Reinforcement Learning in Collectible Card Games: Preliminary Results on Legends of Code and Magic

Ronaldo Vieira, Luiz Chaimowicz
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
Email: {ronaldo.vieira, chaimo}@dcc.ufmg.br

Anderson Rocha Tavares
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
Email: artavares@inf.ufrgs.br

***Abstract*—Games have long been a popular application area for research on artificial intelligence. They provide hard and diverse challenges whose solution often apply to real-life problems. Games like Chess, Go and recently Poker can be played by computers at superhuman level, but this performance is yet to be achieved in more complex games, such as collectible card games. To tackle this problem, we propose a pure reinforcement learning approach to the card game *Legends of Code and Magic*, in contrast to search strategies, most commonly used in this domain. In this paper, we present the intended methodology, preliminary results and the next steps of the project.**

***Keywords*-reinforcement learning; deep learning; game-playing; collectible card games;**

I. Introduction

Throughout the last decade, Artificial Intelligence (AI) has been responsible for numerous technological and scientific breakthroughs, most of them powered by the now feasible deep learning methods. Despite this rapid advancement, there are still many challenges unsolved in the field, especially regarding tasks that require more abstract thinking.

Games have long been a popular application for AI research, since they feature vast state spaces and frequently are beyond NP-hard complexity. Games provide non-trivial challenges for all core areas of AI, often requiring interdisciplinary solutions [1]. The recent solutions to games such as Go [2] and Poker [3] served as milestones on the ability of computers to deal with large state spaces and imperfect information. Alongside these, collectible card games (CCGs) also present the additional complexity of having dynamic rules that are partially determined by the combination of cards in play at each moment, as well as actions with more diverse consequences. These factors make playing them comparable to a task of general game playing [4].

In this paper, we tackle a collectible card game named *Legends of Code and Magic* (LOCM). Besides the aforementioned characteristics, winning a LOCM match also requires a successful interplay of two different tasks: deck-building and playing. In comparison to other collectible card games, LOCM is simpler and provides a friendlier environment for AI research, which can be later expanded to more complex card games.

Instead of the commonly used search strategies, we propose a pure reinforcement learning approach to the game. Both deck-building and playing tasks use policy networks, where action selection is learned directly from the game state representation, and are trained simultaneously by self-play. In the process, a reimplementation of the game engine and OpenAI Gym environments for all phases of the game are also developed. The evaluation plan includes comparison with current state-of-art agents and submission to the Strategy Card Game AI Competition[1].

In Section II, we describe *Legends of Code and Magic* in more detail. Section III contains a discussion on related work, considering card games and other games. We present the intended methodology in Section IV, then discuss preliminary and expected results in Section V. Lastly, Section VI brings the conclusion and directions for future work.

II. Legends of Code and Magic

Legends of Code and Magic is a two-player digital collectible card game (CCG) that implements a subset of the rules found in the popular *Hearthstone* [5] and *The Elder Scrolls: Legends* [6]. It was designed to provide a simpler yet representative environment for AI research on CCGs, where non-determinism is found only in the orderings of cards and in the opponent's actions.

A match of LOCM has two phases: draft and battle. In the draft phase, each player constructs a deck by choosing a card between three random cards presented in each turn, for 30 turns. Although the random cards are the same for both players, their choices are secret. This format eliminates any advantage of using pre-built top-tier decks. In the battle phase, the players take turns until one of them has their health reduced from the initial amount of 30 points to zero.

Each card can represent either a creature or an item. Items are further divided into green (positive effect), red (negative effect), and blue (mixed or neutral effect) item

[1]More info on https://jakubkowalski.tech/Projects/LOCM/COG19/.

cards. Although creatures and items apply their attributes in distinct forms, all cards have the same set of attributes: attack, defense, mana cost, keyword abilities and additional abilities. Keyword abilities affect combat, while additional abilities are triggered when the card is played.

On each turn on the battle phase, the active player draws a card from their deck to their hand. Then three types of actions can be taken: (i) place a friendly creature card on the board from hand; (ii) use a friendly creature on the board to attack the opponent or an enemy creature, reducing health points according to its attack power attribute; and (iii) place an item card on the board from hand, producing an instantaneous effect on a target creature or player.

