The Feasibility of Deep Counterfactual Regret Minimisation for Trading Card Games

David Adams(✉)

University of Western Australia, Perth, WA 6009, Australia
david.adams@uwa.edu.au

Abstract. Counterfactual Regret Minimisation (CFR) is the leading technique for approximating Nash Equilibria in imperfect information games. It was an integral part of Libratus, the first AI to beat professionals at Heads-up No-limit Texas-holdem Poker. However, current implementations of CFR rely on a tabular game representation and hand-crafted abstractions to reduce the state space, limiting their ability to scale to larger and more complex games. More recently, techniques such as Deep CFR (DCFR), Variance-Reduction Monte-carlo CFR (VR-MCCFR) and Double Neural CFR (DN-CFR) have been proposed to alleviate CFR's shortcomings by both learning the game state and reducing the overall computation through aggressive sampling. To properly test potential performance improvements, a class of game harder than Poker is required, especially considering current agents are already at superhuman levels. The trading card game Yu-Gi-Oh was selected as its game interactions are highly sophisticated, the overall state space is many orders of magnitude higher than Poker and there are existing simulator implementations. It also introduces the concept of a meta-strategy, where a player strategically chooses a specific set of cards from a large pool to play. Overall, this work seeks to evaluate whether newer CFR methods scale to harder games by comparing the relative performance of existing techniques such as regular CFR and Heuristic agents to the newer DCFR whilst also seeing if these agents can provide automated evaluation of meta-strategies.

Keywords: Artificial intelligence · Machine learning · Extensive-form games

1 Introduction

Attempting to solve problems of increasing complexity is one of the main goals of artificial intelligence (AI) research. Games are often used as a test bed for such research, as they provide a reasonable environment to evaluate but can also be applicable to the real world. Over time different techniques have been created to address different classes of games, starting with simple perfect-information (whole game state is known at all times) deterministic games like Tic-Tac-Toe, to massive imperfect-information (partial unknown game state) extensive form games like Starcraft. In the case of perfect information games, Monte-Carlo Tree Search (MCTS) [10] and deep neural networks have been used in AIs such as Alpha Zero [21], which surpassed human levels of performance in Chess, Go, and Shogi. In the case of imperfect information games, a technique called Counterfactual Regret minimization (CFR) [6] was used in Libratus [9] to beat top professionals at Heads-up No-limit Texas Holdem Poker. Given that superhuman performance has been achieved at the hardest benchmark for imperfect information games, new harder games are needed to increase benchmarks for existing and future methods. Trading Card Games (TCGs) are a possible direction as, despite having a larger state space, more complex card interactions, and the concept of meta-strategies, they are still feasible to compute in comparison to massive online games like StarCraft or League of Legends, and can still be easily represented as a game tree. This paper will benchmark existing and new game solving methods, such as Deep Counterfactual Regret Minimisation (DCFR) [8], to see if they cope with the demands of more complex games like TCGs and assess whether these methods can evaluate different meta-strategies.

1.1 Foundational Work

In general, games are classified by the following properties:

- Zero-sum: overall reward sums to zero or there is some concept of a winner and loser.
- Information: whether the state is partially or fully known.
- Determinism: whether chance affects the game in any way.
- Sequential: whether actions occur one after another or simultaneously.
- Discrete: whether actions are applied in real time or not.

For simple deterministic perfect-information games with small state spaces, the whole game tree can be evaluated with the classical Minimax [4]. However, for most non-trivial games, an algorithm must decide what part of the game tree to explore. Perhaps the most widely used algorithm is MCTS [10] which was used in DeepMind's AlphaGo [22] to beat the 18-time Go world champion. Instead of hand-crafted game evaluation functions and state selection heuristics that would be required to make Minimax feasible, the algorithm used in AlphaGo used a deep neural network trained with self-play for state evaluation and a MCTS for state selection. Along with its successor AlphaZero [21] AlphaGo serves as the benchmark for perfect-information game playing performance.

Despite this excellent performance in perfect information games there are few real-world scenarios that have perfect information. In fact, most real-world situations, such as **business strategy**, **economic models** or **simple negotiation** can all be modelled as imperfect information games [15]. Whilst extensions can be made to perfect information games to make them imperfect, such as imperfect Chess [19], and simple games like Bridge are used as teaching tools, Poker is the canonical example of an imperfect information game.

