2020 IEEE 9th Data Driven Control and Learning Systems Conference
November 20-22, 2020, Liuzhou, China

A Fast-Convergence Method of Monte Carlo Counterfactual Regret Minimization for Imperfect Information Dynamic Games

Xiaoyan Hu[1], Li Xia*[2], Jun Yang[3], Qianchuan Zhao[3]

1. School of Mathematics, Sun Yat-Sen University, Guangzhou 510275, P. R. China
E-mail: huxy58@mail2.sysu.edu.cn

2. Business School, Sun Yat-Sen University, Guangzhou 510275, P. R. China
E-mail: xiali5@sysu.edu.cn(Corresponding author)

3. Department of Automation, Tsinghua University, Beijing 100084, P. R. China
E-mail: {zhaoqc,yangjun603}@tsinghua.edu.cn

Abstract: Among existing algorithms for solving imperfect-information extensive-form games, $Monte\ Carlo\ Counterfactual\ Regret\ Minimization$ (MCCFR) and its variants are the most popular ones. However, MCCFR suffers from slow convergence due to its high variance in estimating values. In this paper, we introduce Semi-OS, a fast-convergence method developed from $Outcome\text{-}Sampling\ MCCFR$ (OS), the most popular variant of MCCFR. Semi-OS makes two novel modifications to OS. First, Semi-OS stores all histories and their values at each information set. Second, after each time we update the strategy, Semi-OS requires a full game-tree traversal to update these values. These two modifications yield a better estimation of regrets. We show that, by selecting an appropriate discount rate, Semi-OS not only significantly speeds up the convergence rate in $Leduc\ Poker$ but also statistically outperforms OS in head-to-head matches of Leduc Poker, a common testbed of imperfect information games, involving 200,000 hands.

Key Words: Imperfect-information Extensive-form games, Monte carlo counterfactual regret minimization, Fast-convergence, Variance reduction

1 Introduction

Solving imperfect-information games has been a core interest in artificial intelligence. The aim is to find an (approximated) Nash equilibrium in which no single player can benefit by alternating its strategy while other players' strategies remain unchanged. Poker serves as a benchmark in this field [1][2]. In the past few decades, remarkable progress has been made in this field[3][4][5]. $Kuhn\ Poker$, a two-player poker with three cards, is the first application discussed in game theory and was analytically solved by hand[6]. Bowling et al.[7] introduce $Cepheus$ which is trained by $CFR+$ algorithm and essentially weakly solves the $Heads\text{-}Up\ Limit\ Hold'em$ (HULH). $Deep\ Stack$ introduced by Moravčík et al.[8] defeated professional poker players in $Heads\text{-}Up\ No\text{-}Limit\ Poker$ (HUNL) with statistical significance in a study involving 44,000 hands. Brown et al. introduce $Libratus$[9] which beat four leading human professionals in HUNL and $Pluribus$[10] which outperformed top human professionals in six-player no-limit texas hold'em poker.

Among existing algorithms to solve imperfect-information extensive-form games, $Monte\ Carlo\ Counterfactual\ Regret\ Minimization$ (MCCFR) and its variants introduced by Zinkevich[11], Lanctot[12] and Gibson[13] are the most popular ones. $Counterfactual\ Regret\ Minimization$ (CFR) introduced by Zinkevich et al. is an iterative self-play algorithm for solving large games based on regret minimization and has proven to be effective in the domain of poker. However, CFR requires a full game-tree traversal at each iteration and therefore consumes enormous computational resources. To avoid this problem, Lanctot et al. introduce MCCFR, a family of sampled-based algorithms, which leads to a dramatic decrease in the cost per iteration. CFR is also viewed as a member of MCCFR but without sampling. However, MCCFR suffers from high variance in estimating values and therefore slow convergence[13]. To our best knowledge, several variance reduction techniques have been proposed in the past few years. In the online setting of MCCFR, where the preparation time is relatively limited, Lisý et al.[14] propose $Online\ Outcome\ Sampling$ (OOS). OOS makes two modifications to MCCFR. First, OOS incrementally builds the search tree, which is similar to the $Monte\ Carlo\ Tree\ Search$ (MCTS). Second, OOS adopts $In\text{-}Match\ Search\ Targeting$ techniques, such as $Information\ Set\ Targeting$ (IST) and $Public\ Subgame\ Targeting$ (PST), that direct the online sampling towards histories that are more likely to happen during the match currently played. As a result, OOS converges close to Nash equilibrium in $Goofspiel$, $Liar's Dice$, and $Generic\ Poker$, while $Preexisting\ Information\ Set\ Monte\ Carlo\ Tree\ Search$ (ISMCTS) can increase exploitability over time. In the off-line setting, Schmid et al.[15] introduce $VR\text{-}MCCFR$. Similar to the policy gradient method in reinforcement learning, VR-MCCFR reformulates estimated values and updates as a function of sampled values and state-action baselines. VR-MCCFR brings an order of magnitude speedup in convergence and three orders of magnitude decrease in the empirical variance in $Leduc\ Poker$ compared to plain MCCFR. Brown et al.[16] introduce two modifications to MCCFR in $Discounted\ Monte\ Carlo\ Regret\ Minimization$. First, regrets from earlier iterations will be discounted using various functions. Second, a non-standard regret minimizing function is adopted. These modifications lead to superior performance in HUNL compared to vanilla MCCFR.

