# Improve Counterfactual Regret Minimization Agents Training by Setting Limitations of Numbers of Steps in Games

Cheng Yi[1,a)] Tomoyuki Kaneko[1,b)]

***Abstract:***
Counterfactual Regret Minimization (CFR) has been one of the most famous algorithms to learn decent strategies of imperfect information games. Because CFR requires traversing the whole or part of game tree every iteration, it is infeasible to handle games with repetition where the game tree is not finite. In this paper, we introduce two abstraction techniques, one of which is to make the game tree finite and the other one is to reduce the size of game trees. Our experiments are conducted in an imperfect information card game called Cheat and we introduce the notion of "Health Points" a player has in each game to make the game length finite thus easier to handle. We utilize the information sets abstraction technique to speedup the training and evaluate how results from smaller games can improve training in larger ones. We also show Ordered Abstraction can help us increase the learning efficiency of specific agents.

***Keywords:*** Imperfect Information Games, Counterfactual Regret Minimization, Abstraction technique, Curriculum Learning, Card Game Cheat

## 1. Introduction

In the Artificial Intelligence research area, we often see games as our challenging problems and solving them represents the research benchmarks and breakthroughs. There are two kinds of games, perfect information games and imperfect information games. In imperfect information games, such as bridge, Mahjong and most poker games, players do not know everything about their opponents. The hidden information of the play is what makes imperfect information games more challenging. One of the most important concepts in Game Theory is called *Nash Equilibrium.* A Nash Equilibrium is a strategy profile where no player can achieve a better result through converting their strategy unilaterally. The goal of most of the researches in this area is to approximate or reach the Nash equilibrium.

*Counterfactual Regret Minimization* (CFR) has lately become one of the most famous and widely-used algorithms when dealing with imperfect information games. The main idea is to converge to a Nash Equilibrium based on the counterfactual regret calculation of every state on the game tree for the players. One of the shortages of this algorithm is that Vanilla CFR requires a traversal of the whole game tree on each iteration as it becomes infeasible when we are facing extremely large or infinite games. As a result, researchers have been looking for a better way to save computing costs in order to deal with these conditions.

In this paper, we introduce a new approach to create and adjust the training environment of CFR agents to serve our purpose. We limit the number of total steps thus the total length of the game for simplification and aim at using results from simpler games in the larger games to speed up the learning and achieve a better result. We also use another abstraction technique to help us save time and space costs. In the next chapter we will talk about some background knowledge, then Chapter 3 will cover some of the related previous works. Chapters 4 and 5 talk about our proposed methods, details of experiment conduction and the results. In the last chapter, we summarize the whole paper and claim our expectation of the future direction of our research.

## 2. Background

In this chapter, we will introduce the background knowledge which is important in our study, including the notation and terminology of extensive-form games and Nash Equilibrium, the basic rules of the card game Cheat and the main algorithm of our experiments, Counterfactual Regret Minimization.

### 2.1 Extensive-form Games

We followed a standard notation in game theory [5]. A finite extensive game with imperfect information is composed of the following elements:

- A finite-size set of *players*, $\mathcal{P}$. For player $i$, $-i$ rep-

[1] Graduate School of Arts and Sciences, the University of Tokyo
[a)] yi-cheng199@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org
A part of this work was supported by JSPS KAKENHI 21H03570.

resents all the players other than $i$. There is also a Chance player $c$, representing the actions that are not controlled by any player. In this paper, we only focus on two-player games. Therefore, $i$ is either 1 or 2, and $-1$ $(-2)$ is 2 (1).

- A *history* $h \in \mathcal{H}$ is a node on the game tree, made up of all the information at that exact game state. A *terminal history* $z \in \mathcal{Z} \subseteq \mathcal{H}$ is where there are no more available actions and each player will get a payoff value for what they have done following the game tree respectively.
- We use $A$ to denote the *action space* of the whole game and $A(h)$ is the set of all the *legal actions* for players at the history $h$. If history $h'$ is reached after a player chooses action $a \in A(h)$ at history $h$, we can write $h \cdot a = h'$ or $h \sqsubseteq h'$.
- An *information set* (infoset) is a set of histories that for a particular player, they cannot distinguish which history they are in between one another. $\mathcal{I}_i$ represents the finite set of all the infosets for player $i$. Inherently, $\forall\, h, h' \in I, A(h) = A(h') = A(I)$. Note that any history $h \in \mathcal{H}$ must belong to exactly one of the infosets.
- For each player $i \in \mathcal{P}$, there is a payoff function $u_i : \mathcal{Z} \to \mathcal{R}$ and especially in two-player zero-sum games, $u_1 = -u_2$.