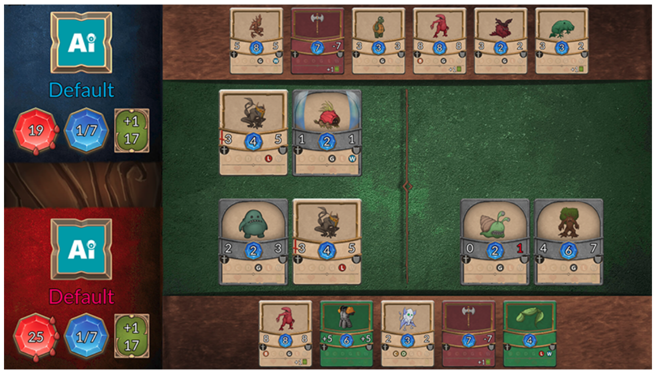

Figure 1. Game state of a turn in the battle phase in *Legends of Code and Magic*. The player in red has four creatures on the board and five cards in hand.

To be placed on the board, cards require spending mana points equivalent to their mana cost attribute. The players start with one mana point that is recharged and increased on each turn to a maximum of twelve mana points. The second player starts with an extra non-rechargeable mana point[2]. Figure 1 shows a snapshot of a *Legends of Code and Magic* match.

III. Related Work

The first classic game to be mastered by a computer was Tic-Tac-Toe, in 1952 [1]. Since then, most of the famous classic board games were tackled by AI researchers and eventually became playable at superhuman level. In 1997, IBM's Deep Blue defeated Chess grandmaster Garry Kasparov using a heavily tuned Minimax algorithm [7]. Ten years later, the game of Checkers became mathematically solved [8]. And recently, in 2017, DeepMind's AlphaGo Zero became the best Go player in history by combining deep reinforcement learning with Monte Carlo tree search (MCTS), and training via self-play [2].

However, classic board games are hardly the most complex scenarios available. Tree search methods are able to achieve superhuman performance on imperfect information games like Texas Hold'em Poker [3], as well as deep convolutional neural networks and reinforcement learning can play video games from the Atari 2600 console using raw pixels [9], but many games still remain as unsolved challenges of game-playing and AI.

Collectible card games are still unsolved and have been increasingly explored in recent years. Work on *Magic: the Gathering*, the most played CCG to date, mainly comprise design and analysis of cards [10]–[12]. However, despite determining the optimal play being non-computable [13], there are also work on playing, using MCTS [4], [14], [15] and Minimax [15] with severe tree pruning strategies to handle the huge branching factor of the game.

Deck-building is more explored on *Hearthstone*, often using evolutionary algorithms [16], [17]. Game-playing is also dominated by tree search approaches and state evaluation using both heuristics [18] and neural networks [19], [20] from selected features. In Hearthstone, more diverse studies such as win rate prediction [21] and finding best card tuning to achieve balance in the metagame [22] are present.

Due to its recent proposal as AI research/competition environment, the state-of-the-art of LOCM is composed of the submissions on previous editions of Strategy Card Game AI Competition and CodinGame's LOCM Marathon[3]. Most successful approaches use statistically-learned card rankings for draft phase and 2-turn-deep Minimax or MCTS for battle phase [23]. Imposing action orderings, using macro-actions (such as attacking opponent with all able creatures) and other tree pruning strategies are also common. The current winner agent uses two separate card rankings for situations in which it plays first or second. Both handmade heuristics and reinforcement learning are used to estimate state values.

IV. Methodology

In this paper, we propose the first pure reinforcement learning approach for *Legends of Code and Magic*. Both draft and battle phases use policy networks that map game states directly to actions and are trained simultaneously by self-play. Both network architectures are yet to be defined. To apply reinforcement learning to each phase separately, they each need to be modeled as a Markov decision process (MDP). In other words, a set of possible game states and possible actions need to be defined, as well as a reward model that expresses the agent's goal. The intended strategy for each phase and their MDP modeling are detailed next.

A. Draft Phase

In the draft phase, each player is required to choose a card between three randomly presented cards each turn, for 30 turns. All chosen cards become the player's deck on the battle phase. On each turn, the game input is formed solely by

[2]For the full set of rules, see https://jakubkowalski.tech/Projects/LOCM

[3]Info and results available at https://www.codingame.com/contests/legends-of-code-and-magic-marathon.

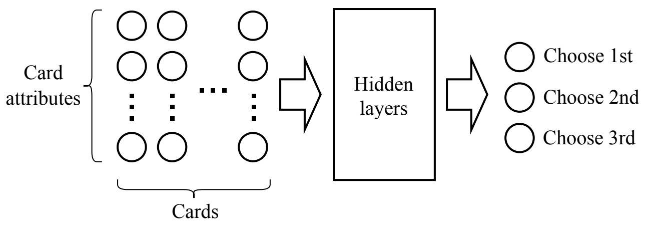

Figure 2. Expected model architecture for draft phase. Takes card attributes from the current choices and past picks as input and outputs probabilities of choosing each of the three cards.

