Towards Finding a Solution for Collectible Card Games with Counterfactual Regret Minimization

Haoyu Zhang[1,a)] Yoshimasa Tsuruoka[1]

Abstract: In recent years, agents trained to cope with various environments have received attention in the machine learning field. An intuitive rational decision-making environment is card games and researchers are especially focused on solving the game of poker. In this work, we focus on Collectible Card Games (CCGs) which have rich diversity of game play and much more randomness compared to poker. In order to solve such complicated card games, we create several simple models of the real game and conduct experiments on these models with Counterfactual Regret Minimization (CFR) algorithms. As results, agents learn equilibrium strategies successfully in the simple model and we confirm the convergence of Counterfactual Regret Minimization (CFR) algorithms.

1. Introduction

Researchers have been studies how to train game AI agents to cope with various environments for many years. The single-agent environments both with perfect information and imperfect information are hot research fields, and researchers have made many excellent contributions in these fields which are studied via Monte Carlo Tree Search or reinforcement learning.There are already some promising approaches of perfect information games in multi-agent environments such as mastering the game of Go [1]. But many problems in imperfect information games remains.

Card games, which are intuitive rational decision-making environments, have multiple players during game process that make card games become systems in which multiple agents share common environments. This is a constantly changing system, in which agents interact with both environment and each participant. Collectible Card Games (CCGs) are a very popular type of card games, and because of the diversity of its game-play, it has become a platform for researchers to do AI research. One of the games, named *Hearthstone*, is a new research target in CCGs. Several algorithms based on perfect information games, such as Monte Carlo Tree Search, are also proposed to solve CCGs.

2. Background

2.1 Extensive Form Game

The extensive-form game is a natural model for sequential decision-making enviroments with multiple agents from game theory [2]. It uses a game tree to represent the entire game process. Tic-Tac-Toe is one of the simplest perfect information games and part of its game tree is shown in Fig. 1. On each non-terminal game state players have several available actions and every terminal state contains rewards for each of players.

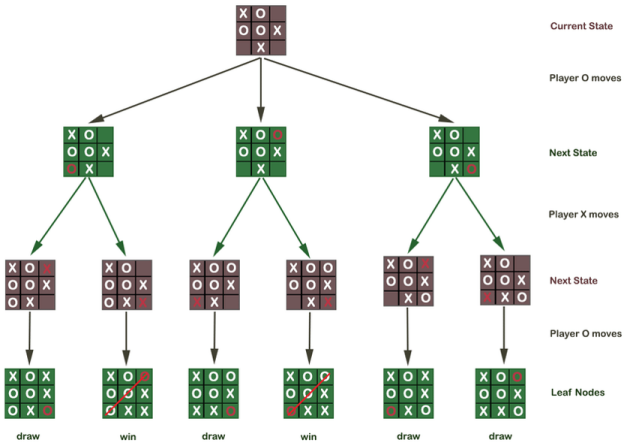


Fig. 1 Part of game tree of Tic Tac Toe

Players of Tic-Tac-Toe know exactly which game state they are on currently, and also understand which game state they would move to via observing the opponent's actions.

However, in imperfect information games, the situation is different because of the unobservability. The key difference is the information sets, which are sets of game states, in which players cannot distinguish which current state exactly they are in and so must choose actions for all such indistinguishable states with the same distribution. Thus the extensive-form of imperfect information games is denoted as:

- N is a finite set of players.
- H is a finite set of sequences which is the histories of actions. $Z \in H$ are the terminal histories.

[1] Department of Information and Communication Engineering, The School of Information Science and Technology, The University of Tokyo
[a)] zhanghaoyu@logos.t.u-tokyo.ac.jp

- $A(h)$ are the actions available after a non-terminal history $h \in H$.
- P is the player function, $P(h)$ is the player who takes an action under that history.
- I_i is a information set of player i belongs to $\mathcal{I}_i$ which is a information partition of player i.
- u_i is a utility function for each player i from the terminal states Z to the regrets R. If $N = (1, 2)$ and u_1 = -u_2, it is a zero-sum extensive game. Define $\Delta_{u,i} = \max_z u_i(z) - \min_z u_i(z)$ to be the range of utilities to player i.

