Effectiveness of MCTS to Develop a Database of Good Decks in TCG

Atsuhiro Yamada
Graduate School of Advanced Mathematical Sciences
Meiji University
Tokyo, Japan
cs192023@meiji.ac.jp

Kazushi Ahara
Graduate School of Advanced Mathematical Sciences
Meiji University
Tokyo, Japan
ahara@meiji.ac.jp

***Abstract*— When we play the Trading Card Game (TCG), we guess the opponent's deck and predict how the game will go by considering the played cards. If we can visualize good combinations of cards, that should be useful. In this paper, we discuss how to develop a database of good decks, the so-called "Best Deck in Format (BDIF)". We point out desirable factors of the database, adopting Monte Carlo Tree Search (MCTS) method. As a result, although our method has several defects, we reveal that it is possible to develop a database of BDIF for MCTS by simulation.**

Keywords—AI agents, Trading Card Game, Deckbuilding Monte Carlo Tree Search

I. Introduction

The Trading Card Game (TCG) is a kind of imperfect information game. TCG has two phases in a set of games. One is the phase where the players build their decks from a pool of cards (deckbuilding phase.) The other is the phase where the players battle each other with cards of their deck (game phase.) When we discuss TCG strategies, we often focus on how to guess the cards in the opponent's deck. Players have a list of pool of cards. But at the beginning of the game phase, a player does not have any information about the opponent's deck. The opponent has a significant invisible area in his/her deck from the start.

Skilled players are good at hearing the opponent's 'voice.' They observe the opponent's playing cards, actions, and strategies through games and learn how the opponent constructs his/her deck. In this paper, we focus on how we know the inclination of deck cards from playing cards using an AI algorithm. Here we build a hypothesis that any rational player knows 'the good combination' implicitly to construct a good deck, or what is called the best deck in format (BDIF.)

This paper deals with the Hearthstone [1] and discusses how to construct a database of the good combinations that maximize the cards' power by AI. Here we adopt the Monte Carlo Tree Search (MCTS). The MCTS is well known as a powerful tool in perfect information games such as Go[2]. Even in imperfect information games as TCG, if we open the contents of the player's deck perfectly by Determinization, MCTS is a useful method to analyze games. See [3].

Here we prepare two players with MCTS to play the game phase of the Hearthstone. We evaluate the decks and combinations from battling simulations by the agents. One uses a deck we would like to evaluate, while the other uses decks developed randomly. A good deck should win more than other decks; therefore, we can create a database.

II. Related works

Cowling et al. [3] focus on optimized agents in *Magic; the Gathering*. First, they fix a pool of cards and deck to play. Thus they apply some methods (MCTS, or Determinization) to this game and optimize the agents' play and verify the effectiveness of these methods.

In [4], García-Sánchez et al. tried to strengthen a deck in the MetaStone platform by using a genetic algorithm (GA). They used only a specific AI agent, compared many kinds of deck, and selected the agent's BDIF. Finally, they succeeded in developing a strong deck under the assumption of fixing an agent.

In [5], Betley et al. prepared classified (in advance) two types of deck in the TCG and fixed an agent. Having done this, they tried to estimate the winning percentage of games.

In [6], Yamada et al. used a GA to formulate an evaluation function. They settled evaluation function as the inner product between the feature vector and the weight vector, and calculated the weight vector by the GA.

III. Monte Carlo Tree Search

MCTS is the application of bandit-arm planning [7] [8] to search for a game tree. This algorithm uses the UCB value shown below to decide the node.

$$UCB = \overline{x_j} + c\sqrt{\frac{\ln n}{n_j}}.$$

In this formula, $\overline{x_j}$ is the average reward from arm j, c is an appropriate constant (we use 1 in this paper), n is the total number of plays, n_j is the number of plays on the node j. MCTS is processed by the four factors illustrated below.

