NTNU

Norwegian University of
Science and Technology

Deckbuilding in *Magic: The Gathering* Using a Genetic Algorithm

Sverre Johann Bjørke
Knut Aron Fludal

Master of Science in Informatics
Submission date: June 2017
Supervisor: Anders Kofod-Petersen, IDI

Norwegian University of Science and Technology
Department of Computer Science

Til mormor
- Knut

Abstract

An important factor when playing *Magic: The Gathering* is choosing which cards one should play with. This is an example of a combinatorial optimization problem in a large and bewildering search space. In this thesis we have conducted a systematic literature review to gain an overview of the different solutions for this type of problem. Based on the findings from the review we have proposed and implemented a system for automatic card selection using a genetic algorithm. This was tested and an expert analysis was performed on the results.

Samandrag

Ein viktig faktor når ein spelar *Magic: The Gathering* er å velje kva kort ein skal spele med. Dette er eit eksempel på eit kombinatorisk optimeringsproblem i eit stort og uoversiktleg søkerom. I denne oppgåva har vi gjennomført ein systematisk litteraturstudie for å kartlegge dei forskjellige løysingane for denne typen problem. Basert på funna frå denne studien har vi foreslått og implementert eit system for automatisk utveljing av kort ved hjelp av ein genetisk algoritme. Denne vart testa og ein ekspertanalyse av resultata vart gjennomført.

Preface

We would like to thank our supervisor, Anders Kofod-Petersen, for his input and guidance during this project.

Thanks to Gjertrud Fludal for the valuable feedback and grammatical nitpicking we sorely needed, albeit not deserved. Thanks to Ole Håvik Bjørkedal for his eagle eyed observations and sharing his LaTeX knowledge. Thanks to Ole Kristian Ekseth for his feedback, encouragement and positive attitude.

Thanks to Sveinung Knudsen Nøding for helping as an MTG Expert by creating decks, and for helping us evaluate the results. Thanks to the open-source MTG community for helping out with our technical problems and for developing the tools we have used.

Table of Contents

List of Tables

List of Figures

Abbreviations

MTG	=	Magic: The Gathering
COP	=	Combinatorial Optimization Problem
SLR	=	Systematic Literature Review
GA	=	Genetic Algorithm
AI	=	Artificial Intelligence

Chapter 1

Introduction

Luck is an important factor in many games, while others rely purely on skill and strategy. A deciding factor when playing trading card games such as *Magic: The Gathering*, *Yu-Gi-Oh!* and *Pokémon* is which cards the players choose to bring to the table. Selecting these cards is often an important part of the game itself, and of the so-called meta game. In trading card games with hundreds or even thousands of available cards, choosing which ones to pick can be a daunting task. This problem of choosing a number of cards from a larger body of cards is an example of a combinatorial optimization problem. This group of problems often deal with large search spaces where an exhaustive search is not tractable, and a better search algorithm is required. In this master's thesis this problem is explored, with *Magic: The Gathering* as a case study.

1.1 Background

Magic: The Gathering (also known as MTG) is a trading card game invented in 1993 by Richard Garfield, PhD, and was the first modern game of its kind. It has been hailed as the origin of all following trading card games. Today, it is by far the most played trading card game in the world, with more than 20 million players globally (Guinness World Records, 2016).

MTG is played both for recreational purposes and in more competitive settings. Tournaments are arranged in many forms, from small local events to large world championships with grand prizes attracting participants and spectators from all across the globe.

In MTG one plays as a powerful wizard, trying to defeat one or more opponents in battle. The game is played by casting spells, summoning creatures or laying down powerful enchantments. This is done by playing different cards from a personal selection of cards known as a *deck*. A player wins the game by reducing the opponent's score to zero.

Players assemble their deck in a process known as deckbuilding. This is an essential part of the game and is where the players do most of the strategizing before a match. How one composes the deck greatly impacts how the game is played. The effects and various abilities of different cards can combine in countless ways, resulting in many different

playstyles and strategies. A well composed deck will mean the difference between defeat and victory.

Most of the combinations of cards utilized by players fall into three widely recognized archetypes. These are aggressive decks, control decks and midrange decks. Each archetype has specialized subtypes that again fall into more or less well known categories. The control and aggressive decks are extremes on the scale, with control focusing on defence and aggressive focusing on offence. Midrange decks falls somewhere in the middle, features aspects from the two other and focuses on adaptability.

There are of writing more than 15.000 unique MTG cards (Wizards of the Coast, 2016). Throughout MTG's 23 year old history, these cards have been published in batches, called sets. Multiple sets with the same thematic setting form so-called blocks. The settings span everything from steampunk, alien invasions and lovecraftian horrors.

1.2 Motivation

As deckbuilding is such a deciding factor in MTG, it is subject to of a lot of discussion, analysis and testing by players. There are many websites and online forums dedicated to cataloguing, showcasing and rating decks that has been utilized during tournaments.

Deckbuilding is a complex problem, with a lot of intricate interactions. MTG cards have multiple attributes, both quantitative and qualitative, as well as natural language text describing different spells and effects. The strength of a card is not always obvious. A card can be underwhelming on its own, but have strong synergies with certain other cards. Discovering these synergies and utilizing them is a considerable part of *Magic: The Gathering*.

Experienced players acquire a knack for building decks, by being skilled at recognizing positive synergies, assessing cards and predicting which cards opponents might choose to bring to the table. Computers, on the other hand, are not able to do this without sophisticated approaches.

In some MTG-formats, deckbuilding is an integral part of the competition, where the players are given a randomly chosen limited pool of cards to construct their deck from. If a player wants to train for this format, he needs to play in a tournament versus real players. A deckbuilding AI would theoretically make it possible to arrange single player tournaments where players can hone their skills in a realistic environment.

A system for building decks is also interesting for a game developer perspective. Testing how a new set of cards can interact with all the previous releases can help identify and mitigate balancing issues.

We therefore propose deckbuilding in Magic: The Gathering as an interesting problem worth exploring.

1.3 Goals

Deckbuilding is not a widely researched problem. In order to identify and draw upon previous related work, the first goal of this thesis is presented as:

G1 Survey the existing implementations for programmatic deckbuilding by conducting a systematic literature review.

As deckbuilding in MTG and other games is an interesting problem to solve programmatically, the second goal for this thesis is presented as:

G2 Propose and implement a system for programmatic deckbuilding using an approach based on the result from the literature review.

This includes figuring out what algorithm we want to use and also how is should be implemented. It will also be required to design a framework for determining the quality of the decks suggested by the algorithm. This will be important in order to achieve Goal 3.

G3 The decks produced by the system must be of a high quality and should be comparable to decks created by seasoned players.

1.4 Contributions

Our work will primarily contribute to the field of deckbuilding in collectible card games and how the problem can be generalized and solved using algorithmic approaches decided by the result from our literature review.

By conducting a systematic literature review we will also contribute to the field of methodical research in computer science. Systematic literature reviews originates from medicinal research methodologies and provides a structured and reproducible method for discovering papers and articles, as well as reviewing them (Kitchenham and Charters, 2007). Thus, the systematic review improves the legitimacy of our findings. Our thesis will illustrate how this methodology can be utilized in computer science research.

1.5 Structure

The remaining six chapters of this thesis are structured as follows: Chapter 2 gives a more thorough introduction to how Magic: The Gathering is played, important aspects of deckbuilding, the different formats, and a generalization of the problem. In chapter three we discuss the structured literature review, our findings and our interpretations. Our proposed implementation of an AI deckbuilder is presented in Chapter 4, while Chapter 5 shows our experiments and their results. Chapter 6 covers the discussion of our results and, and Chapter 7 concludes our thesis by summarizing our results and future work.

Chapter 2

Background

For the benefit of the readers not familiar with Magic: The Gathering we here give a brief introduction to the basics of the game. The emphasis is on the important aspects for building a deck, and not specific game-rules and mechanics. In Section 2.4 we discuss the problem of deckbuilding and how it can be generalized.

2.1 Summary of gameplay

Magic: The Gathering is designed to be played by two or more players, either in teams, or in everyone versus everyone matches. In competitive games the game is primarily played one versus one. Each player start with twenty points known as lives, and uses the cards in their deck to reduce the opponent's life total to zero. There exists a few other winning conditions, but these are rarely invoked, and not typically seen in tournament play.

Each player start the game by shuffling their deck and drawing seven cards to form their hand. The game is then played by the players taking alternating turns playing cards from their hand. A player's turn last until he declares that he is done playing cards. For each turn the players also draw a card from their respective decks, except on the first player's first turn.

Lands are an essential part of MTG. These are a special type of card that is used to produce a resource known as mana. Mana is used to cast spells and activate abilities. Lands does not require mana to be played, but only one land can be played each turn. There are five different types of basic land cards, each producing a different color of mana. *Plains* produce white mana, *swamps* produce black mana, *forests* produce green mana, *mountains* produce red mana, and *islands* produce blue mana. A land produces mana when the player activates it, which is indicated by turning the card sideways. This is known as "tapping" the land. A tapped land is unusable until it is untapped, which happens at the start of ones turn.

Different cards require different amounts and color of mana. A spell might need one red mana, and another spell might require one green, two white, and three additional mana of any color. A 40-card deck usually contains 16 to 18 lands, depending on how aggressive

it is. *Manabase* is a term used to describe what colors a given deck can produce, and how many sources there are of each. Having a sustainable manabase in a deck is important, because of the different color and mana requirements of the cards.

The more mana required to cast a spell, the better the spell usually is. Players are only allowed to play one land every turn. Because of this, a deck usually has a mix of cards with a low mana cost to play early, and more powerful spells to play later in the game. This distribution is known as the curve of a deck, and ensure that the mana is used as efficiently as possible. Figure 2.1, 2.2 and 2.3 show example curves for different deck-archetypes.

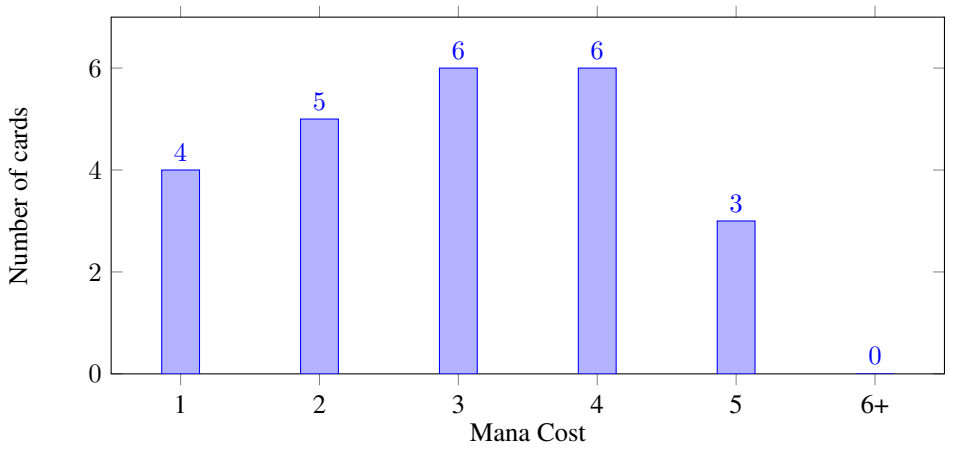

Figure 2.1: Mana curve in an aggro deck with only 16 lands.

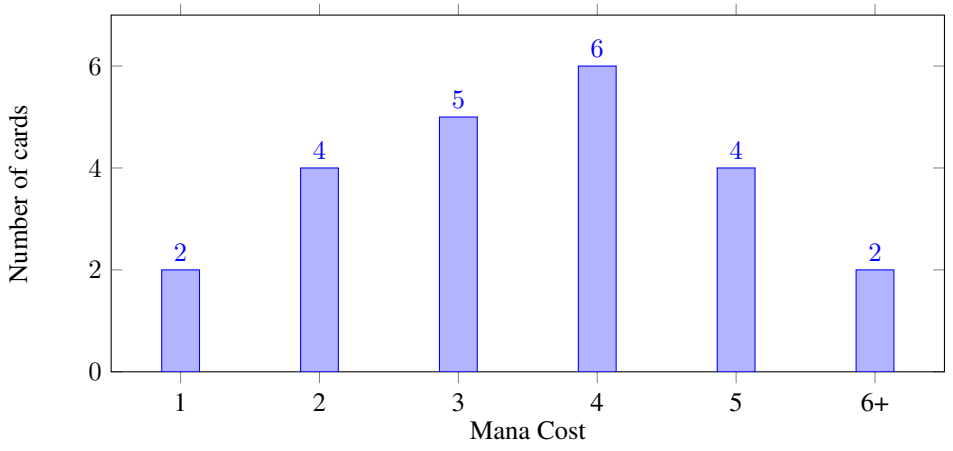

Figure 2.2: Mana curve in a midrange deck with 17 lands.

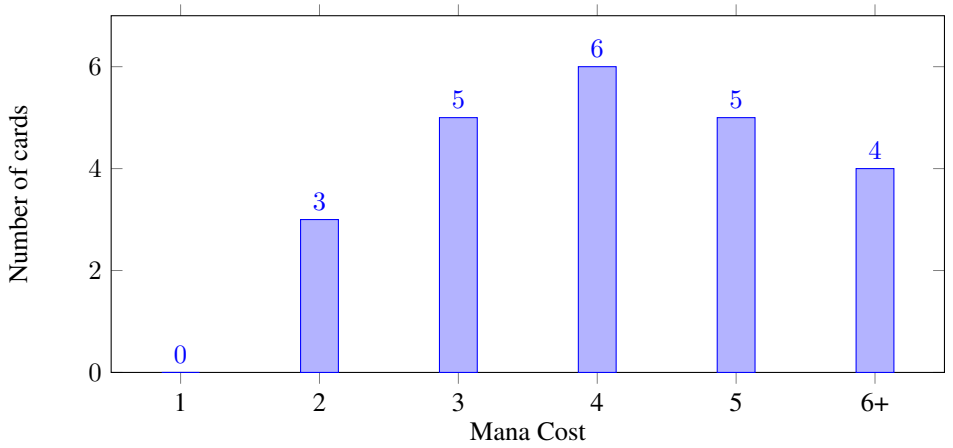

Figure 2.3: Mana curve in a control deck with 18 lands.

