUMELEMENTALSPLAYEDTHISTURN 0
NUMFRIENDLYMINIONSTHATATTACKEDTHISTURN 0
NUMFRIENDLYMINIONSTHATDIEDTHISTURN 0
NUMMINIONSPLAYEDTHISTURN 1
NUMMINIONSPLAYERKILLEDTHISTURN 0
NUMOPTIONSPLAYEDTHISTURN 1
NUMSECRETSPLAYEDTHISGAME 0
NUMSPELLSPLAYEDTHISGAME 0
NUMTIMESHEROPOWERUSEDTHISGAME 0
REMAININGMANA 0
TOTALMANASPENTTHISGAME 1
USEDMANATHISTURN 1
PlayedCard: [Mecharoo]
No more cards played this turn.
```

Figure 3.5 Verbose logs reformatted into an easily parsable format. The reformatted logs are printed only after each turn is complete.

```
Player1: PLAYING / Player2: PLAYING - turn no 8
Hero[P1]: 13 / Hero[P2]: 30
>>>>>>>>>TASK TYPE CHECK (is EOT?): True
>>>>>>>>>TASK TYPE: END_TURN
ALL TURN TASKS:
MinionAttackTask => [Roffle] 'Voidwalker[46]'(MINION) attack 'Gul'dan[4]'
MinionAttackTask => [Roffle] 'Voidwalker[47]'(MINION) attack 'Gul'dan[4]'
PlayCardTask => [Roffle] play 'Fiendish Circle[51]'(SPELL) Option -1
MinionAttackTask => [Roffle] 'Knife Juggler[64]'(MINION) attack 'Gul'dan[4]'
EndTurnTask => [Roffle]
COMPLETE ALL TURN TASKS
CURRENT PLAYER: P2 Roffle
AMOUNTHEALEDTHISTURN 0
HEROPOWERACTIVATIONSTHIS TURN 0
NUMATTACKSTHISTURN 0
NUMCARDSDRAWNTHISTURN 1
NUMCARDSPLAYEDTHISTURN 1
NUMCARDSTODRAW 0
NUMELEMENTALSPLAYEDLASTTURN 0
NUMELEMENTALSPLAYEDTHISTURN 0
NUMFRIENDLYMINIONSTHATATTACKEDTHISTURN 3
NUMFRIENDLYMINIONSTHATDIEDTHISTURN 0
NUMMINIONSPLAYEDTHISTURN 0
NUMMINIONSPLAYERKILLEDTHISTURN 0
NUMOPTIONSPLAYEDTHISTURN 4
NUMSECRETSPLAYEDTHISGAME 0
NUMSPELLSPLAYEDTHISGAME 3
NUMTIMESHEROPOWERUSEDTHISGAME 0
REMAININGMANA 0
TOTALMANASPENTTHISGAME 11
USEDMANATHISTURN 4
PlayedCard: [Fiendish Circle]
No more cards played this turn.
```

Figure 3.6 This sample of reformatted logs shows the current player P2 Roffle executing four additional tasks during the turn.

```
Player1: LOST / Player2: WON - turn no 12
Hero[P1]: -1 / Hero[P2]: 29
>>>>>>>>>TASK TYPE CHECK (is EOT?): False
>>>>>>>>>TASK TYPE: MINION_ATTACK
ALL TURN TASKS:
MinionAttackTask => [Roffle] 'Saronite Taskmaster[61]'(MINION) attack 'Gul'dan[4]'
COMPLETE ALL TURN TASKS
CURRENT PLAYER: P2 Roffle
AMOUNTHEALEDTHISTURN 0
HEROPOWERACTIVATIONSTHIS TURN 0
NUMATTACKSTHISTURN 0
NUMCARDSDRAWNTHISTURN 1
NUMCARDSPLAYEDTHISTURN 0
NUMCARDSTODRAW 0
NUMELEMENTALSPLAYEDLASTTURN 0
NUMELEMENTALSPLAYEDTHISTURN 0
NUMFRIENDLYMINIONSTHATATTACKEDTHISTURN 1
NUMFRIENDLYMINIONSTHATDIEDTHISTURN 0
NUMMINIONSPLAYEDTHISTURN 0
NUMMINIONSPLAYERKILLEDTHISTURN 0
NUMOPTIONSPLAYEDTHISTURN 1
NUMSECRETSPLAYEDTHISGAME 0
NUMSPELLSPLAYEDTHISGAME 3
NUMTIMESHEROPOWERUSEDTHISGAME 0
REMAININGMANA 6
TOTALMANASPENTTHISGAME 12
USEDMANATHISTURN 0
No more cards played this turn.
Game: COMPLETE, Player1: LOST / Player2: WON
```

Figure 3.7 This sample of reformatted logs shows the current player P2 Roffle executing one task to win the game.

```
Player1: PLAYING / Player2: PLAYING - turn no 1
Hero[P1]: 30 / Hero[P2]: 30
>>>>>>>>>TASK TYPE CHECK (is EOT?): True
>>>>>>>>>TASK TYPE: END_TURN
ALL TURN TASKS:
PlayCardTask => [Roffle] play 'Mecharoo[28]'(MINION) to Pos[0] Option -1
EndTurnTask => [Roffle]
COMPLETE ALL TURN TASKS
CURRENT PLAYER: P1 Roffle
AMOUNTHEALEDTHISTURN 0
HEROPOWERACTIVATIONSTHIS TURN 0
NUMATTACKSTHISTURN 0
NUMCARDSDRAWNTHISTURN 1
NUMCARDSPLAYEDTHISTURN 1
NUMCARDSTODRAW 0
NUMELEMENTALSPLAYEDLASTTURN 0
NUMELEMENTALSPLAYEDTHISTURN 0
NUMFRIENDLYMINIONSTHATATTACKEDTHISTURN 0
NUMFRIENDLYMINIONSTHATDIEDTHISTURN 0
NUMMINIONSPLAYEDTHISTURN 1
NUMMINIONSPLAYERKILLEDTHISTURN 0
NUMOPTIONSPLAYEDTHISTURN 1
NUMSECRETSPLAYEDTHISGAME 0
NUMSPELLSPLAYEDTHISGAME 0
NUMTIMESHEROPOWERUSEDTHISGAME 0
REMAININGMANA 0
TOTALMANASPENTTHISGAME 1
USEDMANATHISTURN 1
PlayedCard: [Mecharoo]
No more cards played this turn.
```

Figure 3.8 Each player in SabberStone has a set number of instance attributes available which describe tasks they've executed each turn, and cumulatively each game. The logs are parsed to cleanly display this information. The description of each can be found in Tables 3.1 and 3.2

	TURN_NO	P1_HEALTH	P2_HEALTH	CURRENT_PLAYER	AMOUNTI	NUMATTA	NUMCARI	NUMCARDSPLAYEDTHISTURN
0	1	30	30	P1 Roffle	0	0	1	1
1	2	30	30	P2 Roffle	0	0	1	3
2	3	30	30	P1 Roffle	0	0	1	1
3	4	26	30	P2 Roffle	0	0	1	1
4	5	26	30	P1 Roffle	0	0	1	2
5	6	23	30	P2 Roffle	0	0	1	2
6	7	23	30	P1 Roffle	0	0	1	0
7	8	13	30	P2 Roffle	0	0	1	1
8	9	13	30	P1 Roffle	0	0	1	2
9	10	4	30	P2 Roffle	0	0	1	1
10	11	2	29	P1 Roffle	0	0	2	2
11	12	-1	29	P2 Roffle	0	0	1	0
12	1	30	26	P1 Roffle	0	0	1	1
13	2	30	26	P2 Roffle	0	0	1	2
14	3	30	26	P1 Roffle	0	0	1	0
15	4	26	26	P2 Roffle	0	0	1	2
16	5	26	26	P1 Roffle	0	0	1	3
17	6	20	26	P2 Roffle	0	0	1	1
18	7	20	21	P1 Roffle	0	0	1	1
19	8	10	21	P2 Roffle	0	0	1	3
20	9	10	15	P1 Roffle	0	0	1	2

Figure 3.9 Sample of the raw logs output from SabberStone formatted into a structured table based csv. Each row of the table represents the end of turn statistics for the player described in the CURRENT_PLAYER column.

CurrentPl	PlayerStra	AvgHeale	AvgCardsI	AvgCardsF	AvgFrienc	AvgFrienc	AvgMinio
1	0	0	1.17	1.33	0.67	0.83	1.17
2	0	0	1	1.33	2	0	0.83

Figure 3.10 Sample of a structured game from the raw output logs transformed into a smaller table of game statistics for the two players. Each observation is a player, and the features are the player's game statistics. Full descriptions of the columns can be found in Table 3.10.

1	0	0	1	0.86	2.57	0.29	0.71
2	0	1.17	1.5	1.5	1.17	0.33	0.67
1	0	0	1.14	0.86	1.86	0.43	0.86
2	0	0	1	1.17	1	0	1
1	0	0	1.5	1	1.33	0.17	0.67
2	0	0	1	1.4	2.4	0.2	0.6
1	1	0.83	1.17	1	1.17	0.33	0.83
2	1	0.36	1.91	1.36	0.27	0	0.82
1	1	0.33	1.17	1	1.25	0.42	1
2	1	0.36	2.09	1.18	0.09	0	0.55
1	1	0.14	1.21	1.21	0.64	0.21	1
2	1	0	1.07	1.29	0.86	0.29	1
1	1	0	1.27	1.55	0.64	0.55	1.55

Figure 3.11 Sample of the structured games seen in Figure 3.10 based from the raw output logs transformed into a table of game statistics for all players across a given matchup. Full descriptions of the columns can be found in Table 3.10.

Figure 3.9 shows how the raw logs are streamlined into a clean comma separated values file. Each file has 23 columns (not including the index column), and the logs stored for all N games run for a particular matchup between two simulated agents. Although this data set contains the data for all of the games in a given matchup, additional feature engineering is used to succinctly describe each player as rows in a table based format, where the columns represent their averaged game statistics. Each game's worth of data is transformed into a smaller data table with only two rows of data via Figure 3.10, containing data on the two players involved in the game, and can be collected into a set of players across a set of unique games via Figure 3.11. More details can be found in Table 3.3.

Hearthstone game states are comprised of many features. Which are most important for describing the game depend on the problem to solve and particular heroes and decks in play. For instance, the feature "Base Mana" describes the amount of mana available to a player at the beginning of a turn, discounting additional mana provided through other means. Likely, this feature alone would accurately

Figure 3.12 Blessing of wisdom is a card that when played casts a spell on a minion. Every time that minion attacks another minion or the heroes, the player who cast the spell draws card from its deck.

disambiguate the current turn number if it were the first through tenth turn and the base mana had not been affected by other cards in the game. However, the amount of mana available to a player stops increasing after the tenth turn and could not disambiguate the eleventh turn from others above the tenth. Because players begin each turn by automatically drawing a single card from their decks, the feature "Current Deck Size" could help disambiguate turns one through eleven. However, because some playable cards like "Blessing of Wisdom" shown in Figure 3.12 can draw cards from the deck based on other game features like the number of times a minion attacks, the current deck size is also not sufficient for determining the current turn of a player. Which features and how many are necessary is a problem in feature engineering [43].

While it is unknown exactly how the features of the game will change with game states of different gameplay strategies, Tables 3.1 and 3.2 show 19 game features belonging to SabberStone AI agents which were chosen to differentiate game states played with the hand-coded AggroScore and ControlScore heuristics and those evolved through CMA-ME. Although 19 features were found to belong to AI in SabberStone, there may be more that have not been utilized yet. In addition, because of the choice of hero character and decks used in this thesis, four of the features are unused (noted

in Table 3.2). Appendix B also offers visualizations which compare the distribution of the game features across a collection of players utilizing different scoring functions.

Like the example of differentiating game state data to determine the current turn of a player, different features are more likely to differentiate state data based on the scoring function selected, assuming the strategy picks the top rated states. One distinguishing feature separating these gameplay strategies is the mana curve in that aggro strategies tend to have many low-cost cards. Control strategies on the other hand have a broader range of cards with different costs. Aggro decks often have a large number of low-cost minions that they can play quickly and aggressively, to accomplish their main goal of directly attacking the opponent's hero character. Compared to control strategies, on average, turns played by an aggressive strategy should have a larger number of minions played per turn (logged with the feature "Number of Minions Played this Turn,"), number of minion attacks per turn ("Number of Friendly Minion Attacks this Turn"), and number of cards played per turn ("Number of Cards Played this Turn"). The distributions of these game features compared between aggro and control strategies can be found via Figures B.6, B.4, and B.2 respectively. In the SabberStone simulator, *option* is a term encompassing all of the actions a player can take in the game. With a larger number of low-cost cards and minions on the board, aggro strategies should also have a higher number of options than a control strategy. The feature "Number of Options Played this Turn" is also included, and the comparison across aggro and control can be found via Figure B.3. Because the number of minions played by the aggro strategy is likely higher than the control and cost less, "Number of Friendly Minion Attacks this Turn" and the number of low cost minions that die should be higher as well ("Number of Friendly Minions that Died this Turn" via Figure B.5). The main goal of an aggro strategy is to attack the opponent hero quickly and end the game before it is necessary to play high cost minions and cards. Requiring fewer turns should be reflected in the "Total Mana Spent this Game"

Table 3.1 Game Features[a]

Game Feature	*Description*
AmountHealedThisTurn	Total Health Points a player healed in a turn.
HeroPowerActivations ThisTurn	Total amount of times the player activated their Hero Power in a given turn.
NumAttacksThisTurn	Total amount of times the player's hero character attacked in a given turn.
NumCardsDrawnThisTurn	Total number of cards drawn by the player during a given turn.
NumCardsPlayedThisTurn	Total number of cards played (minion, spell, weapon, and hero) during a given turn.
NumMinionsPlayedThisTurn	Total number of minions played in a given turn.
NumOptionsPlayedThisTurn	Total number of tasks carried out by a player in a given turn. Each task is an OptionNode on the game tree.
NumSpellsPlayedThisGame	Total number of spells played during the current game.
RemainingMana	The amount of available mana the player has leftover after ending their turn.
TotalManaSpentThisGame	Total amount of mana a player spent during the whole game.
UsedManaThisTurn	Total amount of mana a player used during a given turn.

[a]Many variables describe the state of a game after a player completes a turn. These features show the subset logged for experiments in this paper.

Table 3.2 Game Features Continued[a]

Game Feature	*Description*
NumFriendlyMinionsThat AttackedThisTurn	Total number of minions owned by the player that attacked in a given turn.
NumFriendlyMinionsThat DiedThisTurn	Total number of minions owned by the player that died in a given turn.
NumMinionsPlayer KilledThisTurn	Total number of minions owned by the opponent that the current player defeated in a given turn.
NumTimesHeroPower UsedThisGame	Total number of times a hero has used their hero power during a game.
NumElementalsPlayed LastTurn	Total number of elemental cards played during the previous turn by the player. NOTE: This feature is unused because the decks do not have elementals.
NumElementalsPlayed ThisTurn	Total number of elemental cards played during the current turn by the player. NOTE: This feature is unused because the decks do not have elementals.
NumSecretsPlayedThisGame	Cumulative number of secret spells played by the player during the game. NOTE: This feature is unused because the decks do not have secret spells.
NumCardsToDraw	NOTE: This feature is unused - not clear what this does.

[a]Continued table of game features and their descriptions.

shown in Figure B.11, where aggro strategies should have fewer turns and therefore less mana spent than the later game turns of control strategies.

Control strategies on the other hand play tend to play higher cost minions with more health, include removal spells of varying mana cost, and try to kill the opponent's minions before attacking the opponent's hero. If any minions are healed, it may be likely that single attacks do not kill these minions, staying alive long enough to be healed. The game feature "Amount Healed this Turn" logs this property.

While the Warlock hero power is not explicitly rewarded by the AggroScore or ControlScore functions, several features are included to explore whether it is played and in what circumstances including: "Hero Power Activations This Turn," "Number of Times Hero Power Used this Game, " "Used Mana this Turn," "Remaining Mana," and "Number of Cards Drawn this Turn." The "Number of Spells Played this Game" feature is included to see which of the two strategies tend to use spells more. The distributions for the above game features are also included in B. The following additional features are included to explore whether they impact the classification of game states by strategy: "Number of Elementals Played this Turn," "Number of Secrets Played this Game," "Number of Cards to Draw." The former two are not utilized in the experiments in this paper particularly because the decklists included do not include these types of cards, and the latter of the three is shown to have no usage as well, shown in Figure B.15. These game features may be utilized in future work.

However, despite domain knowledge, what is important to consider is that these features may act in unexpected ways. Unknown is whether combined they will result in meaningful differences of game states.

Table 3.3 Succinct Statistics for Supervised Learning[a]

Game Feature	*Description*
PlayerStrategy	Strategy used by the player aggro (0) or control (1).
AvgHealedPerTurn	Average amount of Health healed per turn.
AvgCardsDrawnPerTurn	Average amount of cards drew per turn.
AvgCardsPlayedPerTurn	Average amount of cards played per turn.
AvgFriendlyMinionAttacksPerTurn	Average number of friendly minions that attacked per turn.
AvgFriendlyMinionDeathsPerTurn	Average number of friendly minions that died per turn.
AvgMinionsPlayedPerTurn	Average number of minions played per turn.
AvgNumMinionsKilledPerTurn	Average number of opponent minions killed per turn.
AvgOptionsPlayedPerTurn	Average number of options played per turn.
AvgRemainingManaPerTurn	Average amount of mana remaining after ending each turn.
AvgManaUsedPerTurn	Average amount of mana used per turn.
NumSpellsPlayedPerGame	Total number of spell cards played per game.
NumHeroPowersUsedPerGame	Total number of times the Hero Power was used per game.
TotalManaSpentPerGame	Total (cumulative) amount of mana spent per game.
AvgHeroAttacksPerTurn	Average amount of hero attacks per turn.
AvgNumCardsToDraw	NOTE: This feature is unused - not clear what this does.

[a]These features (excluding the last two) are used in the models for supervised learning, and PCA as well. Each row of the data set represents a player in a game, and each feature is listed above belonging to that player.

CHAPTER 4

EXPERIMENTS

Experiments aim to explore properties of the heuristic scoring functions, including the hand-curated ones described in Section 3.1. In the first set of experiments, these heuristics are compared to those generated with CMA-ME with the aim of finding the highest performing game state evaluator. The second set explores the data gathered from the first with supervised and unsupervised learning techniques.

4.1 Comparing Scoring Functions

While the difficulty of playing with certain classes and archetypes differs between players, because each hero is equipped with unique Hero Powers and class cards, they necessarily require different styles of play. However, AggroScore in Algorithm 3.1 and ControlScore in Algorithm 3.2 are generalized heuristic functions designed for adequate performance across a range of different styles of play for the ten classes.

In addition, Hero Powers differ by class as shown in Table 2.1. The Hero Power for the Hunter class is to spend two mana to damage the opponent's hero character by two Health points, and is accounted for in both heuristics through the consideration of each player's total Health. However, the Warlock class must do two damage to itself to draw a card when using its Hero Power. Because the number of cards in the player's hand is not counted, these functions do not explicitly reward playing the Hero Power (for Warlock), despite its strategic importance in competitive, human-level play. Both of these scoring functions ignore the importance of the number of cards a player has in their hand, such that the Hero Power is only rewarded when the new card drawn can be immediately played. While there are other classes that are indirectly rewarded for playing their Hero Powers by these heuristics (i.e., the Rogue), experiments in this paper focus on the Warlock class because of the direct relationship between the

Hero Power and a feature ignored in these heuristics. The goal of the experiments in this section are to use the Warlock class to compare which of the AggroScore or ControlScore are better as scoring functions. In addition, they are using win rates to compare the performance of ANNs for game state evaluation evolved via CMA-ME to the AggroScore and ControlScore scoring functions.

4.1.1 AggroScore vs. ControlScore

The first experiment explores whether there is an implicit advantage for either scoring function when playing with the Warlock class. Strategies are in part determined by the decks that players select for their classes, so five aggro and five control decks are gathered from websites like `http://hearthstonetopdecks.com`, where human players regularly upload and tag their favorite decks to share with the community (more details on each deck are found in Appendices E and F). For each set of two decks of the ten, four hundred games are played and logged for a total of 45 different matchups between two scoring functions. Each matchup is played with all combinations AggroScore and ControlScore functions such that there are 180 matchups of specific combinations of decks, and strategies. The setup for this experiment is described in Table 4.1. The goal is to determine which scoring function performs best and why, given the Warlock hero class and collection of decks. Because the ControlScore heuristic places less emphasis on the health difference between players, the control Warlock is inadvertently incentivized to play the Hero Power more often. This provides more possibility to play cards by having increased hand size, and therefore plays more successfully. The hypothesis is that the average win rate of aggro players (AggroScore) should be less than the average win rate of control players (ControlScore).

