Dynamic Deck-dependent Strategy Adjustment for Collectible Card Game Arena Mode

(Dynamiczne dostosowywanie strategii wyboru talii dla trybu areny w kolekcjonerskich grach karcianych)

Dawid Barzyk

Praca licencjacka

Promotor: dr Jakub Kowalski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

11.09.2020 r.

1:29184

Abstract

Nowadays, collectible card games (CCGs) are one of the most popular genres of esports games. Due to that, players very quickly developed so called *metagaming*[1] for these games. In my thesis I focus on examining how important it is to consider metagaming during creation of agents. In order to do this, I developed agent with different deck-depending strategies for Legends of Codes and Magic (LoCM). Created for AI research it is a simplified strategic card game based on popular CCGs (like The Elder Scrolls: Legends and Hearthstone).

I examine the idea of determining draft choices and further strategy on cards that are in the already drafted deck. Agents following these strategies are compared against best agent presented at Strategic Card Game AI competitions hosted by IEEE CEC 2020 and IEEE COG 2019.

Although tests of this approach did not result in a strictly better solutions, the approach is promising and worth further research.

Przez ostatnie lata kolekcjonerskie gry karciane stały się jednym z najbardziej popularnych gatunków gier esportowych. Dzięki owemu zjawisku wyjątkowo szybko rozwinął się wśród graczy tak zwany *metagaming*[2]. W swojej pracy skupiam się na zbadaniu jak ważne jest rozważenie metagaming'u przy projektowaniu agentów do owych gier. W tym celu stworzyłem kilku agentów przyjmujących różne strategie do gry Legends of Code and Magic (LoCM). Jest to przeznaczona do badań sztucznej inteligencji, uproszczona strategiczna gra karciana wzorowana na popularnych tytułach takich jak The Elder Scrolls: Legends oraz Hearthstone.

W pracy rozważam uzależnianie doboru nowych kart oraz strategii w dal-

[1] Relations between the game mechanics, and a general opinion about strategies which are better than the others.

[2] Wiedza o istniejących relacjach w mechanice gry oraz o zależnościach między strategiami przyjmowanymi przez graczy.

3:62681

szej części rozgrywki od kart znajdujących się w talii. Analiza opiera się na porównaniu stworzonych agentów w zestawieniu z istniejącymi już agentami, zaprezentowanymi na turniejach organizowanych w ramach konferencji IEEE CEC 2020 oraz IEEE COG 2019.

Pomimo, że wyniki uzyskane podczas tego doświadczenia nie przynoszą dostatecznie zadowalających rezultatów, uważam, że jest to zagadnienie warte dalszych badań.

4:11365

Table of contents

5:23009

6:59643

Chapter 1

Introduction

Nowadays, as collectible card games (CCGs) become more and more popular, people rapidly develop their skills and adjust to mechanics changes, making it hard for AI researchers to develop agents that could compete with the best players. One of the players' basic approaches to CCGs is adjusting strategy of playing to the characteristics of his deck. I have decided to research how this approach could influence the effectiveness of agents by developing an agent for Legends of Code and Magic [1]. I will refer to this game as LoCM. It was created by Jakub Kowalski and Radosław Miernik with AI competition in mind. Up to the day of writing this thesis, there were held four contests [2, 3, 4, 5] in version 1.2 of LoCM.

1.1 Collectible card games definition

A collectible card game is a board game for two or more players in which all players use personal decks of cards. Those cards are later used to perform certain actions (depending on game mechanics and card description) that affect the state of the game. Magic: The Gathering is said to be the forerunner of CCGs since it gained a lot of popularity in the 90s and brought the attention of many players to CCGs. Since then many digital versions of CCGs were created and due to the huge popularity of this genre best players' knowledge and abilities are hard to reach for AI.

7:10906

Even though almost every CCG has its own set of rules and mechanics, most of them have a common core. Matches are mostly played in turns, during which players draw new cards from their decks, place their cards in hand on the board or perform actions using cards that already are on the board.