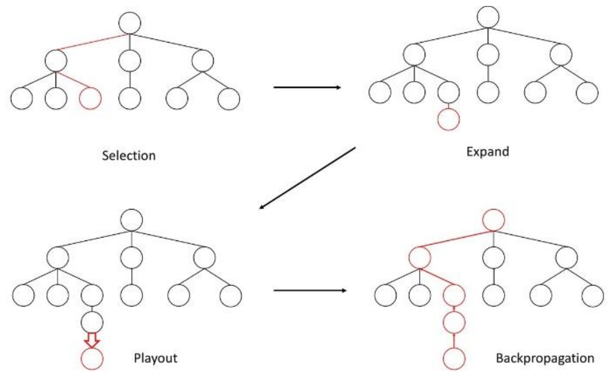
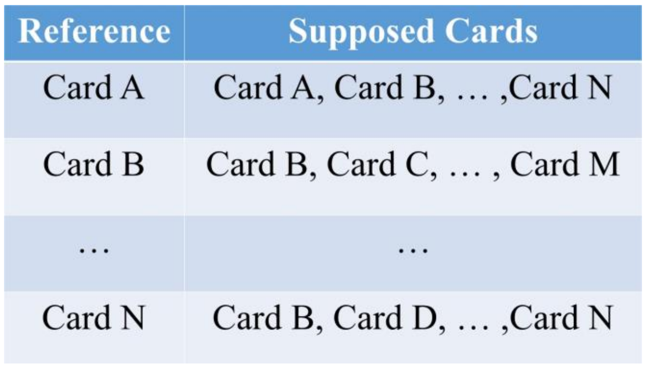


Figure 1 Four steps of MCTS

IV. PROPOSED METHOD

This paper deals with constructing a database of a deck like below.



Reference	Supposed Cards
Card A	Card A, Card B, … ,Card N
Card B	Card B, Card C, … , Card M
…	…
Card N	Card B, Card D, … ,Card N

Figure 2 Sample of database

This database has two columns, 'Reference' and ‘Supposed cards’. First, we explain the usage of the database. In the game phase, suppose that a card appears in the field from the opponent's deck. The agent refers to the column 'Reference' to find the name of the revealed card. The column 'Supposed cards' shows the agent some candidates for the opponent's deck members; the agent assumes these sets of cards in the simulation using MCTS. Next, we show two requirements of the database as follows.

1) *Record BDIF*

Decks in this database should be BDIF, that is, strong decks in that format because the primary purpose of the database is to win the game. Therefore, we should distinguish BDIFs from other weak decks to create a database.

2) *All Cards*

The column 'Reference' includes all cards in the pool so that the agent can retrieve the deck in any case. There is a card in the pool that nobody uses at all. If the database does not include such a card, the agent must be allowed to suppose random cards for the deck.

Last, we show our approach to create the database as follows.

1) *Focus on a specific card.*
2) *Develop decks randomly with the focused card.*
3) *Evaluate the deck by MCTS simulations.*
4) *Record a deck that gets a high score on average in the database.*
5) *Repeat 1)-4) while all cards are in focus.*

We may think that the quality of the deck depends on the character of the opponent. By using this approach repeatedly with the MCTS, the agent can suppose adoptable decks of the opponent.

V. EXPERIMENT

The first author developed the experimental environment and codes in Python. He used ‘Fireplace’ [9], a simulator of Hearthstone. We limited the pool of cards to simplify the experiments and to make it easy to distinguish the strength of the decks. In this experiment, we set the pool of cards consist of only *Murloc* cards and *Beast* cards. Moreover, we reduced the number of deck cards to eight. In this variant of Hearthstone, we focused on the card *Grimscale Oracle.* We took the approach to making a database as above.

We prepared two agents who used MCTS with entirely random game trees. One agent had the target deck (a deck with the specified card), and the other had a random deck. We let the playouts number be 90, let the agents evaluate target deck by repeating the game phase 30 times.

Since the pool of cards included only two races, we defined the good combination of deck in this study as a deck where almost all cards had the same race, *Murloc*. We introduced a 'Race Bias value' (RB) that represented how cards of the *Murloc* gathered in a deck. RB is formulated as

$$\mathrm{RB} = x_M - x_B.$$

Here x_M is the number of *Murloc* cards and x_B is the number of *Beast* cards. We verified the hypothesis that the correlation between RB and the game score was positive in general games. We fluctuated some game parameters (for example, the life parameter or the mana parameter) of the game.