4.1.2 AggroScore and ControlScore vs. ANNs

The second experiment explores whether evolved artificial neural networks (i.e., scoring functions) with access to fifteen different observable game features (shown in Table 2.2) can perform better than the curated scoring functions found in SabberStone. In addition to the game state features considered by AggroScore and ControlScore, the ANNs consider observable properties of the players including the number of cards in each of the players' hands and decks, and the total mana cost of the turn player's hand. Each ANN also inputs the defensive power of the turn player indicated by the total health of all of their minions with taunt. The behavior characteristics for the evolved strategies considered the average number of turns it took to play games during evolution, and the average count of the cards in the player's hand per turn, as these features tend to best differentiate aggro and control playstyles. (Generally, aggro players seek to finish games in a shorter amount of turns than control strategies; see Appendix A for more details. Likewise, aggro players have less cards in their hand because they are rapidly using their cards to aggressively attack the opponent; see Figure B.2 for more details.) The idea is to see whether evolved scoring functions can better estimate the value of game states for Warlock players using the same decks described in the first experiment, *AggroScore vs. ControlScore*. The experimental setup for changing behavior characteristics is described in Table 4.2, whereas the matchups used in this experiment are described in Table 4.3. The hypothesis is that agents playing via evolved scoring functions should have higher average win rates compared to agents playing with SabberStone scoring functions.

Table 4.1 Experiment 1 Configurations[a]

P1Score	*P1Decks*	*P2Score*	*P2Decks*	*NumGames x Matchup*	*Total Num Games*
AggroScore	Aggro	AggroScore	Aggro	400	10000
AggroScore	Aggro	ControlScore	Control	400	10000
ControlScore	Control	ControlScore	Control	400	10000
AggroScore	Aggro	ControlScore	Aggro	400	10000
ControlScore	Control	AggroScore	Control	400	10000
ControlScore	Aggro	AggroScore	Control	400	10000

[a]These are the configurations used for each of the initial matchups.

Table 4.2 CMA-ME Behavior Configurations[a]

Network Name	*Num Games per ANN*	*Num ANNs To Evaluate*	*HandSize [Min,Max]*	*NumTurns [Min,Max]*	*PlayerScore, OpponentScore*
Warlock Net_CC_sm	100	5000	[1,7]	[5,15]	Control,Control
CvsNNC_2.0	100	5000	[1,7]	[25,35]	Control,Control
CvsNNC_ Large	200	50000	[1,9]	[5,45]	Control,Control
Warlock Net_AA_sm	100	5000	[1,7]	[5,15]	Aggro,Aggro
Warlock Net_AA_lg	200	50000	[1,9]	[5,45]	Aggro,Aggro

[a]These are the configurations used for each of the ANN evolutions via CMA-ME.

Table 4.3 Experiment 2 Configurations[a]

P1Score	*P1Decks*	*P2Score*	*P2Decks*	*NumGames x Matchup*	*Total Num Games*
AggroScore	Aggro	Warlock Net_AA_sm	Aggro	400	10000
AggroScore	Aggro	Warlock Net_AA_lg	Aggro	400	10000
ControlScore	Control	Warlock Net_CC_sm	Control	400	10000
ControlScore	Control	CvsNNC_2.0	Control	400	10000
ControlScore	Control	CvsNNC_ Large	Control	400	10000
Warlock Net_AA_lg	Aggro	CvsNNC_ Large	Control	400	10000

[a]These are the configurations used for each of the matchups comparing SabberStone scoring functions to ANNs heuristics.

4.2 Visualizing and Predicting Gameplay Strategies

After playing a number of games using different scoring functions, agents using AggroScore or ControlScore should then also show a difference in the types of turns they take throughout each game. As such, the statistics of player decisions for aggro players should be different than those for control players. Using this assumption, given any game statistics for a player using an unknown scoring function, a supervised learning model should be able to classify the gameplay strategy used as aggro or control. In addition, a projection of the players onto Cartesian space should show a clear separation of aggro players and control players.

4.2.1 Predicting Gameplay Strategies from Game Statistics

As seen in previous papers [8, 22, 57], supervised learning models perform differently based on domain. By testing classification algorithms using scikit-learn [41], at least one may correctly predict if a player is using an aggro or control gameplay strategy. In addition, by using k-Fold Cross Validation, this algorithm can train a model and find the split validation which produces the highest validation accuracy. Combined with exhaustive grid search for hyperparameter selection [2, 4, 5], the models trained via the input data should also show high values across different metrics [11, 14, 18].

All of the game features and their descriptions for the data sets used in this workflow are listed in Table 3.3, while small samples of the data sets can be found via Figures 3.10 and 3.11. The training and validation data sets consist of simulated players using AggroScore and ControlScore scoring functions. Each observation has 15 independent attributes which correspond to the game statistics of the current player. In addition, the dependent attributes are both labels for the current player (1 or 2) and for the gameplay strategy (0 for aggro, 1 for control). For this particular experiment, only the label for the gameplay strategy is considered.

Because the scoring functions evaluate states differently to execute different decisions, supervised learning models can be trained on the data for players using an aggro or control strategy. Five different supervised learning models are selected for learning the patterns of the player data. The hyperparameters for both support vector machine (classifier) and logistic regression include l1 and l2 regularization, as well as the C coefficient (0.1, 1, 10, 100). Random forest is tested with 50, 100, and 200 estimators. A decision tree classifier is tested with search criterion (gini, entropy), methods for determining splits (random, best), and max depth (4,5,6,7,8,9,10). Finally, a stochastic gradient descent classifier is tested with four different loss functions [hinge, log, perceptron and modified huber], l1 and l2 regularization, and alpha (learning rate) values of 1, 0.1, 0.01, 0.001, and 0.0001.

For testing each models' ability to generalize aggro and control strategies, players using ANNs as scoring functions for both aggro and control are used. The goal with this experiment is to determine if supervised learning models can be trained using two types of simulated players with their associated scoring function generalized as aggro and control (the umbrella gameplay strategy used). This label can also be assigned to players using a different scoring function evolved via CMA-ME, but still generalized to aggro or control. The hypothesis is that at least one of the models will be able to confidently score high across different classification metrics to correctly identify if a player is using an aggro or control strategy.

4.2.2 Visualizing Gameplay Strategies via PCA

One of the problems of understanding high dimensional data sets is that it becomes challenging to visualize as the number of dimensions grows [26]. Principal Component Analysis is a feature extraction method [50] commonly used to reduce the number of dimensions in a data set. If the number of Principal Components selected is small (less than four), then the data can be plotted on a PCA score plot [15] in Cartesian space to aid human perception of the data, which has seen success in other types of data like genome sequences [44].

Scoring functions in SabberStone have been shown to evaluate states quite differently. AggroScore for example, leads the agent towards game states that include an attack against the opponent's hero character. Meanwhile, ControlScore leads the agent towards game states that maintain more minions on the board. If agents continue with these types of decisions throughout their games, then their average turn statistics would also appear to be different because the actions they took during each turn would be different. The input data is first standardized such that the features are all on the same scale. This standardized data is separated into a train/validation set and a test set via 70/30 split validation. By fitting both sets onto the first two

principal component axes, they can then be visualized in a two-dimensional Cartesian plane.

Because the labels are known from the transformation, this can be molded into a supervised learning problem [3, 42]. Given the expectation that aggro and control players have different turn statistics, then when projected onto a lower dimension, it's possible that the points show a geometric separation as well. The support vector classification (SVC) model uses a margin of separation to find the maximum spread across sets of points from different classes. As such, the SVC model via *Predicting Gameplay Strategies from Game Statistics* is re-trained on the projected data to find an optimal separation in the principal component space of aggro players and control players. When visualizing the input data on the projected axes, the clear separation of aggro and control can become visually clear. In addition, the the model is tested with players using ANNs as scoring functions. Because the projected data has a lower number of columns, the model may also achieve a higher test accuracy. Using PCA, there exists a visualization which separates aggro and control strategies in Cartesian space. In addition, by training SVC on the reduced data, the model can achieve a higher test accuracy than in the original feature space.

CHAPTER 5

RESULTS

5.1 Experiment 1

Comparing across different matchups (more details in Appendices G and H), agents using ControlScore do perform better than AggroScore when pit against each other. In the mirror matches of aggro versus aggro and control versus control, both result in win rates of approximately 50% as shown in Figures H.1 and H.2. This can be expected because the random sample of games are played using players with the same decks and the same scoring functions. Comparatively, the data shows that when pitting aggro players against control players, the control players won with a 75% win rate shown in Figure H.2.

In addition, this observation carries forward when using players with differing scoring functions than the strategy for the deck they are using. After altering one of the players in the aggro matchup to use the ControlScore function, the agents using ControlScore with aggro decks won on average 77% against the agents using AggroScore with aggro decks, as shown in Figure H.4. When comparing these results with the control mirror matchup, agents using the ControlScore with control decks won on average 64% against the players using AggroScore with control decks. Finally, when analyzing the matchups where both sets of players used the opposite scoring function compared to their decks, the agents using ControlScore with aggro decks won on average 82% against the agents using AggroScore with control decks.

Agents using ControlScore are showing differences in their game features when compared to agents using the AggroScore, especially when considering the Hero Power. Control players are using their Hero Power to draw an additional card much more than the aggro players (Figure B.12). The use of this Hero Power can lead to having more cards in a control player's hand, allowing for the potential to have

increased board control. Even though the main goal behind an aggro deck is to aggressively attack the opponent, the AggroScore function mainly incentivizes these types of decisions, but does not take into account hand or board advantage. On the contrary, while control players generally aren't focused so much on attacking the opponent, the ControlScore still shows a positive weight in game states where the player's Health points are higher than the opponent's. As such, control players still have the opportunity to play aggro decks well, however with the inclusion of the directive on maintaining board control, the control players do not use their cards as heavily in the early turns. This can lead to more wins for the agents using the ControlScore.

5.2 Experiment 2

After using CMA-ME to evolve ANNs to evaluate game states, using agents in matchups against agents using the SabberStone scoring functions show contradicting results. When using the aggro mirror matchup as a baseline (average win rate of 50% shown via Figure H.1), the aggro players using evolved ANNs comparatively win on average 79% against aggro players, and when using a larger search space for ANNs, win on average 82%. The data shows that evolved aggro ANN players are better at using aggro-like strategies; for example, they play more minions per turn as shown in Figure B.6, and they have more minion attacks per turn as shown in Figure B.4. Both of these game features are shown to be more aggro-like, and given that evolved ANN players utilize these features more (combined with their increased performance against AggroScore players) shows that they are also stronger aggro players. This shows that the ANNs evolved via CMA-ME are better game state evaluators than the AggroScore scoring function for aggro players.

However, the results are not consistent when analyzing control players. In the set of mirror matchups of agents using ControlScore, still the performance is

roughly evenly distributed, where the ControlScore agent has an average win rate of about 50% (shown in Figure H.6). When using a candidate ANN found via the Warlock_Net_CC_sm setup, the agent using ControlScore won more than the agent with the ANN, with an average of 66% win rate. This particular setup for CMA-ME mimics that of the evolution for desired aggro players, as shown in Table 4.2. Not only was the search space for candidate ANNs too small, but the behavior characteristics configured a lower number of turns to win the game, and a lower average hand size, which are typically indicative of aggro players. As such, when using ANNs found via the CvsNNC_2.0 and CvsNNC_2.0_Large setups, there is a significant rise in the win rate for the agents using ANN solutions, 65% and 66% respectively. The evolved ANN solutions for control players also seem to be better at using game features which are shown to be more control. For example, these agents are drawing more cards per turn as shown in Figure B.1, likely due to the corresponding increase in the use of the Warlock Hero Power as shown in Figure B.12. This would then lead the evolved ANN control players to have more cards in their hand, which leads to more potential to maintain board control based on a variety of cards in their hand.

To observe that control is better than aggro, the networks from the Warlock_Net _AA_lg and CvsNNC_2.0 _Large setups were pit against each other to see if an evolved aggro or evolved control player would win. In this case, the evolved control player won with an average of 59% win rate, as shown in Figure H.12. When compared to *AggroScore vs ControlScore*, the ControlScore agents won with an average win rate of 75%. Although the players using ControlScore performed better against the players using an evolved control ANN, the difference is likely in part due to the fact that the opponent for the evolved control ANN players were evolved aggro ANN players, who are shown to perform better than the AggroScore players. This plays a strong impact on evolved ANN control players' ability to win in this particular matchup.

5.3 Experiment 3

Based on the results of exhaustive hyperparameter search with 5-fold cross validation, each of the classification models were able to predict with 99% accuracy, among other various other model metrics. This score was achieved on training and validation data using AggroScore and ControlScore players. The results of the hyperparameter search are below:

- Logistic Regression: l2 regularizer, C coefficient of 10
- Random Forest: 100 estimators
- Support Vector Classifier: l2 regularizer, C coefficient of 1
- Decision Tree Classifier: best splitters, max depth of 10, entropy criterion
- SGD CLassifier: modified huber loss, l2 regularizer, alpha 0.001

Table 5.1 Supervised Learning Model Comparison on Train Data[a]

Model	*Accuracy*	*Precision*	*Recall*	*F1 Score*	*AUC*
Logistic Regression	0.9986	0.9981	0.9992	0.9986	0.9986
Random Forest	0.9990	0.9988	0.9992	0.9990	0.9990
SVM	0.9987	0.9983	0.9992	0.9987	0.9987
Decision Tree	0.9981	0.9978	0.9983	0.9981	0.9981
SGD Classifier	0.9988	0.9985	0.9992	0.9988	0.9988

[a]The models are trained on player game statistics for AggroScore and ControlScore, and the goal is to predict aggro (positive) or control (negative).

In practice, not many models can reliably predict with this high of a score. The same models were also tested using different games' data, notably samples where the players were using evolved network scores. When testing these players' data, the models predicted significantly less. The decks used were kept the same, but the samples used were based on players using the evolved ANNs for the scoring functions. Using the same metrics from above, there is a significant dropoff in the performance of the models as shown in Table 5.2. Likely there is bias in the data used that the model can't predict well on completely new samples. It seems like this model has overfit to AggroScore and ControlScore players, but when exposed to simulated players using evolved networks as input for the scoring functions, the models cannot predict as well. Either the models need more players using different types of scoring functions, or the models need players using different types of decks.

Table 5.2 Supervised Learning Model Comparisson on Test Data[a]

Model	*Accuracy*	*Precision*	*Recall*	*F1 Score*	*AUC*
Logistic Regression	0.6106	0.7390	0.3418	0.4675	0.6106
Random Forest	0.6623	0.8201	0.4157	0.5518	0.6623
SVM	0.6120	0.7408	0.3446	0.4704	0.6120
Decision Tree	0.6817	0.7854	0.4999	0.6109	0.6817
SGD Classifier	0.6245	0.7708	0.3543	0.4854	0.6245

[a]The same models are tested on player data retrieved from agents using evolved scoring functions, and the goal is to predict aggro (positive) or control (negative).

5.4 Experiment 4

When analyzing the charts of the original data projected onto a smaller feature space, it's clear that Principal Component Analysis was able to successfully separate the players based on AggroScore and ControlScore (thus aggro and control players). Without using the knowledge of the labels, the points in the lower feature space show a distinct clustering with a region separating the two (shown without color in Figure 5.1). Because the labels of the training data are known, it's shown in Figure 5.2 that red points represent AggroScore (aggro) players, and blue points represent ControlScore (control) players. The separation isn't perfect, and there does appear to be a margin of error where some blue points are in the red cluster, and some red points are in the blue cluster. Likely, those particular points represent players whose turn by turn decisions somewhat overlapped with the opposite strategy of what they were using. For example, if a game ended very quickly, it's possible that a control player had very similar gameplay statistics to that of a defeated aggro player in an aggro mirror match.

Using a Biplot (Figure 5.3) to visualize how each of the input features load to the principal components, each of the summarized game features are also plotted as vectors which represent their load onto both of the principal component axes. Notably, the features that are the farthest apart on the horizontal PC1 axis contribute to most of the variability in the data. Some of these features include Number of Minion Attacks (Per Turn), Spells Played (Per Game), and Hero Power Activations (Per Game). This does make sense especially when looking at Figure B.4, Figure B.13, and Figure B.12. These Boxplots make it easy to see that the statistics vary from aggro to control, when comparing between AggroScore and ControlScore players.

Using support vector classification on the principal component decomposition of the data showed results that improve classification via the original feature space. The model was able to predict AggroScore and ControlScore players via Figure 5.4 with

an accuracy of 98%, which mimics the high validation accuracy shown in *Predicting Gameplay Strategies from Game Statistics*. Because this model was trained using players of AggroScore and ControlScore, the model was also able to predict highly on these players on validation data. However, when the model was tested using games of agents with evolved ANNs transformed onto the Principal Component axes, it could only predict with an accuracy of 74%. This shows an improvement to the test accuracy in the original feature space, which was 61%. Likely, because the data has been reduced to be represented as a linear combination of the features, the model did not suffer from having to learn the patterns of high number of columns.

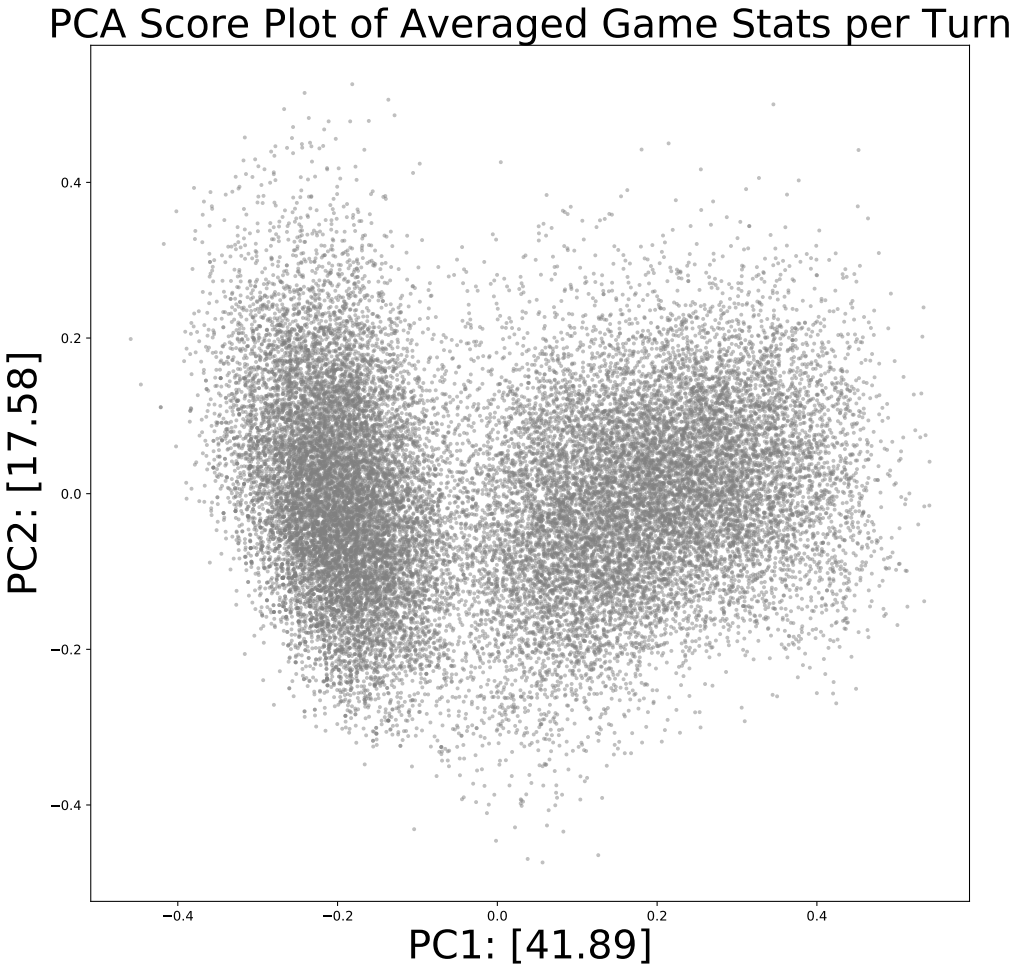

Figure 5.1 Principal Component Analysis decomposition of AggroScore and ControlScore players, shown without color.