A very important part of CCGs is the variety of cards provided to players. For example, Magic: The Gathering currently consists of more than 20000 unique cards, to which hundreds are added each year, Hearthstone has almost 3500 cards, and similarly to MTG this number also keeps growing. Such big amounts of cards provide a great variety of possibilities for creating new decks. Due to the amount of information that each player would need to process, most of them decide to play already created and tested decks built by other players. Since some decks counter each other, people tend to play different but closed variety of decks. Such a situation is called metagaming and will be discussed later.

1.2 Problem description

LoCM is inspired by the arena modes of other CCGs. It consists of two phases:

1. Draft. Before the beginning of this phase engine randomly selects 60 cards from the pool of 160 cards. Later, during each out of 30 turns, both players are introduced with 3 cards out of 60 selected from which they are supposed to pick one for their deck. Both players have the same choice but no knowledge about the opponent's pick.
2. Battle. During this phase, both players aim to reduce the opponent's health bar to 0 or below using cards drafted earlier with mechanics described below.

There are two types of cards each player can play:

1. Creatures that can be summoned by a player to one of two lanes on board. Summoned minions can attack the opponent and his monsters placed on the same lane. Each of them has certain statistics (attack, defense, and cost) and some of them have special abilities, such as:
    - Breakthrough - applies excessive damage dealt to opponent.
    - Charge - allows creature attack in the turn it was summoned.
    - Drain - recovers health of the owner by amount of damage dealt.

8:63221

- Guard - creature with this ability has to be attacked by opponent's creatures on the same lane in first place.
- Lethal - kills every creature that was dealt damage to.
- Ward - protects creature from any damage. As soon as creature with ward is attacked or counterattacked it loses its ward.
- Player/Enemy health change - when placed on board, changes player's/enemy's health points by amount in the description.
- Cards to draw - when placed on board, raises the number of cards drawn by the player at the beginning of the following turn.

2. Spells that are split into three categories:

   - Green - may be used on friendly creatures to change their defense, attack, or give additional abilities.
   - Red - may be used on hostile minions, to kill them, reduce their statistics, or remove abilities.
   - Blue - recover player's health, reduce opponent's health, or deal damage to opponent or his creatures.

Each player has to perform all the actions in limited time which is $1s$ for first round of each phase and $200ms$ for any other round.

More detailed rules description is obtainable on LoCM project page [6]

1.3 Metagaming

Metagaming is generally speaking the knowledge of the current mechanics of the game and how to use them for own advantage. Throughout the years CCGs' players managed to develop great capability to adjust own strategies to changes in game mechanics using their knowledge about existing mechanics and analyzing card synergies. Most of the best players managed to develop special skill in creation of decks adaptive to changes. This skill is very practical in most arena game modes since the player has to choose one out of few cards that suits their deck the most. Because of great popularity of CCGs, players created a lot of guides containing all the knowledge players should have when approaching arena mode for special game. Although most of these games have various game mechanics, people tend to specify few general ideas of deck constructing. Main ideas of those tactics are:

1. Aggro decks are constructed for aggressive strategy. Their main aim is to deal as much damage to opponent as fast as possible while flooding

9:10234

board with minions to make it hard for opponent to control the board or clear player's minions. It is mostly made of cards that deal as much damage as possible and are not hard to kill.

2. Control decks are made of strong, hard to kill creatures along with board clearing cards. Player having this deck should bo more focused on controlling the board rather than dealing damage to the opponent.

3. Mid-range decks. Player with this deck should gain control over the board during the first phase of battle and strike opponent with best and most aggressive in-game creatures in the other half. Here, we can split deck in half where first one would contain creatures that are more directed to killing hostile minions (cards with lethal or high attack-defense ratio) and the other ones that are strong, hard to kill and very effective at dealing damage to the opponent.

Although most of CCGs have cards profiled exactly for specific type of gameplay, in arena it is hard to create a perfect deck for specific strategy (mostly it is impossible due to the randomness during the draft phase). I focused on creating agents that mix general approach along with playing according to strategy their deck was assigned to.

10:10050

Chapter 2

Analysis of existing solutions

In an attempt to prepare for creation of agents, I analyzed solutions existing in other games along with bots that were created for LoCM competitions mentioned earlier.

2.1 Hearthstone

Hearthstone is currently leading in popularity among other CCGs. Although it is one of the fastest developing digital CCGs, its AI leaves much to be desired if it comes to its efficiency. Therefore there is a high demand for the agents to this game.