2.2 Competitive MTG

MTG is played in many different formats. These formats fall into one of two categories, Limited or Constructed. There is one major difference between these two. Playing Constructed requires a deck built in advance, while in Limited the deck creation is part of the game.

2.2.1 Constructed

There are multiple different constructed formats. They all have different rules when it comes to deckbuilding. Some only allow certain types of cards, like Pauper, where only common and uncommon cards can be used. Others, like standard, only allow cards from certain sets, while Vintage allow any card, as long as it is not on a short list of banned and restricted cards. Legacy is similar to Vintage, but with a more extensive list of banned cards.

Constructed decks usually fall within a specific decktype. There exists different categories of decks that have been fine tuned to a given format. This has been done over many years, and *netdecking* is very common. This is the practice of finding complete deck listings online, copying them, and only making minor adjustments. Netdecking is common because it is hard to make new decks that are competitive and can hold their own against other decks.

2.2.2 Limited

In Limited the deck-creation is part of the competition. This is done in two ways. Draft, or drafting, is done in groups of eight, where each player have three booster packs containing 15 cards each. When drafting, a player picks one card from his first pack and passes the pack to the player on the right. That player then picks a card from the pack he receives from

the left, and the process continues for all cards in all packs. The players then create decks with the cards they drafted. These cards are called a cardpool. Sealed is the other way of playing Limited. In Sealed, the players open six booster packs each, which constitutes their cardpool.

2.2.3 Automatic deckbuilding in competitive play

Using electronic devices to gain a strategic advantage during tournament play is prohibited by the tournament rules (Wizards of the coast, 2016, Section 2.12). This is both on a professional and competitive rules enforcement level, and on the regular enforcement level. This means that the potential for creating an app to help create a deck during a competition is low, as it is not allowed, and therefore not something to pursue. However, in online play, such restrictions does not apply. A person playing a sealed event in Magic: The Gathering Online has unlimited time to create a deck from the boosters he or she opens. This means we can theoretically create a system that gives some examples of decks the player can create with his or her card pool.

2.3 Building a better deck builder

As stated in goal G2 we will aim to create a system that can take a card pool and create one or more complete deck suggestions. These deck suggestions need to be of a high quality if they are to be useful. High quality in this context is defined as being comparable to decks created by seasoned players. How to score proposed decks will also be a problem for us to solve.

A deck produced by our proposed solution needs to exhibit some important properties to be viable. It needs to have a good curve. This includes having high-end threats to play, but also early-game spells. The curve must also adhere to the manabase available in the cardpool, so the number of usable colors and color-combinations is limited. The deck must also be able to take advantage of synergy in cards, it can not just consist of a number of cards that fit nicely into a curve. Two weak cards working great together might be better than two good cards working alone.

Conducting a systematic literature review will enable us to draw upon previous relevant work and finding one or more suitable AI approaches. We must also decide if the AI should be general purpose or specialized for a single MTG set. Seeing that sets often features game mechanics and effects that go well together, this might yield better results than trying to construct valid decks from all the available MTG cards.

2.4 Problem generalization

As programmatic deckbuilding in trading card games is not an extensively researched topic, in order to be able to draw upon previous work we need to generalize the problem. Deckbuilding is, as previously stated, a complex problem with many factors. Building a strong deck can be viewed as an optimization problem in a large and complex solution space.

2.4.1 Problem Analysis

In order to be able to reason about the size of the solution space we perform a simplified calculation.The size of a set of MTG cards can range in size from the smallest set, Arabian Night at 92 unique cards, to the largest, Fifth Edition, at 449 unique cards. These sets are outliers, and the size of a set depends on whether it is the first or second set in a block. First and second sets are around 270 and 220 in size, respectively. In sealed, one is normally given 90 cards from one or two sets, where 22-24 non-land cards are to be picked for use in a deck. One should expect to see some duplicates in these 90 cards, but to keep it simple, we will assume they are all unique cards. If we were to create a deck of 23 cards from a card pool of 90, the following equation shows the number of possible unique decks one could make:

$$\binom{90}{23} \approx 1.58 * 10^{21}$$

This is a search space many orders of magnitude higher than one can search using conventional search methods. Assuming one could examine one billion decks per second it would still take close to fifty thousand years to examine the whole search space. One could eliminate the 10 worst cards, and determine what two colors to play so we could eliminate 60% of the remaining cards, since there are five colors in total. This is calculated as follows:

$$\binom{32}{23} = 28048800$$

This is a more realistic search space, but this would still require a lot of computation in order to examine. This could also be view as the best case scenario, and doing these simplifications are not necessarily trivial.

2.4.2 A combinatorial optimization problem

Within the field of mathematical optimization, combinatorial optimization relates to finding an optimal solution in a finite set of discrete objects. Solutions typically include sets, subsets, combinations, permutations as well as concepts from graph theory such as graphs, subgraphs, cliques and cuts.

Combinatorial optimization problems range from the simple minimum spanning tree problem to the well known travelling salesman problem. Many real life problems encountered every day are combinatorial optimization problems, such as finding optimal routes for infrastructure, planning time schedules for classes or even games like Sudoku.

In essence, most search algorithms can be used to solve combinatorial optimization problems, as finding a solution can be reduced to searching for the best item in a set. However, in many cases, this is not feasible. In our concrete case of deckbuilding, finding the optimal deck in the large solution space is as such not computationally tractable with exhaustive search. This is the case with many combinatorial optimization problems.

Chapter 3

Systematic Literature Review

In accordance with our established goal G1, we conducted a systematic literature review in order to survey the existing implementations for programmatic deckbuilding.

The rest of this chapter is structured as follows: In Section 3.1 we introduce the background of systematic literature reviews, the state of systematic literature reviews in the field of computer science and our motivation for conducting one. In Section 3.2 we detail the review process. Our analysis is presented in Section 3.3. In Section 3.4 we evaluate our process and in Section 3.5 we summarize the review.

3.1 Introduction

A systematic literature review is a secondary review where the goal is to extract and combine the knowledge in primary studies related to a specific topic of inquiry (Kitchenham and Charters, 2007). Stemming from the field of medicine, systematic literature reviews are characterized by their extensive use of planning and their adherence to protocol, as well as being objective and transparent. Explicitly defined steps constitutes the review protocol by which the review is executed.

While not widely adopted in computer science, the methodology has some useage, much to the credit of Barbara Kitchenham. Spearheading the usage of systematic reviews in software development, Kitchenham proposes a guideline for undertaking such a study. We have employed the "light" version of the systematic literature review as proposed in *Guidelines for performing systematic literature reviews in software engineering* (Kitchenham and Charters, 2007) .

As an example of a well executed systematic literature review in computer science, we have looked at the work of Lillegraven and Wolden. In their master's thesis *Design of a bayesian recommender system for tourists presenting a solution to the cold-start user problem* (Lillegraven and Wolden, 2010) they conduct a systematic literature review of the recommender system literature focusing on the cold-start user problem. The design of the review protocol is largely based on their work.

3.1.1 Motivation

Most research involves a literature review in order to draw upon previous work. In order for this process to be of high value and relevance, it has to be thorough and fair (Kitchenham and Charters, 2007). Systematic literature reviews are used to achieve this. Through a well defined and systematic process, literature related to a certain question or topic is gathered, reviewed and analysed.

There are several advantages in performing a systematic review. They are reproducible, and easier to peer review. They help with identifying gaps in current research and they provide a solid fundament for performing further research. They make it less likely that the results are biased and help with identifying tendencies across different works (Kitchenham and Charters, 2007).

The main disadvantage of systematic literature reviews, as pointed out by Kitchenham, is that it requires a lot of work compared to traditional reviews.

3.2 The review process

This section describes the different steps performed during this process. These steps are adapted from the guidelines by Kitchenham. They are:

ST1 Define research questions for the systematic literature review

ST2 Write a protocol for the systematic literature review

ST3 Literature search

ST4 Filter on inclusion criteria

ST5 Filter on quality criteria

ST6 Data collection

ST7 Quality Assessment

ST8 Analysis

ST9 Dissemination

All the steps are described in further detail below. An analysis of the results is presented in Section 3.3.

3.2.1 ST1 - Defining the research questions

Research questions are the starting point of any academic work, and are fundamental to methodical research. Research questions are, simply put, questions to be answered by the following research. In this work, the research questions for the systematic literature review were defined as:

RQ1 What is state of the art on deck-building systems in Magic: The Gathering?

RQ1.1 If no current system exists, how can such a system be created?

RQ1.2 If such systems exists, in what way is it possible to improve them?

RQ2 What is the strength of the evidence in support of the different solutions?

RQ3 How can our findings be applied when creating a system for programmatic deck-building?

3.2.2 ST2 - The systematic literature review protocol

The systematic literature review protocol is a guiding document that describes how each step of the review is to be performed. The protocol should be defined in advance and should contain unambiguous instructions. This is done in order to counteract human tendencies towards bias while conducting research, which will compromise the objectivity of the work.

As stated in the review protocol, the protocol has been subject to an iterative process. In some cases we found it necessary to change some parts of it, in order to proceed. The protocol can be viewed in it's final iteration in Appendix A.

3.2.3 ST3 - Literature search

In this step the search for literature that could help us answering our research questions was carried out. To adhere to the principles of the review method, this step was conducted systematically, following the predefined steps in the protocol.

Searched Libraries

The review protocol lists the various online libraries and search engines used during the search step. These were selected based on recommendation by our supervisor, with the addition of Google Scholar. The latter does not host papers and articles itself, but indexes other well known sources, including the rest of the list, and will in some cases, by our experience, return more relevant results than the integrated search engines offered by the libraries. The libraries were distributed between the researchers to share the workload and speed up the process. The list of sources searched during this process is presented in Table 3.1.

Search Terms

Before performing the search we decided on a set of search terms to be used, as well as synonyms and alternative words with the same or similar meaning for these terms. These terms are all selected based on the research questions. The complete search matrix is presented in Table 3.2.

By combining the search items in the search matrix, we were able to construct a search string that targets the overlap of the articles corresponding to each search term. This was done by combining each word within a group with the OR operator, and combining each group with the AND operator. This search string could then be used in the advanced search engine that most of the digital libraries offer. In the cases where such advanced

Table 3.1: The digital libraries and search engines used during the search process.

Library	URL	Assignee
ACM digital library	http://dl.acm.org/advsearch.cfm	Knut
IEEE Xplore	http://ieeexplore.ieee.org/search/advsearch.jsp	Knut
Web of Science	https://apps.webofknowledge.com	Knut
ScienceDirect	http://www.sciencedirect.com/science/search	Knut
CiteSeerX	http://citeseerx.ist.psu.edu/advanced_search	Sverre
SpringerLink	https://link.springer.com/advanced-search	Sverre
Wiley Online Library	http://onlinelibrary.wiley.com/advanced/search	Sverre
Oria	https://bibsys-almaprimo.hosted.exlibrisgroup.com	Sverre
Google Scholar	https://scholar.google.no/	Sverre

Table 3.2: The first search matrix used during the search process.

Group 1	Deckbuilding	Deck building	Combinatorial optimization	COP
Group 2	Evolutionary algorithm	Neural Network		
Group 3	Magic: The Gathering	MTG	Trading card game	TCG
Group 4	Artificial intelligence	AI	Machine learning	

functionality was not supported, we performed the equivalent as separate searches and combined the results. The resulting search string for the search matrix was:

("Deckbuilding" OR "Deck Building" OR "Combinatorial Optimization" OR "COP") AND ("Evolutionary Algorithm" OR "Neural Network") AND ("Magic: The Gathering" OR "MTG" OR "Trading Card Game" OR "TCG") AND ("Artificial Intelligence" OR "AI" OR "Machine Learning")

New Search Matrix

The first search matrix did not yield many results. As deckbuilding in MTG and similar card games is not an extensively researched subject, the strictness of the AND operator meant that most of the searched libraries did not return any results at all. In the few cases where results were found they were with one exception not related to computer science. This forced us to rethink our search terms and generalize the problem, as described in Section 2.4. From this we created a new search matrix and restarted the search step (Table 3.3). The resulting search string:

("Combinatorial optimization" OR "COP") AND ("Optimal subset" OR "subset selection") AND ("Artificial Intelligence" OR "AI" OR "Machine Learning") AND ("Evolutionary Algorithm" OR "Neural Network" OR "Simulated Annealing")

3.2.4 ST4 - Filter on inclusion criteria

Following the search step, the collected studies were filtered based on the inclusion criteria defined in the protocol. The goal of these criteria is to ensure that the collected articles

Table 3.3: The second search matrix used during the search process.

Group 1	Combinatorial optimization	COP	
Group 2	Optimal subset	Subset selection	
Group 3	Artificial intelligence	AI	Machine Learning
Group 4	Evolutionary algortithm	Neural network	Simulated annealing

have a certain relevance for our systematic literature review. The inclusion criteria were:

IC1 The study's main concern is Combinatorial Optimization Problems

IC2 The study is a primary study presenting empirical results.

IC3 The study focuses on finding an optimal subset of a given set.

IC4 The study proposes a general solution.

Each inclusion criteria focuses on a different aspect we felt was needed in order for an article to be relevant. IC1 ensures that the article focuses on the problem we want to solve, along with IC3. IC2 ensures that an article is a primary study, and not a meta-analysis or secondary study.

This step in the process was carried out in two parts, the primary and secondary filtering. The primary filtering was carried out by each of the researches reading the abstract of their assigned articles, considering the two first inclusion criteria, and making a note of whether or not it should be included. Following this, we discussed our verdicts and exchanged brief summaries of the contents of the articles in order to identify any disagreement in the inclusions.

The secondary filtering was executed in a similar fashion, replacing reading of the abstracts with full text filtering and considering the second two inclusion criteria. Again we discussed our findings and reached a mutual decision on which items to proceed with.

3.2.5 ST5 - Filter on quality criteria

A final filtering step was performed, in order to ensure that all the included articles had a high level of credibility and quality. This filtering was performed with two quality criteria.

QC1 Is there is a clear statement of the aim of the research?

QC2 Is the study put into context of other studies and research?

If the answer to any of these questions was no for any of the collected articles it was not included for further study. The nine resulting articles from the filtering steps are presented in Table 3.4.