Attempting to apply the Minimax algorithm or raw MCTS to Poker will lead to poor results as each game state has uncertainty and it is infeasible to enumerate all combinations. Instead, information sets (**infoset**) are used as a proxy for game state and represent the set of all possible states that could be known with the current information. Whilst MCTS can be modified to accommodate information sets, Regret Matching (or regret minimisation) has been shown to have better convergence and results in practise [24]

Intuitively, the action that you regret not taking the most is the one that should have used. A mathematical representation of regret is the difference between the reward of an action that was taken and the action that could have been taken

$$regret = \mu(\text{possible action}) - \mu(\text{action taken})$$

CFR [26] is an extension to regret matching. It deals with scenarios that have multiple steps and allows an agent to know what the regret of not taking an action is at each step. Instead of calculating the regret for an action, the regret is calculated based on a counterfactual value, which is the value of a state multiplied by the probability of reaching that state.

CFR was used to play the hardest variation of Poker (Heads up no limit Texas-holdem having approximately 10^{161} decision points) and successfully to beat top-level human players [9]. Computing a strategy for this game was obviously infeasible. As such, treating groups of scenarios as strategically identical was required. But, it came at the cost of fixing the implementation to a hand-crafted abstraction and a tabular representation. Overall, this means the original CFR techniques would not generalise well to other games, nor would they scale to extremely large games.

1.2 Current Methods

One of the first methods to deal with both tabular solving and abstraction was Deep CFR [8] (Deep CFR). It performed better than all the previously mentioned approaches and stands as one of the few algorithms that can tackle games whose state or action spaces are too large. It works by using a neural network to approximate, with theoretical convergence, the behaviour of CFR. The neural network architecture used can be seen in Fig. 1.

It is unique compared to the previous methods shown in that it does not calculate and accumulate regrets at each infoset, rather it generalises across similar infosets with the function approximation provided by deep neural networks. Unlike tabular CFR it does not require a hand crafted game abstraction and, as such, learns through self-play.

For each iteration, Deep CFR performs a constant number of partial traversals according to Monte-Carlo CFR [16]. At each infoset it plays its current strategy, which was determined by regret matching the output of the neural network. This neural network takes in information sets as input and has the goal of approximating the regret that tabular CFR would have produced. Like regular CFR, when a terminal node is reached, values are propagated back up the tree.

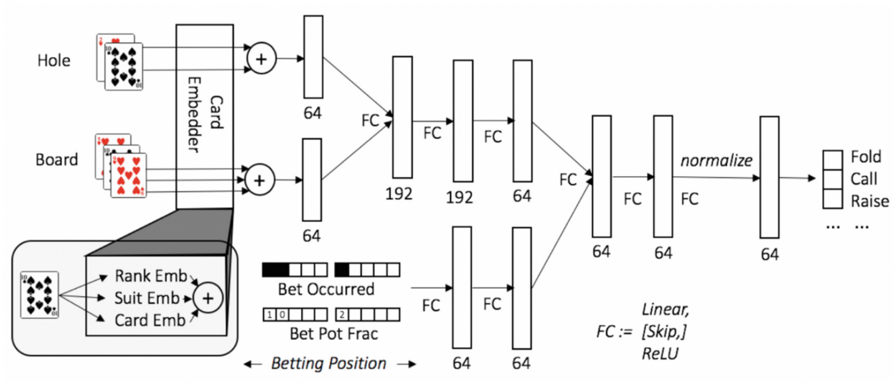


Fig. 1. Neural network architecture of Deep CFR as presented in [8]

These instantaneous regrets are sampled and stored in memory. Then, before the next iteration, a completely new network is trained to minimise the error between the predicted regrets and the samples of regret that have been stored in memory. Once this training is complete, the next iteration can begin.

Despite scaling better than tabular CFR, DCFR is not perfectly scaleable either. The sampling strategies used are simplistic, and introducing more sophisticated methods would likely result in high variance between sampled payoffs. Extensions to Deep CFR such as Variance Reduction Monte-Carlo CFR (VR-MCCFR) [20] and Double Neural CFR (DNCFR) [18] represent the state of the art in solving massive imperfect information games.