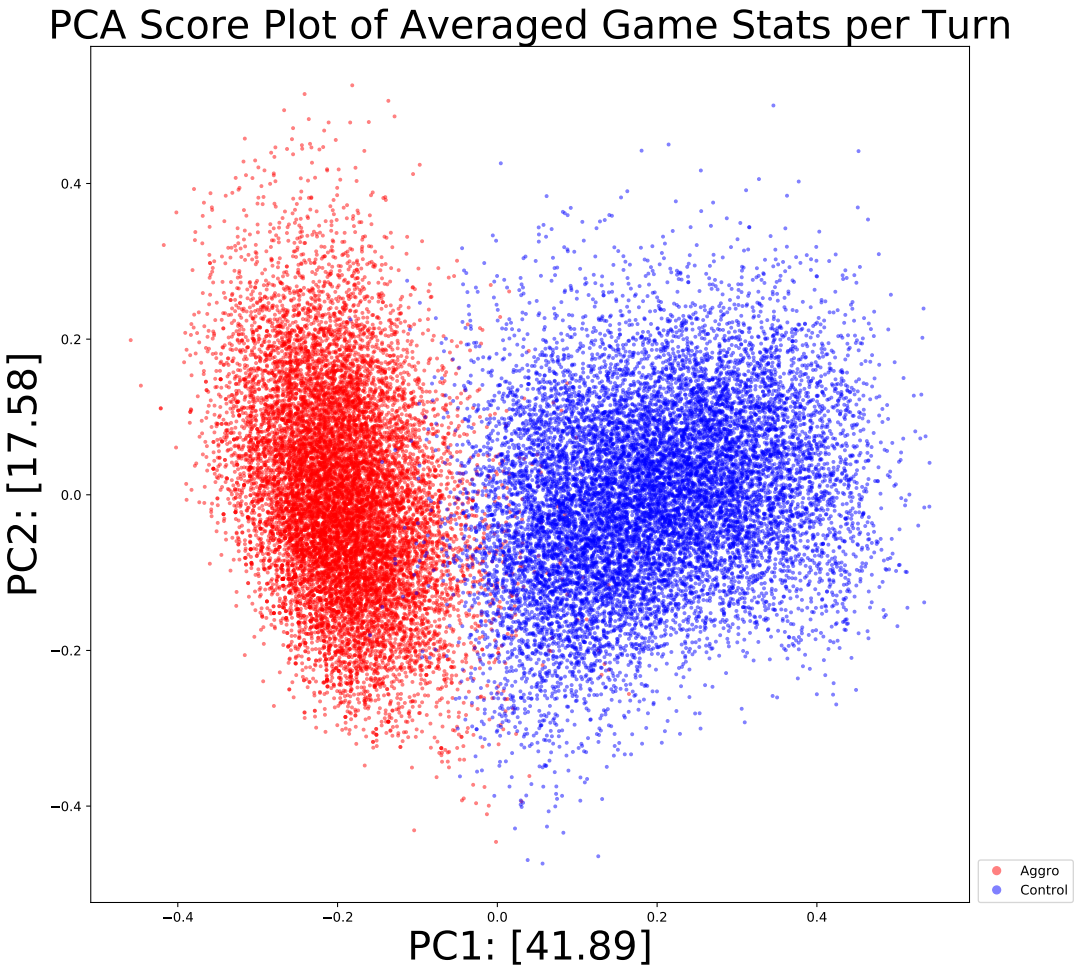

Figure 5.2 Principal Component Analysis decomposition of AggroScore (red) and ControlScore (blue) players.

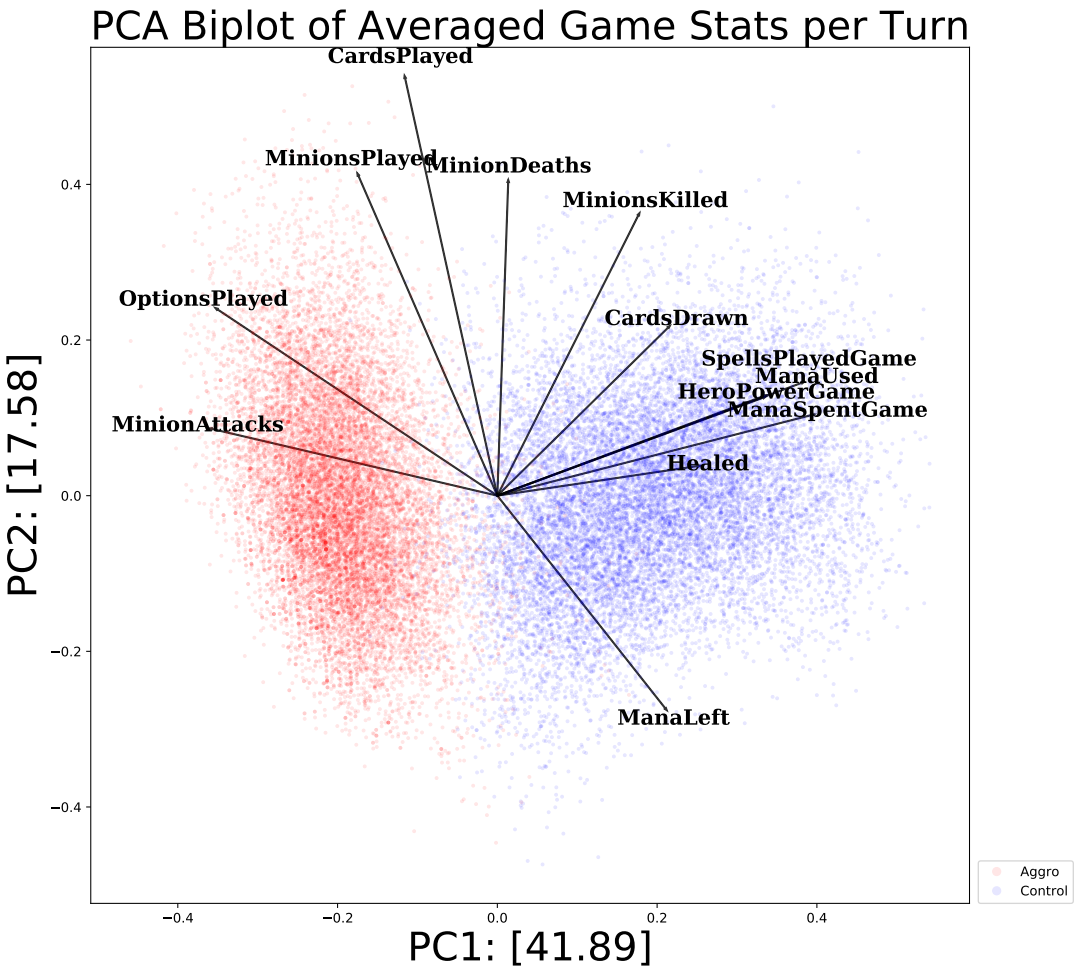

Figure 5.3 Principal Component Analysis Biplot of AggroScore and ControlScore players.

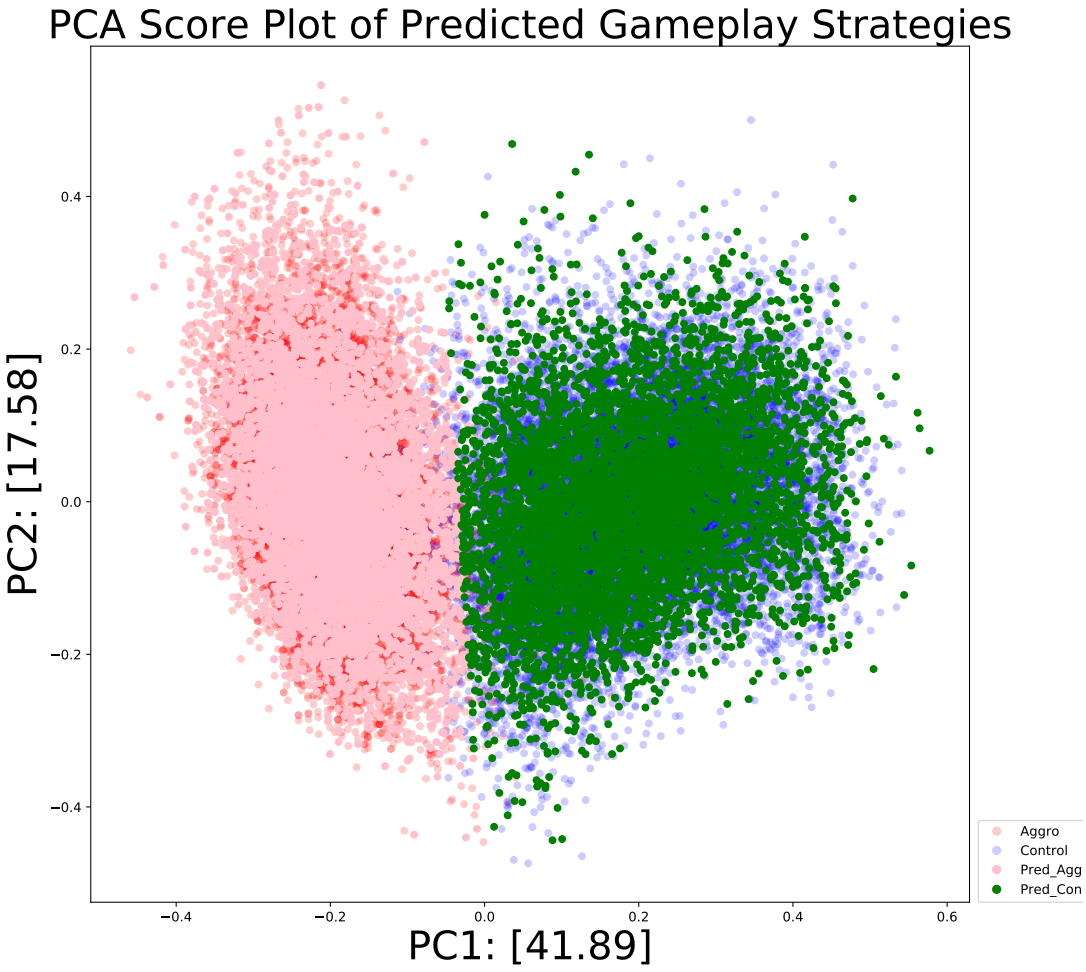

Figure 5.4 Principal Component Analysis Biplot of AggroScore and ControlScore players, and predictions for the corresponding test data as well (predicted Aggro is pink, predicted Control is green).

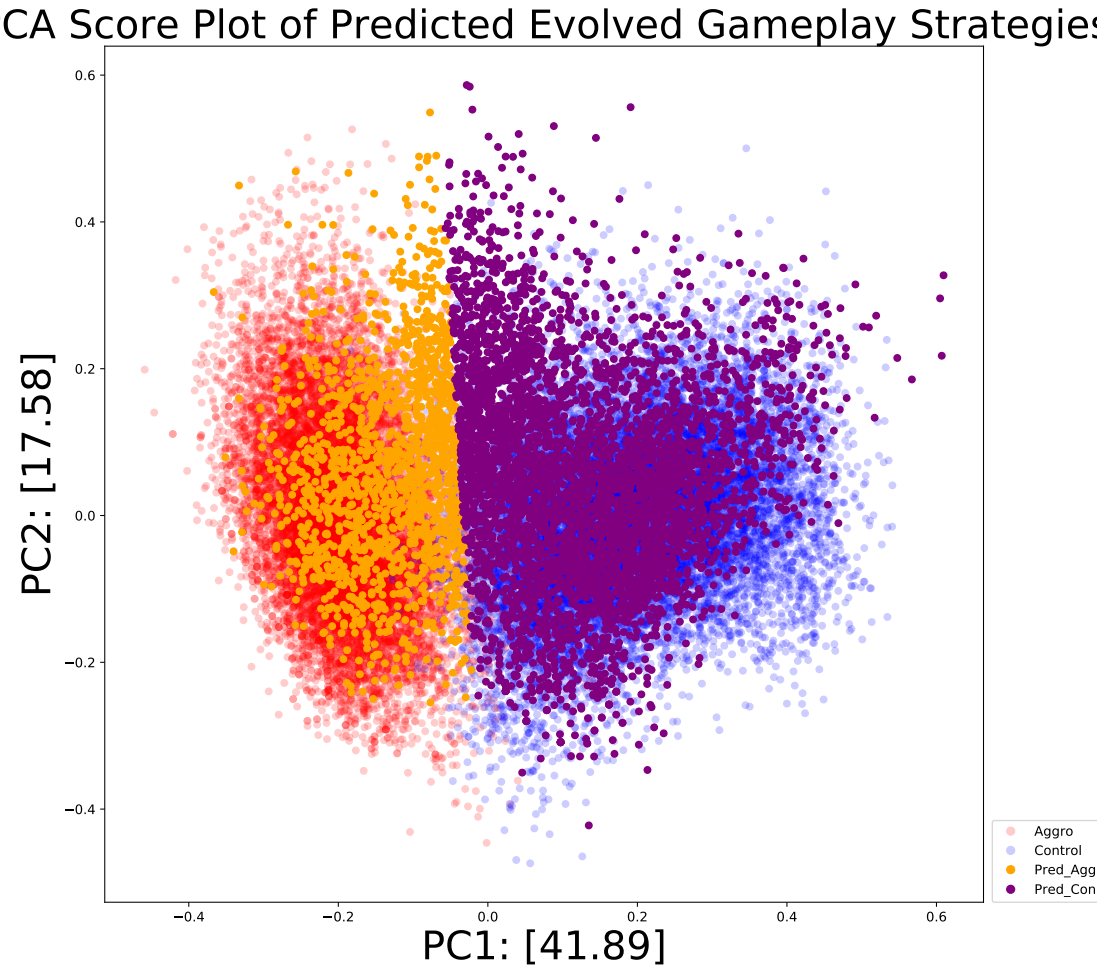

Figure 5.5 Principal Component Analysis Biplot of AggroScore and ControlScore players compared to the predictions of ANN scoring functions (predicted EvoAggro is orange, predicted EvoControl is purple).

CHAPTER 6

CONCLUSION

This thesis utilized several different analysis techniques to determine how separable aggro players and control players are in Hearthstone (particularly SabberStone), and even determined which players are better. In general, the notion from previous papers that control players perform better than aggro players is still true, at least within the Warlock hero class and the decks used in this paper. This notion is refined with the condition that ControlScore is a better heuristic scoring function than AggroScore. But, this also held true when using players with evolved ANNs, as the evolved ANN control players had a higher win rate against evolved ANN aggro players.

Given the performance difference, there must be some underlying structural differences between the games that can numerically separate aggro from control. Several supervised learning models were used including support vector classifier, logistic regression, and random forest. An exhaustive grid search was used to find the optimal hyperparameters for each of the models, including 5-fold cross validation, and each showed an average validation accuracy of 99%. This seems to be overfit for the AggroScore and ControlScore functions, because when tested using games of agents using evolved ANNs for game state evaluation, the test accuracies dropped to about 65%. It's possible the models were not built with enough data, and in order to be more general, the input data would need to include games of agents using evolved ANNs as well. But, this still goes to show that given a game's worth a data, a model can predict with relatively high accuracy whether the player is aggro or control.

In order to visualize the differences between aggro players and control players, Principal Component Analysis was used to reduce the original input data into a feature space which can be easily visualized. Using the players' gameplay strategies as labels, PCA can be used to visualize gameplay strategies, which can then be used as

input for supervised learning models to predict whether a player is aggro or control. Reducing the original feature space also showed a corresponding improvement on test accuracy. This may be useful in player modeling in Hearthstone, which can be used in other agents attempting to identify an opponent's strategy during gameplay execution.

6.1 Future Work

In the logs, there are actually more than 15 features captured from each agent. In future tests of CMA-ME, it would be interesting to try and alter the topology of the hidden layers to test different types of ANNs, and potentially add more observable game features to the input layer. This may lead to more refined heuristic scoring functions. In addition, there may be other ways to approach the supervised learning problem of predicting an opponent's gameplay strategy. First, the models as they are currently implemented should be tested with more input data including players using different decks, different scoring functions, and different hero characters. But, if a game is instead looked at as a series of turns, then the question of predicting a player's gameplay strategy can be molded into a time series problem. For example, at what turn in the game can a model best predict a players gameplay strategy? This can have many use cases, for example developing an AI agent which can execute a desired gameplay strategy in early turns, but then begin to optimize counters once it has determined the opponent's gameplay strategy.

6.2 Final Statement

Although preliminary, this thesis helps quantify differences between aggro and control players. It shows that games can be reduced to visualize the geometric distance between the two gameplay strategies, and models can even be trained to predict a player's gameplay strategy. However, the study would like to grow such that it doesn't encompass the small subset of cards and decks used in this paper. Hearthstone is a constantly changing game due to the rotating standard format, as well as with the booster packs that are released. But with the continued application of AI methods, gameplay strategies can be correctly identified for potential use during gameplay or for post-game analysis and player modeling.

APPENDIX A

HISTOGRAMS FOR THE NUMBER OF TURNS PER SCORING MATCHUP

Figures A.1 to A.9 show the frequency of the number of turns per game over the collections of all games for each matchup pairing all decks using the designated heuristic scoring functions.

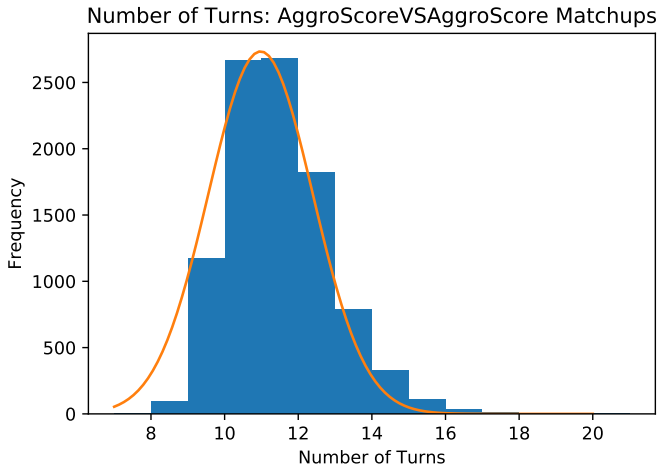

Figure A.1 Distribution of the number of turns per game for AggroScore vs AggroScore matchups.

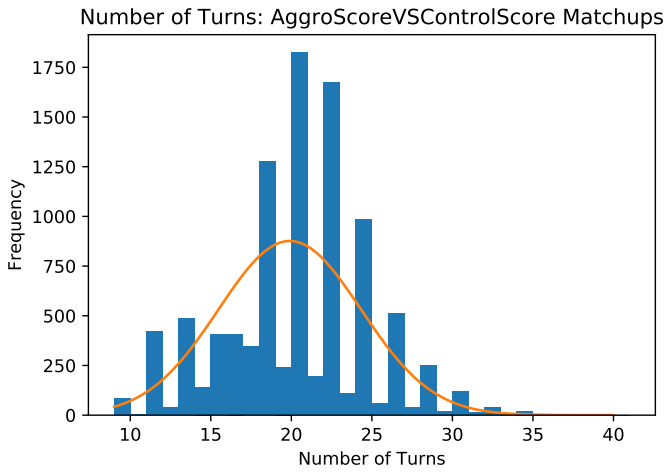

Figure A.2 Distribution of the number of turns per game for AggroScore vs ControlScore matchups.

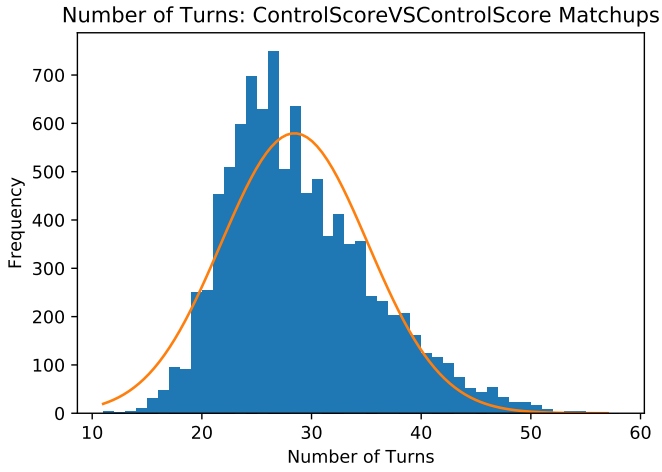

Figure A.3 Distribution of the number of turns per game for ControlScore vs ControlScore matchups.

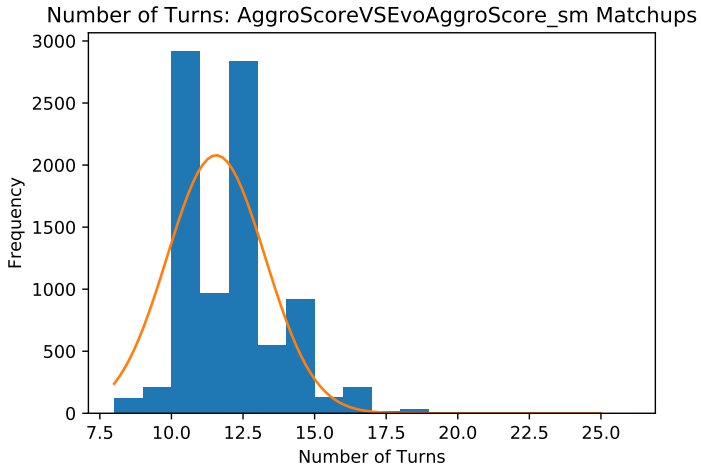

Figure A.4 Distribution of the number of turns per game for AggroScore vs Evolved ANN (aggro) matchups. This ANN was evolved using the Warlock_Net_AA_sm setup.