Table 3.4: The studies collected during the search.

Study ID	Author(s)	Title	Year	Assignee
S1	García-Sánchez, Pablo, et al	Evolutionary Deckbuilding in HearthStone	2016	Knut
S2	García-Martínez, C., Lozano, M., & Rodríguez-Díaz, F. J.	A simulated annealing method based on a specialised evolutionary algorithm	2011	Sverre
S3	Nahar, S., Sahni, S., & Shragowitz, E.	Simulated annealing and combinatorial optimization	1986	Knut
S4	Park, K., & Carter, B.	On the effectiveness of genetic search in combinatorial optimization	1995	Sverre
S5	Kubalík, J.	Evolutionary-based iterative local search algorithm for the shortest common supersequence problem	2011	Knut
S6	Meinl, T., & Berthold, M. R.	Crossover operators for multiobjective k-subset selection	2008	Sverre
S7	Rainville, D., Gagné, C., Teytaud, O., & Laurendeau, D.	Optimizing low-discrepancy sequences with an evolutionary algorithm	2009	Knut
S8	Osaba, E., Carballedo, R., López-García, P., & Diaz, F.	Comparison between Golden Ball Metaheuristic, Evolutionary Simulated Annealing and Tabu Search for the Traveling Salesman Problem	2016	Sverre
S9	Beheshti, Z., Shamsuddin, S. M., & Yuhaniz, S. S.	Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems	2012	Knut

3.2.6 ST6 - Data Collection

In this step, data from each of the articles in the filtered list was extracted. The data to be extracted was chosen with respect to the research questions to be answered, as well as metadata necessary for citation. All articles and corresponding data was catalogued in a new spreadsheet. The extracted data fields were:

- Author
- Title
- Year of publication
- Type of article
- Aims
- Type of COP
- Algorithm type
- Experimental design
- Performance metric used
- Conclusion

3.2.7 ST7 - Quality assessment

To map the quality of the research on the selected topic, and to answer RQ2, we assessed the quality of the collected articles. Every article was scored on ten questions, where every Yes was worth 1 point, every No was 0 points, and any answer in between was scored 0.5. The scoring was done in collaboration between the researchers, and unanimous consensus was reached on the scoring for all the articles.

The full list of quality criteria was used during this process. All of these questions were taken from (Kofod-Petersen, 2014).

QC1 Is there is a clear statement of the aim of the research?

QC2 Is the study put into context of other studies and research?

QC3 Are system or algorithmic design decisions justified?

QC4 Is the test data set reproducible?

QC5 Is the study algorithm reproducible?

QC6 Is the experimental procedure thoroughly explained and reproducible?

QC7 Is it clearly stated in the study which other algorithms the study's algorithm(s) have been compared with?

QC8 Are the performance metrics used in the study explained and justified?

QC9 Are the test results thoroughly analyzed?

QC10 Does the test evidence support the findings presented?

The following table show the scores for all the articles.

Table 3.5: The scores for the included studies on the quality criteria.

Study ID	S1	S2	S3	S4	S5	S6	S7	S8	S9
QC1	1	1	1	1	1	1	1	1	1
QC2	1	1	1	1	1	1	1	1	1
QC3	1	1	1	0.5	1	0.5	1	1	1
QC4	1	1	0.5	1	1	1	1	1	1
QC5	1	1	1	1	1	1	1	0.5	1
QC6	1	1	1	1	1	1	1	1	1
QC7	0.5	1	0.5	1	1	1	1	1	1
QC8	1	1	1	1	1	1	1	0.5	1
QC9	1	1	1	1	1	1	0.5	1	0.5
QC10	1	1	1	1	1	1	1	1	1
Total:	9.5	10	9	9.5	10	9.5	9.5	9	9.5

3.2.8 ST8 - Analysis

In this step we analyzed the results of our literature review. This process was done by thoroughly reading and discussing the collected studies with respect to the research questions. The analysis is detailed in Section 3.3.

3.2.9 ST9 - Dissemination

The systematic review protocol describes dissemination as the final step of the review. The current chapter is our presentation of the review and its execution, the results from the study itself, as well as our experience with performing a systematic literature review.

3.3 Analysis

In this section we analyze the results from our literature review. In Section 3.3.1 through 3.3.4 we discuss our findings with respect to each of the three research questions.

3.3.1 RQ1: What is state of the art on deck-building systems in Magic: The Gathering?

Following our planned search strategy, we found no solutions for automatic deckbuilding in Magic: The Gathering. We did find a solution for deckbuilding for the immensely

popular computer game Hearthstone (S1). Hearthstone is developed and published by Blizzard Entertainment and is a digital collectible card game. Hearthstone shares many similarities with Magic: The Gathering. Players strive to defeat each other using spells and creatures represented by cards that require a resource to use.

To iterate, we were unable to find any existing solutions for programmatic deckbuilding specifically for Magic: The Gathering. We have therefore eliminated RQ1.2 from our further research and we will focus on RQ1.1: *If no current systems exists, how can such a system be created?*

3.3.2 RQ2: What is the strength of the evidence in support of the different solutions?

As a part of our systematic literature review we have evaluated the quality of the reviewed studies, in order to determine the strength of the evidence in accordance with RQ2. This process is detailed in Section 3.2.7. In this section we discuss the results.

All of the collected articles achieved good scores in our quality assessment, with the worst score being 9 out of 10. None of the articles were marked 0 on any of the questions. A good result here is to be expected, as the initial filtering process should have excluded any studies of poor quality.

QC1, QC2 and QC3

All the studies achieve full score on the two first criteria, clearly stating the aim of the research and putting the research into context of other studies. Most of the articles did well to justify their algorithmic design decisions, the exceptions being S4 and S6. In S4 the author does not propose an algorithmic design but tests the effect of the crossover operator in a well known evolutionary algorithm. S6 compares the efficiency of different crossover operators on a well known evolutionary algorithm. Both are rated 0,5 points on this criteria. While they do not explicitly state the rationale behind the examined operators, the choice of algorithm itself is the premise of the papers.

QC4, QC5 and QC6

All of the articles had reproducible data sets, except S3, which used randomly generated test sets. This, combined with the fact that all the included articles describes reproducible algorithms and experimental procedures, shows that the overall reproducibility of the selected papers is good. The only exception here is S8, which does not describe the implementation of the Golden Ball algorithm, but refers the reader to another publication that does.

QC7

Most of the articles clearly state which algorithms they compare, the exception being S1 where no comparison is performed. We have thus rated S1 half point, as it does not technically fail on this criteria.

QC8

The chosen performance metrics are well justified. The only article to be given a half point on this criteria is S8. The performance metrics used are distance and computational time. As these metrics are obvious choices when comparing algorithmic performance on the traveling salesman problem, we agreed that this was not a choice that required much justification.

QC9 and QC10

S7 and S9 does not thoroughly analyze the presented test results. Much can be inferred from the prsented results, as the premise for these articles is a comparison between performance of different techniques. There is some analysis of the results, and the articles have been scored based on the admittedly subjective definition of the word "thoroughly". In all of the articles the presented findings are supported by the test results.

As shown, all the found articles are of high quality, and there seems to be strong evidence in support of the different solutions for the various combinatorial optimization problems examined.

3.3.3 RQ3: How can our findings be applied when creating a system for programmatic deckbuilding?

Deckbuilding is a very specific combinatorial optimization problem. While only one of the included studies directly discuss deckbuilding, and for a different game, all the other studies deals with other forms of combinatorial optimization problems. We here discuss how the knowledge gained from the studies can be applied to the problem of deckbuilding in Magic: The Gathering.

S1 Evolutionary Deckbuilding in HearthStone

In their paper, *Evolutionary Deckbuilding in HearthStone*, García-Sánchez et al. 2016 propose and implement a genetic algorithm for creating decks in trading card games, using Hearthstone as a case study. The produced decks are analysed and tested against strong decks designed by humans. From their experiments they show that their methodology is indeed able to create competitive decks for the two attempted player classes in Hearthstone. While their results does not conclusively transfer to Magic: The Gathering, the strong similarities between the two games hint that this approach could be applicable to deckbuilding in MTG as well. However, there are some important differences between Hearthstone and MTG.

Hearthstone was designed with simplicity in mind, and wile the game has some depth, it does not feature the amount of different mechanics that MTG does. In Hearthstone all cards require the same type of mana. Mana is automatically generated at a fixed rate, with one additional mana added per turn. This is different from MTG, where the player must manage both mana and spell cards together. All cards can be used together in MTG. In Hearthstone many of the cards are limited to one of the different classes that the players choose to play as. This limits the search space considerably.

S2 A simulated annealing method based on a specialised evolutionary algorithm

In this paper, the authors present a novel approach for searching in a solution space using **s**imulated **a**nnealing based on a **s**pecialised **e**volutionary **a**lgorithm, SASEA (García-Martínez et al., 2012). Comparing their proposed algorithm with other hybrid algorithms, traditional techniques and other optimizers they show that their approach of combining evolutionary algorithms with simulated annealing achieves better results on several of the problems chosen for the test. As most optimization problems, including deckbuilding, can be formulated as a search problem, this hybrid method might be applicable to our problem.

S3 Simulated annealing and combinatorial optimization

This paper presents a class for general adaptive heuristics and compare the performance of simulated annealing with the sequence method. The paper concludes that for some problems when using randomization heuristics such as simulated annealing, having a good starting solution will yield a better result than a randomly chosen one, but this difference evens out with longer runtime of the algorithm (Nahar et al., 1986).

S4 On the effectiveness of genetic search in combinatorial optimization

This paper compares the performance of three algorithms on the well known max-clique problem (Cormen et al., 2009). The first is the full procedure of a genetic algorithm, crossover, mutation and selection. The second is a reduced version, with only mutation and selection. The third is an implementation of simulated annealing. This is done in order to evaluate contribution and importance of the crossover operator when solving combinatorial optimization problems.

The results from the experiments performed show that the reduced version of the genetic algorithm outperforms the full version with respect to CPU time, and that found solutions are mostly of the same quality. The simulated annealing approach outperforms both of the genetic algorithms (Park and Carter, 1995).

The paper suggest that the crossover operator contributes marginally for problems where the building block hypothesis does not hold true, and that the additional CPU-time required clearly outweighs the miniscule improvement. In addition simulated annealing might be superior to both in these cases. This suggests that whether or not the building block theorem applies to deckbuilding is important when choosing a method.

S5 Evolutionary-based iterative local search algorithm for the shortest common supersequence problem

This paper suggest two extensions to the POEMS algorithm (Kubalik and Faigl, 2006) for solving the shortest common supersequence problem. This paper is highly specific for this one combinatorial optimization problem, and it shows that evolutionary algorithms can perform well in combinatorial optimization (Kubalík, 2011).

S6 Crossover operators for multiobjective k-subset selection

This paper discusses the k-subset selection problem, the problem of choosing a subset from a larger set while optimizing for some factors (Meinl and Berthold, 2009). The authors compare the effect of different crossover functions when using the multiobjective NSGA-II algorithm (Deb et al., 2002) to solve a k-subset problem. For their experiments, both a real life and a randomly generated data set is used. The compared functions are a single point crossover on binary encoded individuals, a two-point crossover on individuals represented by integer lists, and a pseudolinear crossover also on integer based individuals.

The results of the experiments show that the linear crossover function offers faster convergence than the other methods, but is eventually outperformed on the real life dataset. On the randomly generated dataset the linear crossover is superior to the two other methods with respect both to speed and results.

This paper is especially interesting, as the described problem of k-subset selection is close to the problem of deckbuilding. The choice of crossover and genetic representation is clearly important and needs to be selected according to the problem to be solved.

S7 Optimizing low-discrepancy sequences with an evolutionary algorithm

This paper discusses the application of an evolutionary algorithm to create low-discrepancy sequences with better space filling properties than uniformly distributed random numbers (Rainville et al., 2009). Low-discrepancy sequences are used for sampling. The authors show that their proposed method performs significantly better than the more established methods. While this paper does not directly relate to deckbuilding, it shows that evolutionary algorithms are versatile in their applications.

S8 Comparison between Golden Ball Meta-heuristic, Evolutionary Simulated Annealing and Tabu Search for the Traveling Salesman Problem

This paper presents an algorithm based on Binary Swarm Particle Optimization and Newtonian motion laws for solving combinatorial optimization problems (Osaba et al., 2016). The algorithm is compared with traditional binary swarm optimization and genetic algorithms on the binary 0-1 multidimensional knapsack problem. The results from the comparison show that the new algorithm offers better results and faster convergence. This algorithm also comes with the inherent benefits such as easy implementation and no need for setting algorithmic specific parameters.

S9 Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems

This paper presents an algorithm based on Binary Swarm Particle Optimization and Newtonian motion laws for solving combinatorial optimization problems. The algorithm is compared with traditional binary swarm optimization and genetic algorithms on the binary 0-1 multidimensional knapsack problem. The results from the comparison show that the new algorithm offers better results and faster convergence (Beheshti et al., 2013). This algorithm also comes with the inherent benefits such as easy implementation and no need for setting algorithmic specific parameters.

Distribution of articles

To gain an overview for analyzing the results we categorized the found studies. The search matrix contained three different solution methods, and we here sort the found articles in three columns. In addition there is the "Other" column for the cases where methods not specified in the search matrix were studied. In some of the articles multiple solutions were discussed, these articles are placed in multiple columns.

Table 3.6: The distribution of studies based on solution type.

	Simulated annealing	Evolutionary algorithms	Neural networks	Other
	S2	S1		S8
	S3	S2		S9
	S8	S4		
		S5		
		S6		
		S7		
Total:	3	6	0	2

3.3.4 Conclusion

As seen in Table 3.6 there were a majority of studies related to evolutionary algorithms. S1 shows that a very similar problem, deckbuilding in Hearthstone, can be solved using a genetic algorithm. S5 shows that genetic algorithms can solve the max-clique problem, which is a COP. Since COPs in essence can be reduced to selecting k elements from a set, it also follows that we can use this for solving deckbuilding. S6 shows that a genetic algorithm with linear crossover works well in for the k subset selection problem, which is similar to deckbuilding. There were also some support for simulated annealing, as seen in S2, S3 and S8, as well as the novel methods described in S8 and S9. However, based on the number of articles in support of evolutionary algorithms and the contents of these articles we decided that a genetic algorithm would be the best approach for deckbuilding.