Specifically, VR-MCCFR takes the per-iteration estimated value updates of a MCTS and reformulates them as a function of sampled values and state-action baselines whilst still being unbiased. It should be noted that plain Monte-Carlo CFR (also known as chance sampled CFR) was the precursor to this method, but it was only applied to small games and struggled to compete with tabular agents [16]. A visual representation of the difference between MCCFR and VR-MCCFR can be seen in Fig. 2

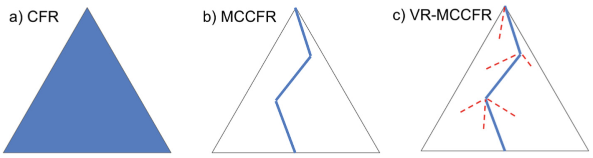


Fig. 2. The tree traversal of VR-MCCFR as compared to regular MCCFR and normal CFR as presented in [20]

VR-MCCFR was not investigated for implementation in this project due to its inability to be accelerated by GPU compute (unlike DCFR and DNCFR), meaning significantly more computational resources would be required for similar results. Similarly, DNCFR was not investigated due to a lack of open source reference implementations.

1.3 Game Selection

Imperfect information games are a harder class of games than perfect information games due to the number of possible game states growing exponentially due to uncertainty. Trading card games (TCG) are a more difficult class of game because there is not only non-determinism in game state, but also uncertainty in card interactions.Furthermore, instead of being completely turn-based games like Poker, card interactions between players can happen on either player's turn. Specifically, many TCG cards have a logical description of the effect they have on the game state, when this can be applied, and any uncertainty or random conditions that need to be met to carry it out. Being significantly harder than Poker, TCG's could provide an environment to test more powerful imperfect information solving methods. An example of a TCG is Yu-Gi-Oh. From Table 1 not only does Yu-Gi-Oh have a much larger card pool but also a significantly larger number of possible actions.

Table 1. Comparison of Yu-Gi-Oh and Poker

Property	Yu-Gi-Oh	Poker
Move types	20	6
Players	2	2
Multi-interaction	✓	✗
Deck size	40–60	52
Card pool	11,892	52

Even compared to other TGC's Yu-Gi-Oh presents a few unique advantages, such as not having mechanics to re-snuffle the starting hand at the beginning of the game, a limited field size, and generally requires more card interactions overall. This means any agent developed does not have to require a hand evaluation system at the start of the game, can have its state represented efficiently, and can learn common patterns of card interaction more easily.

A Brief Description of Yu-Gi-Oh. A player wins a game of Yu-Gi-Oh by reducing their opponent's life points to zero. Both players start with 8000 and they can be reduced either by attacks from an opponent's *Monster* or card effects. A *Monster* card is one of the three main types of Yu-Gi-Oh cards. It can be placed on the field, termed "summoning" by a player on their turn given certain conditions are met. A simple analogy would be Chess pieces, where a *Monster* is a particular piece and a square on the Chess board is a *Monster* zone on the field. There are also *Spell* and *Trap* cards which cannot directly harm an opponent but do influence the state of the game. These cards are stored in a *Deck* which can be between 40–60 cards of the players' choice.

1.4 Similar Work

Most TCGs have large player bases, and some even have international competitions. Such competitions are usually held in person with physical cards. Additionally, there is a dedicated AI competition in the case of Hearthstone [12], but methods so far have focused on perfect information MCTS. This is also the case for Magic the Gathering [25], where even ensemble MCTS tree search methods showed poor results [11]. This is because MCTS, even with information sets struggles to adequately address the inherent imperfect information nature of the games. Some neural network methods have been attempted but have had poor results [13]. At the time of writing there are no works investigating the application of game solving methods to Yu-Gi-Oh.

1.5 Meta-strategies

The terms Meta-game or Meta-strategy have different interpretations depending on context, but from the perspective of Yu-Gi-Oh the so called "meta" is the specific decks that are the best or most successful. One of the most prudent examples of "meta" is that of the 2013 or "Dragon Ruler" format in Yu-Gi-Oh where 95% of all tournament wins and top positions were taken out by two decks, Dragon Rulers and Spellbooks. Furthermore, the world championships of 2013 were comprised entirely of those two decks [5]. Playing any other deck at the time put a player at a serious disadvantage.