2.2 Strategies and Equilibria

A strategy of player i, σ_i in an extensive game, is a function which represents a distribution over $A(I_i)$ to each $I_i \in \mathcal{I}_i$, and $\sum_i$ is the set of strategies of player i. A strategy profile σ consists of strategies for each player, σ_1, σ_2,..., with σ_{-i} referring to all the strategies in σ except σ_i. Let $\pi^\sigma(h)$ be the probability of history h if players choose actions according to σ. It naturaly decomposes $\pi^\sigma(h) = \Pi_{i \in N} \pi_i^\sigma(h)$ into each player's contribution to this probability. According to these notions, the overall value to player i of a strategy profile is the expected payoff $u_i(\sigma) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$, combining with each player would obtain a value at each final state of the game.

A tranditional solution concept of a two-player extensive game is the Nash equilibrium and it refers to such a situation in the game: for each player, as long as the other players do not change their strategies, the player cannot improve his own performance. The Nash equilibrium has been proved to exist in the premise that each player has only a limited number of strategic choices and allows for a mix strategy. This is an important property of extensive games. A Nash equilibrium is a strategy profile σ where

$$\begin{aligned} u_1(\sigma) &\geq \max_{\sigma_1' \in \sum_1} u_1(\sigma_1', \sigma_2), \\ u_2(\sigma) &\geq \max_{\sigma_2' \in \sum_2} u_1(\sigma_1, \sigma_2'). \end{aligned} \tag{1}$$

2.3 Collectible Card Games

Since the release of $Magic : The\,Gathering$ in the 1990s, Collectible Card Games (CCGs) [3] have been one of the most popular and profitable products in the entertainment industry. Recently, the digital version also appeared, $HearthStone : Hero\,of\,Warcraft$ is one of the most popular digital CCGs. CCGs are turn-based card games in which players set their decks in advance and carefully select cards to have the opportunity to take advantages of powerful combinations during real matches. Because each card has a specific function, there are always complex and diverse game modes.

In addition to the game modes, cards are obviously a major component of the game. However, as the number of cards increases, the number of interactions or 'combos' also increases, which makes the strategy of such games depending on the player's constructed deck. Therefore, most of the previous research focuses on the strategic exploration of fixed decks.

2.4 Hearthstone

$HearthStone : Heros\,of\,Warcraft$ is an online digital CCG (DCCG) launched in 2013 by Blizzard Entertainment.

2.4.1 Cards

Players can create a deck consisting of 30 cards from a huge pool of cards. In order to win, the player needs to use several types of cards to reduce the health of opponents from 30 to 0, such as spells that affect the battlefield discarding directly after be played, and the minions, that could be considered as servants, are put into the battlefield and can attack the enemy or other minions. In addition, weapons that allow heroes to use special equipment to attack other characters in a few rounds. An example of minion is shown in Fig. 2 which cost/mana of four with four attack and five health points.

Fig. 2 Example of minion

2.4.2 Mechanism

Each card has an associated cost, also called crystals, equivalent to mana, which is reduced after playing, but replenished at the beginning of each turn. Each player starts with one crystal, and adds one more per turn up to ten crystals, the maximum. Deckbuilding is limited to neutral pools and to the cards belonging to a specific hero that players select for the game from a total of seven heroes. In addition, each hero has a different hero power (costing two crystals), combined with their special cards, making each hero a special prototype. There is a screenshot of Hearthstone match shown in Fig. 3 which shows the various features of the battle.

Fig. 3 Screenshot of Hearthstone match

2.4.3 Challenge

According to the definition, the differences between CCGs and ordinary card games are very obvious. Although Hearthstone and poker are both games of imperfect information, we are familiar with the card features of poker. Each card of poker has a different number which represents the strength of this card. Poker has a total of 54 cards. Players perform an action in one round in most modes of poker games. Similarly, in Hearthstone, players also play cards in hands which draw from a deck containing 30 cards. But the opponent's deck is unpredictable, and unless he finishes all the cards, we cannot fully know his deck. This deck is selected from a larger card pool which contains more than two thousands cards. In addition, poker players can perform only one action to play cards or pass, but Hearthstone players can perform many different sequences of actions in one round. In most CCGs, players have several action options, and their combinations make up the real sequence of actions. The order in which the actions are performed in every unique action sequence is crucial and determines interactions which we call 'combo'. The different card features and performed actions make the game tree too deep and large to handle.