However, due to the great amount of cards, deck creation is really hard and most agents are based on assumption that there exists certain pool of decks that players tend to choose from (so called meta decks). Due to that and the fact that arena in Hearthstone is pay to play [7], there are very little agents to play the arena mode.

Nevertheless, there are some AIs developed especially for building deck to play arena draft phase. While most of the solutions I found focus on evolutionary algorithms [8, 9], the most popular among players is HearthArena Companion [10].

HearthArena Companion uses online database (created by collecting data from every game ever played using this engine) to calculate main value of the card given its win-ratio statistics. Having that, it evaluates card's value based on its cost, current deck mana curve and its prototype along with synergies with other cards. This approach is very similar to approach tested in this

11:11502

thesis.

In case of agent algorithms, best of them are based on MCTS combined with Supervised Learning Algorithms [11].

2.2 Legends of Code and Magic

Since the creation of LoCM in version 1.2, there were organized four contests, three [2, 3, 4] of which were won by agent Coac and the last one [5] was won by Chad.

The base algorithm behind Coac (whose name probably refers to continous orthogonal ant colony [12]) is min-max search of depth 3. During each turn of battle phase agent picks best action based on simulation up to 3 own moves and then evaluating opponent state by picking (locally) best option. This is repeated for as long as possible without time-outing until there are no valid moves. Draft phase is determined by four hard-coded tables with hierarchy of cards. Choice of table that should be used for evaluation depends on the current amount of monsters in deck and whether the agent starts first or second in battle phase.

This agent especially brought my attention because its construction allows me to outright test choice of strategy influence on the quality of the algorithm in comparison with direct approach.

Chad is based on an MCTS algorithm that predicts opponent's cards based on given choices during drafting phase. Draft phase here is also hard-coded, based on pre-made calculations for different agents. For more about this research please refer to thesis of the agent creators [13].

Other totally different approach to this game is represented by ReinforcedGreediness, which uses three neural networks: one for draft, one for battle phase when agent starts as first player, and one for battle phase when player plays second.

2.3 Other games

There are two other CCGs I made a research about. The first one is The Elder Scrolls: Legends, but even though it is third biggest CCG (if it comes to development and popularity), sadly I did not manage to find any relevant contests or documents about AI for this game.

12:11558

The second game worth mentioning is Magic: The Gathering. Even though it is not digital (originally), it is considered one of the hardest games in the world. Last year it was even proved to be Turing Complete [14]. Because of that, creating agents for this is extremely hard since it is impossible to obtain using standard methods. The most common approach is by using neural networks trained on data gathered from sites designed for players to practice [15]. In Zachary Witten's article [16] about AI for drafting Magic: the Gathering, author says that his agents generally picks better cards than human players, although its effectiveness depends on the training dataset.

13:73040

Chapter 3

My solution

At the beginning, I developed basic input/output management in order to be able to test any changes on the code. Then I started implementing a basic game engine that would let me simulate player's and opponent's moves. Having all the important abstraction done and tested, I moved to implementing drafting methods.

3.1 Drafting algorithms

Drafting consists of four evaluations: hard-coded, aggro, mid-range, and control. In order to be able to compare each rating with others, I had to come up with fair evaluation method that in each case will give a reliable score (to avoid favoritism of any strategy).

3.1.1 Hard-coded draft

Since the number of cards in game is rather low (160), I decided to hard-code value of each one accordingly to their evaluation in other algorithms, my own experience, HearthArena Companion and later tests of algorithm effectiveness.

15:72246

3.1.2 Aggro

During aggro evaluation, I decided to reward *straight damage* that could be dealt directly at the opponent. Also, in order to counter some of the enemy's plays, I decided to reward strong creature killers and silencers, such as minions with lethal or items like Staff of Suppression, Decimate or Mighty Throwing Axe [17]. Our desired mana mean in deck is 3.0 because our plan is to have as much cheap creatures as possible, and kill opponent before reaching 12 mana.