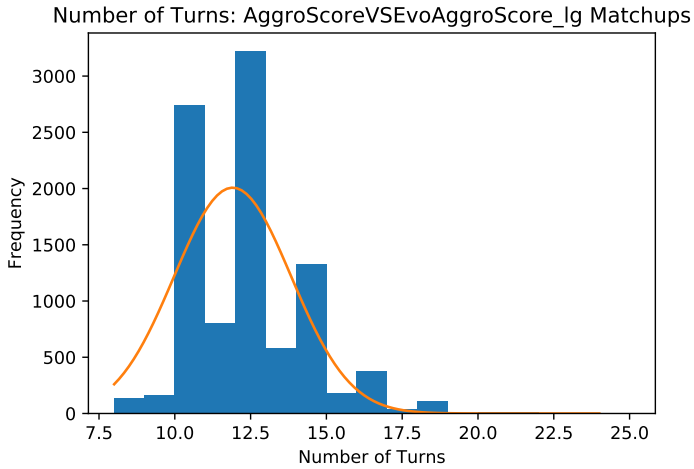

Figure A.5 Distribution of the number of turns per game for AggroScore vs Evolved ANN (aggro) matchups. This ANN was evolved using the Warlock_Net_AA_lg setup.

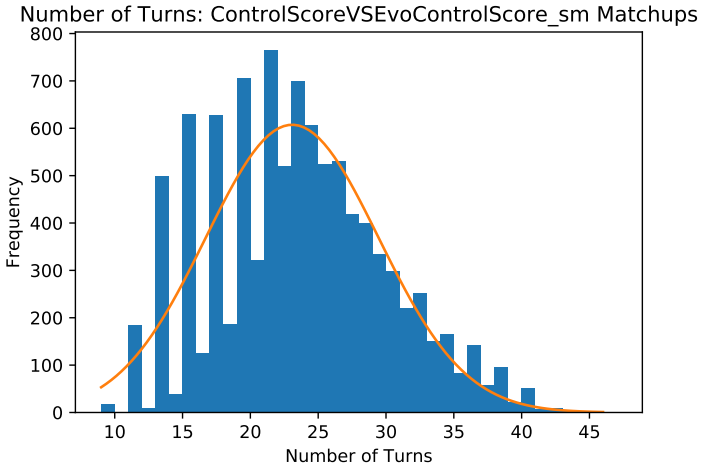

Figure A.6 Distribution of the number of turns per game for ControlScore vs Evolved ANN (control) matchups. This ANN was evolved using the Warlock_Net_CC_sm setup.

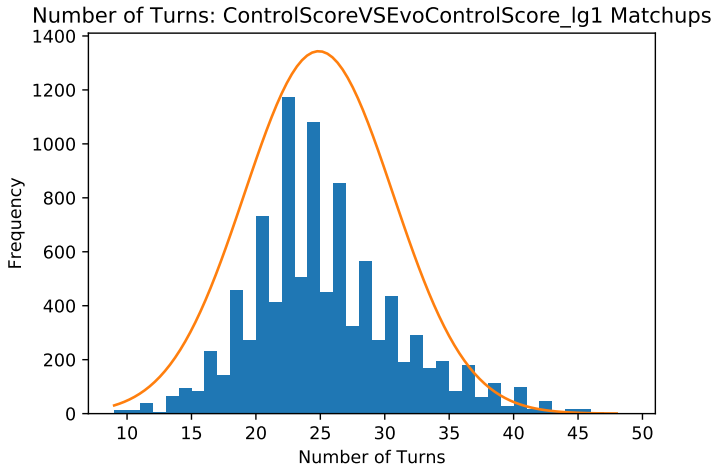

Figure A.7 Distribution of the number of turns per game for ControlScore vs Evolved ANN (control) matchups. This ANN was evolved using the CvsNNC_2.0 setup.

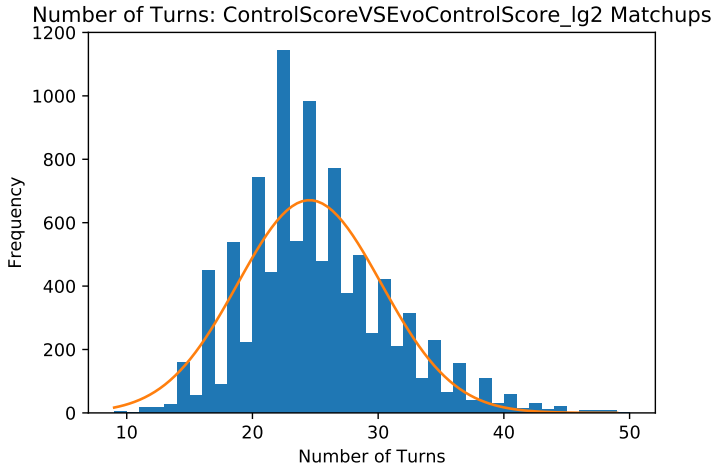

Figure A.8 Distribution of the number of turns per game for ControlScore vs Evolved ANN (control) matchups. This ANN was evolved using the CvsNNC_Large setup.

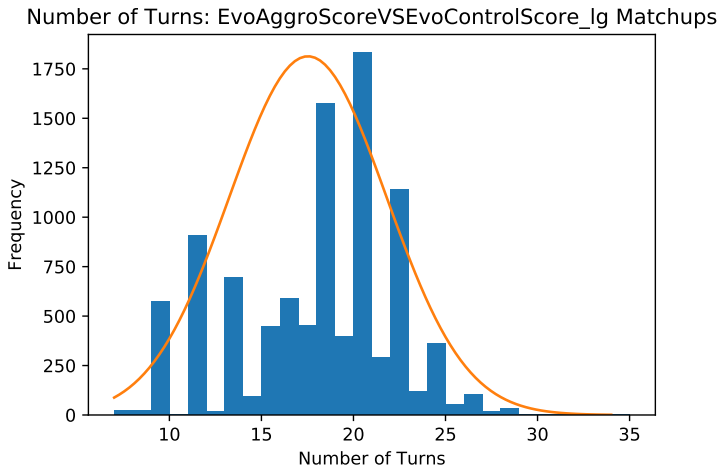

Figure A.9 Distribution of the number of turns per game for Evolved ANN (aggro) vs Evolved ANN (control) matchups. These ANNs were evolved using the Warlock_Net_AA_lg and the Warlock_Net_CC_lg setups.

APPENDIX B

BOXPLOTS FOR COLUMN DISTRIBUTIONS - GAME STATISTICS

Figures B.1 to B.15 show the distribution of values for the numerical features for simulated agents using each scoring function. The statistics for players using evolved aggro and evolved control ANN heuristics are retrieved using the ANNs from the Warlock_Net_AA_lg and Warlock_Net_CC_lg configurations described in Section 4.1.

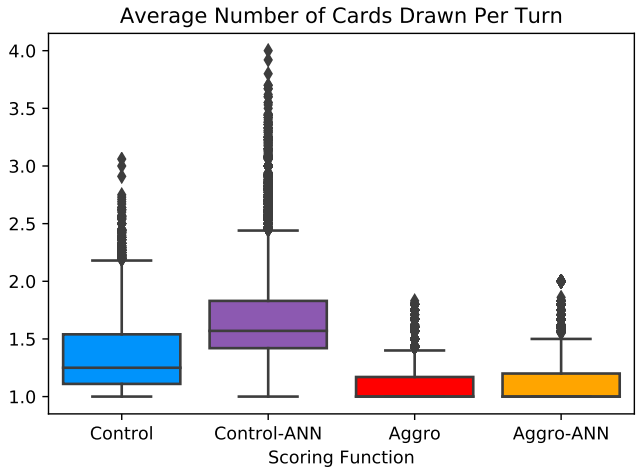

Figure B.1 Boxplot comparing the distribution of number of cards drawn per turn across four scoring functions. Sample size of 9644 players.

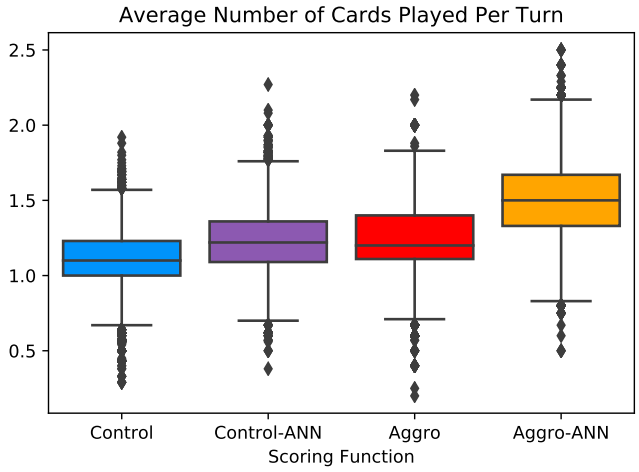

Figure B.2 Boxplot comparing the distribution of number of cards played per turn across four scoring functions. Sample size of 9644 players.

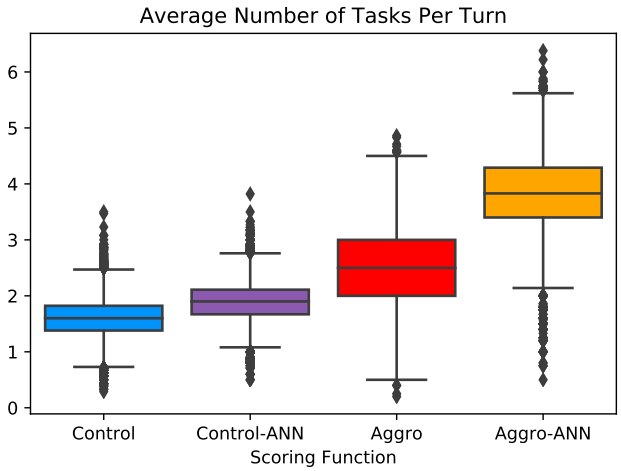

Figure B.3 Boxplot comparing the distribution of options (tasks) played per turn across four scoring functions. Sample size of 9644 players.

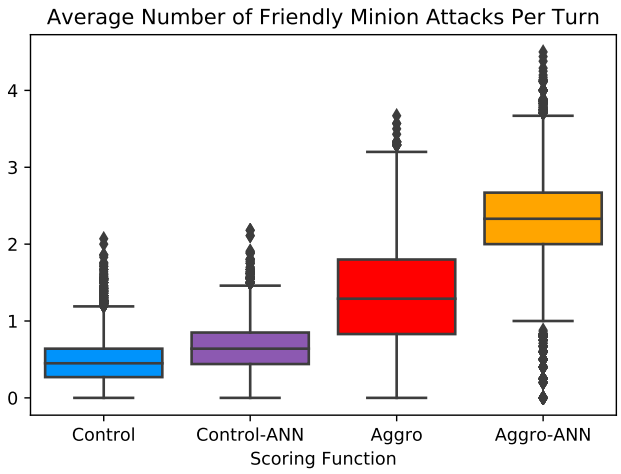

Figure B.4 Boxplot comparing the distribution of number of friendly minions that attacked per turn across four scoring functions. Sample size of 9644 players.

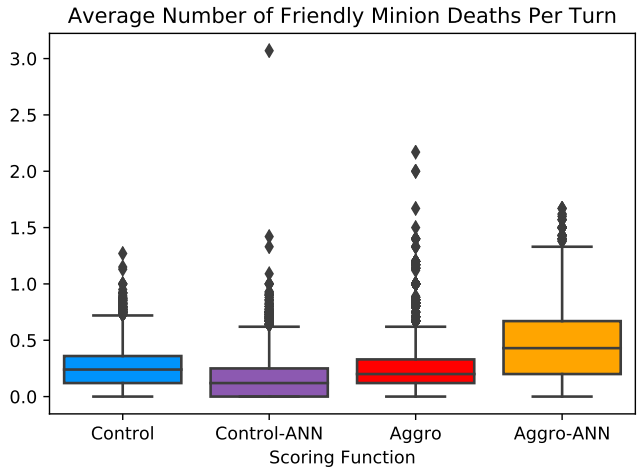

Figure B.5 Boxplot comparing the distribution of number of friendly minion deaths per turn across four scoring functions. Sample size of 9644 players.

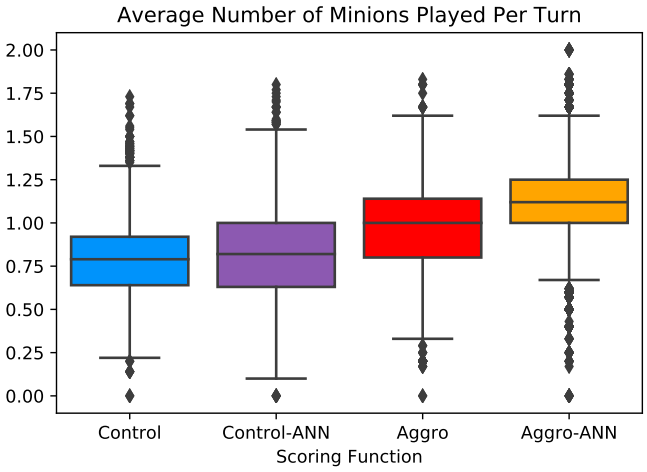

Figure B.6 Boxplot comparing the distribution of average amount of minions played per turn across four scoring functions. Sample size of 9644 players.

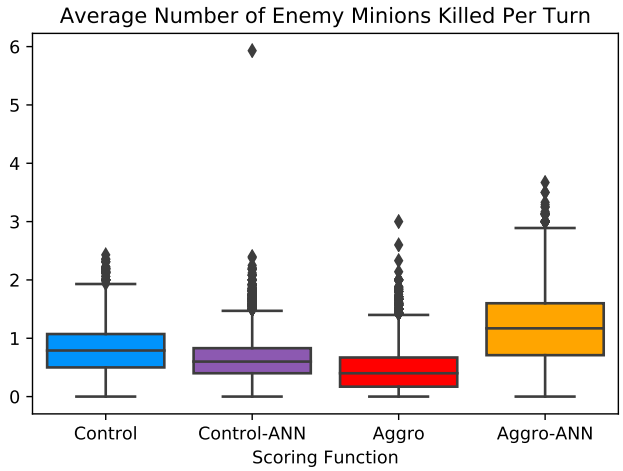

Figure B.7 Boxplot comparing the distribution of average number of opponent minions killed per turn across four scoring functions. Sample size of 9644 players.

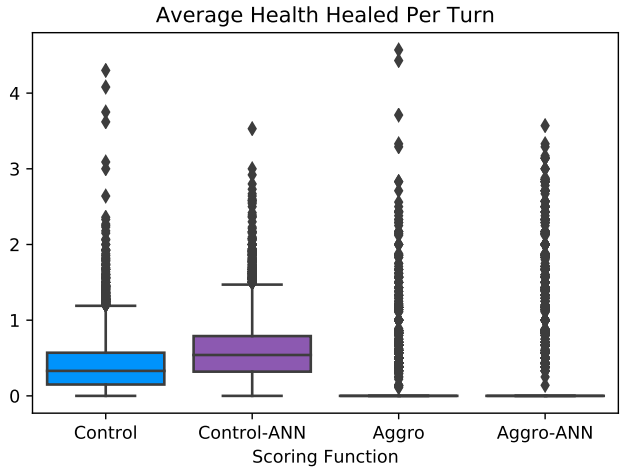

Figure B.8 Boxplot comparing the distribution of average health healed per turn across four scoring functions. Sample size of 9644 players.

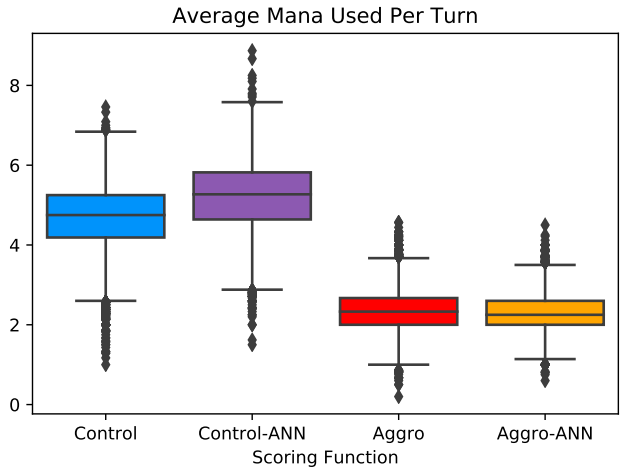

Figure B.9 Boxplot comparing the distribution of average amount of mana used per turn across four scoring functions. Sample size of 9644 players.

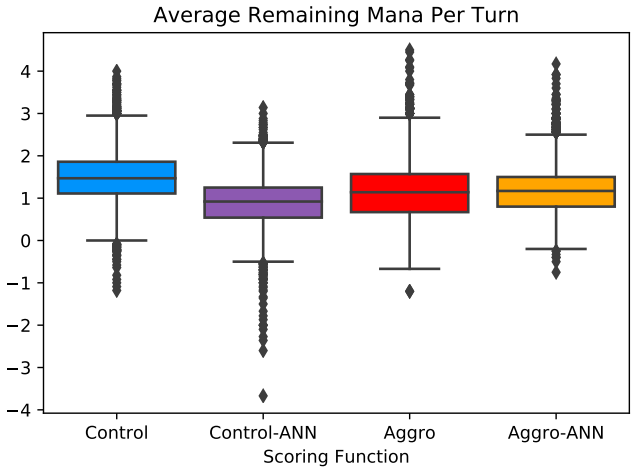

Figure B.10 Boxplot comparing the distribution of average amount of mana remaining per turn across four scoring functions. Sample size of 9644 players.

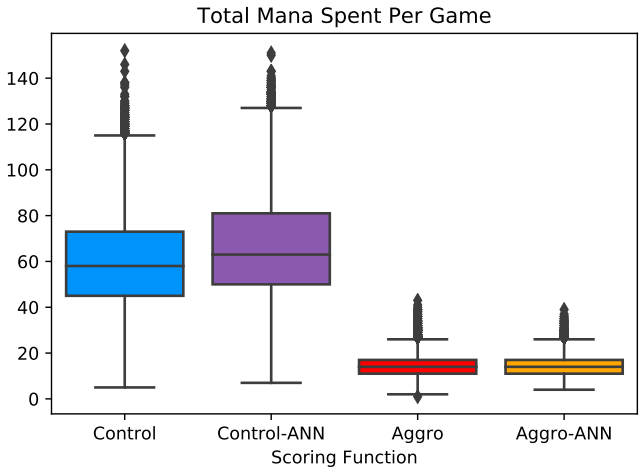

Figure B.11 Boxplot comparing the distribution of total amount of mana spent per game across four scoring functions. Sample size of 9644 players.

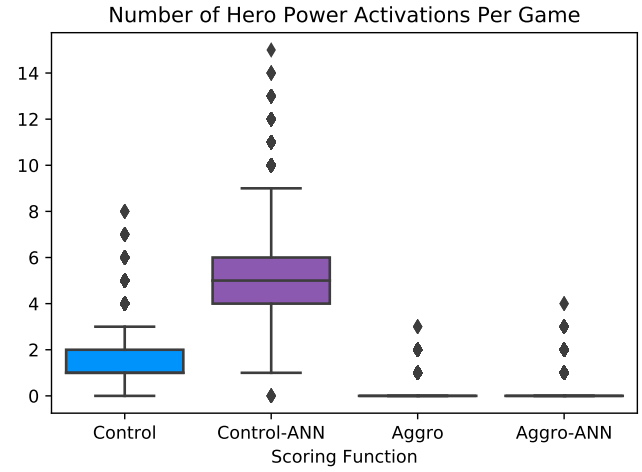

Figure B.12 Boxplot comparing the distribution of total number of Hero Power activations used per game across four scoring functions. Sample size of 9644 players.

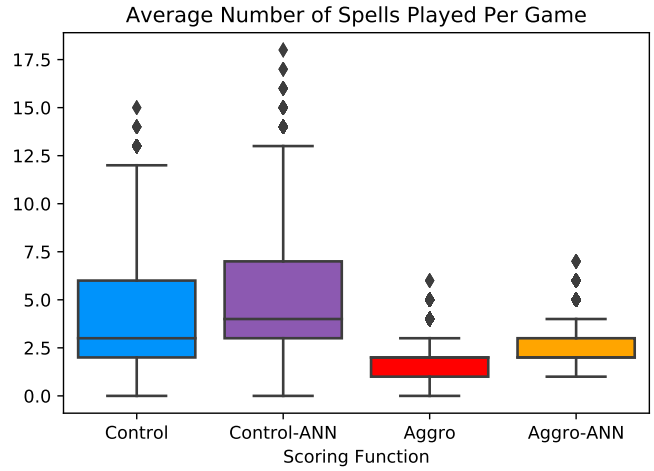

Figure B.13 Boxplot comparing the distribution of total number of spells played per game across four scoring functions. Sample size of 9644 players.