2.5 Previous Methods

For Hearthstone, there are already several mature simulators in the Hearthstone community named *Hearthsim*. Some of those simulators implement the main game mechanism and include several artificial intelligence agents that allows players to play against, or run simulations, and provide statistics after the game is over. Those methods are:

- *Random*
  The agent always chooses one action randomly from all possible actions without any logic.
- *Noaggression*
  The agent never actively attacks the opponent, it just randomly chooses between $PLAY\,CARD$ and $PASS$ which means end-turn.
- *Greedy*
  The agent always chooses the most valuable action, and does not consider too much. Its actions are driven by an algorithm built on several game metrics using an evolutionary approach.
- $GameStateValue$
  The agent makes an action selection based on the value of the current state. It is a recursive Alpha-Beta pruning algorithm controlled by a heuristic algorithm.
- $MonteCarloTreeSearch$
  Researchers have tried to apply the Monte Carlo tree search algorithm, which is very mature in the perfect information game, to Hearthstone, and made some adaptive improvements recently [4]. It makes use of a real player-based deck database to make Monte Carlo tree search possible on imperfect information situations.

Besides those game playing methods, researchers are also trying to solve it in several different aspects. Such as deck building [5], card generation and data mining [7].

3. Proposed Approach

As we mentioned before, the $GameStateValue$ algorithm apparently calculates the value based on fixed decks and the $MonteCarloTreeSearch$ algorithm are not suitable for CCGs which is an imperfect information game. In order to solve such a complicated imperfect information game, we use a mature algorithm Counterfactual Regret Minimization (CFR) [8] which is one of the most efficient methods for computing Nash equilibria in large, zero-sum, in imperfect information games. In the domain of poker, CFR has proven effective, and produced a lot of variants, such as MCCFR [9], CFR+ [10] and DisCFR [11].

3.1 Counterfactual Regret

Counterfactual Regret Minimization is an iterative algorithm that contains a sequence of strategies σ^0, σ^1,..., σ^T, the average strategy $\bar{\sigma}^T$ of which converges to an approximate Nash equilibrium as $T \to \infty$ in the two-player zero-sum game. In order to obtain the average strategy, it should computes counterfactual value at first,

$$v_i(\sigma^t, I) = \sum_{(h,z)\in Z_I} \pi^{\sigma^t}_{-i}(h) u_i^{\sigma^t}(h, z), \tag{2}$$

where $\pi^{\sigma^t}_{-i}(h)$ has been introduced before, the probability, and $u_i^{\sigma^t}(h, z)$ is the value players can get at final state Z. Also there is an action-dependent counterfactual value,

$$v_i(\sigma, I, a) = \sum_{(h,z)\in Z_I} \pi^{\sigma^t}_{-i}(ha) u^{\sigma}(ha, z), \tag{3}$$

where ha is the sequence h followed by action a. So the counterfactual regret for not taking action a at I is:

$$r^t(I, a) = v_i(\sigma, I, a) - v_i(\sigma^t, I). \tag{4}$$

Then the regret can be accumulated as $R^T(I, a) = \sum_{t=1}^{T} r^t(I, a)$.