Deciding on a good meta-strategy is a reflection of a player's skill and is not something that is directly addressed by modern card game AIs. Often when playing an AI player their meta-strategy has been pre-determined and does not change, such is the case with AIs provided with community Yu-Gi-Oh simulators and the official Yu-Gi-Oh online games. Because MCTS, CFR and DCFR agents all learn through some form of self-play and attempt to learn the optimal strategy (or policy) given the deck they have, it should be possible to give one of them different decks, train the same agent against itself, and use the results to draw conclusions about the relative performance of those decks.

2 Experimental Design

Being different to almost all games traditionally studied in game theory and AI generally, the implementation of Yu-Gi-Oh for performing experiments will both be distinct and more complicated. There are relatively few digital Yu-Gi-Oh environments, and none that have been used in the context of AI. To play

Yu-Gi-Oh online, players can either purchase the first party app [17] but must unlock cards and cannot play freely with other players. They can go to the website [1] where they are free to choose cards but must still play and perform all the card interactions manually. The final option is a free and open source simulator, EdoPro (formerly YGoPro) which is a C++ game engine that uses Lua scripts to represent card logic. Being originally designed to function as a game server it provides a reasonable API from which game state and available actions can be captured, thus making it a suitable platform to build and train an agent. Being a community built technology means it has the advantage of being regularly updated and remarkably complete in comparison to the latest release of the game. It is also quite fast, supporting Lua scripting for describing card logic.

The overall simulator used for experiments in this project used parts of the core game engine available [3] and a selection of card scripts from [2] as a base. Parts of it were re-written and other parts were added to make tree searching more practical with what was originally a completely linear state machine. On top of this base, a Python abstraction layer was built and linked to the associated algorithms. This Python layer also allowed for parallelisation across different kinds of compute resources. Considering that the successful agents for Poker ran on a supercomputer [9], to be able to achieve any reasonable results, a reduced game was considered. The rule set, card pool and banned card list were all restricted to the original release of the game. Furthermore, every agent used the same pre-constructed deck. This version of the game still captures the complex interactions and vast card pool without making the game overly complicated or too large.

The following agents were implemented:

- **Heuristic agent**
  The EdoPro simulator [3] provides some built-in AIs that are all heuristic agents hard-coded to respond to certain combinations of cards. For example, always attack the weakest monster, always set trap cards in main phase two, and if the opponent's monster is more powerful, set your own monsters to defence position.
- **ISMCTS Agent**
  A simple information set MCTS agent with the UCB [14] tree selection policy
- **Plain CFR agent**
  A custom game abstraction was implemented and the different phases of a Yu-Gi-Oh turn were divided into buckets
- **Deep CFR agent**
  A deep regret matching network trained through partial iterations of MCCFR [16]

For all experiments, a single duel setting was decided upon (where two players play until one wins) as opposed to a match (best two out of 3 duels) to alleviate the need for side decking and to simplify numerical analysis. The agents played 500 duels, and they both played with the same set of cards. They also started with 8000 life-points, had a starting hand of 5 cards, drew one card per turn, and played under no ban list (as was the case for the Original Yu-Gi-oh release). This round of 500 games was repeated three times for each pair of agents to reduce variance further.

Constraints. To save computational resources, for the implementation of all tree searching methods, the following additional constraints were placed on the state of the game to save computational resources:

- What cards were in the graveyard and what order they were in was not recorded.
- The cards that were in both players' extra decks were not recorded.
- Only cards on the field were recorded not the specific placement or ordering and the same for the hands of both players

Multiple instances of each agent were used for training but each referenced the game memory (the game tree in the case of ISMCTS and counterfactual memory for the CFR methods). In the case of ISMCTS random simulations were limited to 100 actions before the evaluation function was applied.

Evaluation Function. Given the sometimes immense length of Yu-Gi-Oh games, waiting until a terminal state in the roll out stage of ISMCTS and CFR would lead to poor performance. As such, the simulated games were cut off and an evaluation function was applied, which is meant to approximate the overall value or result of that state.

The following function was used:

$$v(s) = 1.5 * (cc - oc) + 2 * mf + \frac{cl}{ol}$$

where $cc \in [0, 20]$ is the number of cards the agent controls $oc \in [0, 20]$ is the number of cards the opponent controls $cl \in [0, 8000]$ is the agent's life points, $ol \in [0, 8000]$ is the opponents' life points and $mf \in [0, 5]$ is the number of Monsters the current player has on the field.