In a game, a *strategy* for player $i$ is $\sigma_i$ which assigns a distribution over their action space to each infoset of player $i$, particularly, $\sigma_i^t(I, a)$ for player $i$ maps the infoset $I$ and the action $a \in A(I)$ to the probability that player $i$ will exactly choose action $a$ in the infoset $I$ on iteration $t$. $\Sigma_i$ denotes the set of all strategies of player $i$. A strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ is a tuple of all the players' strategies with one entry for each player where $\sigma_{-i}$ represents the strategies in $\sigma$ except $\sigma_i$. Let $\pi^\sigma(h)$ denote the *reach probability* of reaching the game history $h$ while all the players follow the strategy profile $\sigma$. The contributions of player $i$ and all the players other than $i$ to this probability are denoted by $\pi_i^\sigma(h)$ and $\pi_{-i}^\sigma(h)$ respectively.

### 2.2 Nash Equilibrium

One of the most important concepts in Game Theory is Nash Equilibrium. A *Nash equilibrium* (NE) is a strategy profile where no player can achieve a better result through converting their strategy unilaterally. It means that when one is playing following the NE strategy, they can be seen as "no lose". If we can find the exact NE of a game, then we can say the game is strongly solved. Let $(\Sigma, u)$ be a game with $n$ players, where $\Sigma = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_n$ is the set of strategy profiles and $u(\sigma) = (u_1(\sigma), \ldots, u_n(\sigma))$ is its payoff function defined over $\sigma \in \Sigma$. So Nash equilibrium can be now expressed as a strategy profile $\sigma^*$, in which every player is playing the best response. Formally, a strategy profile $\sigma^* \in \Sigma$ is a *Nash Equilibrium* if,

$$\forall i, \sigma_i \in \Sigma_i : u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*) \qquad (1)$$

### 2.3 The Game *Cheat*

*Cheat* is a card game of lying and bluffing while also detecting opponents' deception. This game is often played among three or more players. At the beginning of the game, all the cards are well shuffled and dealt to the players as equally as possible. There are two phases in one turn of this game, the *Discard* phase and the *Challenge* phase. The first player to discard is chosen randomly and the "current rank" which all the players share is set to be Ace.

In the *Discard* phase, the discard player discards card(s), puts them facing down on the table and makes a claim, including the number of cards they just discarded and the current rank. Players are supposed to discard cards only of the current rank but they can lie about their cards - either bluffing it out when they do not hold any correct cards or choosing other cards even if they have the correct ones. Then in the *Challenge* phase, if any other player thinks the discard player is lying, they can challenge them by saying "Cheat!". When there is a challenge, the last discarded card(s) will be revealed to all players to see whether they are consistent with the claim. If the accused player did lie then they must take all the cards on the table back to their hands, otherwise, the challenger takes the pile. If no one challenges, the card(s) remain(s) in the pile.

After the challenge phase, we move to the discard phase in the next turn. The current rank increases by one (K is followed by Ace) and the player sitting right to the former discard player is then supposed to discard card(s). The one who first discards all the cards from their hand and survives the last challenge wins the game.

### 2.4 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) was first proposed in 2008 by Zinkevich et al. in the study [6] where the idea that claims minimizing overall regret can be used for approximating a Nash equilibrium in extensive games with incomplete information was demonstrated and proved. The basic steps of one iteration of Vanilla CFR are the following: first, it keeps a record of the regret values, $R^t(I, a)$, for all actions $a \in A(I)$ (all zeros at the beginning) in each infoset $I \in \mathcal{I}_i$ where $t$ denotes iteration; second, the values are used to generate strategies, s.t., $\sigma^{t+1}(I, a) \propto \max(R^t(I, a), 0)$; third, the regret values are updated based on the new strategies. After all iterations, the average strategy $\bar{\sigma}(I, a) = \sum_t \pi^{\sigma_{-i}^t} \sigma^t(I, a)$ obtained by normalizing overall actions belonging to the action space of this infoset, weighted by counterfactual reach probability, is proved to converge to the best strategy as time tends to infinity.