We changed the parameter 'life of player' in Experiment A, and we changed the parameter 'max mana' in Experiment B. In Experiment A, we let the player's life parameter be 5, 10, and 15 (case I, II, III, respectively.) In Experiment B, we let the max mana parameter be 10, and 1. In the second parameter of Experiment B, the durable deck should have low-cost cards, because high-cost cards are useless.

VI. Results

a) Experiment A

We plot data of results with RB as the x-axis and score as the y-axis.

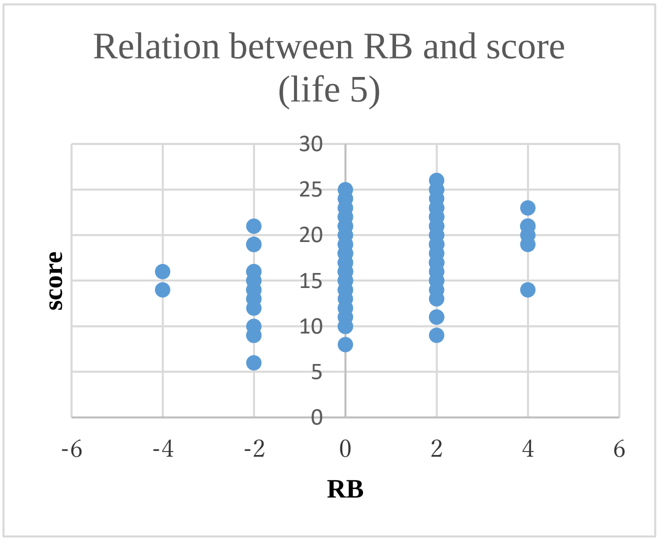


Figure 3 RB and score in case I

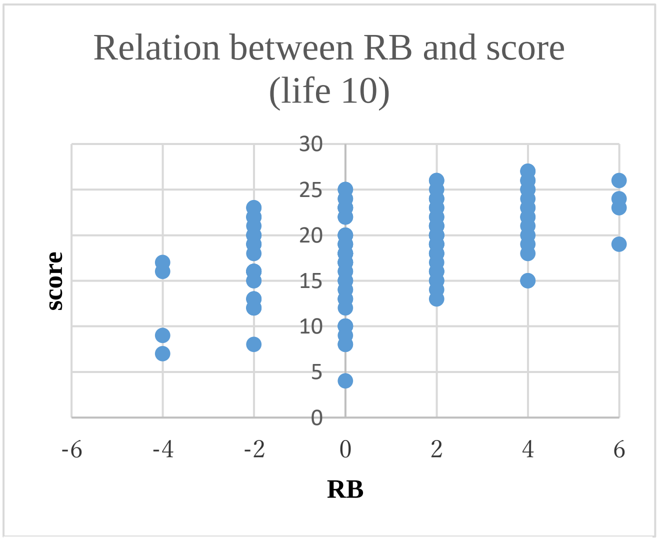


Figure 4 RB and score in case II

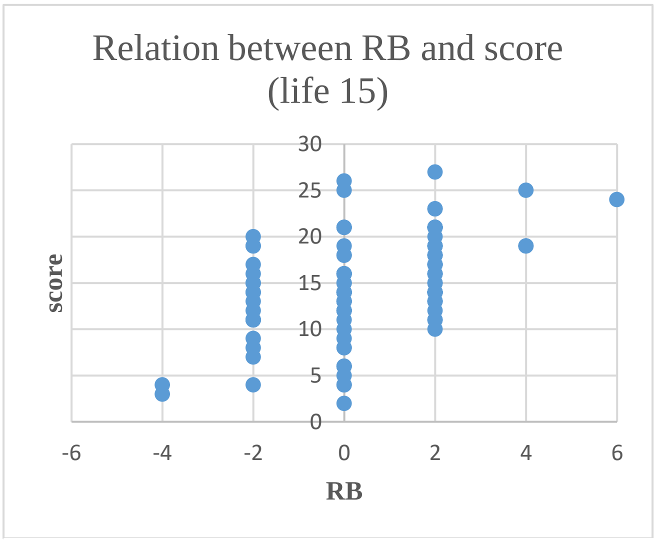


Figure 5 RB and score in case III

RB and Score had a positive correlation in any case as shown in the figures. These results suggest the agent can distinguish BDIF from other decks. Thus, we can meet requirement 1 in chapter IV to develop the database.