This function was chosen based on experience and preliminary testing. It seemed to capture the three main aspects of Yu-Gi-Oh that led to an overall advantage:

- Having more life points than the opponent.
- Having Monsters to attack the opponent with.
- Having more cards than the opponent to play with.

Time constraints limited empirical determination of good values for the weighting of various factors, so crude values were chosen based on experience with the game.

Statistical Significance. Independent sample t-tests were performed to compare two agents in each of the three experiment runs.

Meta-strategy Evaluation. A way of assessing whether CFR methods can evaluate decks at a high level would be to compare them to human evaluations of decks. A common notion among the Yu-Gi-Oh player base is tiers, where if a deck is in tier 1, it is one of the best and is expected to win most major events. If a deck is in tier 3, it is not expected to win much, but is still competitive. In the early history of Yu-Gi-Oh there were no real archetypes (groups of cards that followed a theme and worked well together), so there were only a few popular decks that people played with subtle variations from player to player. Because of this, and the fact that records of top performances during the early 2000s are difficult to find, decks from that time period cannot be used.

The first format to have reasonable records and well defined tiers is that of early 2011. Table 2 shows which decks were selected after compiling popular decks from the Pojo community forums [23] and what tier they are.

Table 2. Decks chosen for Meta-Strategy evaluation and their relative performance

Deck	Tier
Agents	1
Tengu-Plants	1
GraveKeepers	2
Six Samurai	2
Worms	3
Gem-Knights	3

A DCFR agent was trained for each deck. Training consisted of instances of the six agents playing against a random opponent (who was one of the six decks) for a period of four days. They were then placed in a tournament scenario which closely resembles real-life Yu-Gi-Oh tournaments.

3 Results

A summary of the overall win percentages of playing various agents against each other can be seen in Table 3.

3.1 Baseline

Overall the Heuristic Agent beat the random Agent 75% of the games. As was expected, reasonable heuristics crafted by an expert player easily outperformed random play. The difference between the agents was significant; t (4) = 10.885, $p < 0.001$. This experiment provided a baseline for the examination of the other agents (Table 4).

Table 3. Overall head to head win percentage of three 500 game matches between different agents.

Deck	Random	Heuristic	ISMCTS	CFR	DCFR
Random	50%	30%	45%	39%	22%
Heuristic	70%	50%	70%	61%	47%
ISMCTS	55%	30%	50%	42%	35%
CFR	61%	39%	65%	50%	37%
DCFR	78%	53%	65%	63%	50%

Table 4. Number of wins, mean and variance of three 500 game matches between the Custom Heuristic Agent and a random Agent

Agent	Round 1	Round 2	Round 3	μ	σ
Random	125	170	130	142	24
Heuristic	375	330	350	352	22

These results are likely due to the fact that heuristics are effective at playing most scenarios with simple decks and most of the actions available do not involve the more complex game mechanics that would require more detailed heuristics.

3.2 Existing Methods

The ISMCTS Agent won 55% of the games when playing the Random Agent (Table 5). The differences between ISMCTS, Random and Heuristic agents were all statistically significant. It fared significantly worse against the Heuristic Agent only winning around 30% of the games played on average. Whilst the ISMCTS Agent was slightly better than random play, it was no match for the expert heuristics, indicating raw MCTS methods are not a good fit for solving Yu-Gi-Oh. The CFR Agent won 61% of the games against the Random Agent (Table 5). However, it was also easily beaten by the Heuristic Agent, only winning 37% of the games. The differences between CFR, Random and Heuristic Agents were all significant. These results indicate that the CFR Agent performs better than the ISMCTS Agent in both random and heuristic scenarios. Therefore, it is still not an ideal fit for Yu-Gi-Oh, especially considering the large training resources required.