3.2 Regret Minimization

If we define $(x)^+ = \max(x, 0)$, so the strategies can be

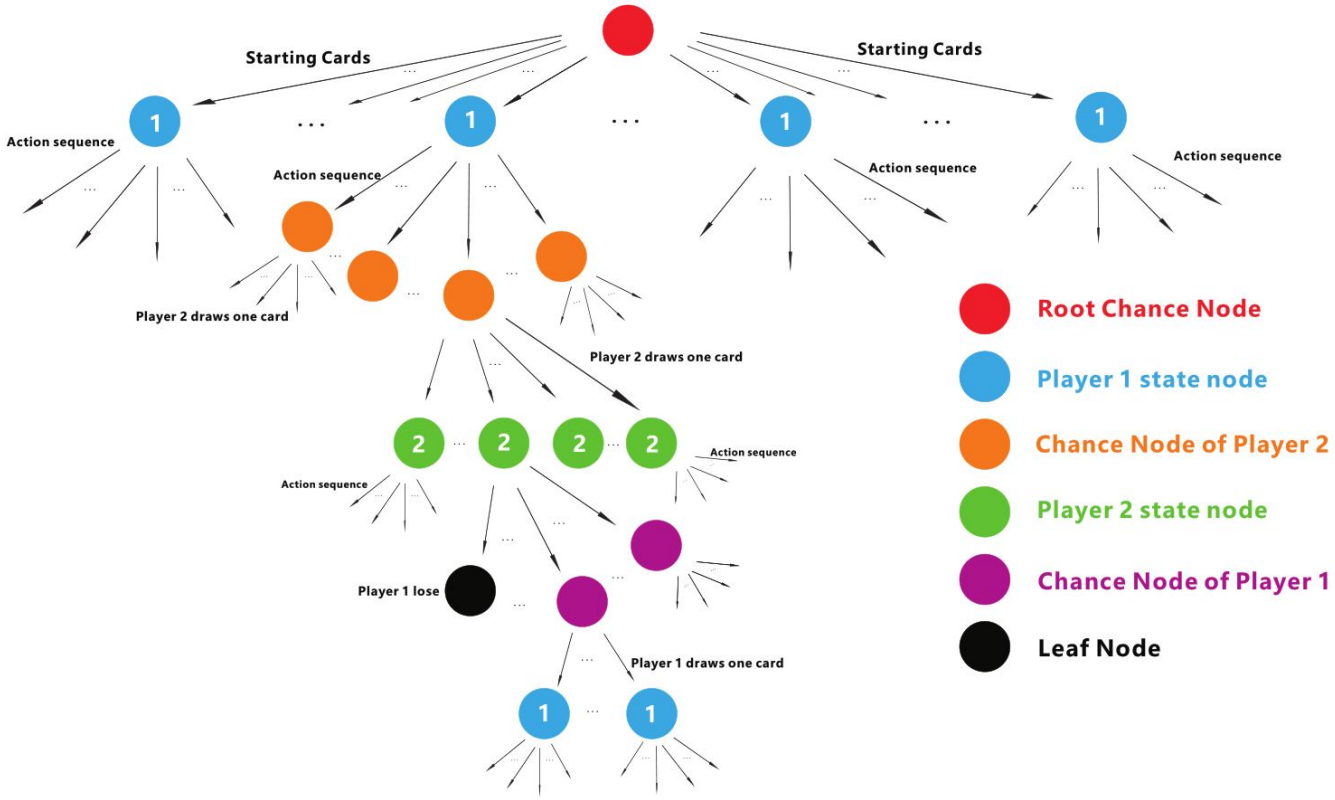


Fig. 4 Game Tree

updated as:

$$\sigma^{T+1}(I,a) = \frac{(R^T(I,a))^+}{\sum_{a\in A(I)}(R^T(I,a))^+}. \qquad (5)$$

If we run CFR for T iterations, then we can get overall regret of the strategies as:

$$R_i^T = \max_{\sigma_i^{'}} \sum_{t=1}^{T} (v_i(\sigma_i^{'}, \sigma_{-i}^t) - v_i(\sigma^t)). \qquad (6)$$

CFR gives an assurance that $R_i^T/T \to 0$ as $T \to \infty$, which means when two players minimize regret, it guarantees to converge to a Nash equilibrium.

4. Experiment

The final target of solving Hearthstone is too complicated to ensure convergence of algorithms of CFR family. So we follow the previous research route which is used to solve Texas hold'em. That is we simplify the original large game into a small model and increases its complexity step by step. In this paper, we conduct our experiments on several simple models.