However, methods mentioned above require either significantly additional computational resources such as targeting information sets to determine the direction of sampling in OOS or previous research such as finding an optimal base-

978-1-7281-5922-5/20/$31.00 ©2020 IEEE

line in VR-MCCFR. The objective of our work is to make modifications that maintain the simplicity of MCCFR but yield better performance on variance reduction.

In this paper, we introduce $Semi\text{-}OS$, a fast-convergence algorithm developed from $Outcome\text{-}Sampling\ MCCFR$ (OS) which is the most popular algorithm among variants of MCCFR. Semi-OS is similar to OS in the following ways: both techniques sample one single trajectory at each iteration and only update regrets at information sets that the sampled trajectory goes through. However, Semi-OS makes two novel modifications to OS. First, after each time we update the strategy, Semi-OS requires a full game-tree traversal whose function will be specified later. Second, Semi-OS stores all histories at each information set and calculates the counterfactual regret by its full definition in CFR.

The main contributions of this paper are summarized as follows. First, in $Leduc\ Poker$ which serves as a common testbed imperfect-information game, Semi-OS significantly speeds the convergence by using an appropriate discount rate. Second, Semi-OS also statistically outperforms OS in head-to-head matches of Leduc Poker involving 200,000 hands.

The rest of the paper is organized as follows. In Section 2, we first introduce basic concepts and definitions in extensive-form games with imperfect information. In Section 3, we describe the theory of CFR and MCCFR, especially Outcome-Sampling MCCFR. In Section 4, we introduce the algorithm and mechanism of variance reduction in Semi-OS. Our main experimental results are obtained in Section 5. Finally, we conclude this paper with Section 6.

2 Notation and Background

2.1 Extensive-form Games with Imperfect-Information

A finite imperfect-information extensive-form game is defined as a tuple $(\mathcal{N}, c, \mathcal{A}, \mathcal{H}, \mathcal{Z}, P, u, \mathcal{I})$.

$\mathcal{N} = \{1, ..., n\}$ is a finite set of n players and we denote $the\ chance$ by c. $\mathcal{A}$ is a finite set of actions. $\mathcal{H}$ contains all possible histories of actions and every prefix of a history in $\mathcal{H}$ is also in $\mathcal{H}$. Plus, we use the notation $h \sqsubseteq h'$ to represent that h is a prefix of h'. $\mathcal{Z} \subset \mathcal{H}$ is a finite set of terminal histories that are not prefixes of any history. $P : \mathcal{H} \backslash \mathcal{Z} \rightarrow \mathcal{N} \cup \{c\}$ is the player function that maps each nonterminal history h to a member in $\mathcal{N} \cup \{c\}$, indicating that the next action is taken by one of the players or $the\ chance$. $u : \mathcal{N} \times \mathcal{Z} \rightarrow \mathcal{R}$ is the utility function that maps each terminal history to a real number as a reward for each player.

For each player $i \in \mathcal{N}$, the $information\ partition$ of player i is a finite set of histories denoted by $\mathcal{I}_i = \{h \in \mathcal{H} | P(h) = i\}$. That is, it is player i's turn to take action when each history in $\mathcal{I}_i$ occurs. An $information\ set$ of player i is a finite set of histories $I \subset \mathcal{I}_i$ such that h, h' can not be distinguished by player i, for all h, $h' \in I$. One possible method to define an information set in poker is that histories in the same information set have uniform private cards and a uniform public action sequence. At each information set I, $A(I) \subset \mathcal{A}$ denotes the set of available actions. The strategy of player i is denoted by σ_i which assigns a distribution over $A(I)$ at each information set I. $\sigma(a|I)$ denotes the probability to take action a at information set I. We denote $\sum_i$ as the set of all possible strategies of player i. A strategy profile denoted by σ represents the combination of strategies of all players in $\mathcal{N}$. And we denote σ_{-i} as the combination of strategies in σ except σ_i. Let $\pi^\sigma(h)$ be the probability of the occurrence of history h given that players take actions according to σ. $\pi^\sigma(h, h')$ denotes the probability of the occurrence of history h' under the strategy profile σ given that h is reached. Then, we define the probability that is contributed to $\pi^\sigma(h)$ by player i as $\pi_i^\sigma(h)$ and the probability that is contributed by players except player i and $the\ chance$ as $\pi_{-i}^\sigma(h)$. Thus, we can obtain that $\pi_i^\sigma(h)\pi_{-i}^\sigma(h) = \pi^\sigma(h)$. For each information set I, we define $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$ as the arriving probability of this information set under the strategy profile σ. $u_i^\sigma(h)$ denote the reward of player i given a terminal history h is reached along which all players choose actions according to the strategy profile σ. We define the utility of player i given that all players take actions according to the strategy profile σ as the expected reward of terminal histories, which is defined as $u_i(\sigma) = u_i(\sigma_i, \sigma_{-i}) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$.