3.1.3 Control

This was the hardest evaluation to program because, as the results show, the game is generally constructed to favor aggressive type of strategies. Although unsatisfactory results, the main aim of this type of deck is to obtain most value from creatures. This allows agent lose as little health as possible while still controlling the board. Hence, I decided to prioritize abilities such as guard and ward, while penalizing ability of breakthrough (unneeded for this strategy yet still increasing the price of a card). Also, a card is granted more value for higher statistics such as attack and defense since they are what makes creatures most useful in this kind of strategy. In this case, the most wanted mana mean is 5.0 since we need to have as much material as possible so that opponent runs out of cards faster than this agent.

3.1.4 Mid-range

Mid-range decks are very simple to develop since their main focus is to play control strategy in first half of the match and then switch to aggro strategy mid-game. Hence, I decided that evaluation for cards of cost up to 5 would be similar to control draft and those of cost above 5 would be similar to aggro. Their main aim is to play strategy in between aggro and control, so their best mana mean would be 4.0.

3.1.5 More on drafting

General idea of this drafting algorithm is to at first evaluate basic rating of the card (similarly to the way it is done by HearthArena Companion) and then add it to the score of each specialization evaluation summed with corresponding current deck rating. More specific, while evaluating score of card A for aggro, agent gets hard-coded score of A, adds it the value of current deck score based on aggro evaluation of cards and then adds to it value of

16:10057

aggro evaluation of card A multiplied by trust factor (number of cards picked over 20.0). Trust factor was introduced to reduce importance of a card score at the beginning of draft and increase it at the end phase of draft.

3.2 Choice of strategy

After the initial phase of draft, agent evaluates his choices and picks strategy that suits his deck most. It is done by evaluating each card in the deck from the perspective of each strategy and later (after subtracting mana compensation) picking the one with the highest score.

3.3 Battle agent

Similarly to Coac I decided to implement min-max with alpha-beta pruning and focus my attention on score of the board.

3.3.1 ProtoCoac

At the beginning of a turn, ProtoCoac searches throughout legal moves for the best one by simulating own 3 moves and opponent's (only the ones that bring him benefit). The move is then scored with a state evaluation function, thar depends on the chosen strategy. After finding a move with the highest score, the agent applies it and repeats the procedure until he runs out of time or legal moves.

17:97835

3.3.2 State evaluation

Evaluation of the state is calculated similarly for every strategy. The agent looks through all of the cards and rates them depending on chosen strategy. Strategies are scored as follows:

1. Aggro. The main aim of this strategy is to flood the board with minions. Hence, the agent must ensure that he does not run out of cards. For that reason each card to draw that does not exceed the hand maximum capacity limit, is awarded. Each card on the board is either penalized or rewarded depending on its owner. Calculation of the score depends on card type:
    - Friendly creatures earn points depending mostly on their attack since we want to deal as much damage to the opponent as possible. They get additional points for their abilities. Creatures with guard are rewarded for the amount of friendly monsters they protect.
    - Hostile creatures penalize score by their general statistics. Their crucial ability is lethal. The second most important for the opponent are minions with guard since they block the player's creatures from dealing direct damage. Hence, they change the general rating of the board by the amount of their defense.
    - Items in hand are rewarded as long as they could have been played (but were passed instead).
2. Control. With this strategy, I focus on rewarding the agent mostly for the amount of minions on the board since we want to be dominating both lanes. In this strategy, agent prioritizes his own health rather than dealing damage to the opponent because he wants the battle to last as long as possible. Agent is also rewarded for the amount of cards to draw as long as they fit in the maximum capacity limit. Again score of each card depends on its type:
    - Friendly creatures gain points according to their main statistics (both attack and defense are important). Furthermore, minion is more valuable if it has guard, drain, or lethal since these abilities are useful in clearing the board, protecting the player, and keeping control over the opponent's minions.
    - Since we prioritize protecting our health, the player should be rewarded for killing minions with the highest attack. Furthermore, to keep the strongest friendly minions alive as long as possible, the agent is additionally penalized for each opponent's minion that has lethal.

18:85478

- In this evaluation of the state, items are equally scored as in the aggro strategy.

3. Mid-range. Since this strategy is a mix of both strategies mentioned earlier, for the first seven turns the agent follows the control strategy and later it switches to the aggro strategy. Scores of the game state for both phases are evaluated accordingly to suitable tactic.