3.4 Challenges met

While conducting a systematic literature review provides many benefits to the researcher, it is the experience of these researches that it is not a task without challenges.

Increased workload

The strength of the methodology comes from its adherence to a predefined protocol and a strict framework. While this has great benefits with respect to reproducibility and reviewability, it is our experience this also makes conducting this type of review more demanding on the available resources compared to more traditional reviews. A significant portion of the allotted time for this thesis was spent on familiarizing ourselves with the methodology, developing the protocol, evaluating the protocol and updating it when required.

Too rigorous?

The strictness of the process itself was also perceived as a challenge for the researchers. Continuously making sure that each step was carried out in accordance with the method was mentally taxing. This made the process seem like drudgery. We also experienced that focusing on adhering to the method had the effect of making us lose focus of the real issue to be researched. Put differently, focusing too hard on how the process was to be performed made us lose track of what we were trying to accomplish.

Challenges concerning computer science

Performing a systematic review in computer science comes with some challenges, compared to more established fields. There are many different naming conventions in computer science, and acronyms are prevalent. Compared to the field of medicine where terminology is more rigid (Lillegraven and Wolden, 2010), coming up with the right search terms might be more difficult in computer science.

Inexperienced researchers?

The mentioned problems might in part be caused by the lack of experience with this form of work in the researchers, and not necessarily an inherent problem with the method. We postulate that the methodology might not be suited for everyone, and will work better for PHD students or students that excel at systematic and structured research.

3.5 Summary

In this review we have surveyed existing solutions for deckbuilding in Magic: The Gathering, and generally for combinatorial optimization problems, in order to find an approach to generating strongly performing decks in Magic: The Gathering.

A protocol for conducting a systematic literature review was defined. The protocol was followed and the resulting review is presented in this chapter.

A selection of related work was collected from several digital libraries. These were filtered based on inclusion and quality criteria defined in the review protocol. The resulting list of articles was included and further reviewed.

We have evaluated the quality of the selected articles in order to judge the strength of the evidence. Each collected article has been thoroughly read and evaluated on ten different quality criteria. As a result, we found that the literature on our branch of combinatorial optimization holds a high level of quality.

We have discussed the results of the review with respect to the research questions defined in the review protocol and compared the different proposed solutions for the different combinatorial optimization problems.

Based on the reviewed articles we have concluded that an evolutionary approach to deckbuilding in Magic: The Gathering can be viable. This will be pursued in the following chapters.

Chapter 4

Designing a deckbuilder

In this chapter we describe the proposed solution for programmatically building decks in Magic: The Gathering, and how it was implemented.

4.1 Proposal

Based on the findings in our literature review, we have concluded that a genetic algorithm can be utilized to solve the optimization problem that deckbuilding is. We therefore propose to use a genetic algorithm for optimizing decks under the limitations of the sealed format in Magic: The Gathering.

Starting with randomly drawn decks from a card pool emulating a real sealed scenario, an evolutionary process is applied, mutating and combining the strongest performing decks. A genetic algorithm is not guaranteed to find the global maximum of the search space, but will find solutions that in some cases can be good enough.

We made the choice to restrict the scope of the solution to the sealed format due to the reduction in solution space, as well as the interesting constraints it puts on the potential decks. A sealed pool is the result of a stochastic process, and in order to produce strong decks one might need to utilize card combinations that otherwise would be overlooked.

4.2 Requirements

To help us with designing and developing our solution we defined four high level requirements for our system.

R1 The system should take a card pool as input.

As our proposed design is tailored for the sealed scenario, it should take a sealed pool as input. This input is the search space in which the algorithm will perform its task of finding strong performing decks.

R2 The system should output one or more decks.

The goal of the system is to find well performing decks in the search space, and return the strongest solution.

R3 The system should be able to evaluate the individuals in parallel.

It was evident from the start that the evaluation step would be the bottleneck in our implementation, and in order to make running the algorithm within reasonable time, we needed to be able to perform the evaluation step concurrently.

R4 The system should log and visualize the performance during the process.

In order for us to be able to reason about the performance of the algorithm, the system should provide insight into how the strength of the candidates evolves during the process.

4.3 Genetic Algorithms

A genetic algorithm is a metaheuristic partly comparable to biological evolution, and is an example of the class of evolutionary algorithms. Genetic algorithms are a tried and tested learning method, and has been applied with success to many learning and optimization problems (Mitchell, 1997, p 249). Genetic algorithms work well in solution spaces where the intricate interactions between elements can be challenging to model and predict, as certainly is the case with decks in MTG.

After first generating the initial generation of solutions, either by guessing or based on a hypothesis, genetic algorithms are applied in primarily two main steps. These are the genetic operators step and the selection step. The genetic operators are crossover and mutation. The selection step itself consists of two parts, evaluating the fitness of the population and selecting the individuals for the next generation. These four components are all called from a main loop driving the algorithm and incrementing the generation. A simple visualization of this process can be seen in Figure 4.1.

4.3.1 Fitness evaluation

A fitness function is a way to measure the performance, or "strength", of a particular individual in an evolutionary algorithm. It is crucial in an algorithm like this, because evolutionary algorithms need to be able to rank any population of individuals from best to worst. The fitness values can either be relative to each other, or on an absolute scale. There are many different ways to calculate fitness, and they usually has to be tailored to the problem being solved. In the case of rating decks in Magic: The Gathering we considered various options, including those listed below.

Expert evaluation

Having an expert player judge a deck is a pretty good way of determining its strength. An experienced player will know what types of cards are required in order for a deck to

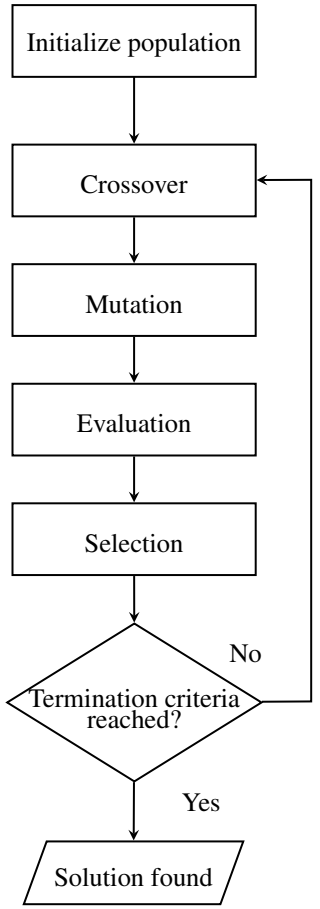

Figure 4.1: Flowchart showing an example of an evolutionary process.

perform well. Expert players can also create decks that rely on positive synergy between cards as well.

A drawback with this approach is that it is labour intensive, and not well suited for being incorporated in an automated system. The time needed to judge a deck is depending on a number of things, including skill, complexity of set and the order of accuracy wanted, but anything less than five minutes is not to expected. Ordering the list of decks from strongest to weakest would also be challenging, as humans may have trouble with accurately sorting a large set of complex entities.

Custom evaluation criteria

It is feasible that one could create a set of criteria, which all strong decks would need to follow. Factors like how many creatures it should contain, the number of removal spells, and what type of mana curve it should feature, could all potentially be used to gauge

whether a deck is good or not. The required ruleset for this approach is complex, and would require extensive research into the finer intricacies of the metagame in Magic: The Gathering. It would also be subject to constant change, as the metagame of MTG is in constant flux with new cards and mechanics being introduced.

Playtesting

When comparing the strength of decks in other tournament formats, tournament results are often used as basis for discussion. For formats such as Legacy and Modern, there are multiple websites which record and store such data. For the sealed format, however, this is not the case. Compared to the other tournament formats where the player is free to construct decks from almost all available cards, the Sealed format puts considerable constraints on the cards available to the player. Therefore, decks used in this format will vary greatly from the decks that are featured in the tournament results.

A solution to this would be to arrange tournaments where some players would use the generated decks, reporting their results for each generation. This could be done using a web app or other software, but would still be subject to challenges regarding time and human resources. Alternatively, one could employ artificial intelligence agents to play each other. This method is best suited for our purpose, and is expanded on in Section 4.4.6.

4.3.2 Selection

For each generation in the algorithm a subset of the current population has to be selected for further breeding and mutation, while the others are culled. Generally the most fit individuals are selected. There exist several strategies for selecting individuals, and like the fitness function, it might need to be chosen based on the problem.

4.3.3 Mutation

The mutation operator is how the genetic algorithm explores the search space and maintains genetic diversity in the population. Mutation is also important to avoid converging towards a local optima. The workings of the mutation operator will depend on the genome representation, but will, in general, change one or more of the properties of the individual.

4.3.4 Crossover

As mutation covers the exploration part of the algorithm, the crossover function covers the exploitation behaviour of the algorithm. Crossover, also known as recombination or breeding, is the process of taking elements from two or more parent individuals and creating children by combining the chromosomes. The idea behind this is that the building block hypothesis is valid for the given problem. If this is not the case, then there will be little point in utilizing a crossover (Park and Carter, 1995). The building block hypothesis, as described by (Goldberg, 1989, p. 41), states:

> *Short, low order, and highly fit schemata are sampled, recombined [crossed over], and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata [the building blocks], we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.*

4.4 Design and Implementation

We here detail our implementation of the genetic algorithm and the various components that constitutes the deck builder.

4.4.1 Genetic algorithm

The genetic algorithm is implemented using DEAP, Distributed Evolutionary Algorithms in Python (Fortin et al., 2012). DEAP is a simple to use framework for evolutionary algorithms. It has broad selection of options regarding representation, methods for performing selection, mutation and crossover and other means of configuration.

In our implementation there are a few parameters that can be varied to explore different results. The various parameters and their function are shown in the Table 4.1.

Table 4.1: The different parameters used for the experiments.

Mutation Rate	The probability that a given individual mutates
Crossover Rate	The probability that two individuals mate
Number of Generations	The number of generations before terminating
Number of games per opponent	Number of games against each opponent
Opponents	Array of opponent decks, closer described in 4.4.6.

The genetic operators are applied by using DEAP's varAnd-method. This method both applies breeding and mutation independently based on the corresponding probabilities. In other words, one, both or neither genetic operator can occur in the same individual.

As the fitness function (Section 4.4.6) is rather computation intensive and time consuming, we needed to be able to do parallel evaluation of individuals for any experimentation to feasible finish within the time constraints of this project. As DEAP is built with this in mind, it works well with frameworks for distributed computing. One such framework is SCOOP, Scalable COncurrent Operations in Python (Hold-Geoffroy et al., 2014).

SCOOP allows for distributing task across multiple worker machines over network using SSH, with one central broker coordinating the process. We chose to implement our cluster on Google Compute Engine[1], as it offers easily configured virtual machines which allowed us to set up one machine and rapidly duplicate it, creating the worker nodes. As the individuals can be evaluated independently, we chose to have one worker node for each individual in the population, thus achieving full parallelization of the evaluation step. Google Compute Engine is a paid service, but all users get 300 USD worth of credits

[1] https://cloud.google.com/compute/

upon signing up. We chose to create 10 worker nodes and one broker node, and chose an population size of 10 for all tests and experiments.

4.4.2 Card pool

When playing the sealed format in Magic: The Gathering, the players compose their deck from a pool of cards, obtained by opening six booster packs, each containing 15 cards. In our implementation, we use card pools generated by a sealed pool simulator[2]. Card pools are represented as text files, containing a number and a card name for each card. These files are read and parsed. The pool is represented as a hashmap, with sequential integers as keys, and a tuple with the card name and the quantity of that card in the pool as value. We know a MTG deck needs lands in order to play cards. Therefore, 14 lands of each color were also added to the card pool. This way, every individual in the first generation will statistically get some amount of basic lands. This helps reduce the run time, because without any starting mana the individuals would need to mutate in a land card in order to use a single spell.

For our implementation and experiments, we primarily used card pools generated for the Aether Revolt set. This set was chosen because it was the most recent standard set released. The set being the second set in a block, also meant that the sealed pools would contain cards from both sets, which would increase variety, and make the problem more interesting. It is also a set where synergies between cards are important, and whether our solution can take advantage of them or not, will be important to it's success.

4.4.3 Chromosome representation

Typically, individuals in genetic algorithms are represented using an array of bits, with each bit representing an individual attribute (Mitchell, 1997, p. 252). The individuals in our population, the decks, consists of various cards from the card pool, and duplicates of some cards are present. We needed to be able to represent this. Therefore, we opted for using an array of integers. The integers each corresponds to a card in the card pool.

By contrast, having n bits representing the n allowed instances of each card would require more complex generation of individuals. By representing individuals as fixed length integer arrays, we avoided potential errors during development, and made the code easier to reason about. As DEAP supports representation using almost any imaginable data structure, this was simple to implement.

The initial generation is generated by drawing random cards from the card pool, using the randint function in Python's random module and checking if the amount of any given card was within the bounds given by the card pool, until the individuals contained 40 cards.

4.4.4 Crossover

When applying the crossover function, we wanted to retain the validity of the decks by not having any of the resulting offspring containing more than the allowed number of copies of a card, determined by availability in the card pool. This could be the case if crossing

[2] http://www.magicdrafting.com/aether-revolt-sealed

over two individuals using more traditional methods, such as a one or two point crossover. To counter this, we implemented an adaptation of a linear crossover function.

The function takes two individuals, combine their integer arrays and sort the combined array. The two resulting offspring are then produced by selecting the elements on the odd positions for the first child and the even positions for the second child. This ensures that no offspring ends up with more than the limited amount of each card in the pool.

4.4.5 Mutation

In our implementation, mutation is done by randomly trying to swap one card for another card from the card pool until a legal swap, that is, one that does not invalidate the deck by having too many of any given card, has been done. As with the initial generation of individuals, this is done using the randint function, first selecting a random mutation site, and then replacing that card with a randomly drawn card. The mutated individual is then returned.