4.1 Environment

The goal of simplification is to reduce the number of information sets for each player so that the small model can be solved.

4.1.1 Model

Firstly we create one-card game models in which each player has only one card, which is a minion in the deck and can play this card after drawing it. Besides this simplest model, we also conduct experiments on two-card game models in which each player has two minion cards. One is in hand at the beginning of one battle and draw another one from the deck later. After that, we adjust these models to an unbalance situation in which players may have different numbers of cards or Hero have different health points. In our simple model, players only have minions but no spells or weapons. We also remove hero power and mechanics of minion cards. Another significant difference between poker and CCGs is judgment on winning and losing. In normal poker games, after the players play cards according to certain rules, the players win or lose within a certain number of rounds. But there is an infinite round of games in Hearthstone when players choose 'pass' all the time. To prevent this unexpected situation, we limit the maximum number of rounds of game. The game tree is shown in Fig. 4.

4.1.2 Information set

As Hearthstone has more than one actions in one round such as 'attack', 'play cards', 'cast spells' or 'pass' and has a few types of cards such as summon 'minions', 'spells' and 'weapons', we consider action sequences as variable that determines state transition in the game tree instead of actions in normal poker games. Summon minions could be considered as servants. Spells affect the battlefield which are discarded after being played. Weapons which give heroes strength to attack. Creating a node for every action would cause the game tree to be too deep. The information set for each indistinguishable group of nodes not only contains