Vanilla CFR requires traversals of the whole game tree in every iteration. The game length of the original Cheat (one deck of poker cards played between 2 players) is possibly infinite and the game still has about $10^{120}$ decision points even we ignore the possibility of repetitions, so traversing the entire game tree even once is impossible and the computation is beyond the calculation power of ordinary computers. Another variant called Chance-sampled CFR (CS-CFR) is more common in practice, especially when dealing with poker or cards games. We see the results of dealing cards as Chance player's actions, and on each iteration, we only sample the action of the Chance player.

## 3. Related Works

There have been many methods to help us tackle large games. For example, the Blueprint strategy is introduced in Libratus [3] and then improved in Pluribus [4]. First, an abstraction of the whole game is defined and the solution to this abstraction is called Blueprint strategy. This strategy only has specific details for the early stage of the game and an approximation for later parts. The approximation will then be refined at the runtime of the game and after the agent gets to know more about the opponents' actions.

Most of the technique helps us save time and space for the whole game but remains unchanged at the early stage of the game. In 2015, Brown et al. first propounded an algorithm called *simultaneous abstraction and equilibrium finding (SAEF)* [1] which does not rely on any domain knowledge but is only applicable in specific conditions. Then in 2016, a refined version of SAEF called *Strategy-Based Warm Starting* was introduced in the study [2]. The new method expands the power of SAEF and is capable to skip the early expensive iterations of the game. Although warm starting and our curriculum learning have some similarities, our method is simpler because both initial strategy and regret are transferred while warm starting involves a sophisticated procedure to recover substitute regret from a given strategy.

## 4. Proposed Methods

### 4.1 Limitations on Numbers of Steps in Games

The rule states that cards are discarded and taken back during the game and it might lead to repetitions of game states and thus infinite game lengths, which is one of the difficulties we have to overcome when dealing with this game. We will now introduce our main contribution of this paper called Ordered Abstraction.

#### 4.1.1 Ordered Abstraction for training

To handle a subset of infinite games with CFR, we present Ordered Abstraction. The basic idea is to make a finite variant of an original game by introducing a condition to terminate the game in a finite number of steps. Then, we run CFR to obtain a strategy in this finite variant with the abstraction. We hope that the learned strategy would also work well in the original game, but it crucially depends on the design of the abstraction. To remedy such difficulties, we present an effective heuristic of a curriculum learning with an abstraction with numbering. We design our curriculum learning as follows:

(1) design a finite variant, $G_n$, of a game, associated with integer $n$ such that
- a variant with a smaller $n$ is easier thus a stronger restriction (i.e., having a shorter game length and a smaller subset of infosets), and it asymptotically recovers the original game as $n \to \infty$.
  We assume that for all $n < n'$, $\mathcal{H}^{G_n} \subseteq \mathcal{H}^{G_{n'}}$ and $|\mathcal{I}_i^{G_n}| \leq |\mathcal{I}_i^{G_{n'}}|$ for each player $i \in \mathcal{P}$ and that any non-terminal history is also non-terminal in a larger game, $((\mathcal{H}^{G_n} \setminus \mathcal{Z}^{G_n}) \cap \mathcal{Z}^{G_{n'}}) = \emptyset$. Usually, there are some histories that are terminal in $G_n$ and non-terminal in $G_{n'}$ to make variant $G_n$ strictly smaller. We use superscript $X^{G_n}$ to denote property $X$ in variant $G_n$.
- each infoset for a variant with $n+1$ is included in exactly one infoset with $n$. Note that the inclusion is well-defined because an infoset is defined as a set of histories. That is, for all $n > 0$, for all $I \in \mathcal{I}_i^{G_{n+1}}$ there exists unique $I' \in \mathcal{I}_i^{G_n}$ such that $I = I'$.