Table 5. Number of wins, mean, variance and t score of three 500 game matches between the ISMCTS Agent, CFR Agent, Heuristic Agent and a Random Agent

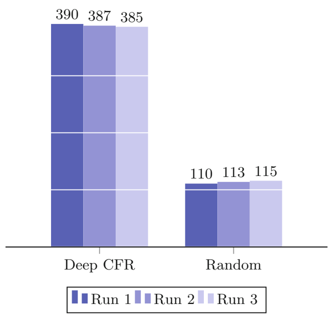
Match	Round 1	Round 2	Round 3	μ	σ	t ($p < 0.001$)
ISMCTS/Random	275/225	270/230	280/220	275/225	5	12.247
ISMCTS/Heuristic	150/350	141/359	152/348	148/352	6	−42.779
ISMCTS/CFR	175/325	173/357	177/323	175/325	2	13.272
CFR/Random	305/195	306/194	300/200	304/196	3.2	40.894
CFR/Heuristic	200/300	159/341	190/310	183/317	21	−9.9625

3.3 Deep Counterfactual Regret Minimisation Agent

Deep CFR performed the best out of all the agents, winning 53% of games against the Heuristic Agent and 78% against the Random Agent (Fig. 3). The difference between Deep CFR and the Random agent was significant. However, the narrow difference between Deep CFR and the Heuristic Agent was not statistically significant; t (4) = 2.746, $p < 0.052$ (Table 6).

Table 6. Number of wins, mean and variance of three 500 game matches between the Deep CFR Agent, the Heuristic Agent and a Random Agent

Match	Round 1	Round 2	Round 3	μ	σ	t ($p < 0.001$)
DCFR/Random	390/110	387/113	385/115	387/112	2	133.670
DCFR/Heuristic	265/235	250/250	270/230	262/238	10	N/A



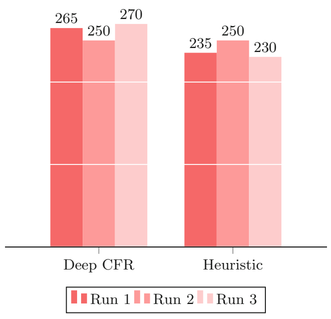


Fig. 3. Visualisation of the number of wins of three 500 game matches between the Deep CFR agent, the Heuristic Agent and a Random Agent

Given this lack of significance, a further test run of three 1000 game matches between the same DCFR and Heuristic agents was run (See Table 7). This run was statistically significant with t $(4) = 0.0154$, $p < 0.001$ confirming the superior performance of the DCFR agent.

Table 7. Number of wins, mean and variance of three 1000 game matches between the Deep CFR Agent and the Heuristic Agent

Agent	Round 1	Round 2	Round 3	μ	σ
DCFR	565	542	560	555	12.1
Heuristic	435	458	440	444	12.1

3.4 Meta-strategy Evaluation

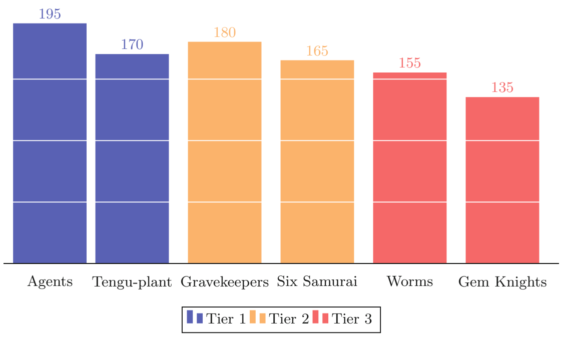


Fig. 4. Number of tournament wins of various decks

The most dominant deck, Agents, won the most games by a reasonable margin (See Fig. 4). However, the tier two deck Gravekeepers came in second place, followed by the other tier one deck, Tengu-Plant. Whilst this slightly matches the trend indicated by Table 2, the tier 1 decks are not as dominant and the tier 3 decks are not completely overpowered.

4 Discussion

4.1 Game Abstractions

The game abstraction used for the CFR agent was both simplistic and small. Looking at some of the Duel replays, it is apparent it hindered the agents' performance in some areas. For example, when the opponent had a powerful Monster like the CFR player would attempt to play a card that only destroys spell cards, thinking it was one that destroyed all cards, leading to a worse position. Furthermore, as the general preprocessing was hand crafted, it is not applicable to other Decks making the agents less general.

Future experiments should incorporate larger game abstractions such as the hierarchical methods outlined in [7] and post-processing methods.