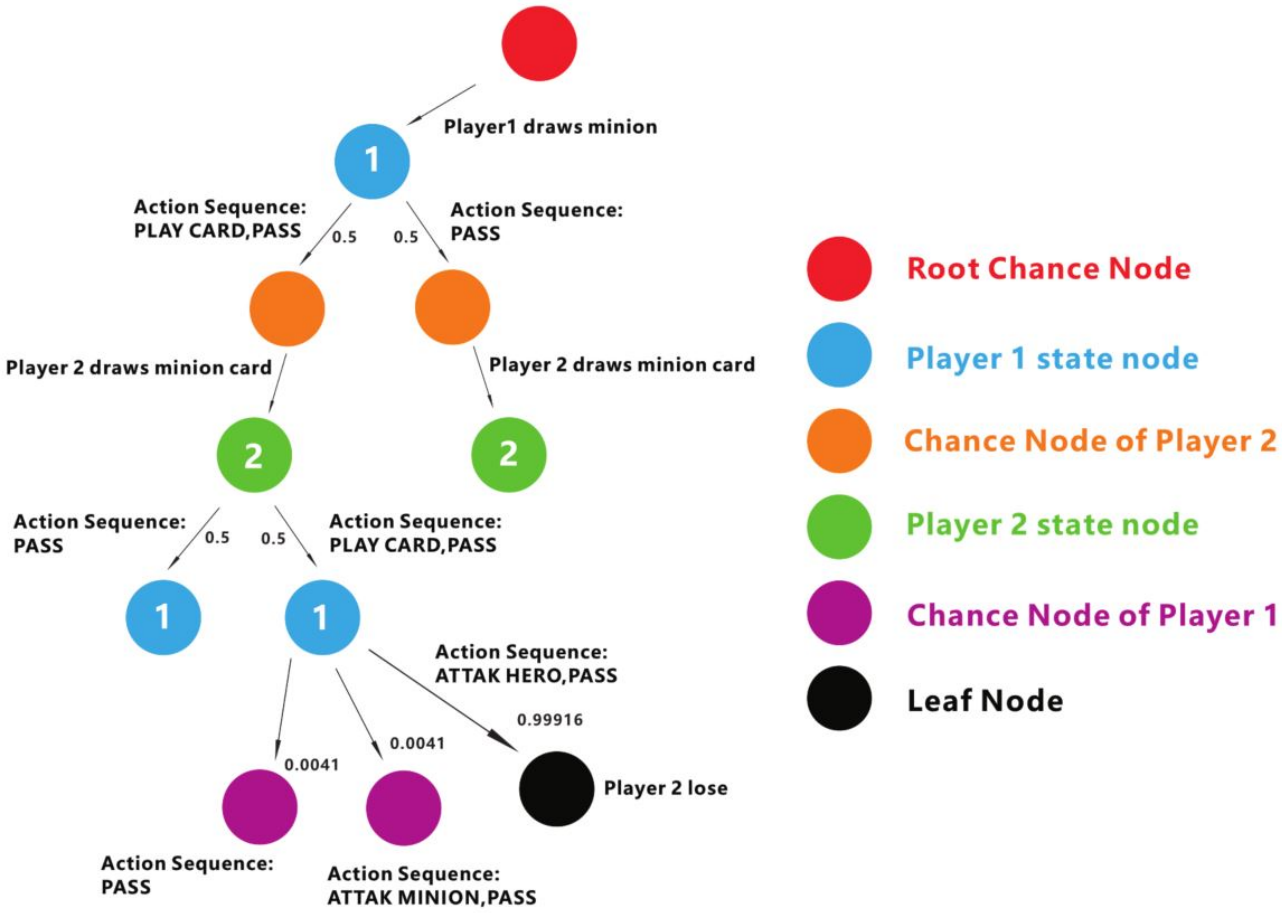


Fig. 5 Game Tree of One-card Game

the action sequences token before, but also includes hand information which means what cards are in hand.

4.1.3 Vanilla CFR and Chance Sampling CFR

We apply two different CFR algorithms from the CFR family to every model. The vanilla algorithm would traverse the whole game tree while the chance sampling algorithm just samples one action sequence from the available action sequences. We plot two results of the same model in one table in different colors.

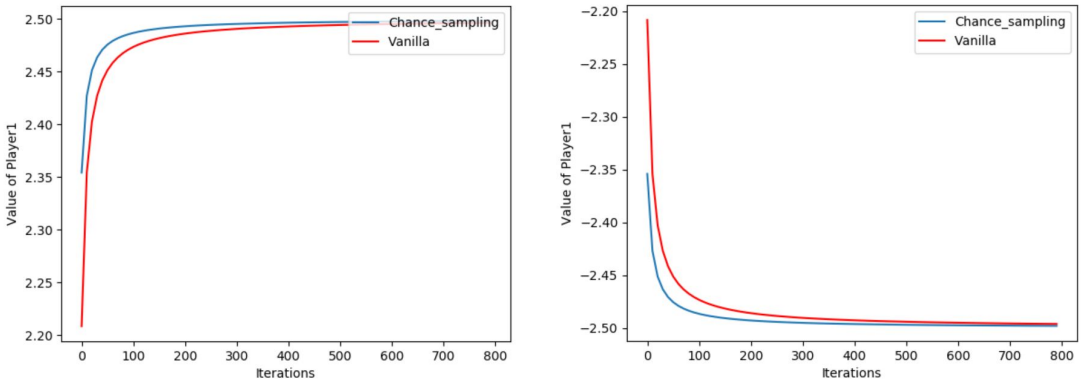
4.2 Result

In order to let the agent learn to avoid draw in limited rounds which cause many draws, we make the agent regret reaching a draw and become much more positive when it reduces opponents hero's health point below zero. We conduct experiments on several one-card games which have different cards. Part of game tree of one model is shown in Fig. 5. Agents learned probability distributions for every action available and we show the final probabilities which also are strategies in the tree. As we mentioned before, in order to obtain the strategies, we compute counterfactual value at first. So there is a value for every node and in order to verify the convergence and observe it, we calculate the value of Player 1 at the root node and plot these values over iterations.

The results of two one-card game models are shown in Fig. 6. In Fig. 6(a), two players have the same card so Player 1 wins in most branches. In Fig. 6(b), Player 1 has a minion card that costs two crystals and Player 2's card costs one crystal. That makes Player 1 lose in most branches.

Another two results of two-card game models are shown in Fig. 7. Both players have two cards that cost one and two crystals for Fig. 7(a). Player 1 has two cards that cost two and three crystals while Player 2 has two cards that cost one and two crystals for Fig. 7(b). In Fig. 7(c), Player 1 has two cards that cost one and three crystals while Player 2 has two cards that cost two and three crystals.