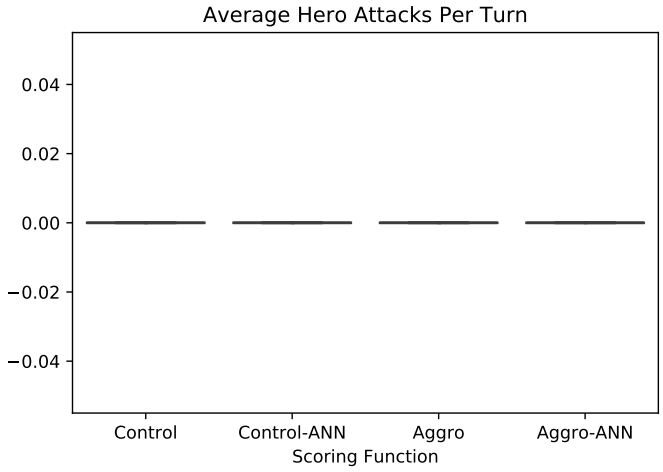

Figure B.14 Boxplot comparing the distribution of average amount of hero attacks per turn across four scoring functions. Sample size of 9644 games. (NOTE: None of the decks for Experiments 1-3 have a weapon, so the hero cannot attack. This column is removed.)

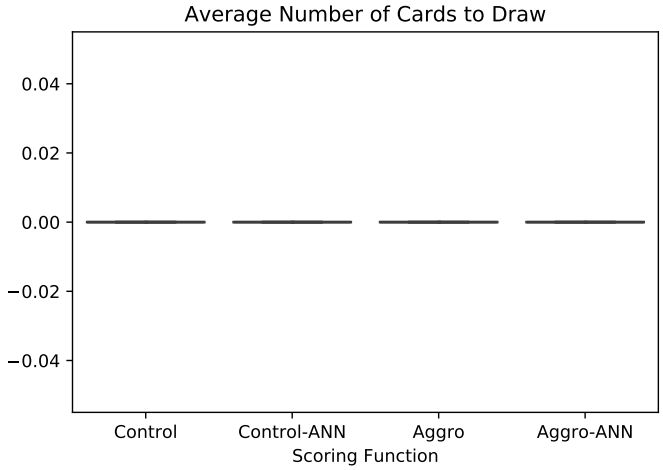

Figure B.15 Boxplot comparing the distribution of average number of cards left to draw across four scoring functions. Sample size of 9644 games. (NOTE: Because this feature is never used, this column is also removed.)

APPENDIX C
CMA-ME HEATMAPS

Figures C.1 to C.12 show the candidate ANNs plotted on a grid (similar to MAP-Elites), where the color hue represents win rate of the ANN across N games (specified in each figure), and the axes correspond to two behavior characteristics designed for the set of candidate solutions. This is explained further in Section 2.6.

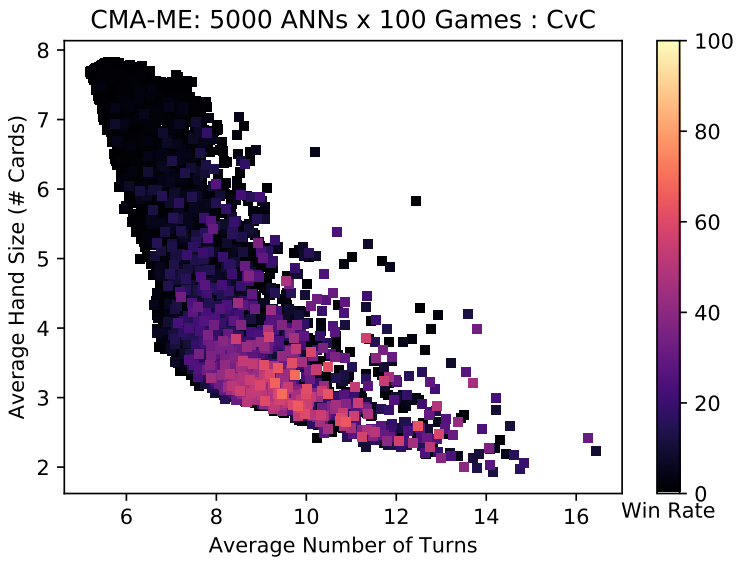

Figure C.1 Candidate solutions found via the Warlock_Net_CC_sm configuration.

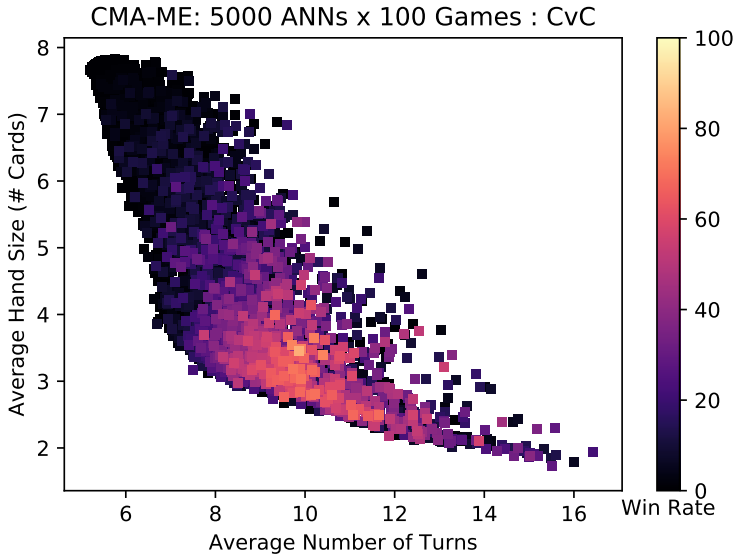

Figure C.2 Candidate solutions found via the CvsNNC_2.0 configuration.

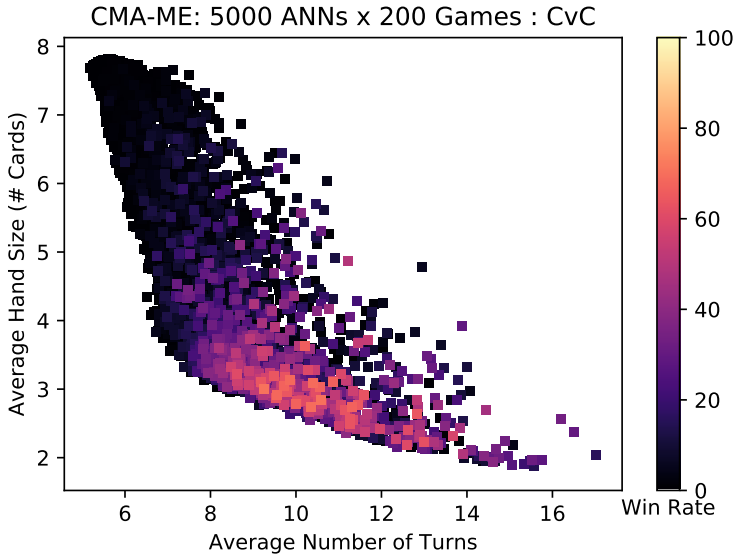

Figure C.3 Candidate solutions found via the CvsNNC_Large configuration.

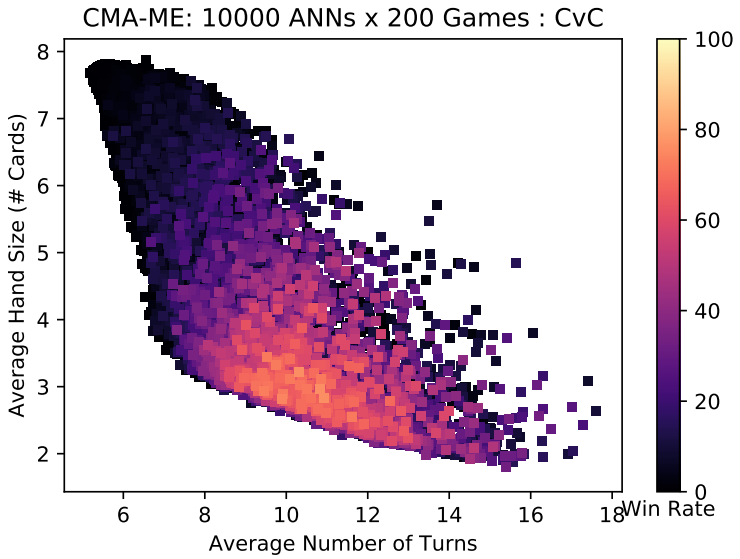

Figure C.4 Candidate solutions found via the CvsNNC_Large configuration.

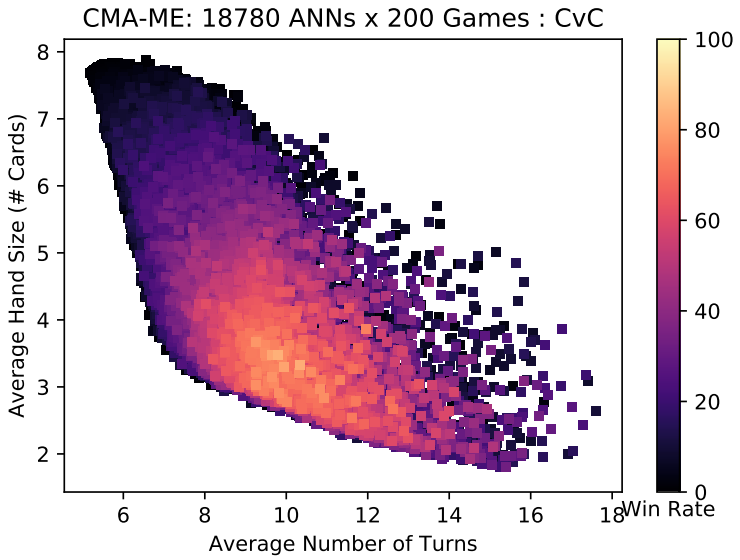

Figure C.5 Candidate solutions found via the CvsNNC_Large configuration. This run stopped short early, but should have evaluated 50000 ANNs.

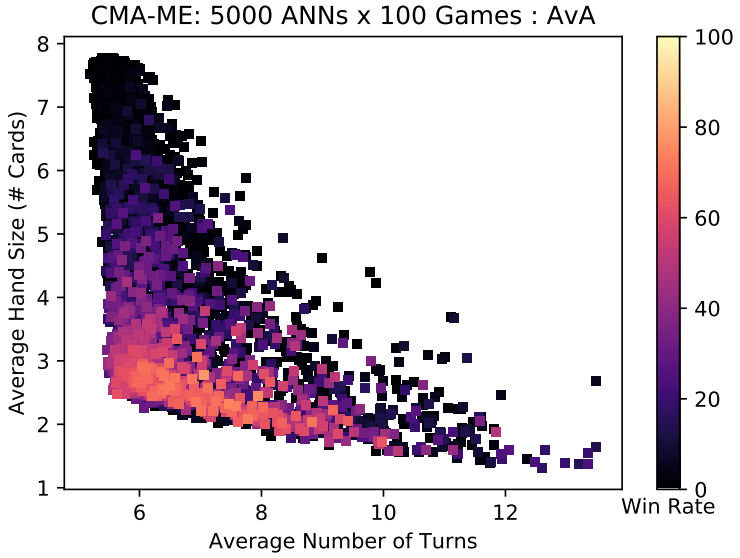

Figure C.6 Candidate solutions found via the Warlock_Net_AA_sm configuration.

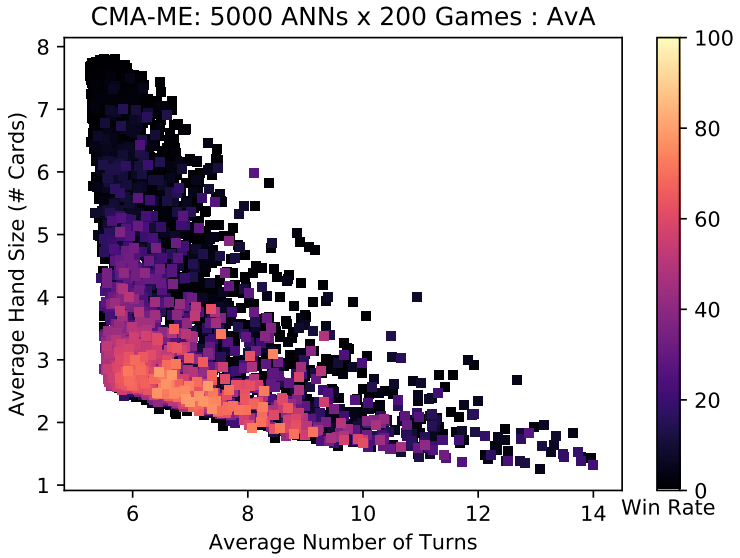

Figure C.7 Candidate solutions found via the Warlock_Net_AA_lg configuration.

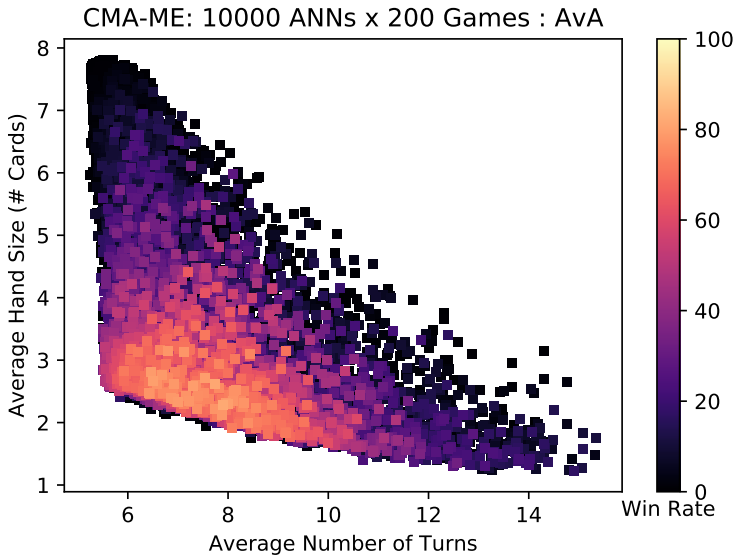

Figure C.8 Candidate solutions found via the Warlock_Net_AA_lg configuration.

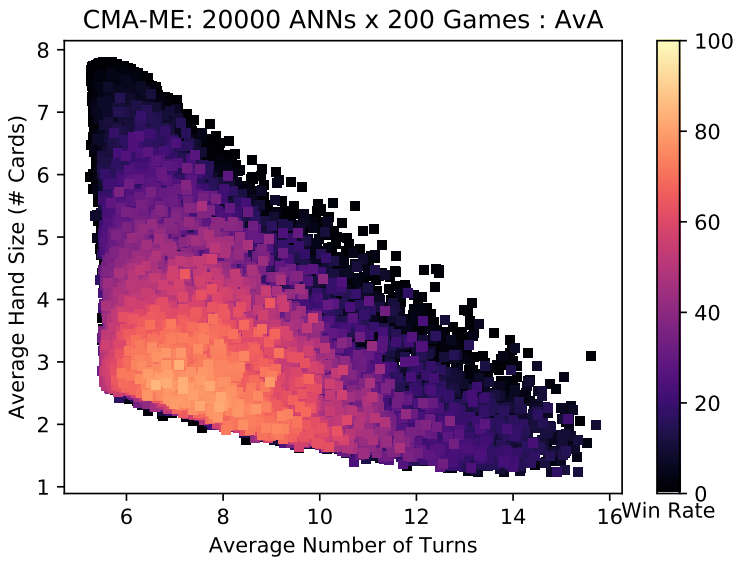

Figure C.9 Candidate solutions found via the Warlock_Net_AA_lg configuration.

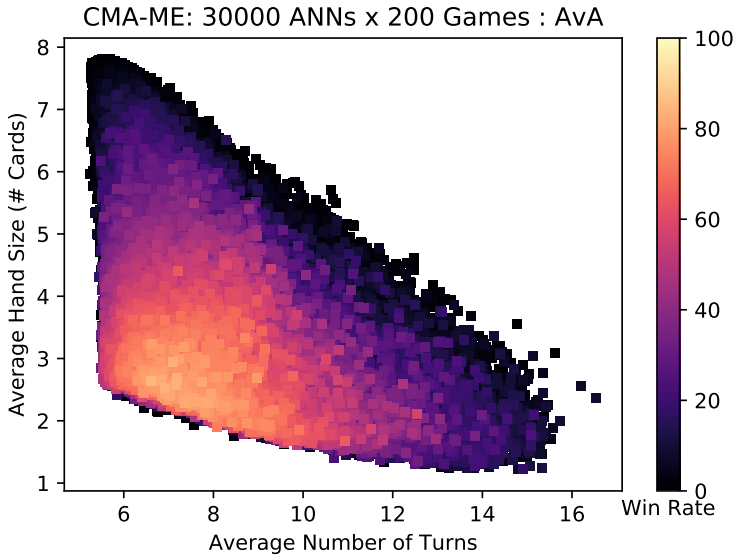

Figure C.10 Candidate solutions found via the Warlock_Net_AA_lg configuration.

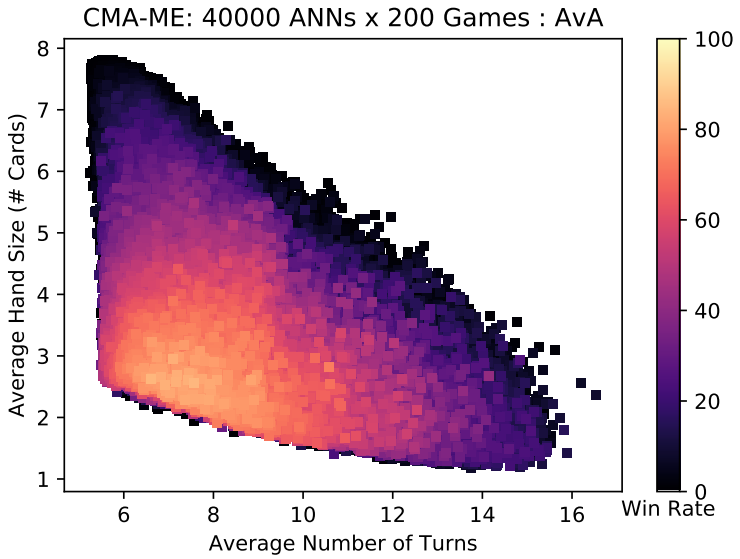

Figure C.11 Candidate solutions found via the Warlock_Net_AA_lg configuration.

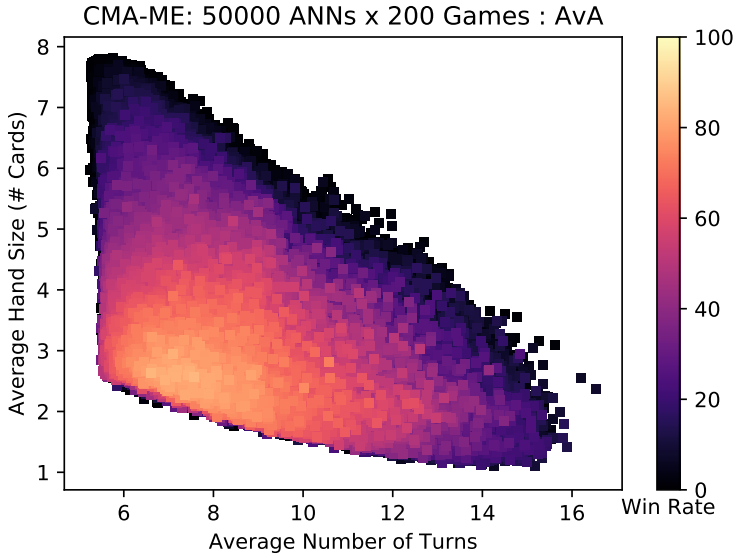

Figure C.12 Candidate solutions found via the Warlock_Net_AA_lg configuration.

APPENDIX D
RISE OF SHADOWS DECKLISTS MANA CURVES

Figures D.1 to D.10 show the Mana Curves for the decklists used during the experiments. Mana Curves are described more in Section 2.3.

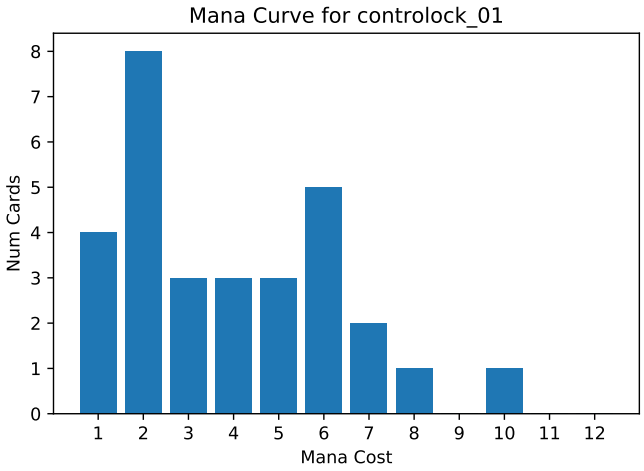

Figure D.1 Mana Curve for control deck 1 via Table E.1

.