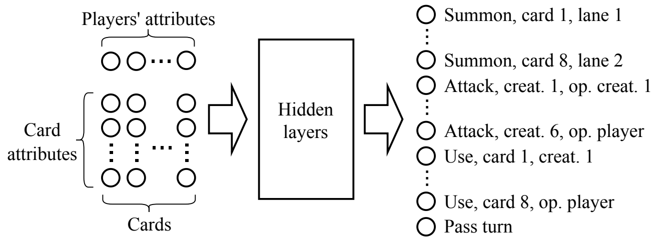

Figure 3. Expected model architecture for battle phase. Takes card attributes from all visible cards and both players' information as input and outputs probabilities of choosing each action.

the three card choices and their attributes. The active player is then expected to output the index of the chosen card.

In the draft phase's MDP model, an episode is composed by all 30 turns. As state representation, we consider not only the attributes of the current choices but also all past picks, to provide context for deck-building strategies to be learned. In this phase, there are always three possible actions: choose the first, second or third card.

The main challenge is to find a good reward model. Initially, we consider the simplest case of receiving 1 as reward at the end of the episode if the match is won and -1 otherwise. In most matches, though, not all cards in deck are used in battle phase. Therefore, a direct optimization is to eliminate any reward on actions that selected unused cards, since they have not contributed to the result. An alternative is to add some form of intermediate reward other than the match result, at the cost of possibly falling into local maxima.

The proposed reinforcement learning model for draft phase uses a neural network that takes as input all attributes of the cards chosen in the previous turns plus the three card choices in the current turn. As output, three neurons represent the probability of selecting each of the three cards. As playing first or second is significantly different [23], using a distinct model for each case is also considered. Figure 2 depicts the model architecture for the draft phase.

B. Battle Phase

The game input on a battle turn presents information of both players including current health and mana points, followed by the amount of cards in the opponent's hand and the actions performed by them last turn. Finally, the attributes of all cards in the board and in the player's hand are listed. The active player is then expected to output all desired actions for the turn.

In battle phase's MDP model, an episode is composed by all battle turns until the game ends. As state representation, we consider information from all players plus the attributes of all cards visible by the active player. 163 distinct actions are possible, including every combination of origins and targets for the summon, attack, use and pass turn actions.

The proposed reinforcement learning model for battle phase also uses a neural network and takes as input the game state and gives as output the probabilities of choosing each of the 163 possible actions. Not all actions are valid on every state, thus the output probabilities are adjusted accordingly after the prediction. We also consider using policy improvement strategies such as in AlphaGo Zero [2]. Figure 3 depicts the model architecture for the battle phase.

V. Preliminary and Expected Results

To speed-up experiments and facilitate further research, we developed an open-source reimplementation of the game engine.[4] It follows the OpenAI Gym [24] interface, as to increase compatibility with reinforcement learning algorithms. Three Gym environments are present: draft phase only, battle phase only and full game.

As preliminary result, we studied the contribution of each phase to winning the game. Every combination of random and state-of-the-art card ranking drafting; and random and state-of-the-art fixed-depth MCTS battling is tested against a baseline for 2048 matches. Table I shows the win rates of each combination.

	Random battle	MCTS battle
Random draft	50.00%	99.63%
Card ranking draft	61.08%	99.73%

Table I
Win rates of each combination of draft and battle strategies against a baseline (random draft and random battle).

Switching from random to a better strategy on both phases significantly increases the win rate. This is positive evidence that both draft and battle phase strategies are important and contribute to the final result of the match, though the battle phase seem to be more relevant.

By considering past picks on each choice, the draft phase models are expected to achieve better results than the fixed card rankings used by the best LOCM agents currently. Similarly, by considering full information of the board, the battle phase model is expected to perform better than limited-depth tree search strategies. On the other hand,

[4]Available at https://github.com/ronaldosvieira/gym-locm

more information as input implies a bigger state space and, consequently, longer training and tuning times.

The proposed models are to be compared with equally-tuned implementations of baseline and state-of-the-art agents. We plan to evaluate the draft and battle models both jointly and individually, using win rate over other agents as main metric. Multiple proposed agents varying on network architecture and modeling may be used on the experiments.