4.4.6 Fitness Function

In order to rate the fitness value of each candidate deck, we chose to test how well they performed in play. As playing actual matches with the generated decks were out of the question for obvious reasons, we opted for simulating the games using artificial intelligence agents playing with the candidate decks, as well as opponent decks. Designing and implementing such an MTG AI would constitute the workload of a master's thesis on it's own, if not several. Fortunately, there exists many open source AI engines for Magic: The Gathering, as seen in Table 4.2.

Table 4.2: List of MTG AIs, taken from: https://www.slightlymagic.net/wiki/List_of_MTG_Engines

Name	**AI**	**AI vs AI**	**Implemented cards**
Forge	Simple	Yes	16697
Incantus	None	?	2583+
XMage	Mad, DraftBot	During Tournament	23929[3]
BotArena	Minimax	?	10744
Magarena	Minimax/Monte Carlo/Vegas	Yes	11404
Multiverse	None	No	1500+
Wagic	Simple	Limited	9000+
Manalink 3.0	Simple	No	12869
Magicgrove	Minimax	Yes	690

Comparing the different engines, we first decided on using Magarena, as it is the most sophisticated engine, and widely recognized as the strongest open source MTG AI. Our rationale here was that it would provide the most realistic testbed for our candidate decks.

[3] XMage has more cards implemented than there exists because reprints are counted as different cards.

It also sports the most active developer community and have frequent releases, which could become important during the implementation phase.

Unfortunately, upon testing we discovered that the Magarena engine, while being a strong player, uses a prohibitive amount of time per match. Depending on the chosen strength level for the AI, a match could take upwards of fifteen minutes. While this might not seem like an unreasonable amount of time, we wanted to run several matches for each individual in the population against each of our opponent decks. There are several elements of a match in MTG that are influenced by chance, and we wanted to average out the results to avoid decks receiving a higher or lower fitness value than deserved. Magarena also lacked over 2000 cards in its implementation at the time this was written. Since replacing unimplemented cards with other cards would not correctly represent a realistic sealed pool, we require an engine with every card from the desired set.

We finally decided on using Forge as our chosen AI. It has many of the same benefits as Magarena, with the addition of being much faster. While the AI is not as advanced, each match can be completed in seconds instead of minutes. Forge is distributed as a full featured game, but can also be used as a command line tool without any graphical user interface. This was mandatory in order for us to integrate it with the genetic algorithm. Forge also has more cards implemented than the other available engines, reducing the need for screening the used sealed pools for cards that are not supported. The list of implemented cards includes almost all of the Kaladesh and Aether Revolt cards.

A downside of using Forge is the lack of insight into how the algorithm works. As the development process has been moved between different hosts during its development, it is hard to find any reliable documentation on how the AI is implemented. Unlike Magarena, where there are plenty of documentation, and an open github repository.

Python's subprocess module is used to start Forge and read the resulting output from each match. When evaluating the fitness of the individuals, each candidate deck is matched against our opponent decks, with the number of matches determined by a parameter of the algorithm. The output from the matches is parsed, and the fitness values are then calculated as the total win percentage of all the matches. Using SCOOP, this process is distributed across the cluster, with each individual being evaluated on a separate node in parallel.

Opponents

An important aspect when designing our fitness function was deciding upon which opponents our candidate decks should be tested against. García-Sánchez et al. (2016) used an external site to determine what opponents the AI should play against. That was based on the most predominant decks in a specific season. As mentioned before, there is no available and similar datasets for the Sealed format. Instead, we opted for generating opponent decks by asking multiple seasoned players to create sealed decks based on randomly drawn Sealed pools using a website [4] that generates sealed pools.

Since we wanted our opponents to represent what one could realistically meet in an actual tournament, we had 8 different decks created. One of these was removed due to strong similarities with another, and one deck was deemed too poorly made. The six remaining decks are a varied mixture of aggro, midrange and control decks, and have quite

[4] http://www.magicdrafting.com/aether-revolt-sealed

distinctive features, and all the five colors are represented. Not all decks were used for every test. This was partly because some of the tests were performed before the external experts asked to create decks for us responded, and also because we in some instances wanted to test the performance against specific sets.

We here present an overview of each of the six used opponent decks. For a complete list of cards for these decks, see Appendix B. The names are based on the primary colors of the decks. W for **w**hite, U for bl**u**e, B for **b**lack, R for **r**ed and G for **g**reen. Lower case letters indicate a small, but not insignificant, number of cards in an additional color. This is known as a splash color.

Table 4.3: Opponents used in the experiments.

Name	Description
GB	A green and black control deck. This deck performs decently in the early game, and a well during mid-game. This deck is favored to win if the game goes on.
UWg	A primarily blue and white control deck, with some green. The green cards are very strong, which weighs up for the inconsistency from including a third color.
RG	A red and green aggro deck. This uses cheap and efficient creatures to defeat the opponent early in the game.
GBw	This is a green and black midrange deck, with some white in it. This was not constructed from a good sealed pool, and was expected to do poorly. This is essentially a weaker version of GB.
UR	This is a red and blue midrange deck. This deck has a good mix of cheap and expensive creatures, and can do well both in the early and late game.
RW	This is a red and white aggro deck. This deck was built from a very good card pool, and was expected to be a strong opponent.

A tournament between the opponent decks using the Forge AI was held. This was to gauge the difference in power-level between them, to get an indication of the best decks, and to see if the Forge-engine favored one type of decks over the others. Every deck played 1250 games versus every other deck, which means that the win percentage is derived from 2500 games between different decks, and 1250 mirror-matches. The mirror-matches, matches between the same decks, should yield an approximate 50 % win percentage, unless the program does not perform symmetrical between equal opponents. The results can be seen in Table 4.4.

Table 4.4: The win rate for each deck in the first column versus every subsequent column.

	GB	**UWg**	**RG**	**BGW**	**UR**	**RW**	**Average vs everyone else**
GB	49.52	29.4+	26.32	53.24	44.04	13.92	33.38
UWg	70.60	48.56	55.24	80.16	65.40	44.60	63.20
RG	73.68	44.76	52.80	77.00	52.32	42.56	58.06
GBw	46.76	19.84	23.00	50.40	38.48	14.24	28.46
UR	55.96	34.60	47.68	61.52	50.32	30.00	45.95
RW	86.08	55.40	57.44	85.76	70.00	48.08	70.94

It is clear that some matchups were better than others. RW performed the best, winning every matchup more than 50% of the times, except against itself. BGw on the other hand, won less than 50% in every matchup, having an average win percentage at 28.5%. We are unsure if the great variance in performance among the decks are due to a fault in Forge, or because the decks themselves just are very uneven quality wise. A more thorough study on these results and Forge would be ideal.

4.4.7 Selection

Selecting which individuals to be carried over to the next generation is done by using DEAP's *selTournament* method. This is a tournament selection, which takes the current generation and parameters k and t, and returns a collection of k individuals to serve as the next generation.

The tournament selection works by randomly selecting a subset of size t, and then selecting the strongest individual in that tournament. This is repeated k times. This ensures that a good solution is much more likely to be included in the next generation, while the t-1 weakest individuals will always be culled.

4.4.8 Termination strategy

The generational process is terminated when one of the following conditions are met:

- The defined number of generations is reached.
- The median fitness score have reached a threshold of 60%.
- The algorithm is plateauing and has not found a better solution for 60 generations.

Any of the termination conditions will end the main loop and the algorithms proceeds to logging and visualization.

4.4.9 Logging and visualization

After termination the results are written to a log file, containing the used parameters, top, median and lowest fitness score for each generation, the card list for the strongest performing deck, as well as the time spent. A graph showing the fitness score for each generation

is also plotted, using the widely used matplotlib library. For convenience the log file and the graph are sent by email to the researchers. For archive purposes, each candidate deck for each generation is also saved in a folder structure on the main node.

Chapter 5

Experiments

To test the performance of our solution we conducted several experiments. In this chapter we describe how we tested different parameters of the algorithm in order to calibrate it, how we performed our experiments and of the results of those experiments.

5.1 Introduction

Although we achieved good parallelization by evaluating all individuals concurrently, the runtime of each experiment was still long, spanning several days. Therefore, we were only able to perform a limited amount of experiments while varying the parameters of the algorithm in order to find the optimal configuration. In Section 5.2 we outline our planned strategy for running tests in order to calibrate the parameters. The experiments, the parameters used and the results are presented in Section 5.3. In Chapter 6 we discuss our findings.

5.2 Testing Strategy

As this work was conducted under limited resources with respect to time and and budget, a well defined plan for running the testing was required. We here detail our plan and how well we were able to adhere to this plan.

5.2.1 Planned tests

In the interest of determining which combinations of parameters that would produce the best decks, we planned on running several test runs. For each test we would use different parameters and log the results. If some combination seemed exceedingly promising we would adjust our plan accordingly in hopes of finding even better results. The number of tests planned was based on the average time each of the initial tests ran for and the remaining time of the project. The initial plan for the tests are shown in Table 5.1.

Table 5.1: Planned tests with different values for crossover and mutation probability. X denotes values that were to be determined based on previous tests.

Nr	Crossover	Mutation
1	0.2	0.2
2	0.5	0.2
3	0.7	0.2
4	0.9	0.2
5	0.2	0.5
6	0.2	0.7
7	0.2	0.9
8	0.X	0.X
9	0.X	0.X
10	0.X	0.X

5.2.2 Deviation from the plan

We experienced problems when performing our tests. Due to technical issues with memory overflow on our nodes we ended up spending a significant time running tests that never terminated. Time was also spent trying to fix this issue. Running 11 nodes on Google's cloud service is expensive, and we quickly ran out of the free credits we received when we signed up. Another problem we encountered was that the fitness evaluation process would hang on some nodes, never terminating. Unfortunately, the developers of SCOOP has as of writing yet to implement how to handle timeouts, and as a result our algorithm would never proceed to the next generation. We were unable to figure out the cause of this problem, and were not able complete all the planned tests.

5.3 Experiments

We performed two experiments after the testing. These are presented with focus on the deck from each experiment that achieved the overall best fitness score. We chose this over using an individual from the last generation based on the assumption that our fitness function correctly assessed the individuals and that a subsequent generation might be weaker than the one before due to the genetic operators.

We enlisted the help from an expert[1] to examine the two selected decks and evaluate them with respect to the card pools they were generated from, as this is a substantial limiting factor. The expert also constructed a deck from each of the card pools in order to provide a baseline for what a strong deck could look like and how it would perform. These are included in Appendix D. The expert gave a rating to every card in the generated deck, and compared it to the cards in baseline deck, while also rating them on how good of an inclusion it was in the deck. The ratings were:

[1] This expert was Sveinung Knudsen Nøding. He is the reigning norwegian Legacy champion, and also a seasoned Draft and Sealed player. He has played MTG actively for more than a decade.

- **Good card:** This card appear in the optimal deck in the sealed pool created by the expert.
- **Potentially good:** A card was given this rating if it was a good card, but a bad inclusion in the generated deck.
- **Bad:** This card should not be included in any deck created from this sealed pool, or it is a good card, but very suboptimal in the generated deck.

5.3.1 Experiment 1

This experiment was conducting on the sealed pool found in Appendix C.1. The parameters used can be found in Table 5.2.

Table 5.2: The parameters for Experiment 1.

Mutation rate	0.5
Crossover rate	0.7
Number of generations	200
Matches per opponent	50
Opponents	GB, UWg, RG, GBw, UR, RW
Completion time	63 hours

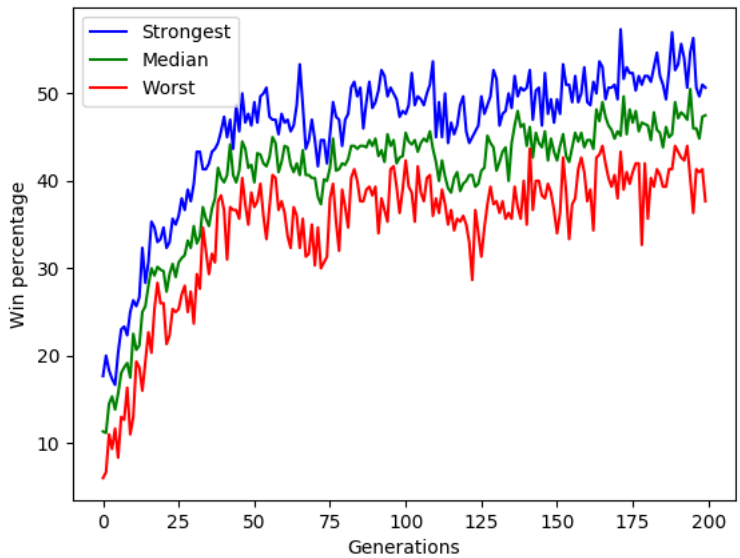

Figure 5.1: Graph of the strongest, median and weakest individual for each generation.

Some things can be observed from the graph. Major improvements happen during the first 50 generations, before the growth rate tapers off. The graph also shows some tendencies to overall climb during the last 150 generations.

This is the cardlist of the best performing deck from this experiment. The overall win rate from the fitness evaluation was 57.3%.

Creatures (21)
1x Accomplished Automaton
1x Aether Poisoner
1x Aethertorch Renegade
2x Audacious Infiltrator
1x Augmenting Automaton
1x Countless Gears Renegade
1x Dawnfeather Eagle
1x Filigree Familiar
1x Freejam Regent
1x Glint-Sleeve Artisan
1x Kari Zev, Skyship Raider
1x Maulfist Squad
1x Ovalchase Daredevil
1 Reckless Fireweaver
1x Scrapper Champion
1x Thriving Rhino
1x Vengeful Rebel
2x Verdant Automaton
1x Weldfast Wingsmith

Instants (1)
1x Shock

Artifacts (1)
1x Aethersphere Harvester

Lands (17)
6x Mountain (red mana)
4x Swamp (black mana)
4x Plains (white mana)
1x Island (blue mana)
2x Forest (green mana)

Here is the expert's ratings of the cards in the deck.