However, we should pay attention to specific cases; for example, *Grimscale Oracle* is the only *Murloc* card and the others are *Beast* cards. Such a deck is not appropriate for registering the database with the 'Reference' column *Grimscale Oracle* because the deck should be retrieved by other *Beast* cards. Therefore, we have to set other rules for recording data 'Reference' and 'Supposed cards'.

b) Experiment B

We plot dots of results with the number of low-cost cards as the x-axis and score as the y-axis. Here low-cost means the cost is 0 or 1.

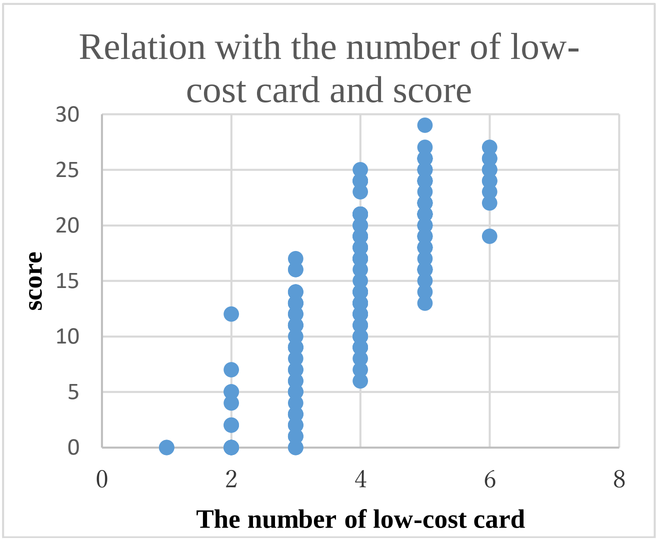


Figure 6 low-cost card and score

There were positive effects between the numbers of the card whose cost was 1 or 0 and Score. These results also indicate that our method has potential to unravel combinations of the cards. In this situation, we cannot use a card whose cost is more than 2. We argue that a database should include such cards as 'Reference' although nobody adopts such a card. Therefore, we should focus on all cards when developing a database.

VII. Conclusion and Future Work

In this paper, we discussed the importance of a decklist database of BDIF created using MCTS. We also examined the usefulness of the proposed method by simulation. We clarified that an agent can evaluate decks properly through simulation in this limited pool of cards. Moreover, we have to discuss the disadvantages of our method. It is not sure that retrieving results is entirely appropriate in the sense we describe in chapter VI. Thus, in future work, we should try to find an efficient way to record proper data, and we try to guess the calculation amount of complete a useful database for another specific pool of cards and confirm its effectiveness.

References

[1] Hearthstone. (https://playhearthstone.com/en-us/ retrieved 2020/05/16)
[2]C.-S. Lee et al. (2009). “The computational intelligence of

MoGo revealed in Taiwan’s computer Go tournaments”. IEEE Transactions on Computational Intelligence and AI Games,(73-89) ,1(1)

[3] Peter I. Cowling et al. (2012). “Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering”. IEEE Transactions on Computational Intelligence and AI in Games (241-257),4(4)

[4] García-Sánchez et al. (2016). “Evolutionary deckbuilding in hearthstone”. IEEE Conference on Computational Intelligence and Games doi: 10.1109/CIG.2016.7860426

[5] Jan Betley et al.(2018). “Predicting winrate of Hearthstone decks using their archetypes”. Federated Conference on Computer Science and Information Systems Doi: 10.15439/2018F362

[6]Atsuhiro Yamada, Kazushi Ahara. (2018). ” Consideration of Learning Model with Neural Network to Build Decks and Develop Agents in Trading Card Game” Game Programing workshop-18 (128-132)

[7]P. Auer, et al.(2002). “Finite-time analysis of the multi-armed bandit problem”. Machine Learning, (235-256),47(2)

[8]Katehakis, M. N, Veinott, A. F. (1987). "The Multi-Armed Bandit Problem: Decomposition and Computation". Mathematics of Operations Research. doi:10.1287/moor.12.2.262

[9]Github.(https://github.com/demilich1/metastone retrieved 2019/05/10)