VI. Conclusion

Games are ideal environments for research on artificial intelligence, and collectible card games provide additional challenges in comparison to currently solved classic board games. In this paper, we proposed a pure reinforcement learning approach to a new card game, *Legends of Code and Magic*, that was designed to facilitate research. The intended modeling, solution and alternatives were described.

Initial experiments show that the strategy used on both draft and battle phases significantly affect the match result, meaning that any prospective state-of-the-art agent should then care about both phases. The proposed models are to be compared to baselines and the state-of-the-art. With this work, we expect to show that pure reinforcement learning approaches are viable and achieve good results in the domain of collectible card games.

For future work, a stateful recurrent neural network could be tested as replacement of considering the past picks in the state representation of the draft phase. Moreover, *Hearthstone* and *The Elder Scrolls: Legends* also figure a game mode similar to LOCM's draft phase. This enables our draft strategy to be ported to these games. Lastly, future work could address the uncertainty in the game (opponent's hand and next draws) by, for example, using a Partially Observable Markov Decision Process (POMDP) instead of an MDP.

References

[1] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018.

[2] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[3] N. Brown and T. Sandholm, "Superhuman AI for multiplayer Poker," *Science*, 2019.

[4] P. I. Cowling, C. D. Ward, and E. J. Powley, "Ensemble determinization in Monte Carlo tree search for the imperfect information card game Magic: The Gathering," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 4, no. 4, pp. 241–257, 2012.

[5] Blizzard Entertainment, "Hearthstone: Heroes of Warcraft," 2014.

[6] Bethesda Softworks, "The Elder Scrolls: Legends," 2017.

[7] M. Campbell, A. J. H. Jr., and F. Hsu, "Deep Blue," *Artif. Intell.*, vol. 134, no. 1-2, pp. 57–83, 2002.

[8] J. Schaeffer *et al.*, "Checkers is solved," *Science*, vol. 317, no. 5844, pp. 1518–1522, 2007.

[9] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] F. Zilio, M. O. R. Prates, and L. C. Lamb, "Neural networks models for analyzing Magic: the Gathering cards," *CoRR*, vol. abs/1810.03744, 2018.

[11] A. J. Summerville and M. Mateas, "Mystical tutor: A Magic: The Gathering design assistant via denoising sequence-to-sequence learning," in *AIIDE, 2016, Burlingame, California, USA.*, 2016, pp. 86–92.

[12] G. Zuin and A. Veloso, "Learning a resource scale for collectible card games," in *CIG*, 2019.

[13] A. Churchill, S. Biderman, and A. Herrick, "Magic: The Gathering is turing complete," *CoRR*, vol. abs/1904.09828, 2019.

[14] C. D. Ward and P. I. Cowling, "Monte Carlo search applied to card selection in Magic: The Gathering," in *CIG*, 2009, pp. 9–16.

[15] "Magarena is a single-player fantasy card game played against a computer opponent," http://github.com/magarena/magarena, accessed: 2019-07-14.

[16] P. García-Sánchez *et al.*, "Evolutionary deckbuilding in Hearthstone," in *CIG*, 2016, pp. 1–8.

[17] Z. Chen *et al.*, "Q-DeckRec: A fast deck recommendation system for collectible card games," in *CIG*, Aug 2018, pp. 1–8.

[18] A. Santos, P. A. Santos, and F. S. Melo, "Monte Carlo tree search experiments in hearthstone," in *CIG*, 2017, pp. 272–279.

[19] M. Swiechowski, T. Tajmajer, and A. Janusz, "Improving Hearthstone AI by combining MCTS and supervised learning algorithms," in *CIG*, 2018, pp. 1–8.

[20] D. Wang and T. Moh, "Hearthstone AI: oops to well played," in *ACMSE*, 2019, pp. 149–154.

[21] Q. H. Vu *et al.*, "Predicting win-rates of Hearthstone decks: Models and features that won AAIA'2018 Data Mining Challenge," in *FedCSIS 2018*, 2018, pp. 197–200.

[22] F. de Mesentier Silva *et al.*, "Evolving the Hearthstone meta," *CoRR*, vol. abs/1907.01623, 2019.

[23] "Legends of Code & Magic (CC05) - Feedback & Strategies," https://www.codingame.com/forum/t/legends-of-code-magic-cc05-feedback-strategies/50996/63, accessed: 2019-07-14.

[24] G. Brockman *et al.*, "OpenAI Gym," 2016.