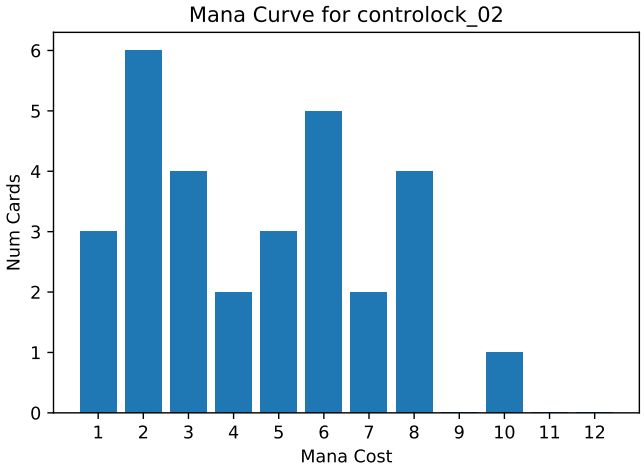

Figure D.2 Mana Curve for control deck 2 via Table E.2.

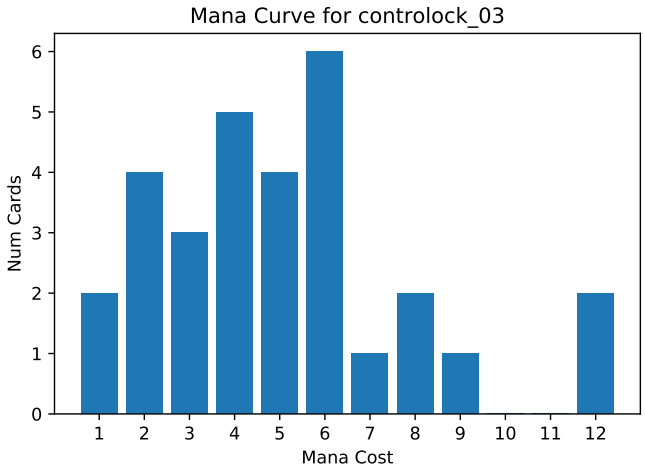

Figure D.3 Mana Curve for control deck 3 via Table E.3.

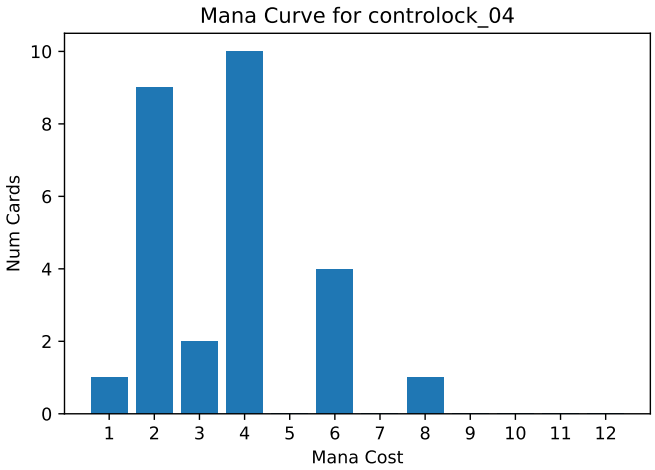

Figure D.4 Mana Curve for control deck 4 via Table E.4.

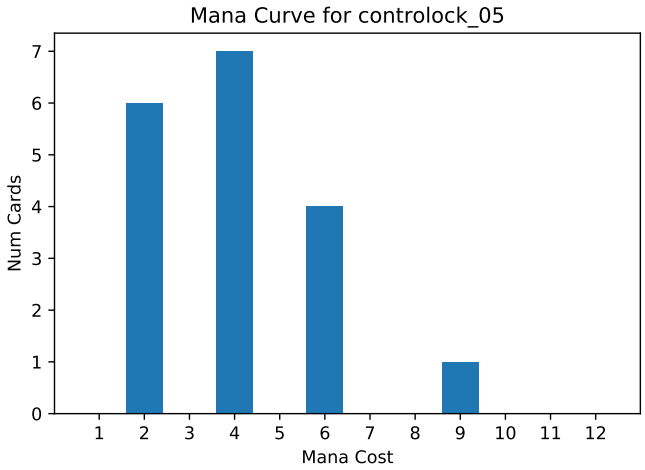

Figure D.5 Mana Curve for control deck 5 via Table E.5.

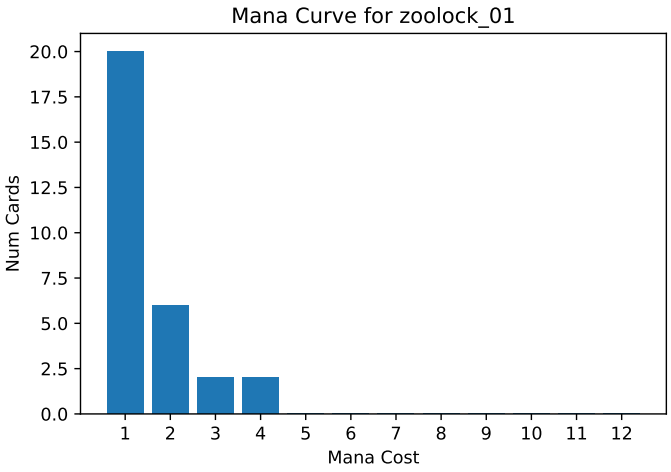

Figure D.6 Mana Curve for aggro deck 1 via Table E.6.

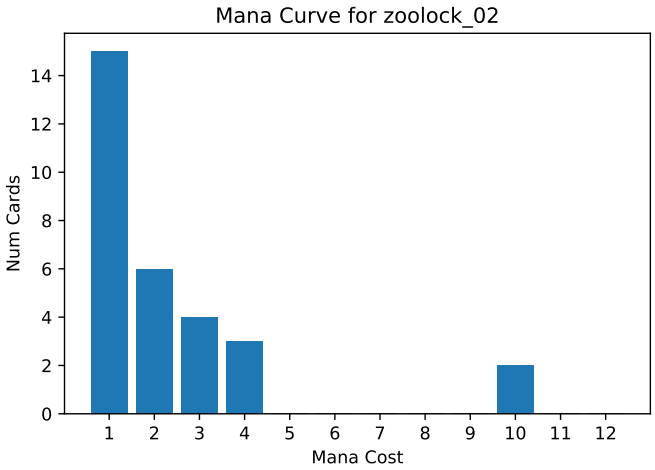

Figure D.7 Mana Curve for aggro deck 2 via Table E.7.

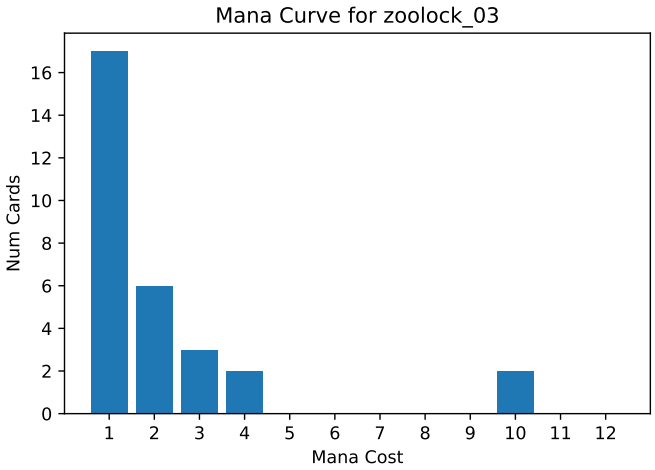

Figure D.8 Mana Curve for aggro deck 3 via Table E.8.

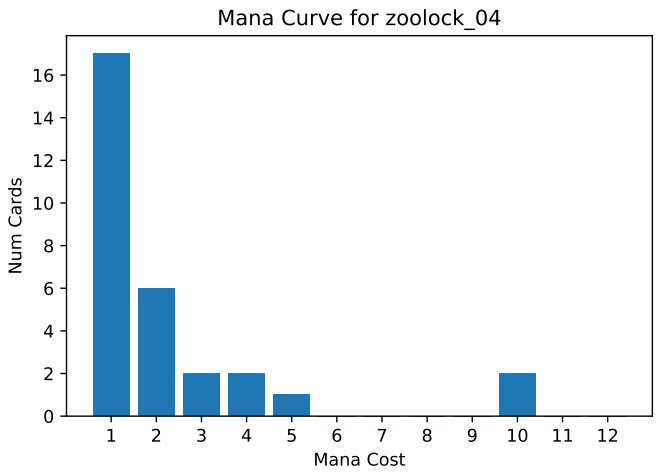

Figure D.9 Mana Curve for aggro deck 4 via Table E.9.

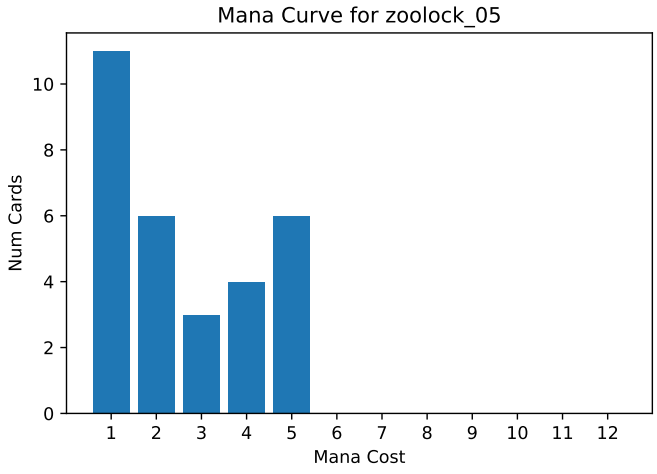

Figure D.10 Mana Curve for aggro deck 5 via Table E.10.

APPENDIX E
RISE OF SHADOWS DECKLISTS DESCRIPTIONS

Tables E.1 to E.10 show the descriptions for the decklists used during the experiments.

Table E.1 Decklist Control 01[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Mortal Coil	2	1	Spell	Warlock
Shriek	2	1	Spell	Warlock
Curse of Weakness	2	2	Spell	Warlock
Plot Twist	2	2	Spell	Warlock
Reckless Diretroll	2	2	Minion	Warlock
Doomsayer	2	2	Minion	Neutral
Sense Demons	2	3	Spell	Warlock
Augmented Elekk	1	3	Minion	Neutral
Hellfire	2	4	Spell	Warlock
High Priestess Jeklik	1	4	Minion	Warlock
Zilliax	1	5	Minion	Neutral
Rotten Applebaum	2	5	Minion	Neutral
Soulwarden	2	6	Minion	Warlock
Mossy Horror	1	6	Minion	Neutral
Aranasi Broodmother	2	6	Minion	Warlock
Lord Godfrey	1	7	Minion	Warlock
Arch-Villain Rafaam	1	7	Minion	Warlock
Fel Lord Betrug	1	8	Minion	Warlock
Hakkar, the Soulflayer	1	10	Minion	Neutral

[a]This is the decklist corresponding to control deck 1. Received via `https://www.hearthstonetopdecks.com/decks/shadows-fatigue-warlock-ft-hakkar/`

Table E.2 Decklist Control 02[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Mortal Coil	2	1	Spell	Warlock
The Soularium	1	1	Spell	Warlock
Plot Twist	2	2	Spell	Warlock
Acidic Swamp Ooze	2	2	Minion	Neutral
Doomsayer	2	2	Minion	Neutral
Sense Demons	1	3	Spell	Warlock
Shadow Bolt	1	3	Spell	Warlock
Voodoo Doll	2	3	Minion	Neutral
Hellfire	2	4	Spell	Warlock
Zilliax	1	5	Minion	Neutral
Rotten Applebaum	2	5	Minion	Neutral
Aranasi Broodmother	2	6	Minion	Warlock
Siphon Soul	2	6	Spell	Warlock
Safeguard	1	6	Minion	Neutral
Lord Godfrey	1	7	Minion	Warlock
Arch-Villain Rafaam	1	7	Minion	Warlock
Fel Lord Betrug	1	8	Minion	Warlock
Deranged Doctor	1	8	Minion	Neutral
Twisting Nether	2	8	Spell	Warlock
Hakkar, the Soulflayer	1	10	Minion	Neutral

[a]This is the decklist corresponding to control deck 2. Received via `https://www.hearthstonetopdecks.com/decks/control-hakkar-warlock-rise-of-shadows-thijs/`

Table E.3 Decklist Control 03[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Mortal Coil	2	1	Spell	Warlock
Curse of Weakness	2	2	Spell	Warlock
Doomsayer	2	2	Minion	Neutral
Sense Demons	1	3	Spell	Warlock
Shadow Bolt	2	3	Spell	Warlock
Hellfire	1	4	Spell	Warlock
Shadowflame	2	4	Spell	Warlock
Twilight Drake	2	4	Minion	Neutral
Omega Agent	2	5	Minion	Warlock
Big Game Hunter	1	5	Minion	Neutral
Barista Lynchen	1	5	Minion	Neutral
Siphon Soul	2	6	Spell	Warlock
Mossy Horror	2	6	Minion	Neutral
Aranasi Broodmother	2	6	Minion	Warlock
Lord Godfrey	1	7	Minion	Warlock
Twisting Nether	2	8	Spell	Warlock
Lord Jaraxxus	1	9	Minion	Warlock
Mountain Giant	2	12	Minion	Neutral

[a]This is the decklist corresponding to control deck 3. Received via `https://www.hearthstonetopdecks.com/decks/omega-control-warlock-shadows-post-nerf-top-500-legend-novamograph/`

Table E.4 Decklist Control 04[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Mortal Coil	1	1	Spell	Warlock
Sunfury Protector	2	2	Minion	Neutral
Doomsayer	2	2	Minion	Neutral
Ancient Watcher	2	2	Minion	Neutral
Acidic Swamp Ooze	1	2	Minion	Neutral
Plot Twist	2	2	Spell	Warlock
Faceless Rager	2	3	Minion	Neutral
Twilight Drake	2	4	Minion	Neutral
Shadowflame	2	4	Spell	Warlock
Hellfire	2	4	Spell	Warlock
Proud Defender	2	4	Minion	Neutral
Spellbreaker	2	4	Minion	Neutral
Siphon Soul	2	6	Spell	Warlock
Aranasi Broodmother	2	6	Minion	Warlock
Arch-Villain Rafaam	1	7	Minion	Warlock
Twisting Nether	1	8	Spell	Warlock
Mountain Giant	2	12	Minion	Neutral

[a]This is the decklist corresponding to condtrol deck 4. Received via `https://outof.cards/hearthstone/decks/1132-control-warlock-old-type`

Table E.5 Decklist Control 05[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Doomsayer	2	2	Minion	Neutral
Sunfury Protector	2	2	Minion	Neutral
Acidic Swamp Ooze	2	2	Minion	Neutral
Faceless Rager	2	3	Minion	Neutral
Earthen Ring Farseer	2	3	Minion	Neutral
Twilight Drake	2	4	Minion	Neutral
Defender of Argus	2	4	Minion	Neutral
Hellfire	2	4	Spell	Warlock
Shadowflame	1	4	Spell	Warlock
Omega Agent	2	5	Minion	Warlock
Rotten Applebaum	2	5	Minion	Neutral
Zilliax	1	5	Minion	Neutral
Mossy Horror	1	6	Minion	Neutral
Siphon Soul	1	6	Spell	Warlock
Aranasi Broodmother	2	6	Minion	Warlock
Lord Godfrey	1	7	Minion	Warlock
Lord Jaraxxus	1	9	Minion	Warlock
Mountain Giant	2	12	Minion	Neutral

[a]This is the decklist corresponding to control deck 5. Received via `https:// www. hearthstonetopdecks.com/decks/handlock-to-legend/`

Table E.6 Decklist Aggro 01[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Flame Imp	2	1	Minion	Warlock
Grim Rally	2	1	Spell	Warlock
Soul Infusion	2	1	Spell	Warlock
Soulfire	2	1	Spell	Warlock
Voidwalker	2	1	Minion	Warlock
Witchwood Imp	2	1	Minion	Warlock
Abusive Sergeant	2	1	Minion	Neutral
Argent Squire	2	1	Minion	Neutral
Mecharoo	2	1	Minion	Neutral
Saronite Taskmaster	2	1	Minion	Neutral
Dire Wolf Alpha	2	2	Minion	Neutral
Knife Juggler	2	2	Minion	Neutral
Scarab Egg	2	2	Minion	Neutral
Doubling Imp	2	3	Minion	Warlock
Fiendish Circle	2	4	Spell	Warlock

[a]This is the decklist corresponding to aggro deck 1. Received via `https://www.hearthstonetopdecks.com/decks/budget-zoo-warlock-deck-list-guide-rise-of-shadows/`

Table E.7 Decklist Aggro 02[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Flame Imp	2	1	Minion	Warlock
Grim Rally	2	1	Spell	Warlock
The Soularium	1	1	Spell	Warlock
Voidwalker	2	1	Minion	Warlock
Witchwood Imp	2	1	Minion	Warlock
Abusive Sergeant	2	1	Minion	Neutral
Crystallizer	2	1	Minion	Neutral
Mecharoo	2	1	Minion	Neutral
Dire Wolf Alpha	2	2	Minion	Neutral
Knife Juggler	2	2	Minion	Neutral
Scarab Egg	2	2	Minion	Neutral
Magic Carpet	2	3	Minion	Neutral
Microtech Controller	2	3	Minion	Neutral
Fiendish Circle	2	4	Spell	Warlock
Defender of Argus	1	4	Minion	Neutral
Sea Giant	2	10	Minion	Neutral

[a]This is the decklist corresponding to aggro deck 2. Received via `https://www.hearthstonetopdecks.com/decks/zoo-warlock-shadows-post-nerf-7-legend-viper/`

Table E.8 Decklist Aggro 03[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Flame Imp	2	1	Minion	Warlock
Grim Rally	2	1	Spell	Warlock
The Soularium	1	1	Spell	Warlock
Voidwalker	2	1	Minion	Warlock
Witchwood Imp	2	1	Minion	Warlock
Abusive Sergeant	2	1	Minion	Neutral
Argent Squire	2	1	Minion	Neutral
Crystallizer	2	1	Minion	Neutral
Mecharoo	2	1	Minion	Neutral
Dire Wolf Alpha	2	2	Minion	Neutral
Knife Juggler	2	2	Minion	Neutral
Scarab Egg	2	2	Minion	Neutral
Magic Carpet	2	3	Minion	Neutral
SN1P-SN4P	1	3	Minion	Neutral
Fiendish Circle	2	4	Spell	Warlock
Sea Giant	2	10	Minion	Neutral

[a]This is the decklist corresponding to aggro deck 3. Received via `https://www.hearthstonetopdecks.com/decks/zoo-warlock-shadows-post-buff-13-legend-pizza/`

Table E.9 Decklist Aggro 04[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Flame Imp	2	1	Minion	Warlock
Mecharoo	2	1	Minion	Neutral
Crystallizer	2	1	Minion	Neutral
Abusive Sergeant	2	1	Minion	Neutral
Witchwood Imp	1	1	Minion	Warlock
Argent Squire	2	1	Minion	Neutral
The Soularium	1	1	Spell	Warlock
Soulfire	1	1	Spell	Warlock
Grim Rally	2	1	Spell	Warlock
Voidwalker	2	1	Minion	Warlock
Dire Wolf Alpha	2	2	Minion	Neutral
Knife Juggler	2	2	Minion	Neutral
Scarab Egg	2	2	Minion	Neutral
Magic Carpet	2	3	Minion	Neutral
Fiendish Circle	2	4	Spell	Warlock
Leeroy Jenkins	1	5	Minion	Neutral
Sea Giant	2	10	Minion	Neutral

[a]This is the decklist corresponding to aggro deck 4. Received via https://www.hearthstonetopdecks.com/decks/zoo-warlock-shadows-post-nerf-early-3-legend-solegit/

Table E.10 Decklist Aggro 05[a]

Card Name	*Quantity*	*Mana Cost*	*Type*	*Class*
Flame Imp	2	1	Minion	Warlock
Grim Rally	2	1	Spell	Warlock
The Soularium	1	1	Spell	Warlock
Voidwalker	2	1	Minion	Warlock
Abusive Sergeant	2	1	Minion	Neutral
Mecharoo	2	1	Minion	Neutral
Scarab Egg	2	2	Minion	Neutral
Knife Juggler	2	2	Minion	Neutral
Dire Wolf Alpha	2	2	Minion	Neutral
Magic Carpet	2	3	Minion	Neutral
SN1P-SN4P	1	3	Minion	Neutral
Fiendish Circle	2	4	Spell	Warlock
Explodinator	2	4	Minion	Neutral
Omega Agent	2	5	Minion	Warlock
Barista Lynchen	1	5	Minion	Neutral
Wargear	2	5	Minion	Neutral
Zilliax	1	5	Minion	Neutral

[a]This is the decklist corresponding to aggro deck 5. Received via `https://www.hearthstonetopdecks.com/decks/omega-zoo-shadows-hof-25-14-wabeka/`

APPENDIX F

COMMON CARDS BETWEEN EACH DECK

Table F.1 Common Cards Across Decklists[a]

	C1	*C2*	*C3*	*C4*	*C5*	*A1*	*A2*	*A3*	*A4*	*A5*
C1	30	18	12	10	11	0	0	0	0	1
C2	18	30	14	14	13	0	1	1	1	2
C3	12	14	30	15	16	0	0	0	0	3
C4	10	14	15	30	17	0	0	0	0	0
C5	11	13	16	17	30	0	1	0	0	3
A1	0	0	0	0	0	30	20	22	22	18
A2	0	1	0	0	1	20	30	27	26	21
A3	0	1	0	0	0	22	27	30	28	22
A4	0	1	0	0	0	22	26	28	30	21
A5	1	2	3	0	3	18	21	22	21	30

[a]This shows how many cards are shared across each of the pairs of decks, control decks 1-5 (C1-C5), and aggro decks 1-5 (A1-A5)

APPENDIX G
HEATMAPS OF WIN RATES FOR DIFFERENT MATCHUPS

Figures G.1 to G.12 show the win rates for each pairing of decks described by the experiments in Section 4.1.