(2) run CFR $T$ iterations in the easiest variant, $G_1$, to obtain a decent strategy profile $\bar{\sigma}^{t=T,G_1}(I,\cdot)$ and regrets $R^{t=T,G_1}(I,\cdot)$ for each infoset $I$,

(3) run CFR with variant $G_n$ after completing CFR with variant $G_{n-1}$, initializing the strategy as well as regret for each infoset by using the results obtained for variant $G_{n-1}$ to speed up learning, i.e., $\sigma^{t=1,G_n}(I,a) \leftarrow \bar{\sigma}^{t=T,G_{n-1}}(I',a)$ and $R^{t=1,G_n}(I,a) \leftarrow R^{t=T,G_{n-1}}(I',a)$ where $I = I'$ for $I \in \mathcal{I}_i^{G_n}$ and $I' \in \mathcal{I}_i^{G_{n-1}}$.

Because CFR with a sequence of variants, $(G_1, G_2, \ldots)$ is enhanced by the initialization using the former results in step 3, we call our method a curriculum learning. A primary advantage of the ordered approach is in iterative improvement. Usually, we cannot expect how well a strategy learned for $G_i$ behaves in the original game before any enhancement. Therefore, it is effective to start the smallest variant $G_1$, gradually improve the strategy along with a larger $G_n$, and stop once a sufficient variant is obtained.

#### 4.1.2 Application to Cheat

We explain an example of our method in the application to Cheat.

By analyzing the game rule, we can see that in order to win the game, we want to not only keep as few cards as we can in our hand, but also win more challenges. In other words, we want to challenge when we are more confident and discard cards more cleverly. Based on this thought, we bring "Health Point" (HP) into this game.

In the original game, there is no restriction on how many times a player can lose challenges as long as no one discards all their cards. Now suppose each player has $n$ HP, which means they only has $n$ chances to lose the challenge. More specifically, when HP equals 1, it means if a player loses in the challenge once, the HP becomes 0 thus they loses the entire game (even if their opponent has not discarded all the cards). We call Cheat with $n$ HP, Cheat-$n$. Through this, we introduce a sequence of smaller variants of Cheat, Cheat-$k$, where Cheat-1 is the smallest and Cheat-$\infty$ equals the original Cheat.

By limiting HP, we created a technique of the Ordered

Abstraction. We propose to compute a smaller and easier version of the game, solve this game then map the strategies into a larger game, i.e. sequentially solve Cheat-1, Cheat-2, ..., to get the strategy for the original Cheat, Cheat-$\infty$.

When evaluating the playing performance of a strategy trained with Cheat-$n$ in Cheat-$n'$ where $n' > n$, an agent may face with an unknown situation, i.e., an infoset with health point $n''$ where $n' \geq n'' > n$. For such cases, we use the strategy learned in case $n'' = n$. In this sense, we argue that our method is a kind of abstraction.

### 4.2 Training Agents

We test four variations of our methods in the experiments: (1) General: there is no abstraction in the information that this agent can obtain during the game and at the same time, it is the baseline of all variations; (2) History-Aware (HA): it is aware of the game history but not the Health Points; (3) HP-Aware (HPA), which is aware of the Health Points but not the game history; (4) Memoryless (M), which does not include either the game history or HP information in its infosets.

In Section 4.3, we will introduce Infosets Abstraction and in our experiments, there are more variants when we combine these four types of agents with different abstraction methods.

### 4.3 Infosets Abstraction

To avoid confusion of the word 'Abstraction', we will specify the more widely-known Abstraction method as *Infosets Abstraction*. There are two Infosets Abstractions we use in our experiment, Card Abstraction and History Abstraction.