Good inclusions (24):
1x Accomplished Automaton
1x Aether Poisoner
1x Aethersphere Harvester
1x Aethertorch Renegade
1x Augmenting Automaton
1x Filigree Familiar
1x Freejam Regent
1x Kari Zev, Skyship Raider
1x Maulfist Squad
6x Mountain
1x Ovalchase Daredevil
1x Reckless Fireweaver
1x Scrapper Champion
1x Shock
4x Swamp
1x Vengeful Rebel

Potentially good inclusions (9):
2x Audacious Infiltrator
1x Countless Gears Renegade
1x Dawnfeather Eagle
1x Glint-Sleeve Artisan
4x Plains

Bad inclusions (7):
2x Forest
1x Island
1x Thriving Rhino
2x Verdant Automaton
1x Weldfast Wingsmith

Expert's verdict

This deck had seven bad inclusions. Those are cards that would never be added by a real player. There is no reason to increase the number of colors in this deck up to five, in order to add these cards. To cast the card "Weldfast Wingsmith", the single island in the deck is needed in play. That is a suboptimal card, even if one can cast it, and in this deck one almost never can. The same goes for the other cards marked as bad. The expert marked every white card as potentially good, instead of good. This is done, because all the white cards should be replaced by blue or black cards in order to make the deck more consistent. This does not mean the white cards are bad, just that they don't fit in the deck.

To more accurately evaluate the strength of the deck under the Forge AI we ran a set of 1000 matches against each opponent. The same was done with the reference deck designed by the expert to compare their performance.The win percentage against each opponent deck is shown in the Table 5.3. The deck created by the expert can be found in Appendix D.1

Table 5.3: Win rates for the generated deck and the expert created deck.

	GB	**UWg**	**RG**	**GBw**	**UR**	**RW**	**Average**
Generated deck	0.712	0.303	0.493	0.733	0.549	0.37	0.526
Expert deck	0.781	0.451	0.551	0.858	0.702	0.464	0.635

5.3.2 Experiment 2

This experiment was conducting on the sealed pool found in Appendix C.2. The parameters used can be found in Table 5.4.

Table 5.4: The parameters used for Experiment 2.

Mutation rate	0.5
Crossover rate	0.5
Number of generations	350
Matches per opponent	50
Opponents	GB, UWg, RG, GBw, UR, RW
Completion time	43 hours

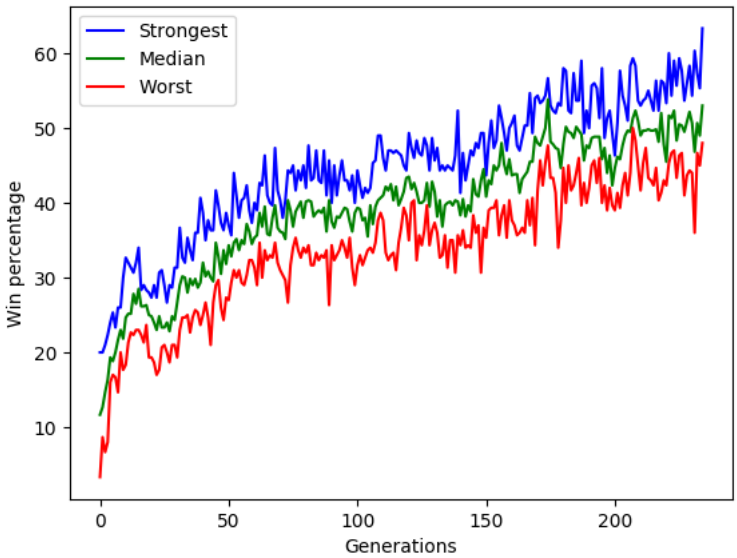

Figure 5.2: Graph showing the performance of the strongest, median and weakest deck for each generation.

The experiment was set to run for 350 generations, but it terminated after 235 generations by the termination criteria fulfilled when there was no improvement to the median score over 60 generations. The graph again shows rapid improvement the beginning, and a more gradual improvement until termination of the experiment.

The generated deck:

Creatures (15)
1 Aether Poisoner
2 Audacious Infiltrator
1 Countless Gears Renegade
2 Dawnfeather Eagle
1 Dukhara Scavenger
1 Filigree Familiar
1 Foundry Screecher
1 Herald Of Anguish
1 Hinterland Drake
1 Shipwreck Moray
1 Spire Patrol
1 Vedalken Blademaster
1 Welder Automaton
1 Wind-Kin Raiders

Artifacts (4)
1 Bomat Bazaar barge
1 implement of Malice
1 Planar Bridge
1 Renegade Map

Instants (1)
1 Skywhaler's Shot

Enchantments (2)
2 Caught in the Brights

Lands (18)
5 Island (blue mana)
2 Mountain (red mana)

6 Plains (white mana)

5 Swamp (black mana

The rating of the cards from the expert:

Good inclusions (25):
1x Aether Poisoner
1x Bomat Bazaar Barge
2x Caught in the Brights
1x Filigree Familiar
1x Herald of Anguish
5x Island
6x Plains
1x Renegade Map
1x Skywhaler's Shot
5x Swamp
1x Wind-Kin Raiders

Potentially good inclusions (12):
2x Audacious Infiltrator
1x Countless Gears Renegade
1x Dawnfeather Eagle
1x Dukhara Scavenger
1x Foundry Screecher
1x Hinterland Drake
1x Implement of Malice
1x Planar Bridge
1x Shipwreck Moray
1x Spire Patrol
1x Vedalken Blademaster

Bad inclusions (3):
2x Mountain
1x Welder Automaton

Expert's verdict

This deck only had three bad cards, which is better than the first generated deck. This deck is almost exclusively three colored, with the exception of two mountains and a single creature card, which makes the deck more consistent than the one from Experiment 1. The expert said that a control deck with only blue and black colored cards and no white would be more optimal than the three color deck generated by the solution. Some of the cards that got marked as potentially good is still included in the deck created by the expert. He said this was due to them having bad synergy in the generated deck, but that in a suited deck they would be good. Some cards white cards like "Caught in the Brights", are cards that are marked as good inclusion, even though they don't appear in the expert deck. This is because he has deemed white colored cards unnecessary in his deck, but if his deck were white, these cards would be excellent. The expert also said that the deck is not very good, but it is not unrealistic to see a deck of equal quality in a tournament.

Again a set of 1000 matches against each opponent was performed for both the generated and the expert's deck. The deck created by the expert can be found in Appendix D.2

Table 5.5: Win rates for the generated deck and the expert created deck.

	GB	**UWg**	**RG**	**GBw**	**UR**	**RW**	**Average**
Generated deck	0.717	0.400	0.421	0.75	0.596	0.346	0.538
Expert deck	0.452	0.317	0.259	0.578	0.516	0.205	0.387

Interestingly, the deck created by the expert performed worse than the solution deck generated in this experiment. This is discussed further in Chapter 6

Chapter 6

Discussion

In which we discuss our results, including the strengths and limitations of the proposed solution and the implementation.

6.1 Results

To reason about Goal 3 and the strength of the generated decks, the results from the experiments need to be analyzed. We'll begin by analyzing the results from the individual experiments, then we will compare them to the decks the expert created, and discuss what improvements could be done to the solution. The decks created in Experiment 1 and Experiment 2 will be analyzed and discussed with regard to synergy, curve, land count, colors and overall focus.

6.1.1 Experiment 1

The implemented solution generated a deck with all five colors of mana, where only three of them were predominant. Our expert proposed that a deck with only two colors, red and black, would be better (Appendix D.1). Every red and black card in the generated deck would have been included in the deck created by the expert. However, there are some weaknesses with the generated deck, and they are outlined in more detail in the sections below.

Colors:

The deck features all five colors of mana. This is very suboptimal, because most decks have at most three colors. Removing two of them would improve the deck immensely, because it would become more consistent. The two colors with the least amount of cards in the deck, blue and green, drags the rest of the deck down, and makes it more unreliable, without bringing in any powerful cards to make up for the inconsistency. The expert would also have removed all white cards from the deck, to further make it more consistent.

Synergy:

The deck features some synergy between the cards. It has multiple cards that create "energy", a mechanic specific to the Kaladesh and Aether Revolt sets. It also features multiple cards that benefit from the energy mechanic in different ways, creating flexibility. This is similar to the deck created by the expert. The expert deck contains the same number of energy cards, only in more appropriate colors. The generated deck also contains multiple ways to create so-called "1/1 tokens", which is an important aspect of this set. However, it lacks some ways to benefit in full from these tokens. It is likely that there would be some synergy no matter what, in the deck, just due to the nature of the game. However, the synergy in this deck is similar to that of the professional deck.

Mana curve:

The curve of this deck is very suboptimal. It has more creatures costing two mana than any other cost. This would normally mean that the deck is very aggressive. However, the deck also contains multiple cards costing more than four. This is bad. A player would either end up running out of cards very quickly, or end up having expensive cards in its hand, which are never cast, because the opponents wins before he accumulates the required amount of mana. In comparison, the deck created by the expert has the same expensive cards, but the rest of the curve is centered around four mana instead of two, which is much better.

Lands:

There are 17 lands in the solution deck. This is good, because that is the normal amount when creating a sealed deck. The distribution of colors among the lands is however suboptimal. To maximize the chance of drawing a specific land color, multiple copies are needed. This deck only have six mountains to produce red mana, and then four or less of each other basic land type. In addition, the two forest and the island cards reduce the usable mana base. There should ideally be minimum 4 lands of the rarest color, unless there are special circumstances. Having less than this amount reduces the probability of having the necessary land type when you want to play cards of that color, which in turn affects your win rate and consistency.

Focus:

As mentioned previously, the deck lacks focus, and that makes it weak. It has many cards suited for early and aggressive play, while also featuring cards which will not be played before turn eight or nine at the earliest, due to the high mana cost. At that point most games will have ended if one plays with an aggressive deck, or against one. Generally it is a bad idea for a deck to try to do two things half-ways instead of committing fully to one strategy.

6.1.2 Experiment 2

Overall, this deck seems stronger than the one created in Experiment 1. It has fewer colors, and seems more balanced. It only has three cards which the expert rated as a bad inclusion.

That means the deck, with the exception of those three cards, could potentially have been created by a novice player. This is still weaker than the original ambition, but is quite a bit better than the result from Experiment 1.

Colors:

This deck features primarily three colors. This is a great improvement over the deck created during Experiment 1. As in Experiment 1, the expert suggested that a deck featuring only two colors, black and blue (Appendix D.2), would be optimal, given the underlying card pool. Both of these colors are represented in the generated deck, which is good. Removing white from the generated deck would make it more consistent and focused. The expert noted that having three colors could be justified, if one of them was less predominant than the others, and only to include very powerful cards. That is not the case for this deck however, which has cheap creatures in all three colors.

Synergy:

The deck has some good synergies, while also having some cards with poor synergy. An example of good synergy is "Renegade Map", which helps the deck being more consistent with its three colors by finding basic lands from the deck, while also helping trigger the mechanic "revolt" which some of the other cards feature. An example of bad synergy is "Implement of Malice", which requires several cards with a certain mechanic to become useful. The deck created by the expert contains five of those cards, which is why the card is marked as a suboptimal inclusion in the solution deck, while at the same time is included in the deck created by the expert.

Mana curve:

The curve is centered around cards costing three mana, while also having some good expensive cards. Some of the more expensive cards also has a mechanic which makes them cheaper to cast. The curve could benefit from being smoothed out a bit, like in the deck created by the expert, but it is decent.

Lands:

This deck has 18 land cards. Considering that the deck has three colors, and some late game cards like "Herald of the Anguish", this is a decent number, although 17 would be more fitting. There are two mountains among the land cards, and they should obviously be replaced for the same reasons as in Experiment 1. Trading them for another white and blue colored land would be ideal.

Focus:

This deck also lacks focus, and appears to be trying to do multiple things. As in Experiment 1, this deck contains cards that are best suited for aggressive decks, while also having late game cards, which is suboptimal in a aggro deck. An example is the card "Dukhara

Scavenger", which is only a suitable inclusion if there are enough high power cards in the deck.

6.1.3 General analysis

Every deck output by the solution, during both testing and experiments, contained cards of at least three different colors. The expert deemed this suboptimal in both experiments. Removing a color from a deck might require adding cards that are suboptimal short term. As this would be unfavorable for the algorithm, it got stuck in local maxima. This is a common problem with genetic algorithms (Laumanns et al., 2002). To ensure that the generated decks only contained two colors the initial generation could have been seeded with predetermined land cards. However, this would partly defeat the purpose of this experiment, since figuring out what colors to compose the deck of is an important aspect of deckbuilding.

There also seem to be less synergy in the generated decks, than in the expert created decks. This can be explained by the way mutation is implemented in our solution. Mutation swaps one card for another. This means in order for an individual to gain two cards with good synergy together, it must either mutate favorably twice, or be mated with an individual containing a match. Cards with strong synergy can often be poor on their own. This is a problem in a solution like this, where a bad card will affect the result of the fitness function, and a lower score means that there is a smaller likelihood that the individual will live on to the next generation.

The difference in the strength of the decks produced in Experiment 1 and 2 is interesting. Experiment 1 produced a deck which could not have been mistaken for even a mediocre player's deck. It has a bad curve, too many colors, bad land distribution between the colors and a general inconsistency. Experiment 2 yielded a more promising deck, which, with exception of some cards, could have been created by a player. It is hard to tell what the cause of this improvement is. The only difference in parameters for the algorithm are 0,5 in mutation rate instead of 0,7 and a higher generation limit. This experiment was set to run for 350 generations instead of 200. It only ran around 240 generations before it met a termination criteria. However, the improved result could also be due to running the experiment on another sealed pool, or just pure chance.

In one regard, the solution performed very well. It always converges towards the correct number of lands overall. The solution yielded the optimal or near optimal number of lands in every experiment and test we ran. Even though the decks did not always have the correct color distribution of lands, there were never lands without at least one card of that color. This means that the Forge AI works to some extent.

6.2 The proposed solution

Genetic algorithms can be subject to premature convergence (García-Martínez et al., 2012), and this seems to be the case in our work. While the fitness value across the population is shown to be increasing with time the growth decelerates early and none of the generated decks reached more than 60% win rate against the opponent decks, even when running for

hundreds of generations. We suspect that the population quickly developed into a monoculture due to the small population size (10) and the nature of the implemented selection method. The tournament selection always ends up killing off the two weakest individuals (for tournsize = 3) and replacing them with copies of more fit individuals to keep the population size constant.