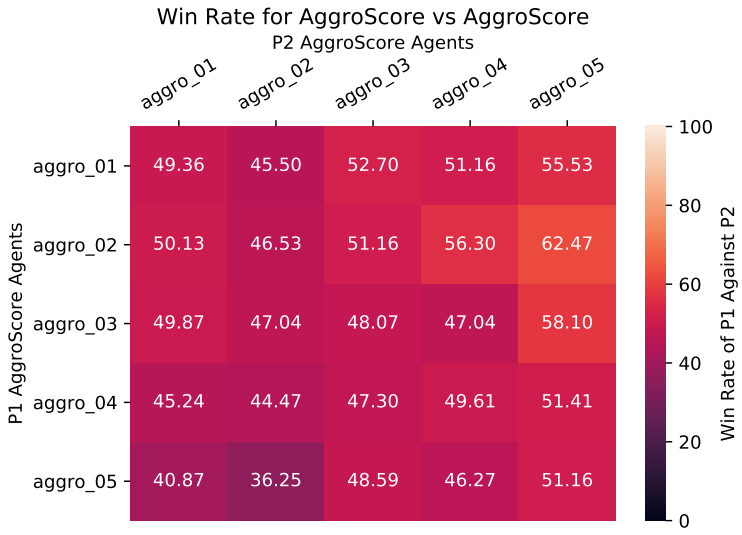

Figure G.1 Each of the five aggro decks were independently paired against each other while playing with the AggroScore function. The lowest win rate is deck 5 against deck 2 at 36.25%, while the highest is deck 2 against deck 5 at 62.47%. The average win rate is 49.29% with standard deviation 5.27.

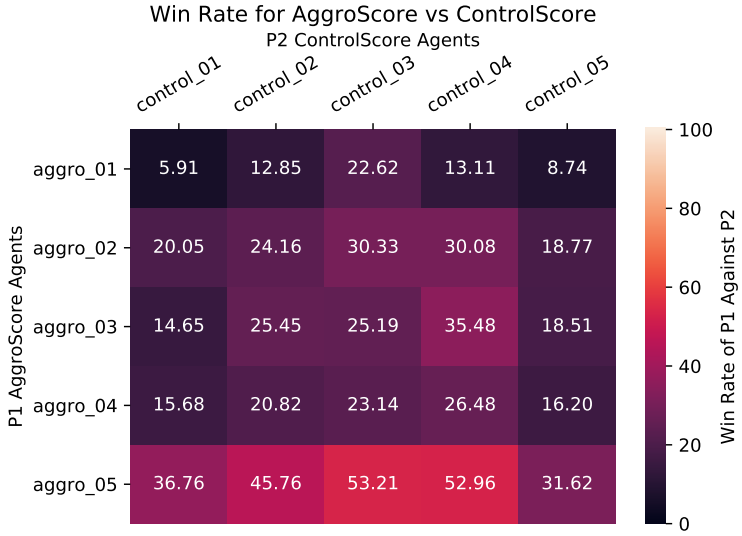

Figure G.2 Each of the five aggro decks played using the AggroScore function were independently paired against each of the control decks using the ControlScore function. The lowest win rate is aggro deck 1 against control deck 1 at 5.91%, while the highest is aggro deck 5 against control deck 3 at 53.21%. The average win rate is 25.14% with standard deviation 12.13.

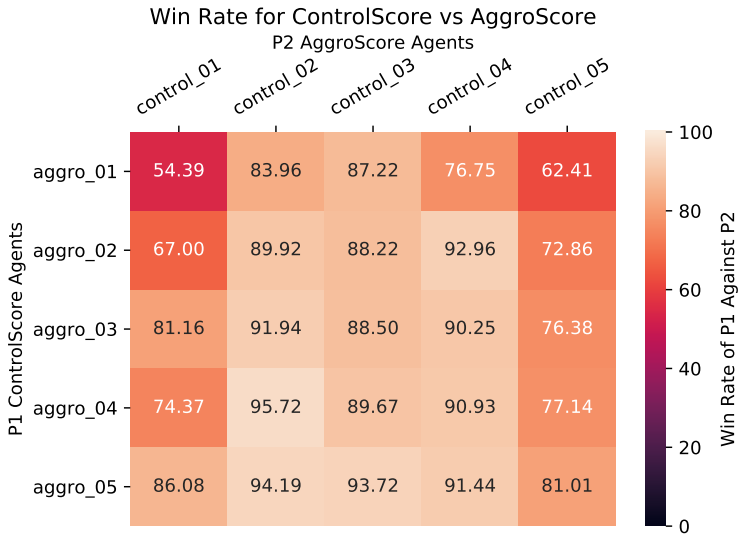

Figure G.3 Each of the five aggro decks played using the ControlScore function were independently paired against each of the control decks using the AggroScore function. The lowest win rate is aggro deck 1 against control deck 1 at 54.39%, while the highest is aggro deck 5 against control deck 3 at 95.72%. The average win rate is 83.13% with standard deviation 10.48.

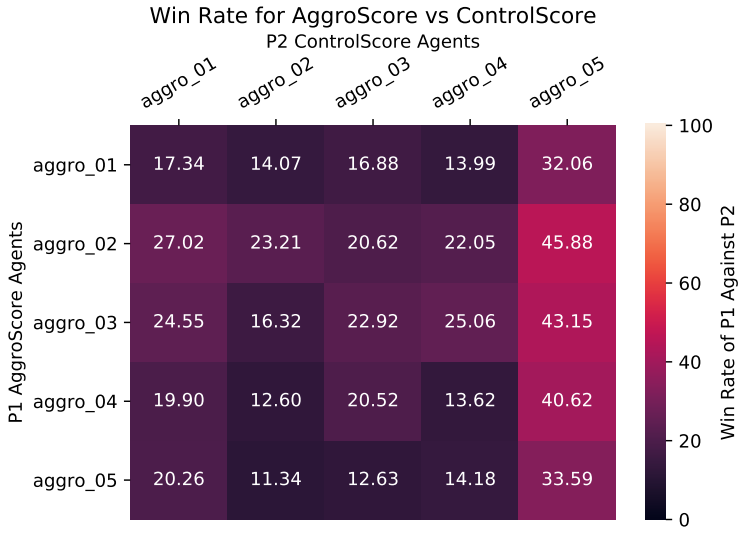

Figure G.4 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using the ControlScore function. The lowest win rate is aggro deck 5 against aggro deck 2 at 11.34%, while the highest is aggro deck 2 against aggro deck 5 at 45.88%. The average win rate is 22.58% with standard deviation 9.54.

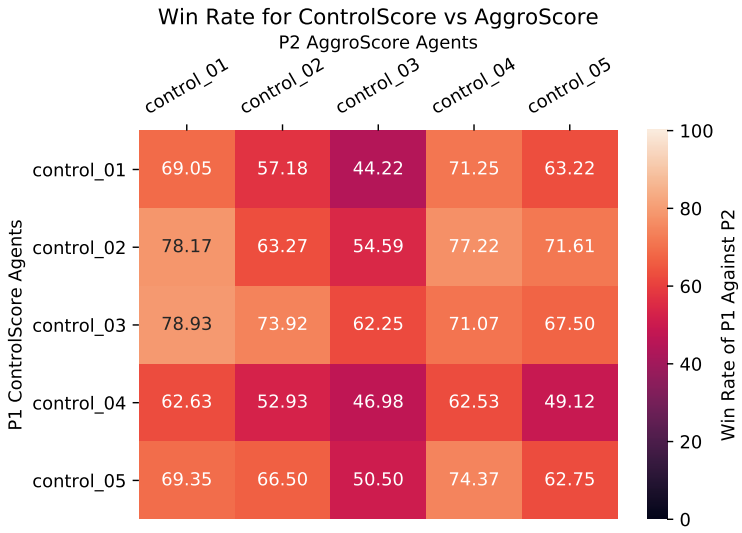

Figure G.5 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using the AggroScore function. The lowest win rate is control deck 1 against control deck 3 at 44.22%, while the highest is control deck 3 against control deck 1 at 78.93%. The average win rate is 64.04% with standard deviation 9.78.

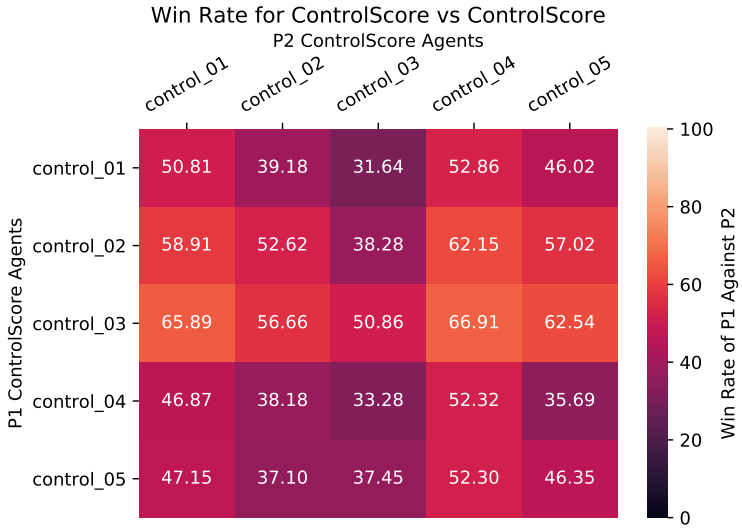

Figure G.6 Each of the five control decks were independently paired against each other while playing with the ControlScore function. The lowest win rate is deck 1 against deck 3 at 31.64%, while the highest is deck 3 against deck 4 at 66.91%. The average win rate is 48.76% with standard deviation 10.18.

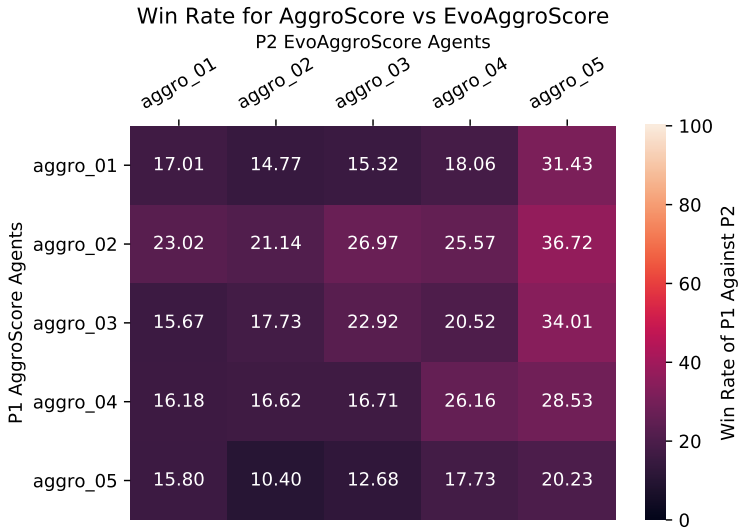

Figure G.7 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using an ANN evolved using the Warlock_Net_AA_sm setup. The lowest win rate is deck 5 against deck 2 at 10.40%, while the highest is deck 2 against deck 5 at 36.72%. The average win rate is 20.88% with standard deviation 6.60.

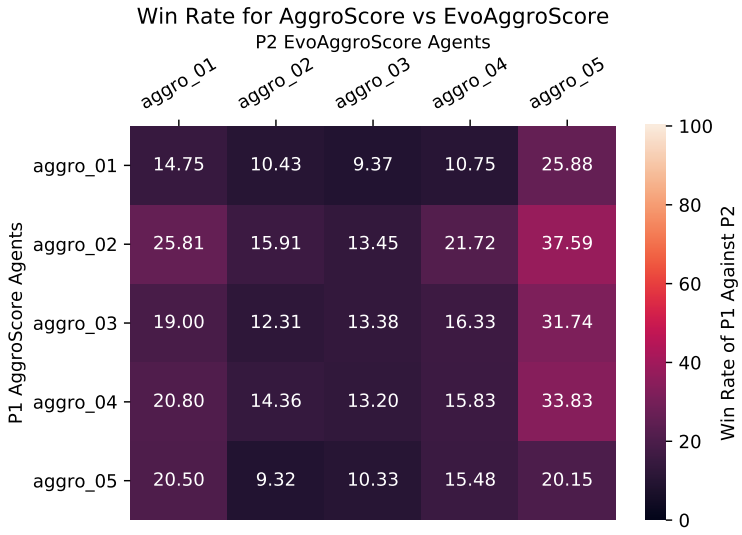

Figure G.8 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using an ANN evolved using the Warlock_Net_AA_lg setup. The lowest win rate is deck 5 against deck 2 at 9.32%, while the highest is deck 2 against deck 5 at 37.59%. The average win rate is 18.09% with standard deviation 7.58.

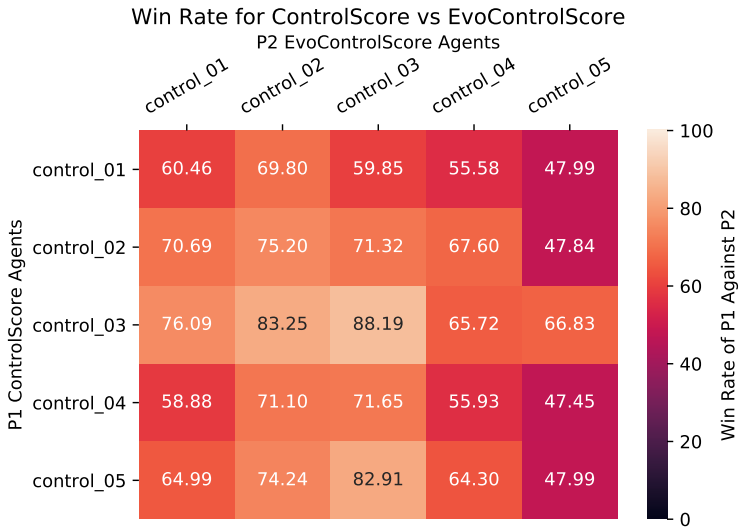

Figure G.9 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the Warlock_Net_CC_sm setup. The lowest win rate is deck 4 against deck 5 at 47.45%, while the highest is deck 3 against deck 3 at 88.19%. The average win rate is 65.83% with standard deviation 11.16.

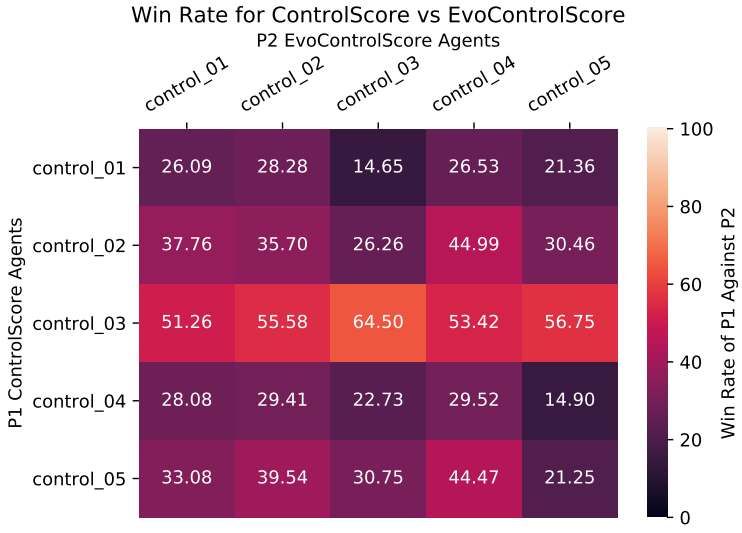

Figure G.10 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0 setup. The lowest win rate is deck 1 against deck 3 at 14.65%, while the highest is deck 3 against deck 3 at 64.50%. The average win rate is 34.69% with standard deviation 13.20.

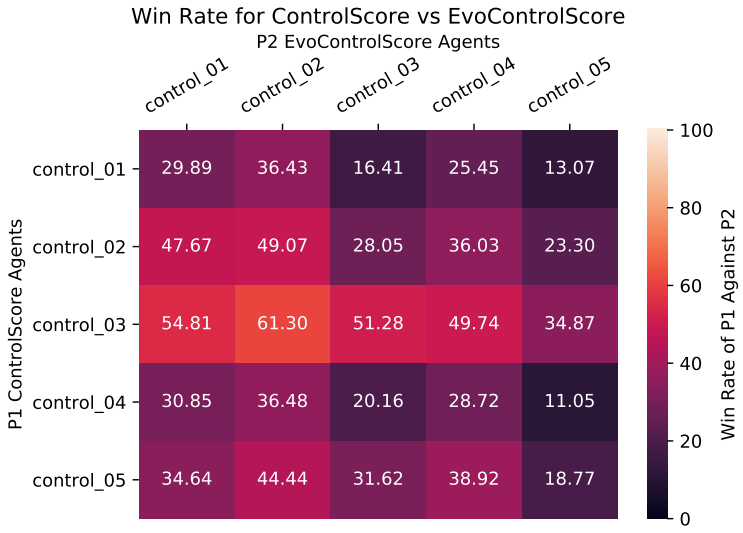

Figure G.11 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0_Large setup. The lowest win rate is deck 4 against deck 5 at 11.05%, while the highest is deck 3 against deck 2 at 61.30%. The average win rate is 34.12% with standard deviation 13.07.

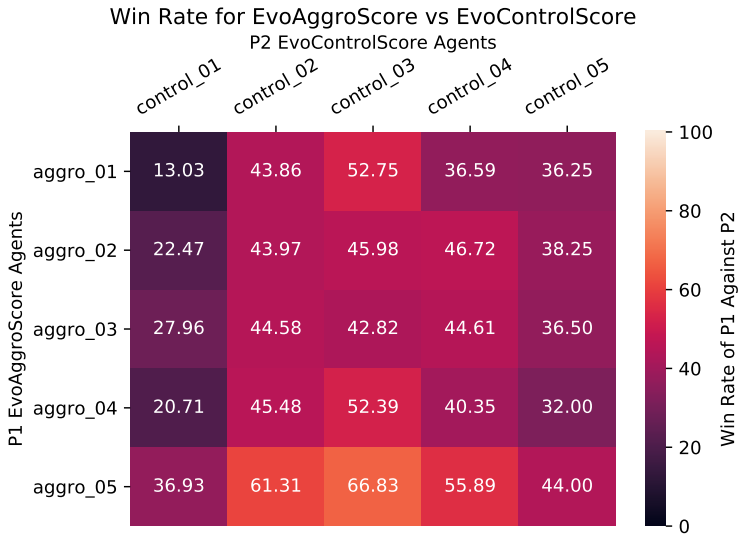

Figure G.12 Each of the aggro decks using an ANN evolved using the Warlock_Net_AA_lg setup were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0_Large setup. The lowest win rate is aggro deck 1 against control deck 1 at 13.03%, while the highest is aggro deck 5 against control deck 3 at 66.83%. The average win rate is 41.29% with standard deviation 11.96.

APPENDIX H
PIE CHARTS OF WIN RATES FOR DIFFERENT MATCHUPS

Figures H.1 to H.12 show the win rates for each pairing of scoring functions described by the experiments in Section 4.1.

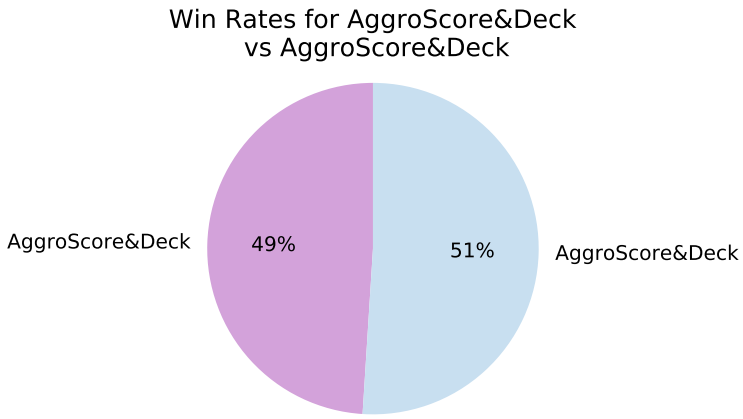

Figure H.1 Each of the five aggro decks were independently paired against each other while playing with the AggroScore function. This pie chart corresponds to the heatmap found in Figure G.1.