19:98723

Chapter 4

Experiments

4.1 Process of experiments

Testing was split into two parts. First, I decided to test draft methods by running agent with three different ways of picking cards.

1. Making a choice based only on deck-dependent evaluations of cards.
2. Picking cards using only hard-coded table. Determining battle strategy after the draft phase.
3. Using both hard-coded and strategy-directed evaluations.

The second part of the testing was to check how well does my agent work in *perfect* environments. In order to do that, I picked subsets of cards most suitable for each of the strategy, and then ran clashes against other bot (with strategy for my agent explicitly set to corresponding card subset).

In order to test the efficiency of my solution, I decided to confront my agent with Coac since it relies on the same algorithm with different approach. All the results were based on 400 games per test with random seeds.

21:60847

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	30.5%	69.5%	Second
Second	15.0%	85.0%	First
Total	20.25%	**79.75%**	

Tabela 4.1: Hard-coded draft is off.

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	56.0%	44.0%	Second
Second	23.5%	76.5%	First
Total	39.75%	**60.25%**	

Tabela 4.2: Hard-coded draft only.

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	61.0%	39.0%	Second
Second	34.5%	65.5%	First
Total	47.75%	**52.25%**	

Tabela 4.3: All evaluations on.

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	66.5%	33.5%	Second
Second	44.5%	55.5%	First
Total	**56.5%**	43.5%	

Tabela 4.4: *Perfect* aggro draft.

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	52.0%	48.0%	Second
Second	13.0%	87.0%	First
Total	27.5%	**72.5%**	

Tabela 4.5: *Perfect* control draft.

ProtoCoac as:	ProtoCoac wins	Coac wins	Coac as:
First	76.5%	24.5%	Second
Second	40.0%	60.0%	First
Total	**58.25%**	41.75%	

Tabela 4.6: *Perfect* mid-range draft.

22:27502

4.2 Data analysis

4.2.1 Standard draft test

From Table 4.1, Table 4.2, Table 4.3, I can deduce that draft phase has big impact on further flow of the game. As expected, draft based only on strategy evaluation brings the worst results. It can be explained by the fact that it was not made to calculate the rating of the card but its suitability for the strategy. The second thing that caught my attention is that there is noticeable improvement between hard-coded draft and draft dynamically adaptive to the current deck.

Table 4.2 contains results of clash between agents with very similar drafting methods since my hard-coded draft is partially based on Coac's drafting table. Hence, I can conclude that my implementation of search algorithm is slightly worse than Coac's. It is very likely that improving performance of the implementation of the search algorithm could strongly affect results to ProtoCoac's advantage.

4.2.2 Perfect environment test

In Table 4.4, Table 4.5, Table 4.6 there is a noticeable anomaly for *perfect* control draft. It might be explained in many ways but the results bring me to the conclusion that LoCM is more oriented towards an aggressive style of play. This statement might be supported by the fact that there is a huge difference between wins as the first player and wins as the second player. The other reason for the failure of the control agent might be the fact that it is really hard to control the board on two lanes without any cards for clearing board from enemy minions. Furthermore, guard ability is very weak in comparison to other games since it protects the player from only half of the dangers on the enemy side of the board. The other reason for the failure of this experiment is that it is hard to define the perfect draft. In case of these tests, I defined cards that are most suitable for the specific style of play. Although, these strategies are created so that they counter other strategies. In a perfect draft, the agent would have to choose from cards perfectly fitting his deck and those that might be countered by his strategy, given fair arena scenario.

23:17745

4.3 Conclusions

With the analysis above, I concluded that the dynamic deck-dependent draft strategy might significantly improve efficiency of agent that adjusts his game plan to the drafted deck. Although my solution did not bring satisfying results, the outcome of the tests shows that this approach is worth further research. Moreover, I think that a deeper analysis of LoCM metagaming could give successful results. Possibly, analyzing and improving the performance of the control strategy, or even cutting it out of the agent, could positively affect the agent's efficiency.

Appendix A

User manual

ProtoCoac battle agent is meant to be run by the LoCM referee [18] with another agent as an opponent. It takes input as described in LoCM rules. [6]

To build the agent, go to ProtoCoac directory and run command: **cmake . && make** (cmake at version 3.10 and GCC supporting c++17 required).