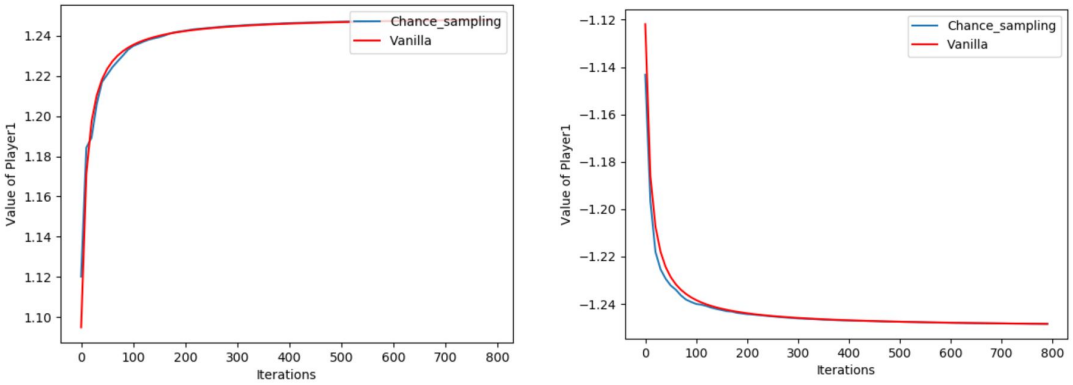
Fig. 8 shows the results of two players in the unbalanced situation in which players has different number of cards or different health points. In Fig. 8(a), Player 1 has two cards that cost one and two crystals while Player 2 has one card that costs one crystal. In Fig. 8(b), Player 1 has three cards that cost one, two and three crystals respectively while Player 2 has two cards that cost one and two crystals. In Fig. 8(c), two players have same card that costs one crystal but Player 1 only has one health point and Player 2 has two. In Fig. 8(d), Player 1 has three cards that cost one, two and three crystals while Player 2 has two cards that cost one and two crystals, and Player 1 has one health point but Player 2 has two health points.