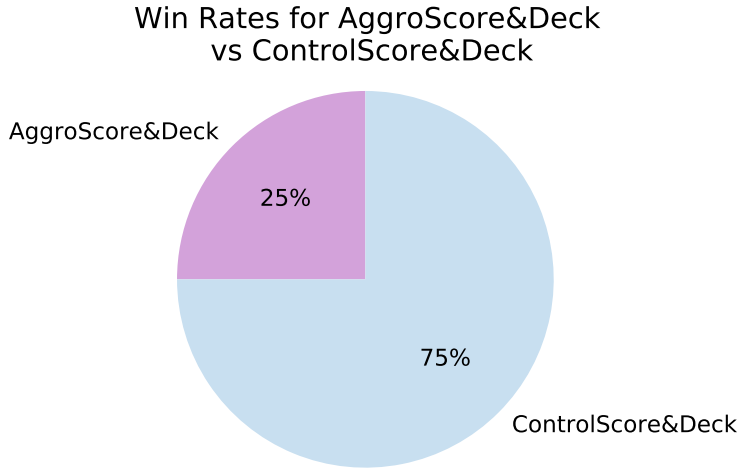

Figure H.2 Each of the five aggro decks played using the AggroScore function were independently paired against each of the control decks using the ControlScore function. This pie chart corresponds to the heatmap found in Figure G.2.

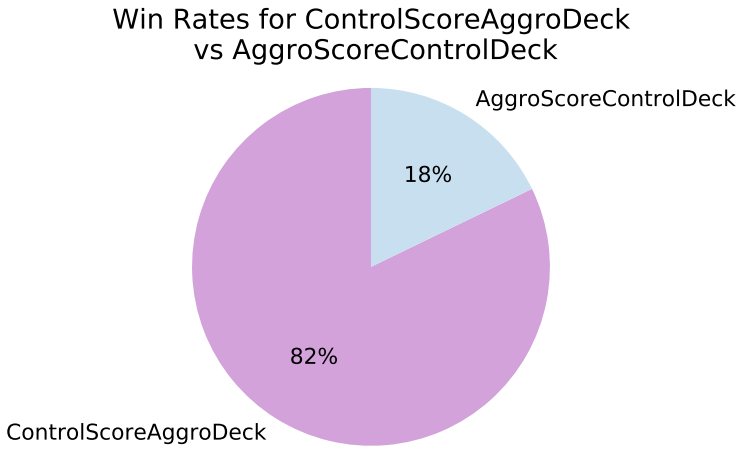

Figure H.3 Each of the five aggro decks played using the ControlScore function were independently paired against each of the control decks using the AggroScore function. This pie chart corresponds to the heatmap found in Figure G.3.

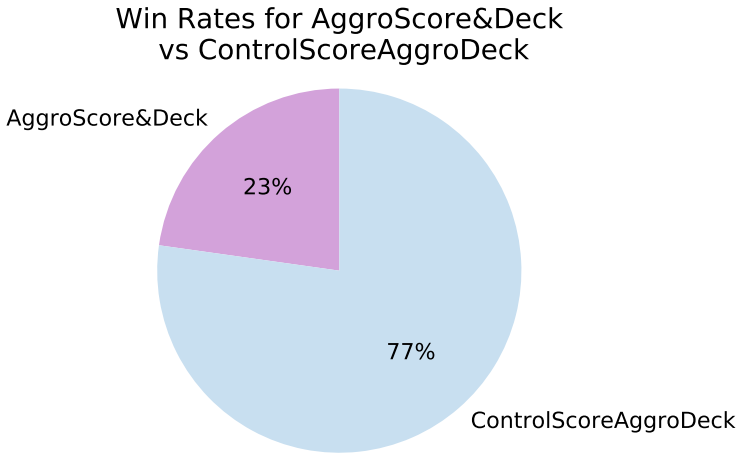

Figure H.4 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using the ControlScore function. This pie chart corresponds to the heatmap found in Figure G.4.

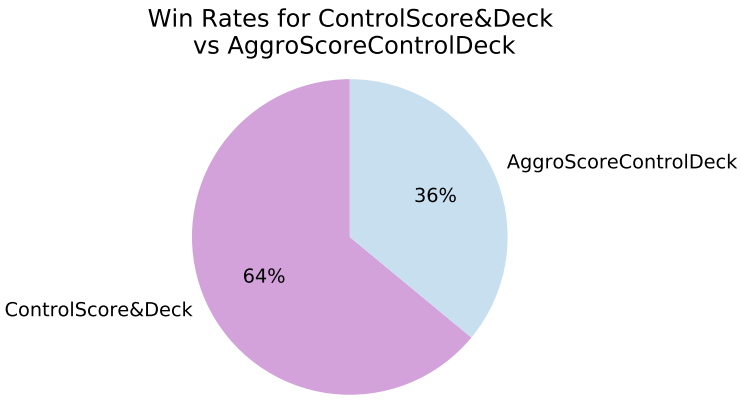

Figure H.5 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using the AggroScore function. This pie chart corresponds to the heatmap found in Figure G.5.

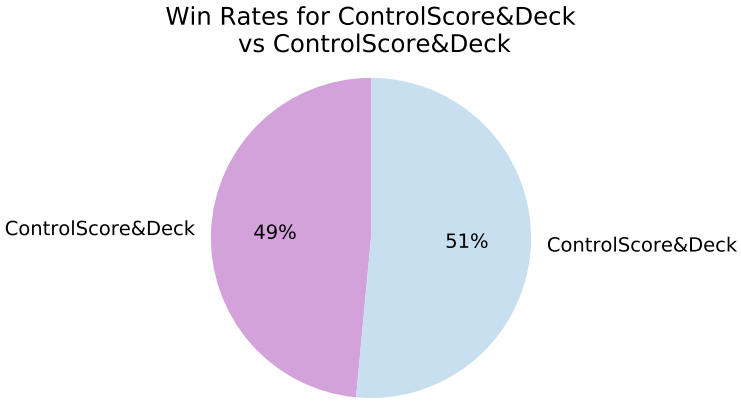

Figure H.6 Each of the five control decks were independently paired against each other while playing with the ControlScore function. This pie chart corresponds to the heatmap found in Figure G.6.

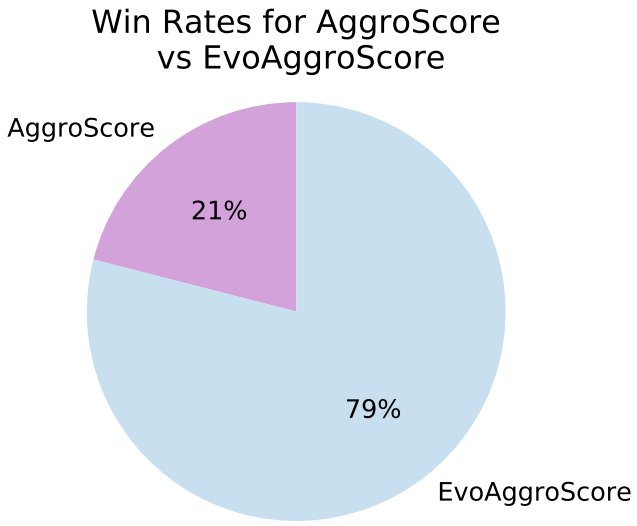

Figure H.7 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using an ANN evolved using the Warlock_Net_AA_sm setup. This pie chart corresponds to the heatmap found in Figure G.7.

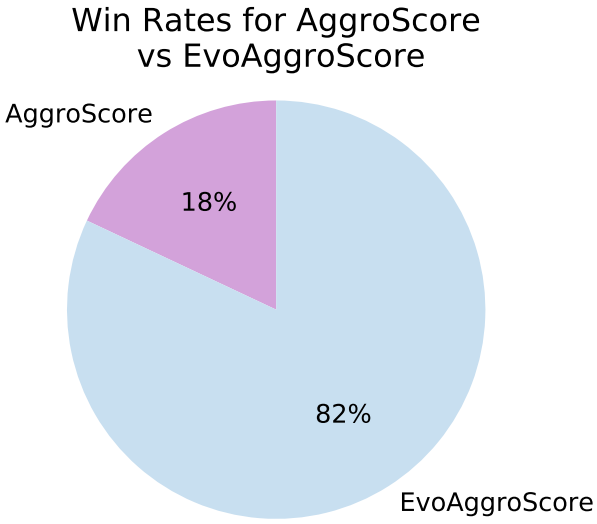

Figure H.8 Each of the five aggro decks played using the AggroScore function were independently paired against each of the aggro decks using an ANN evolved using the Warlock_Net_AA_lg setup. This pie chart corresponds to the heatmap found in Figure G.8.

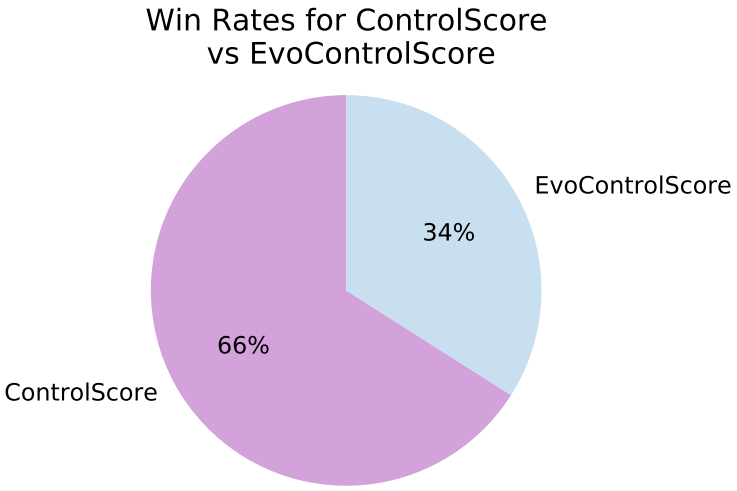

Figure H.9 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the Warlock_Net_CC_sm setup. This pie chart corresponds to the heatmap found in Figure G.9.

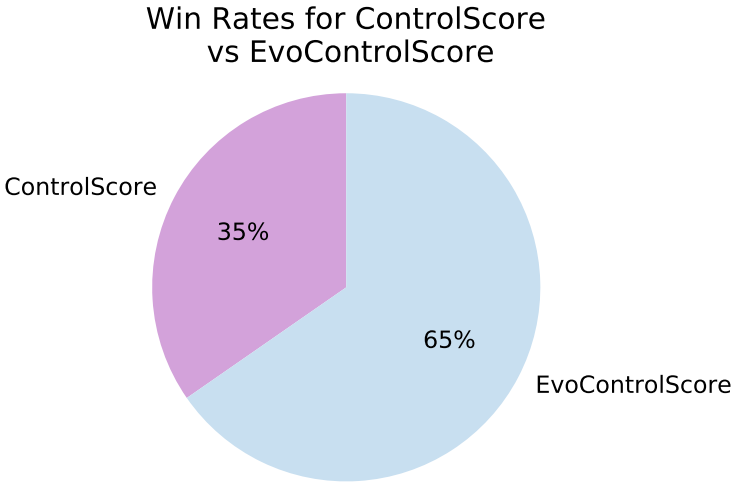

Figure H.10 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0 setup. This pie chart corresponds to the heatmap found in Figure G.10.

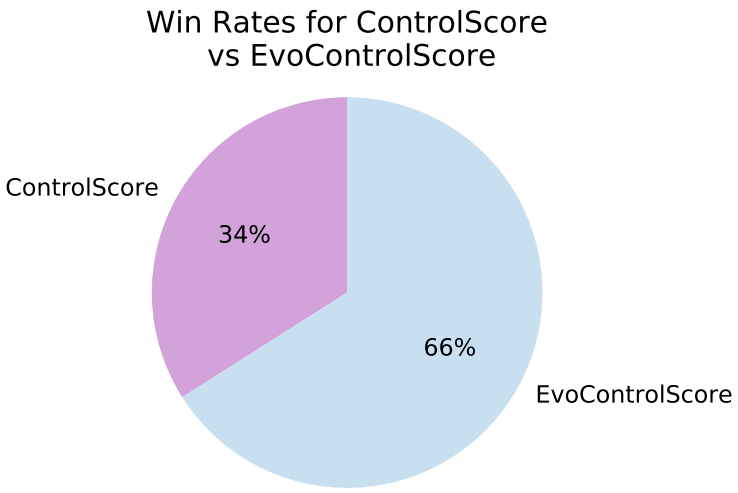

Figure H.11 Each of the five control decks played using the ControlScore function were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0_Large setup. This pie chart corresponds to the heatmap found in Figure G.11.

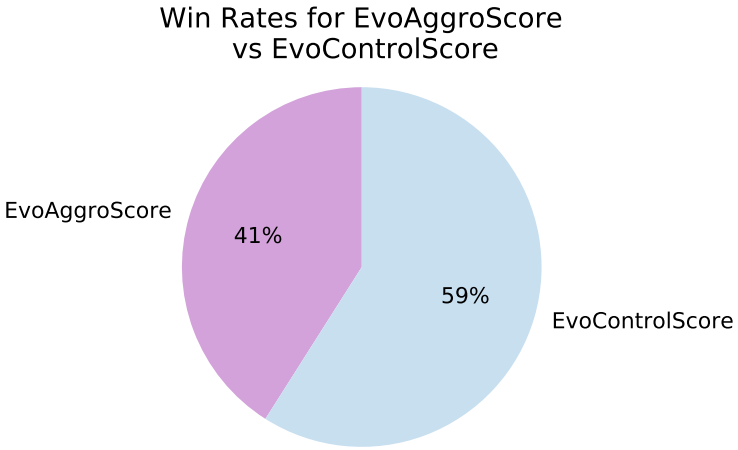

Figure H.12 Each of the aggro decks using an ANN evolved using the Warlock_Net_AA_lg setup were independently paired against each of the control decks using ANNs evolved using the CvsNNC_2.0_Large setup. This pie chart corresponds to the heatmap found in Figure G.12.

BIBLIOGRAPHY

[1] `https://playhearthstone.com/en-us/heroes`. Online; accessed 26 March 2020.

[2] Anguita, D., Ghio, A., Ridella, S., and Sterpi, D. K-fold cross validation for error rate estimate in support vector machines. In *DMIN* (2009), pp. 291–297.

[3] Barshan, E., Ghodsi, A., Azimifar, Z., and Jahromi, M. Z. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition 44*, 7 (2011), 1357–1371.

[4] Bergstra, J., and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research 13*, Feb (2012), 281–305.

[5] Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems* (2011), pp. 2546–2554.

[6] Bhatt, A., Lee, S., de Mesentier Silva, F., Watson, C. W., Togelius, J., and Hoover, A. K. Exploring the hearthstone deck space. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (2018), ACM, p. 18.

[7] Blizzard Entertainment. Hearthstone, 2014.

[8] Brazdil, P. B., and Soares, C. A comparison of ranking methods for classification algorithm selection. In *European conference on machine learning* (2000), Springer, pp. 63–75.

[9] Brown, N., and Sandholm, T. Superhuman ai for multiplayer poker. *Science* (2019), eaay2400.

[10] Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. Deep blue. *Artificial intelligence 134*, 1-2 (2002), 57–83.

[11] Caruana, R., and Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (2006), pp. 161–168.

[12] Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. Monte-carlo tree search: A new framework for game ai. In *AIIDE* (2008).

[13] Ciancarini, P., and Favini, G. P. Monte carlo tree search in kriegspiel. *Artificial Intelligence 174*, 11 (2010), 670–684.

[14] Çığşar, B., and Ünal, D. Comparison of data mining classification algorithms determining the default risk. *Scientific Programming 2019* (2019).

[15] Cozzolino, D., Power, A., and Chapman, J. Interpreting and reporting principal component analysis in food science analysis and beyond. *Food Analytical Methods 12*, 11 (2019), 2469–2473.

[16] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. Robots that can adapt like animals. *Nature 521*, 7553 (2015), 503–507.

[17] Decoster, C., and Seong Bjorn Choe, J. SabberStone. Online; accessed 09 August 2019.

[18] Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research 7*, Jan (2006), 1–30.

[19] Dockhorn, A., Frick, M., Akkaya, Ü., and Kruse, R. Predicting opponent moves for improving hearthstone ai. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (2018), Springer, pp. 621–632.

[20] Dockhorn, A., and Mostaghim, S. Introducing the hearthstone-AI competition. *arXiv preprint arXiv:1906.04238* (2019).

[21] Dummett, M. The history of card games. *European Review 1*, 2 (1993), 125–135.

[22] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research 15*, 1 (2014), 3133–3181.

[23] Fontaine, M. C., Togelius, J., Nikolaidis, S., and Hoover, A. K. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO)* (New York, NY, USA, 2020), GECCO '20, Association for Computing Machinery.

[24] García-Sánchez, P., Tonda, A., Squillero, G., Mora, A., and Merelo, J. J. Evolutionary deckbuilding in hearthstone. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (2016), IEEE, pp. 1–8.

[25] Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., and Teytaud, O. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM 55*, 3 (2012), 106–113.

[26] Grinstein, G., Trutschl, M., and Cvek, U. High-dimensional visualizations. In *Proceedings of the Visual Data Mining Workshop, KDD* (2001), vol. 2, Citeseer, p. 120.

[27] Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems* (2014), pp. 3338–3346.

[28] Ha, D. A visual guide to evolution strategies. *blog.otoro.net* (2017).

[29] Hansen, N. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[30] Hansen, N. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[31] Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation* (2010), pp. 1689–1696.

[32] Hoover, A. K., Togelius, J., Lee, S., and Silva, F. d. M. The many ai challenges of hearthstone. *arXiv preprint arXiv:1907.06562* (2019).

[33] Igel, C., Suttorp, T., and Hansen, N. A computational efficient covariance matrix update and a (1+ 1)-cma for evolution strategies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (2006), pp. 453–460.

[34] Jakubik, J. A neural network approach to hearthstone win rate prediction. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2018), IEEE, pp. 185–188.

[35] Janusz, A., Tajmajer, T., and Świechowski, M. Helping ai to play hearthstone: Aaia'17 data mining challenge. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2017), IEEE, pp. 121–125.

[36] Janusz, A., Tajmajer, T., Świechowski, M., Grad, Ł., Puczniewski, J., and Ślęzak, D. Toward an intelligent hs deck advisor: Lessons learned from aaia'18 data mining competition. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2018), IEEE, pp. 189–192.

[37] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[38] McKinney, W. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 51 – 56.

[39] Mouret, J.-B., and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[40] OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning.

[41] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[42] Perantonis, S. J., and Virvilis, V. Input feature extraction for multilayered perceptrons using supervised principal component analysis. *Neural processing letters 10*, 3 (1999), 243–252.

[43] Przybyszewski, P., Dziewiątkowski, S., Jaszczur, S., Śmiech, M., and Szczuka, M. Use of domain knowledge and feature engineering in helping AI to play Hearthstone. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2017), IEEE, pp. 143–148.

[44] Ringnér, M. What is principal component analysis? *Nature biotechnology 26*, 3 (2008), 303–304.

[45] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall Press, USA, 2009. horizon effect.

[46] Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development 3*, 3 (1959), 210–229.

[47] Santos, A., Santos, P. A., and Melo, F. S. Monte carlo tree search experiments in hearthstone. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)* (2017), IEEE, pp. 272–279.

[48] Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. Checkers is solved. *Science 317*, 5844 (2007), 1518–1522.

[49] Shannon, C. E. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 41*, 314 (1950), 256–275.

[50] Shlens, J. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100* (2014).

[51] Silva, F. d. M., Canaan, R., Lee, S., Fontaine, M. C., Togelius, J., and Hoover, A. K. Evolving the hearthstone meta. *arXiv preprint arXiv:1907.01623* (2019).

[52] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature 529*, 7587 (2016), 484.

[53] Stiegler, A., Dahal, K. P., Maucher, J., and Livingstone, D. Symbolic reasoning for hearthstone. *IEEE Transactions on Games 10*, 2 (2017), 113–127.

[54] Świechowski, M., Tajmajer, T., and Janusz, A. Improving hearthstone ai by combining mcts and supervised learning algorithms. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)* (2018), IEEE, pp. 1–8.

[55] Trammell, A. Magic: The gathering in material and virtual space: An ethnographic approach toward understanding players who dislike online play. *Meaningful Play 2010 proceedings* (2010), 1–21.

[56] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* (2017).

[57] Wainer, J. Comparison of 14 different families of classification algorithms on 115 binary datasets. *arXiv preprint arXiv:1606.00930* (2016).

[58] Ward, C. D., and Cowling, P. I. Monte carlo search applied to card selection in magic: The gathering. In *2009 IEEE Symposium on Computational Intelligence and Games* (2009), IEEE, pp. 9–16.

[59] Xu, N. Understanding the reinforcement learning. In *Journal of Physics: Conference Series* (2019), vol. 1207, IOP Publishing, p. 012014.