4.2 Agent Results

According to the results against the Heuristic agent, Deep CFR is the most scalable and best performing agent. There are a few factors that were responsible for this success. The first is how it uses the computational resources. Both MCCFR and CFR were limited to using only the CPU for calculations and storage of regret values, whilst the Deep CFR agent was able to make use of 2 GPU's for training its neural networks and some storage, putting it at a significant advantage. This is especially the case when compared to the MCTS agent, which ran out of memory during training and struggled to complete enough iterations to become competitive in terms of win rate. As Yu-Gi-Oh cannot easily be represented as a vector with reasonable memory requirements and hence a matrix game, being able to scale some component of the system to better hardware could be considered a desirable quality in the case of Deep CFR.

In the case of the CFR agent, having a more efficient variant such as CFR+ or Linear CFR could have led to much better results. However, it is likely that both a sampling strategy and advanced abstraction will be required for the best results. These will help alleviate memory issues as the standard CFR agent ran out of memory on multiple occasions and had to cut nodes from the game tree in some cases. Overall, despite being the fastest computationally, CFR does not provide a path to scale to the full game.

A small number of games were played between the Deep CFR Agent and a human player. The Deep CFR Agent was able to win some of these games and made relatively few obvious mistakes, which, whilst expected due to the stochastic nature of Yu-Gi-Oh and the simplified game, is a promising indicator that the agent had achieved a level of human-like play. Future experiments should aim to play more games against humans, perhaps making use of the online facilities provided by EdoPro [3], to ascertain how competitive Deep CFR and agents like it are.

4.3 Meta-strategy Evaluation

Whilst the results are promising for meta-strategy evaluation, there are a few issues that likely led to less than perfect results. The first is training the DCFR agent on much more complicated decks. In the simplified game, the agents hardly ever had to deal with chained effects or extra deck Monsters, yet for the 2011 decks, the usage of extra deck Monsters was critical. This partially explains why the Tengu-Plant deck did not perform as well as expected. Being a strategy that heavily revolves around the extra deck and requires complicated card combinations to be successful, it was probably too much for the Deep CFR agent to learn perfectly. This could also explain why the Gravekeeper deck did better than expected, coming in second place, as its strategy revolves more around controlling the field and playing as few cards as possible. Seeing that the Deep CFR agent prefers more simple strategies, it raises the question of what improvements could be made algorithmically. A possible solution could be focusing on expanding more of the game tree related to the current player's turn as opposed to the opponent's move to encourage exploration of combination moves on a single turn. Despite its shortcomings, the fact that Tengu-Plants came in 3rd place is an indication that Deep CFR can learn complex strategies, and if improvements to the algorithm were made, or if more computational resources were applied, it is likely that it would perform even better.

5 Conclusion

Current CFR methods are almost universally tested using the game of Poker which, whilst providing a complex stochastic imperfect information environment, does not capture complex logic interactions between cards or players. Yu-Gi-Oh, in contrast, is in a class of harder games where card interactions can be stochastic, conditional, or temporally inter-dependent at the same time, and player interaction incredibly situational. These properties make Yu-Gi-Oh a better representation of real world strategic interactions and more apt at addressing modern challenges in AI, such as complex logic, massive state space and hidden information, than Poker. To test Yu-Gi-Oh using CFR, within the bounds of modern computational power, a slightly simplified version of the game that still captures the logic and state space requirements was constructed.

Of the methods tested, Deep CFR and a simple Heuristic Agent performed the best. This indicates that techniques such as MCTS and tabular CFR with custom abstractions are not well suited to address the amount of hidden information Yu-Gi-Oh presents and that even custom abstractions do not capture the relationships between information sets well. Also of concern is that, with the exception of the Heuristic Agent all methods struggled under the CPU, memory, and disk resource limitations of the experimental environment. Future work should look at scaling the experiments to larger computational resources to investigate if MCTS and CFR can perform better under such conditions, especially in the case of sampling variants such as CFR+ and VR-MCCFR.

In the case of using CFR as a way of evaluating meta-strategy, the results are positive but inconclusive. Using Deep CFR as the evaluation system results in similar trends as in real tournament play. Considering that the decks and rules used in those experiments were far more advanced than the simple deck and that each agent had to learn how to play against multiple meta-strategies, the fact that the best deck in the format came out on top is promising. Future work, similarly to the evaluation of CFR methods, should look at what the algorithms do with more training time but also at how different algorithms learn to play different meta-strategies or combinations of them. Furthermore, advanced variants such as Single Deep CFR and Double Neural Deep CFR should be considered for their better computational performance and ability to utilise computational accelerators.