- History Abstraction
  This Abstraction only can be applied to agents which include history information in their infosets, i.e. History-Aware and General Agent. Due to the property of the game, the ranks are played repeatedly, it is natural to consider the former history from last round less important than the current round. As a result, when we apply History Abstraction, if we are playing with $k$ ranks, we only save the history of the last $k$ rounds of the game instead of the whole game history from the beginning.
- Cards Abstraction
  We design the Cards Abstraction especially for card games like Cheat. In every turn, we are more willing to deal the cards of the *current rank* so we introduce the idea of *relative position representation* which is based on this rule. Since the shared rank forms a circulation (we begin with rank Ace, rise to rank K then from Ace again...), at each rank what players really care about is the relative position of the current rank. By "relative", we mean that we do not store the specific ranks players share or those of the cards, but how many steps they are away from the current rank.
  For example, if we have cards Ace, 2, 4, 5 in our hands, and the current rank is 4, the game uses 6 ranks in total, the abstracted cards representation will be $[3, 4, 0, 1]$ ($[0, 1, 3, 4]$ in the memory storage). Similarly, if we have cards 2, 3, 5, 6 and the current rank is 5, the representation of our cards after the Cards Abstraction will also be $[0, 1, 3, 4]$. Although the cards we are holding are different, they have the "same" power when facing the conditions we mentioned above respectively.
  With the help of this abstraction technique, we can further combine similar situation during the game play. Moreover, unlike the History Abstraction, Cards Abstraction can be applied to all the four agents.

We will compare the effects of these abstractions on the four agents in the next chapter.

## 5. Experiments

### 5.1 Mini-Cheat

The whole experiments were conducted in a simplified version of Cheat, naming *Mini-Cheat*. In Mini-Cheat, we use cards of 3 ranks and 2 cards for each rank, i.e. 6 cards in total. There are 2 players in the game and we deal 2 cards to each player to eliminate the possibility of perfect information. Although only a subset of cards is used in Mini-Cheat, it inherits an important property of infinite game length with repetition from the original.

In the following paragraph, "Cheat-$n$" means Mini-Cheat with $n$ health points for each player, unless stated otherwise. Moreover, we found that the average number of challenges in one game without any restriction is about 3.5, so we will start with Cheat-3, a simple but still strategically complex version of the game. Similar to the naming of game environments, we call Memoryless agents trained in Cheat-$n$, Memoryless-$n$. The same thing works for all the other agents.

### 5.2 Testing bots

To evaluate how our agents perform in different environments under various ways of training, we built two testing bots: Random bot and Heuristic-perfect bot. The Random bot chooses all actions randomly with equal probability. On the other hand, the Heuristic bot was built based on human knowledge. It memorizes all the cards that were revealed in the game and keeps a record of where they go if someone takes the pile on the table. As a result, it is quite strong and more accurate when challenging other players. As a result it behaves as the perfect player once all the cards are memorized.

### 5.3 Results

We compare all the agents in four aspects: the first one is cost, including time consumption and storage space; the second one is winning rates against the two testing bots; the third one is its generalization ability in different game environments; the fourth is the effect of its results on other agents' training process. We will include the results of agents with and without Infoset Abstraction in the first and second part, then in the rest two parts, we only show the results of ones with Infoset Abstraction method.

### 5.3.1 Time and Space Cost

Table 1 and 2 show the cost of four agents without and with Infoset Abstraction after 100 iterations of training with Chance-Sampled CFR. We can see that the time costs of four agents all increase exponentially as the Health Point increases. Time costs of agents with Infoset Abstraction are a bit larger than those without abstraction. We believe this comes from the calculation of Cards Abstractions.

The numbers of infosets of Memoryless agents stay constant. The growth of numbers of infosets of HP-Aware agent is exponential in powers of 2 while those of History-Aware and General agents are almost in power of 10. We also find that although Infoset Abstraction does not save much time, the numbers of infosets decrease and the more complex the game, the larger the difference becomes.