(a) Players have same card

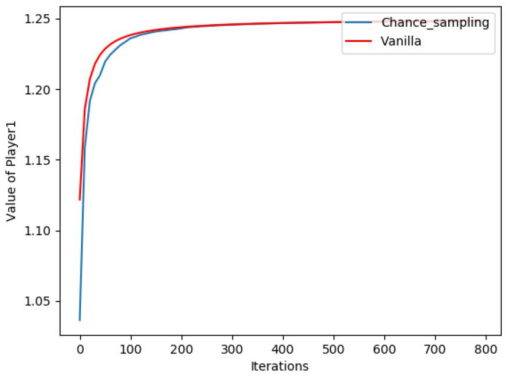
(b) Players have different card

Fig. 6 One-card game



(a) Players have same cards

(b) Players have different cards



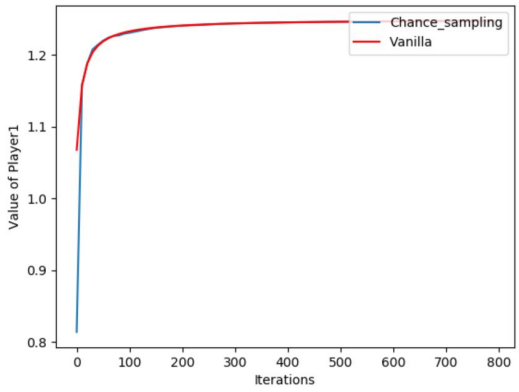
(c) Players have different cards

Fig. 7 Two-card game

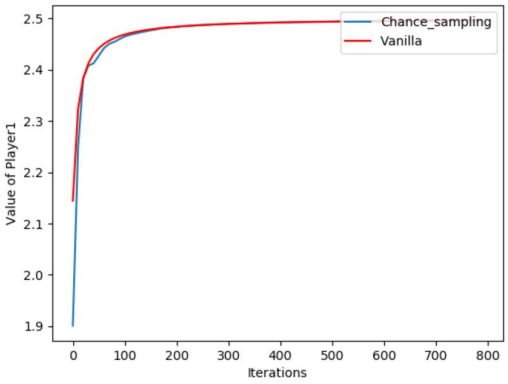
5. Conlusion

In this work, we propose a novel approach to solve Collectible Card Games (CCGs) from a traditional imperfect information perspective. In order to solve the complicated card games like Hearthstone, we create several battle environments based on Hearthstone rules and apply Counterfactual Regret Minimization (CFR) algorithms on those models. In each model, we calculate the value of player that makes the action first. Although the model tested is small, we confirm the convergence of CFR algorithms.

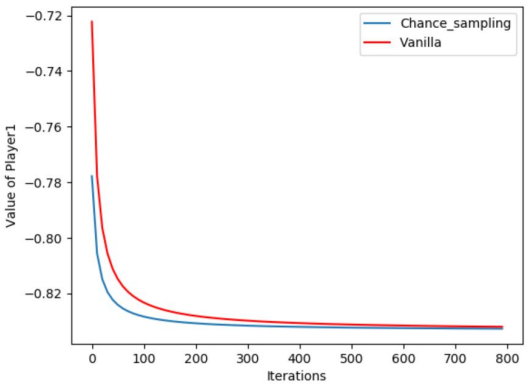
For future work, we would like to find equilibrium strategies in larger or more complicated CCGs like Hearthstone. But we think there are some difficulties to apply traditional CFR algorithms to the real game which has a larger action space and not traversable game tree. In that case, we would like to try some newer methods in imperfect information research fields [13][12].



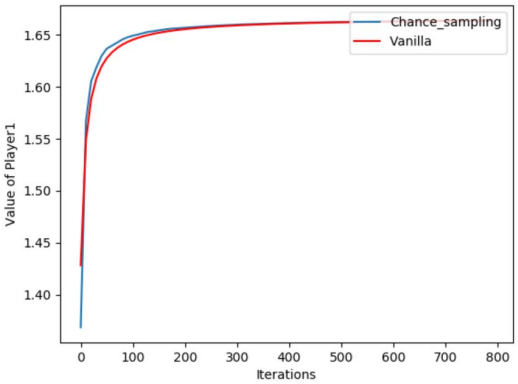
(a) Players have different number of cards



(b) Players have different number of cards



(c) Players have different health points



(d) Players have different number of cards and health points

Fig. 8 Unbalanced game

References

[1] Mastering the game of Go with deep neural networks and tree search, David Silver and Huang, 2016.

[2] *Game Theory*, Drew Fudenberg and Jean Tirole, 1991.

[3] Monte Carlo Search Applied to Card Selection In Magic: The Gathering, C.D. Ward and Peter Cowling, 2009.

[4] Monte Carlo Tree Search Experiments In Hearthstone, A. Santos, P. A. Santos, and F. S. Melo, 2017.

[5] Predicting winrate of Hearthstone decks using their archetypes, Jan Betley, Anna Sztyber, and Adam Witkowski, 2018.

[6] Investigating Similarity between Hearthstone Cards: Text Embeddings and Interchangeability Approaches, Andrzej Janusz and Dominik Slezak, 2018.

[7] Helping AI to Play Hearthstone: AAIA'17 Data Mining Challenge, Andrzej Janusz, Tomasz Tajmajer and Maciej Swiechowski, 2017.

[8] Regret Minimization Iin Games With Incomplete Information, Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione, 2008.

[9] Monte Carlo Sampling for Regret Minimization in Extensive Games, Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling, 2009.

[10] Solving Large Imperfect Information Games Using CFR+, Oskari Tammelin, 2014.

[11] Solving Imperfect Information Games via Discounted Regret Minimization, Noam Brown and Tuomas Sandholm, 2018.

[12] Superhuman AI for multiplayer poker, Noam Brown and Tuomas Sandholm, 2019.

[13] Deep Counterfactual Regret Minimization, Noam Brown, Adam Lerer, Sam Grossand and Tuomas Sandholm, 2019.