In the proposed solution a third party AI is used for evaluating the fitness of the generated individuals. The assumption was that the AI would perform close enough to how humans play MTG, and that this would encourage the evolution of decks that would conform to the patterns present in human created decks. As seen from the test performed with the expertly created deck in Experiment 2, this does not hold true for all decks.

The difference in win rate between the generated deck and the expert deck is nearly 15 percent points. There is no reason to believe that the expert deck should perform worse than the generated deck. It is better with regard to consistency, curve, land composition and overall focus than the deck generated by the solution. This has multiple implications. The first implication is that the Forge AI does not accurately represent a real player. If it did, the expert deck should have performed better than the generated deck. This means that Forge favors some types of decks over others. If Forge is unable to map a good deck to good performance, then the result is never going to live up to real player's standard. However, the algorithm is still able to create decks that perform well in Forge, and both decks generated in the experiments had an average win rate above 50%. This is a decent win rate considering that the opponent decks were all built by seasoned players. This suggests that the algorithm itself works, even though Forge is a suboptimal substitute for a player. Using a better MTG AI would most likely yield better decks as a result. One possible explanation for the weak performance by the expert deck could be that Forge doe not exploit the synergies in the deck to the fullest.

The time aspect of running an experiment severely hampers the usability. The current solution has a runtime of over 48 hours when run on 11 nodes. The majority of time is spent on the fitness evaluation, relying on simulating several games for each iteration. At its current state, this implementation would be unsuitable for any real time purpose.

6.3 Conclusion

To conclude the thesis, we need to look at the goals we set when we started on the thesis. We need to compare these to the results we have discussed in the previous sections, and conclude whether we have achieved our goals.

> **G1** Survey the existing implementations for programmatic deckbuilding by conducting a systematic literature review.

We achieved this goal when we performed the Systematic Literature Review detailed in Chapter 3. We were unable to find any research on implementation of deckbuilding directly related to Magic: The Gathering. We did however find other research that helped us come to the conclusion that we should use genetic algorithms to implement this. Due to the nature of an SLR, we can determine that the articles we found are of high standard, which in turn strengthens the findings of this thesis.

G2 Propose and implement a system for programmatic deckbuilding using an approach based on the result from the literature review

During the SLR, the majority of articles we retrieved was related to genetic algorithms in one way or another. This led us to implement a genetic algorithm. We used an external AI in the implementation for the fitness function. Any decisions regarding the implementation is described in Chapter 4.

G3 The decks produced by the system must be of a high quality, when compared to decks created by seasoned players.

We have showed that it is possible to create decks using genetic algorithms. The decks performed well under the AI we used, and had an average win rate of over 50% versus six widely different opponents. However, the decks we produced lacked many of the defining qualities of the decks created by the expert. We believe this can be improved, and suggest some further work in Section 7.2.

Chapter 7

Summary and further work

Befitting the last chapter of our thesis, we here briefly summarize the process of our work. We also discuss points that would benefit from further research.

7.1 Summary

During this project we have explored the problem of deckbuilding in the popular trading card game Magic: The Gathering. We have shown that this problem can be interpreted as a combinatorial optimization problem, and that it is an interesting testbed for applying different strategies for solving such problems.

We have conducted a systematic literature review, researching the various approaches for solving combinatorial optimization problems. During this review, we have catalogued different methods, synthesized the knowledge from the studies found and analyzed the results in the context of our research questions.

We have proposed and designed a system that uses a genetic algorithm to create playable decks from a given card pool that simulates the constrictions under which the sealed format is played.

We have implemented our proposed solution, using the DEAP and SCOOP frameworks for creating a distributed genetic algorithm that uses simulated matches played with the open source MTG AI Forge to evaluate the fitness of the generated individuals.

In order to test our proposed and implemented design, two experiments have been carried out, and the results analyzed. These have been presented, along with a discussion of the project as a whole.

7.2 Further work

As often is the case with time constrained projects, there is always more that could have been done. We here present some of the points that could have been expanded upon.

More testing

As stated in Chapter 5, we were not able to perform all the tests and experiments that we wanted. Given more time and resources, we would have performed more tests, in order to identify which combinations of parameters that yielded the best results, as well as better gauging the quality of the decks produced by our implementation.

Different genetic operators

In our solution we have implemented a simple one point mutation operator, and a linear crossover operator. It would be prudent to explore if and how other types of operators, such as one or two point crossover functions and different mutation types, would influence the quality of the output from the algorithm.

The effect and importance of crossover in genetic algorithms is also a subject of research (Park and Carter, 1995; Meinl and Berthold, 2009). An implementation fully reliant on only the explorative nature of mutation could easily be implemented and tested to determine the importance of crossover in our solution.

Adaptive parameters

The various parameters that influence the implemented algorithm are all static, in that they do not change during the generational process. Genetic algorithms with adaptive parameters are shown to outperform traditional algorithms (Srinivas and Patnaik, 1994; García-Martínez et al., 2012). Implementing various strategies for self adaptive parameters to explore how they could affect the results would be a logical next step.

Pareto optimization with a multi-objective fitness function

As of now, the fitness score is aggregated from the win statistics against different opponents. With the goal of producing decks that perform well against a diverse selection of opponents, it is not hard to see that this approach can be flawed. As an exaggerated example one can imagine a scenario where a candidate deck defeats all opponents consistently apart from one archetype, which it loses to every time. The overall win rate is still high, but this would not be considered a well rounded deck.

An alternative to our approach would be for the fitness function to return the individual win statistics for each opponent, and use a different selection algorithm. DEAP features the NSGA-II algorithm from (Deb et al., 2002) which attempts to create pareto optimal solutions with genetic evolution. Pareto optimal solutions are optimal in the sense that there are no other solution that are stronger when all objectives are considered (Zitzler and Thiele, 1999). This would help optimizing the decks more evenly for all opponents.

Another approach could be to optimize for both win percentage against an AI opponent as well as different factors, such as the mana curve and color distribution. As we have seen in our experiments, balancing these factors seems to be one of the problems that limit the strength of the generated decks.

Different AI engines in the fitness evaluation

Using an AI to evaluate the fitness of the individuals comes with a limitation, as the algorithm will unavoidably optimize for the different strengths and weaknesses of the chosen AI. Depending on the chosen AI, this may or may not accurately reflect how humans play the game. To mitigate this, one could include multiple AIs in the fitness calculation, and either calculate the fitness as an aggregate of this, or implement an multi-objective fitness rating optimizing for all AIs.

Population diversity measure

Currently, we do not measure the diversity in our populations. As discussed, it is a real possibility that the population quicly develops into a monoculture and that the search space is not thoroughly explored (Laumanns et al., 2002). If we could detect this, countermeasures could be implemented in order to improve the generated results.

Present multiple decks as solutions

Our solution only outputs one deck for every run. The solution could potentially present multiple of the individuals from the last generation as output. It would be possible to write a function to recognize what colors each individual is. This could be used to present the strongest deck from each color combination created during the run. Presenting these would help to create a more diverse results. It would also give a higher chance at giving the player a deck he likes to play with, since he could choose a deck that better fits his style.

Different card sets

The mechanics in Magic: The Gathering vary with different sets. As stated, we have limited our experiments to card pools drawn from the Kaladesh and Aether Revolt sets. In order to establish the validity of our proposed solution as a general solution for building decks in the sealed format of Magic: The Gathering, tests with different card sets should be performed.

Other tournament formats

For this work we limited our scope to the MTG format Sealed, as this would greatly limit the search space, as well as putting interesting constraints on the production of decks. However, it would indeed be interesting to see how our proposed solution would perform within a search space an order of magnitude larger. This could also lead to the discovery of previously unseen combinations of cards spanning multiple sets that might be very potent.

Generation of specific deck types

The algorithm proposed in this paper might be biased towards specific types of decks due to the available cards in the supplied card pool, or the Forge AI favoring one type of deck more than the others. Unlike the implementation described in (García-Sánchez et al., 2016)

however, it does not steer the generated decks towards a specified type. Bootstrapping the first generations with land cards of a specific color, or cards that lean towards a given deck type might yield stronger decks.

Code refactoring and dissemination

The resulting source code from our project has been built in several stages, with parts added as they were required and conceived. As a result, the code repository has a lot of potential for a systematic overhaul in order to raise the standard of the code. This should be prioritized, as we would very much like to present our code and findings to the overlap among the MTG community and those with an interest in computer science. A codebase that is easier to understand, maintain and use will probably be better received.

Other approaches for programmatic deck building

Only one technique for building decks in Magic: The Gathering has been tested in this work. Genetic algorithms have been around for a long time (Barricelli, 1957), but there are many novel and more recent approaches for combinatorial optimization that seem promising (Osaba et al., 2016; García-Martínez et al., 2012; Kubalík, 2011). It is the researchers' opinion that exploring different methods for solving this problem is interesting, both from an academic perspective, as well as from one with interest in trading card games, and that it is well worth pursuing.

Bibliography

Barricelli, N. A. (1957). Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9(35-36):143–182.

Beheshti, Z., Shamsuddin, S. M., and Yuhaniz, S. S. (2013). Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems. *Journal of Global optimization*, 57(2):549.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*, pages 1086–1087. MIT press Cambridge.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

García-Martínez, C., Lozano, M., and Rodríguez-Díaz, F. J. (2012). A simulated annealing method based on a specialised evolutionary algorithm. *Applied Soft Computing*, 12(2):573–588.

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

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, page 41. Addison-Wesley Professional.

Guinness World Records (2016). Most played trading card game. `http://www.guinnessworldrecords.com/world-records/most-played-trading-card-game/`. Accessed: 2016-09-28.

Hold-Geoffroy, Y., Gagnon, O., and Parizeau, M. (2014). Once you scoop, no need to fork. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 60. ACM.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kofod-Petersen, A. (2014). How to do a structured literature review in computer science. *Document released as a guide to performing a Structured Literature Review at NTNU. Retrieved December*.

Kubalík, J. (2011). Evolutionary-based iterative local search algorithm for the shortest common supersequence problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 315–322. ACM.

Kubalik, J. and Faigl, J. (2006). *Iterative Prototype Optimisation with Evolved Improvement Steps*, pages 154–165. Springer Berlin Heidelberg, Berlin, Heidelberg.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282.

Lillegraven, T. N. and Wolden, A. C. (2010). Design of a bayesian recommender system for tourists presenting a solution to the cold-start user problem. Master's thesis, NTNU, Institutt for datateknikk og informasjonsvitenskap.

Meinl, T. and Berthold, M. R. (2009). Crossover operators for multiobjective k-subset selection. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1809–1810. ACM.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Nahar, S., Sahni, S., and Shragowitz, E. (1986). Simulated annealing and combinatorial optimization. In *Proceedings of the 23rd ACM/IEEE design automation conference*, pages 293–299. IEEE Press.

Osaba, E., Carballedo, R., López-García, P., and Diaz, F. (2016). Comparison between golden ball meta-heuristic, evolutionary simulated annealing and tabu search for the traveling salesman problem. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1469–1470. ACM.

Park, K. and Carter, B. (1995). On the effectiveness of genetic search in combinatorial optimization. In *Proceedings of the 1995 ACM symposium on Applied computing*, pages 329–336. ACM.

Rainville, D., Gagné, C., Teytaud, O., Laurendeau, D., et al. (2009). Optimizing low-discrepancy sequences with an evolutionary algorithm. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1491–1498. ACM.

Srinivas, M. and Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667.

Wizards of the Coast (2016). Gatherer. `http://gatherer.wizards.com/`. Gatherer is the Magic Card Database. Accessed: 2016-10-14.

Wizards of the coast (2016). Magic: The gathering tournament rules. `http://www.wizards.com/contentresources/wizards/wpn/main/documents/magic_the_gathering_tournament_rules_pdf1.pdf`. This is the rule set for tournament play. Accessed: 2016-10-26.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

Appendix A

SLR Protocol

This is the protocol for the performed systematic literature review. The protocol was used as a guidance document during the SLR phase of the thesis work to ensure that the review was conducted in a structured and reproducible way. It is based on the guidelines proposed in ”Guidelines for performing Systematic Literature Reviews in Software Engineering” (Kitchenham and Charters, 2007). We have continuously reviewed this document during the process, and it has such been subject to change.

Background

As a part of our master thesis at NTNU (Norwegian University of Science and Technology) we are to conduct a Systematic Literature Review. This systematic literature review is being performed in the period of autumn 2016 and spring 2017. A Systematic Literature Review (SLR) is a method to gather and process available research on a given topic or set of research questions. This includes identifying, evaluating and collecting primary studies, assessing the quality of these studies and extracting and synthesizing relevant data.

The topic for our master's thesis is deckbuilding in the Magic: The Gathering sealed format. We want to create a system that uses AI to build these decks. The main objective for this SLR is to learn more about the topic of Artificial Intelligence methods in card games like Magic: The Gathering. The reason we use an SLR is because our results and findings need to be a of a high quality, and because our method needs to be reproducible.

Magic: The Gathering is a card game where you build decks, and use them to battle against opponents. There are several different card types, and they all have different roles in a deck. You need land cards to produce mana in order to cast spells. These spells can do different things, like summoning a creature, or lay down enchantments. In order to have a good deck you need a good mixture of all of these card types.

The main challenges with building decks is choosing cards that compliment each other,

balancing early and late game cards, building a mana base that supports the chosen cards well and having the overall deck fit into a specialized playstyle.

Research Questions

RQ1 What is state of the art on deck-building systems in Magic: The Gathering?

RQ1.1 If no current system exists, how can such a system be created?

RQ1.2 If such systems exists, in what way is it possible to improve them?

RQ2 What is the strength of the evidence in support of the different solutions?

RQ3 How can our findings be applied when creating a system for programmatic deck-building?

Search Strategy

During the search process, we will go through multiple digital libraries in order to attain the knowledge we seek. We will do this in a structured and controlled manner. Based on the recommendations in the supervisor's paper (Kofod-Petersen, 2014) on conducting SLRs in computer science, we have selected the following databases and search engines.

Table A.1: List of search engines and databases.

Name of database /search engine	Assignee
ACM digital library	Knut
IEEE Xplore	Knut
ISI web of knowledge	Knut
ScienceDirect	Knut
CiteSeerX	Sverre
SpringerLink	Sverre
Wiley Inter Science	Sverre
Oria	Sverre
Google Scholar	Sverre

To shorten the list of papers needed to process, we will follow these simple guidelines when adding articles to the list of potential studies:

- If we encounter multiple releases of the same article, we will keep the newest one, on the assumption that it is the most up to date.
- In cases where an author has written multiple articles on the same subject, we will use the most comprehensive one.