Overall, CFR methods appear to be able to handle the demands of larger and more complicated games. They can produce competitive results when compared to a tuned domain-specific agent by learning similar general behaviours and, if given more resources, would likely outperform them. Furthermore, CFR methods appear to be a promising tool for investigating the construction and evaluation of meta-strategies and, with future research, could lead to intelligent systems that are both able to calculate what resources are required to solve a problem as well as how to best use them when doing so. Such systems do not currently exist for imperfect information contexts, but if they did, they could be revolutionary for business and military strategy, negotiation interactions, and complex planning problems.

References

1. Dueling book. https://www.duelingbook.com/
2. Project ignis card scripts for edopro. https://github.com/ProjectIgnis/CardScripts
3. Project ignis: Edopro. https://github.com/ProjectIgnis/EDOPro
4. Best-first minimax search: Artif. Intell. **84**(1), 299–337 (1996). https://doi.org/10.1016/0004-3702(95)00096-8
5. Akira: Yu-gi-oh! world championship 2013. https://roadoftheking.com/yu-gi-oh-world-championship-2013/
6. Bowling, M., Burch, N., Johanson, M., Tammelin, O.: Heads-up limit Hold'em poker is solved. Science **347**(6218), 145–149 (2015)
7. Brown, N., Ganzfried, S., Sandholm, T.: Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas Hold'em agent. In: Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
8. Brown, N., Lerer, A., Gross, S., Sandholm, T.: Deep counterfactual regret minimization. CoRR abs/1811.00164 (2018). http://arxiv.org/abs/1811.00164
9. Brown, N., Sandholm, T.: Superhuman AI for heads-up no-limit poker: libratus beats top professionals. Science **359**(6374), 418–424 (2018). https://doi.org/10.1126/science.aao1733, https://science.sciencemag.org/content/359/6374/418
10. Browne, C., et al.: A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–43 (03 2012). https://doi.org/10.1109/TCIAIG.2012.2186810
11. Cowling, P.I., Ward, C.D., Powley, E.J.: Ensemble determinization in Monte Carlo tree search for the imperfect information card game magic: the gathering. IEEE Trans. Comput. Intell. AI Games **4**(4), 241–257 (2012)
12. Dockhorn, A., Mostaghim, S.: Introducing the hearthstone-AI competition. arXiv preprint arXiv:1906.04238 (2019)
13. Grad, Ł.: Helping AI to play hearthstone using neural networks. In: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 131–134 (2017). https://doi.org/10.15439/2017F561
14. James, S., Konidaris, G., Rosman, B.: An analysis of Monte Carlo tree search. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
15. Blar, J., Mutchler, D., Liu, C.: Games with imperfect information (1993)
16. Johanson, M., Bard, N., Lanctot, M., Gibson, R.G., Bowling, M.: Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In: AAMAS, pp. 837–846. Citeseer (2012)

17. Konami: Yugioh duel links. https://www.konami.com/yugioh/duel_links/en/
18. Li, H., Hu, K., Zhang, S., Qi, Y., Song, L.: Double neural counterfactual regret minimization. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=ByedzkrKvH
19. Matros, A.: Lloyd shapley and chess with imperfect information. Games Econ. Behav. **108**, 600–613 (2018). https://doi.org/10.1016/j.geb.2017.12.003, https://www.sciencedirect.com/science/article/pii/S0899825617302221, special Issue in Honor of Lloyd Shapley: Seven Topics in Game Theory
20. Schmid, M., Burch, N., Lanctot, M., Moravcik, M., Kadlec, R., Bowling, M.: Variance reduction in Monte Carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2157–2164 (2019)
21. Silver, D., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm (2017)
22. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
23. SinL0rtuen: New format's top tiers - let's make a list together ii. https://www.pojo.biz/board/showthread.php?t=991471
24. Syrgkanis, V., Agarwal, A., Luo, H., Schapire, R.E.: Fast convergence of regularized learning in games. arXiv preprint arXiv:1507.00407 (2015)
25. Ward, C.D., Cowling, P.I.: Monte Carlo search applied to card selection in magic: the gathering. In: 2009 IEEE Symposium on Computational Intelligence and Games, pp. 9–16 (2009). https://doi.org/10.1109/CIG.2009.5286501
26. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1729–1736 (2007)