**Table 1** Time and Space Costs of four agents without Infoset Abstraction

| Agent | | Game Environment | | |
|---|---|---|---|---|
| | | Cheat-3 | Cheat-4 | Cheat-5 |
| Memoryless | Time | 60 | 860 | 17887 |
| | Space | 292 | 292 | 292 |
| HP-Aware | Time | 52 | 1057 | 19034 |
| | Space | 1331 | 2557 | 4257 |
| History-Aware | Time | 171 | 5514 | 18371 |
| | Space | 10689 | 113331 | 1209336 |
| General | Time | 224 | 5054 | 20315 |
| | Space | 11433 | 120512 | 1523174 |

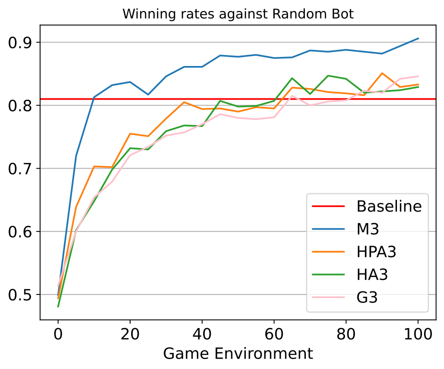
**Table 2** Time and Space Costs of four agents with Infoset Abstraction

| Agent | | Game Environment | | |
|---|---|---|---|---|
| | | Cheat-3 | Cheat-4 | Cheat-5 |
| Memoryless$^\dagger$ | Time | 60 | 1238 | 20759 |
| | Space | 100 | 100 | 100 |
| HP-Aware$^\dagger$ | Time | 80 | 1346 | 21208 |
| | Space | 604 | 1063 | 1660 |
| History-Aware$^\dagger$ | Time | 159 | 4031 | 14428 |
| | Space | 3668 | 46827 | 528371 |
| General$^\dagger$ | Time | 187 | 4231 | 17891 |
| | Space | 4220 | 52723 | 659312 |

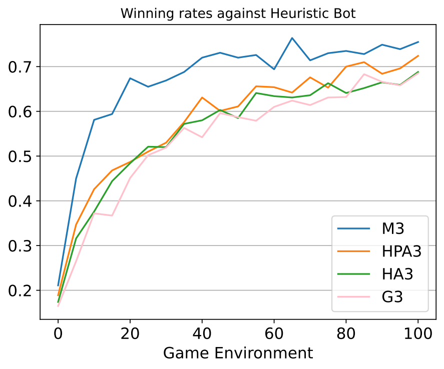
[1,4] To make it distinguishable, we add $^\dagger$ to mark the name of the agents with Infoset Abstraction.
[2] Time is in seconds in wallclock time.
[3] Space is represented in the number of infosets.

### 5.3.2 Winning Rates

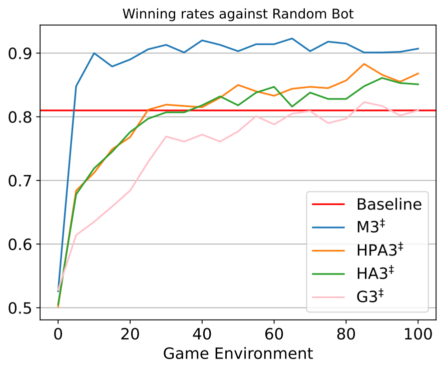
To evaluate the learning efficiency and performance, we measure the winning rate of Heuristic bot against Random bot as our baseline which is approximately 80% (slightly varies in different game environments). The baseline will



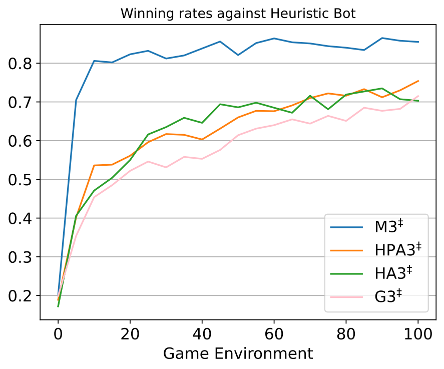
(a) Agents Without Abstraction v.s. Random



(b) Agents Without Abstraction v.s. Heuristic



(c) Agents With Abstraction v.s. Random



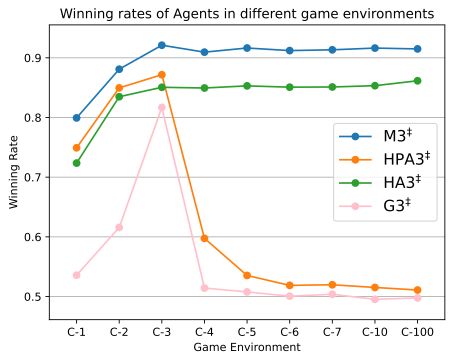
(d) Agents With Abstraction v.s. Heuristic

**Fig. 1** Winning rates of four variants against two testing bots: x-axis is the number of training iterations. The red line represents the baseline.