Table A.2 lists the terms we will use when searching for articles. They are grouped up to ensure high prescision and recall. In other words, to ensure that we minimize irrelevant results, while hopefully getting all the relevant articles. This is the revised search matrix. This is elaborated further on in section 3.2.3 in the thesis.

Table A.2: Matrix with search terms used during the search step.

Group 1	Combinatorial optimization	COP	
Group 2	Optimal subset	Subset selection	
Group 3	Artificial intelligence	AI	Machine Learning
Group 4	Evolutionary algortithm	Neural network	Simulated annealing

In our search phase we will use the following search string, made by combining the terms in our search matrix using boolean operators.

("Combinatorial optimization" OR "COP") AND ("Optimal subset" OR "subset selection") AND ("artificial intelligence" OR "AI" OR "Machine Learning") AND ("evolutionary algorithm" OR "neural network" OR "Simulated annealing")

If this should not return any usable results we will exclude one of the groups of terms and repeat the search, resulting in four different searches. This is illustrated in figure A.1, where green represents the results of the full search string above and blue represents the result of any search string where you exclude one group of terms.

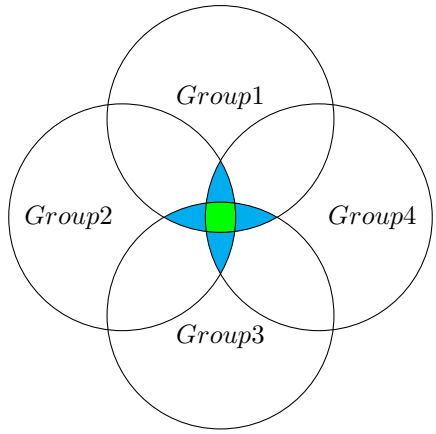

Figure A.1: Venn diagram showing the targeted studies. Green is the primary target, blue is only targeted if needed.

Study Selection Process

We have chosen two different sets of criteria for us to use when reading and evaluating the collected studies. The first set is to ensure that the studies are of interest and relevance. These are called the inclusion criteria. The second set is the quality criteria, which is designed to ensure a high academic quality in the studies. A study can be rated either one point, half a point or zero points on the criteria.

IC1 The study's main concern is Combinatorial Optimization Problems.

IC2 The study is a primary study presenting empirical results.

IC1 and IC2 are the most important inclusion criteria, and will therefor need to be at least partly fulfilled. IC3 and IC4 are less important, and one of these can be unfulfilled.

IC3 The study focuses on finding an optimal subset of a given set.

IC4 The study proposes a general solution.

In addition a full score on the two first quality criteria are mandatory for inclusion.

QC1 Is there is a clear statement of the aim of the research?

QC2 Is the study put into context of other studies and research?

Quality Assessment

To assess the quality of the research related to our problem we will score each collected study on ten different questions. This will be done by both researchers in unison after thoroughly reading all articles.

QC1 Is there is a clear statement of the aim of the research?

QC2 Is the study put into context of other studies and research?

QC3 Are system or algorithmic design decisions justified?

QC4 Is the test data set reproducible?

QC5 Is the study algorithm reproducible?

QC6 Is the experimental procedure thoroughly explained and reproducible?

QC7 Is it clearly stated in the study which other algorithms the study's algorithm(s) have been compared with?

QC8 Are the performance metrics used in the study explained and justified?

QC9 Are the test results thoroughly analysed?

QC10 Does the test evidence support the findings presented?

Data extraction

During the data extraction, we will collect this information from each paper. The extracted data will be stored in a spreadsheet document for later usage.

- Author
- Title
- Year of publication
- Type of article
- Aims
- Type of COP
- Algorithm type
- Experimental design
- Performance metric used
- Conclusion

Data Analysis

After the data collection step we will analyze our findings with respect to our research questions. The basis for this discussion will be the collected data and the quality assessment. This process will be undertaken by both researchers together.

Dissemination

The results from this review will be used in our master's thesis, where we will propose and implement a solution for building decks in Magic: The Gathering. A chapter in our thesis will detail the process of the literature review, our findings, our analysis and a summary.

Appendix B

Opponent Decks

These were the decks used in the fitness evaluation.

B.1 GB

1x Aid from the Cowl
1x Caught in the Brights
1x Conviction
1x Cowl Prowler
1x Daring Demolition
1x Dhund Operative
8x Forest
1x Fretwork Colony
1x Gifted Aetherborn
1x Irontread Crusher
1x Lawless Broker
1x Lifecraft Cavalry
1x Maulfist Revolutionary
1x Narnam Renegade
1x Night Market Aeronaut
1x Peema Outrider
2x Plains
1x Prophetic Prism
1x Renegade Rallier
1x Revoke Privileges
1x Rishkar's Expertise
1x Scrapheap Scrounger
7x Swamp
1x Thopter Arrest
2x Unbridled Growth

B.2 UWg

1x Aether Inspector
1x Aether Swooper
2x Aethertide Whale
1x Airdrop Aeronauts
1x Bastion Mastodon
1x Caught in the Brights
1x Chief of the Foundry
1x Consulate Skygate
1x Dawnfeather Eagle
3x Forest
1x Fumigate
1x Hinterland Drake
7x Island
1x Leave in the Dust
1x Master Trinketeer
7x Plains

1x Renegade Map
1x Restoration Specialist
1x Revoke Privileges
1x Rishkar, Peema Renegade
1x Rogue Refiner
1x Thopter Arrest
1x Treasure Keeper
1x Whirlermaker
1x Wind-Kin Raiders

B.3 RG

1x Attune with Aether
1x Blossoming Defense
1x Brazen Scourge
1x Chandra's Pyrohelix
1x Cultivator of Blades
1x Elegant Edgecrafters
1x Fairgrounds Trumpeter
1x Fateful Showdown
1x Fleetwheel Cruiser
8x Forest
1x Highspire Artisan
8x Mountain
1x Oviya Pashiri, Sage Lifecrafter
1x Renegade Freighter
1x Shock
1x Sky Skiff
2x Spireside Infiltrator
1x Thriving Rhino
2x Voltaic Brawler
1x Wayward Giant
2x Welding Sparks
1x Whirlermaker
1x Wild Wanderer

B.4 GBw

1x Appetite for the Unnatural
1x Battle at the Bridge
1x Caught in the Brights
1x Countless Gears Renegade
1x Daring Demolition
1x Druid of the Cowl
1x Elegant Edgecrafters
1x Fatal Push
7x Forest
1x Gifted Aetherborn
1x Lifecraft Cavalry
1x Maulfist Squad
1x Nissa, Vital Force
1x Pacification Array
4x Plains
1x Prakhata Pillar-Bug
1x Prophetic Prism
1x Revoke Privileges
1x Scrounging Bandar
1x Silkweaver Elite
6x Swamp
1x Thopter Arrest
1x Thriving Rhino
1x Tidy Conclusion
1x Unbridled Growth
1x Wispweaver Angel

B.5 UR

2x Aether Meltdown
1x Aethersquall Ancient
2x Bastion Mastodon
2x Chandra's Pyrohelix
1x Gearseeker Serpent
2x Glassblower's Puzzleknot
1x Glimmer of Genius
1x Hightide Hermit

1x Insidious Will
10x Island
1x Long-Finned Skywhale
1x Malfunction
2x Minister of Inquiries
7x Mountain
1x Padeem, Consul of Innovation
1x Skyship Stalker
1x Tezzeret's Ambition
1x Thriving Turtle
1x Weldfast Monitor
1x Welding Sparks

B.6 RW

1x Bomat Bazaar Barge
2x Brazen Scourge
1x Chief of the Foundry
1x Combustible Gearhulk
1x Consul's Shieldguard
1x Fleetwheel Cruiser
1x Fragmentize
3x Gearshift Ace
1x Glint-Sleeve Artisan
1x Inventor's Apprentice
9x Mountain
8x Plains
1x Propeller Pioneer
1x Renegade Freighter
1x Revoke Privileges
1x Scrapheap Scrounger
1x Sky Skiff
1x Skyship Stalker
1x Thriving Grubs
1x Veteran Motorist
1x Visionary Augmenter
1x Weldfast Monitor

Appendix C

Experiment Sealed Pools

C.1 Experiment 1

1x Accomplished Automaton
1x Aether Poisoner
1x Aether Swooper
1x Aether Theorist
1x Aethersphere Harvester
1x Aethertorch Renegade
1x Alley Evasion
2x Audacious Infiltrator
1x Augmenting Automaton
1x Bastion Inventor
1x Caught in the Brights
2x Chandra's Revolution
1x Chandra, Torch of Defiance
1x Commencement of Festivities
1x Consulate Turret
1x Countless Gears Renegade
1x Cruel Finality
1x Curio Vendor
1x Daredevil Dragster
1x Daring Demolition
1x Dawnfeather Eagle
1x Deft Dismissal
2x Dispersal Technician
1x Druid of the Cowl
1x Efficient Construction
1x Elegant Edgecrafters
1x Embraal Gear-Smasher
1x Enraged Giant
1x Failed Inspection
2x Fen Hauler
2x Filigree Crawler
1x Filigree Familiar
1x Fortuitous Find
1x Foundry Assembler
1x Freejam Regent
1x Glint-Sleeve Artisan
1x Gonti's Machinations
1x Hidden Stockpile
1x Highspire Artisan
1x Hijack
1x Hinterland Drake
1x Implement of Combustion
2x Implement of Examination
1x Indomitable Creativity
1x Invigorated Rampage
1x Kari Zev's Expertise
1x Kari Zev, Skyship Raider
1x Lost Legacy
1x Maulfist Squad
2x Negate
1x Night Market Lookout
1x Ovalchase Daredevil

1x Reckless Fireweaver
1x Renegade Freighter
1x Renegade Rallier
2x Renegade's Getaway
1x Reservoir Walker
1x Resourceful Return
1x Revoke Privileges
1x Scrapper Champion
1x Sequestered Stash
1x Shock
1x Silkweaver Elite
1x Take into Custody
1x Thriving Rats
1x Thriving Rhino
1x Unlicensed Disintegration
1x Vedalken Blademaster
1x Vengeful Rebel
2x Verdant Automaton
1x Weldfast Wingsmith
1x Wily Bandar
2x Wind-Kin Raiders
1x Workshop Assistant
1x Wrangle

C.2 Experiment 2

1x Aerial Modification
1x Aether Poisoner
1x Aether Theorist
1x AEther Tradewinds
1x Aethertorch Renegade
2x Alley Strangler
2x Audacious Infiltrator
1x Aviary Mechanic
1x Barricade Breaker
1x Bomat Bazaar Barge
1x Call for Unity
2x Caught in the Brights
1x Chandra's Revolution
1x Cogwork Assembler
1x Consulate Crackdown
1x Consulate Turret
1x Countless Gears Renegade
1x Daring Demolition
2x Dawnfeather Eagle
1x Defiant Salvager
1x Dispersal Technician
1x Dukhara Scavenger
1x Eddytrail Hawk
1x Electrostatic Pummeler
1x Embraal Gear-Smasher
1x Engineered Might
1x Fen Hauler
1x Filigree Familiar
1x Fireforger's Puzzleknot
1x Foundry Screecher
1x Fourth Bridge Prowler
1x Frontline Rebel
1x Ghirapur Osprey
1x Herald of Anguish
1x Highspire Artisan
1x Hinterland Drake
1x Hungry Flames
1x Impeccable Timing
1x Implement of Ferocity
1x Implement of Malice
1x Inventor's Goggles
1x Invigorated Rampage
2x Irontread Crusher
1x Kujar Seedsculptor
1x Live Fast
1x Make Obsolete
1x Mobile Garrison
1x Narnam Renegade
3x Negate
1x Night Market Aeronaut
1x Nimble Innovator
1x Ornamental Courage
1x Peema Aether-Seer
1x Planar Bridge
1x Prey Upon
1x Prophetic Prism
1x Renegade Map
1x Revolutionary Rebuff

1x Rishkar's Expertise
1x Saheeli's Artistry
1x Salivating Gremlins
1x Salvage Scuttler
1x Shipwreck Moray
2x Silkweaver Elite
1x Skywhaler's Shot
1x Spire Patrol
1x Take into Custody
1x Unbridled Growth
1x Universal Solvent
1x Untethered Express
1x Vedalken Blademaster
1x Watchful Automaton
1x Welder Automaton
1x Weldfast Monitor
1x Welding Sparks
1x Wind-Kin Raiders
1x Winding Constrictor

Appendix D

Expert decks

These are the decks created by the expert from the two sealed pools used for Experiment 1 and Experiment 2.

D.1 Experiment 1

1x Accomplished Automaton
1x Aether Poisoner
1x Aethersphere Harvester
1x Aethertorch Renegade
1x Augmenting Automaton
2x Chandra's Revolution
1x Chandra, Torch of Defiance
1x Daring Demolition
1x Enraged Giant
1x Filigree Familiar
1x Freejam Regent
1x Implement of Combustion
1x Kari Zev, Skyship Raider
1x Maulfist Squad
9x Mountain
1x Ovalchase Daredevil
1x Reckless Fireweaver
1x Renegade Freighter
1x Scrapper Champion
1x Shock
8x Swamp
1x Thriving Rats
1x Unlicensed Disintegration
1x Vengeful Rebel

D.2 Experiment 2

1x Aether Poisoner
1x Aether Theorist
1x AEther Tradewinds
1x Barricade Breaker
1x Bomat Bazaar Barge
1x Cogwork Assembler
1x Daring Demolition
1x Fen Hauler
1x Filigree Familiar
1x Herald of Anguish
1x Hinterland Drake
1x Implement of Malice

1x Inventor's Goggles
8x Island
1x Live Fast
1x Make Obsolete
1x Negate
1x Nimble Innovator
1x Prophetic Prism
1x Renegade Map
1x Saheeli's Artistry
8x Swamp
1x Universal Solvent
1x Untethered Express
1x Weldfast Monitor
1x Wind-Kin Raiders