Agent might be run with few different arguments:

- **"./ProtoCoac"** - runs agent with dynamic deck-dependent strategy of drafting and playing cards.
- **"./ProtoCoac aggro"** - runs agent with aggressive strategy.
- **"./ProtoCoac control"** - runs agent with control strategy.
- **"./ProtoCoac mid-range"** - runs agent with mid-range strategy.
- **"./ProtoCoac hardcode"** - runs agent with draft strategy based only on hard-coded draft.
- **"./ProtoCoac hardcode_off"** - runs agent with turned off hard-coded score evaluation during draft phase.

25:35020

26:35628

Appendix B

Cardlists and code structure

B.1 Cardlists

In folder ProtoCoac there is folder named *cardlists* which contains cardlists used during tests mentioned in second part of testing. The folders are made as follows:

- *cardlistAggro.txt* contains cards used for perfect draft for aggressive strategy.
- *cardlistControl.txt* contains cards used for perfect draft for control strategy.
- *cardlistMidrange.txt* contains cards used for perfect draft for mid-range strategy.
- *cardlistOrig.txt* contains all cards in the game engine

In order to run tests on specified cardlist the file should be renamed to *cardlist.txt* and replaced in LoCM referee .jar file.

27:45085

B.2 Project structure

Program is split into 7 files containing:

- Action.h - definition of Action structure and functions for printing.
- Card.h - definition of Card structure.
- Constants.h - definition of basic constants that are bounding in-game agent.
- Player.h - contains Player structure.
- State.h - inludes all of the above and creates State structure that simulates the state of the game and evaluates state of the board.
- DeckBuilder.h - contains DeckBuilder structure that is used for picking cards during drafting phase and choosing an appropriate type of strategy on the first round of the battle phase.
- Agent.h - defines basic interface of battle agent and implements logic behind action-picking algorithm.

Bibliografia

[1] Radosław Miernik Jakub Kowalski. *Legends of Code and Magic*. `https://legendsofcodeandmagic.com`, 2020.

[2] Legends of Code and Magic CEC19. `https://legendsofcodeandmagic.com/CEC19/`.

[3] Legends of Code and Magic COG19. `https://legendsofcodeandmagic.com/COG19/`.

[4] Legends of Code and Magic CEC20. `https://legendsofcodeandmagic.com/CEC20/`.

[5] Legends of Code and Magic COG20. `https://legendsofcodeandmagic.com/COG20/`.

[6] Legends of Code and Magic rules. `https://legendsofcodeandmagic.com/rules.html`.

[7] Official Hearthstone modes site. `https://playhearthstone.com/en-us/ways-to-play/`.

[8] Jakub Kowalski and Radosław Miernik. Evolutionary approach to collectible card game arena deckbuilding using active genes. *Congress of Evolutionary Computation*, 2020.

[9] Pablo García-Sánchez, Alberto Tonda, Giovanni Squillero, Antonio Mora, and Juan J Merelo. Evolutionary deckbuilding in hearthstone. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[10] HearthArena Companion. `https://www.heartharena.com/`.

[11] Maciej Świechowski, Tomasz Tajmajer, and Andrzej Janusz. Improving hearthstone ai by combining mcts and supervised learning algorithms. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

29:56827

[12] Ant colony optimization algorithm. https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms.

[13] Marcin Witkowski Łukasz Klasiński, Wojciech Meller. *Implementation of Collectible Card Game AI with Opponent Prediction*, 2020.

[14] Alex Churchill, Stella Biderman, and Austin Herrick. Magic: The gathering is turing complete. *arXiv preprint arXiv:1904.09828*, 2019.

[15] Magic Flooey draft viewer. https://magic.flooey.org/draft/upload.

[16] Zachary Witten. *Teaching an AI to Draft Magic: the Gathering or, How To Train Your Robot To Train Your Dragon.* 2019.

[17] Complete list of cards in Legends of Code and Magic with descriptions. https://jakubkowalski.tech/Projects/LOCM/cardlist.html.

[18] Legends of Code and Magic referee. https://github.com/acatai/Strategy-Card-Game-AI-Competition/tree/master/referee-java.