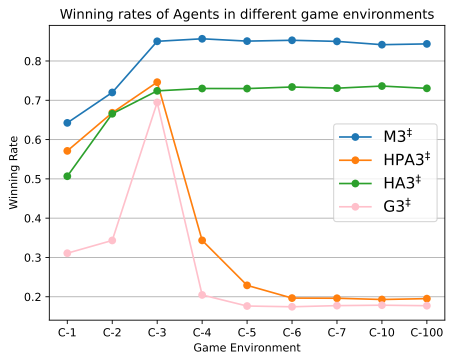
be represented in the red line in the following graphs.

We first test our agents against two testing bots every 5 iterations and then compare among different agents. Fig. 1(a) and 1(b) reveal the trends of winning rates of four agents in Cheat-3 respectively. The x-axis is the number of training iterations, while the y-axis is the winning rate. We notice that after 100 iterations, most agents become strong enough to exceed the baseline and especially, Memoryless-3 reaches almost 90%. Meanwhile, agents even beat Heuristic bot with winning rates over 60% while Memoryless-3 reaches 75%. We also notice that most agents have a much steeper learning efficiency at the beginning of the training and is more steady in the later iterations.

On the other hand Fig. 1(c) and 1(d) show the winning rates of agents with Infosets Abstraction. We can see three in four agents have exceeded the baseline against Random bot. The winning rates of all agents when playing against Heuristic Bot are about 5-10% higher than those without Infoset Abstraction. As a result, this abstraction technique not only helps us save space cost but also improves the performance of four agents. In the following experiments, we only focus on the agents with Infosets Abstraction.



(a) Against Random Bot



(b) Against Heuristic Bot

**Fig. 2** Generalization ability of four agents in different variants of Cheat

### 5.3.3 Generalization Ability

Fig. 2 demonstrates how agents perform in different game environments, from Cheat-2 to Cheat-100 (represented in C-2 to C-100 for abbreviation) of Mini-Cheat. We can see that the Memoryless and History-Aware agents can perform better in the games that have larger numbers of HP while the HP-Aware and General agents only excel in the game environment that it was trained. This is because they involve Health Point information in their infosets while in larger games, they hardly face the state they used to face during training process. The generalization ability of Memoryless player is best of the four while General player appears much weaker.

### 5.3.4 Effect on curriculum learning

We then test the effect of agents trained in smaller games on those trained in larger games. Lighter lines represent agents trained from nothing while darker lines represent agents trained in Cheat-$n$ based on the infosets data from Cheat-$(n-1)$. For example, the darker blue line in Fig. 3(a) is the winning rate of Memoryless-4 using Memoryless-3's final strategy profile at the beginning of the training. In other words, instead of starting from scratch where the initial strategy and regret of each infoset are zero, we use the data from Memoryless-3's infosets at the beginning of Memoryless-4's training.

From Fig. 3 we can see that, abstractions provided by Ordered Abstraction training with a smaller game serves as a good approximation of that with a larger game for Memoryless, History-Aware and General agents since the darker lines start at higher winning rates and always higher than the lighter ones. On the other hand, it is less useful for HP-Aware agents because the trends of lines of the same type are almost the same.

## 6. Conclusion

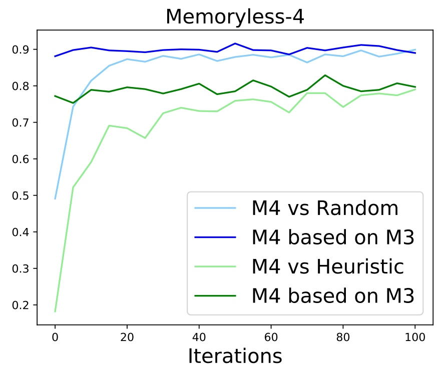
Hereby, we introduce Ordered Abstraction, an abstraction of limiting game length effectively in imperfect information games with a large or possibly infinite game length, such as Cheat. It generates a variant where the game is forced to terminate in a finite number of steps and enable us to train in a much smaller and simpler version of the game. Also, by relaxing the condition of forced termination, we design a curriculum learning with a series of variants; from the most abstracted variant toward the original game.

In Cheat, we introduced a new term called "Health Point" which is used to limit the number of challenges a player can lose in one game. With help of this method, we first designed smaller variants of Cheat so that training of Chance-sampled CFR agents becomes feasible. We also use Infosets Abstraction to further speed up the learning and save memory storage. We see four agents perform differently when playing against testing bots and in various game environments. Moreover, we also demonstrated that we can utilize strategy profiles obtained in smaller games in the training of larger ones and the experiments show that there is an increase in the learning efficiency of specific agents.

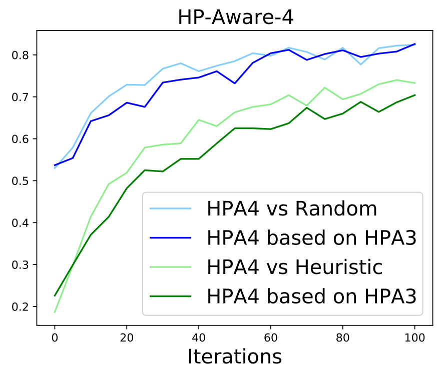
For the future, we are also interested in including Monte Carlo Sampling methods to further improve learning efficiency especially training time, to tackle the original Cheat between two and even more players. Theoretical foundation and the generalizability to other games would also be an interesting line of further research.

## References

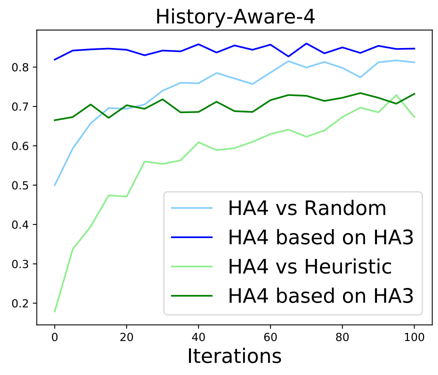
[1] Brown, N. and Sandholm, T.: Simultaneous abstraction and equilibrium finding in games, *Twenty-fourth international joint conference on artificial intelligence* (2015).
[2] Brown, N. and Sandholm, T.: Strategy-based warm starting for regret minimization in games, *Thirtieth AAAI Conference on Artificial Intelligence* (2016).
[3] Brown, N. and Sandholm, T.: Superhuman AI for heads-up no-limit poker: Libratus beats top professionals, *Science*, Vol. 359, No. 6374, pp. 418–424 (2018).
[4] Brown, N. and Sandholm, T.: Superhuman AI for multiplayer poker, *Science*, Vol. 365, No. 6456, pp. 885–890 (online), DOI: 10.1126/science.aay2400 (2019).
[5] Myerson, R. B.: *Game theory: Analysis of Confllict*, Harvard University Press (1997).
[6] Zinkevich, M., Johanson, M., Bowling, M. and Piccione, C.: Regret minimization in games with incomplete information, *Advances in neural information processing systems*, pp. 1729–1736 (2008).



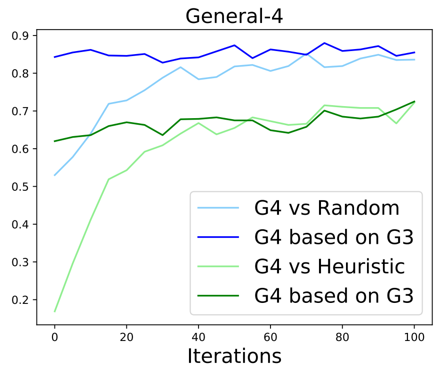
(a) Memoryless-4



(b) HP-Aware-4



(c) History-Aware-4



(d) General-4

**Fig. 3** Effect on curriculum learning of four agents: Blue lines represent winning rates against Random bots; Green lines represent winning rates against Heuristic bots. Lighter lines show training from scratch while darker lines show training based on former data.