2.2 Nash Equilibrium and Exploitability

$Nash\ equilibrium$ [18] is traditionally used in solving a two-player game. A Nash equilibrium is a strategy profile σ such that:

$$u_1(\sigma) \geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \tag{1}$$

$$u_2(\sigma) \geq \max_{\sigma_2' \in \Sigma_2} u_2(\sigma_2', \sigma_1) \tag{2}$$

Respectively, an ϵ-Nash equilibrium is defined as follows:

$$u_1(\sigma) + \epsilon \geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \tag{3}$$

$$u_2(\sigma) + \epsilon \geq \max_{\sigma_2' \in \Sigma_2} u_2(\sigma_2', \sigma_1) \tag{4}$$

A $best\ response$[18] to strategy σ_{-i} is the player i's strategy denoted by $BR_i(\sigma_{-i})$ that satisfies:

$$BR_i(\sigma_{-i}) = \arg\max_{\sigma_i' \in \sum_i} u_i(\sigma_i', \sigma_{-i}) \tag{5}$$

We further denote $b_i(\sigma_{-i}) = u_i(BR_i(\sigma_{-i}), \sigma_{-i})$ as the $best\text{-}response\ value$. In game theory, we use $exploitability$ to measure the deviation of the strategy profile σ from the Nash equilibrium. In particular, in a two-player zero-sum game, the exploitability of a strategy profile σ is defined as:

$$\epsilon_\sigma = b_1(\sigma_2) + b_2(\sigma_1) \tag{6}$$

However, exploitability is impractical to calculate especially in large games since it is difficult to compute the best response[19]. Thus in this paper, we adopt the Local Best Response (LBR) technique[20] to compute a lower bound of exploitability. Here, we describe the basic idea of LBR as follows. After each action taken by the opponent, LBR uses the Bayes' rule to update its belief on the opponent's private cards, which is presented as a probability distribution over all possible private cards of the opponent, with access to the opponent's strategy. Then, LBR takes the action that maximizes its expected utility under the assumption that the game will be called/checked until the end.

3 Counterfactual Regret Minimization

$Counterfactual\ regret\ minimization$ (CFR) is an iterative self-play algorithm for solving imperfect-information extensive-form games based on regret minimization. We define the strategy profile at the tth iteration is defined as σ^t. The $average\ overall\ regret$ of player i after T iterations is defined as:

$$R_i^T = \frac{1}{T}\max_{\sigma_i^* \in \Sigma_i}\sum_{t=1}^{T}(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)) \tag{7}$$

At each information set I and for each action $a \in A(I)$, we define the $average\ strategy$ of player i within T iterations as:

$$\overline{\sigma}_i^T(a|I) = \frac{\sum_{t=1}^{T}\pi_i^{\sigma^t}(I)\sigma_i^t(a|I)}{\sum_{t=1}^{T}\pi_i^{\sigma^t}(I)} \tag{8}$$

An important theorem on average regret, average strategy and the ϵ-Nash equilibrium is proved by Waugh[21]:

Theorem 1 *In a two-player zero-sum game after T iterations, if both players' average overall regret is less than ϵ, then their average strategy profile $(\overline{\sigma}_1^T, \overline{\sigma}_2^T)$ is a 2ϵ-Nash equilibrium.*

Let $Z_I = \{z \in \mathcal{Z} | \exists z[I] \sqsubseteq z, z[I] \in I\}$ denote the set of terminal histories that "go through" information set I. In CFR, the $counterfactual\ value$ of player i at information set I given the strategy profile σ is defined as:

$$v_i(\sigma, I) = \sum_{z \in Z_I}\pi_{-i}^{\sigma}(z[I])\pi^{\sigma}(z[I], z)u_i(z) \tag{9}$$

The $immediate\ counterfactual\ regret$ after T iterations is defined as:

$$R_{i,imm}^T(I) = \frac{1}{T}max_{a \in A(I)}\sum_{t=1}^{T}(v_i(\sigma_{(I\to a)}^t, I) - v_i(\sigma^t, I)) \tag{10}$$

where $\sigma_{(I\to a)}$ denotes the strategy profile identical to σ except that player i always takes action a at information set I. Let $x_+ = max(x, 0)$. The relation between $R_{i,imm}^T$ and R_i^T proved by Zinkevich et al.[11] is stated as follows:

Theorem 2 $R_i^T \le \sum_{I \in \mathcal{I}_i} R_{i,imm}^+(I)$

Theorem 2 implies that the average overall regret can be bounded by the sum over counterfactual regrets at each information set, which is key to the CFR algorithm. Further, for each $a \in A(I)$, we define its $counterfactual\ regret$ as:

$$R_i^T(I, a) = \frac{1}{T}\sum_{t=1}^{T}(v_i(\sigma_{(I\to a)}^t, I) - v_i(\sigma^t, I)) \tag{11}$$

After each iteration, one method to update strategies is the $Regret\text{-}Matching$ (RM) algorithm:

$$\sigma^{T+1} = \begin{cases} \frac{R_+^T(I,a)}{\sum_{a' \in A(I)} R_+^T(I,a')} & if \sum_{a'} R_+^T(I, a') > 0 \\ \frac{1}{|A(I)|} & otherwise \end{cases} \tag{12}$$

The following theorem introduced by Cesa-Bianchi et al.[22] shows the mechanism of how the minimization of counterfactual regrets finally leads to minimizing overall regret.

Theorem 3 *If we update the policy according to RM, then $R_{i,imm}^T(I) \le \Delta_{u,i}\sqrt{|A_i|}/\sqrt{T}$ and consequently $R_i^T \le \Delta_{u,i}|\mathcal{I}_i|\sqrt{|A_i|}/\sqrt{T}$, where $|A_i| = \max_{h:P(h)=i}|A(h)|$.*

In CFR+, Tammelin[23] proposes $Regret\text{-}Matching+$ (RM+) where the negative regrets will be set to zeros after each iteration:

$$\sigma^{T+1} = \begin{cases} \frac{max\{R_+^T(I,a),0\}}{\sum_{a' \in A(I)} R_+^T(I,a')} & if \sum_{a'} R_+^T(I, a') > 0 \\ \frac{1}{|A(I)|} & otherwise \end{cases} \tag{13}$$

3.1 Monte Carlo Counterfactual Regret Minimization

In $Monte\ Carlo\ Counterfactual\ Regret\ Minimization$ (MCCFR) , we denote a set of subsets of Z as $\mathcal{Q} = \{Q_1, ..., Q_r\}$ which spans the Z. The $sampled\ counterfactual\ value$ when updating on $\mathcal{Q}_i$ is:

$$\widetilde{v}_i(\sigma, I|i) = \sum_{z \in Q_j \cap Z_I}\frac{1}{q(z)}u_i(z)\pi_{-i}^{\sigma}(z[I])\pi^{\sigma}(z[I], z) \tag{14}$$

where $q(z)$ is the probability of sampling the history z. By simple calculation, we can obtain that[12]:

Theorem 4 $\mathbf{E}_{j\sim q_j}[\widetilde{v}_i(\sigma, I|j)] = v_i(\sigma, I)$

The $sampled\ immediate\ counterfactual\ regret$ of action a is defined as:

$$\widetilde{r}(I, a) = \widetilde{v}_i(\sigma_{(I\to a)}^t, I) - \widetilde{v}_i(\sigma^t, I) \tag{15}$$

$Outcome\text{-}Sampling$ MCCFR (OS) is the most popular one in the family of Monte Carlo CFR minimizing algorithms. In OS, $\mathcal{Q}$ is chosen such that $|Q_i| = 1, \forall Q_i \in \mathcal{Q}$. Hence, the $sampled\ immediate\ counterfactual\ regret$ of action a alternates to the following form:

$$\widetilde{r}(I, a) = \begin{cases} w_I \cdot (1 - \sigma(a|z[I])) & if\ (z[I]a) \sqsubseteq z \\ -w_I \cdot \sigma(a|z[I]) & otherwise \end{cases} \tag{16}$$

where $w_I = \frac{u_i(z)\pi_{-i}^{\sigma}(z)\pi_i^{\sigma}(z[I]a, z)}{\pi^{\sigma'}(z)}$.

A well-known theorem on the regret bound for Outcome-Sampling MCCFR is proved by Lanctot et al.[12]:

Theorem 5 *Let $\overrightarrow{a}_i$ denote a subsequence of a history such that it only contains player i's actions in the history and $\overrightarrow{A_i}$ denote the set of all such player i action subsequences. Let $\mathcal{I}_i(\overrightarrow{a}_i)$ denote the set of all information sets where player i's action sequence up to that information set is $\overrightarrow{a}_i$. Then for any $p \in (0, 1]$, when using Outcome-Sampling MCCFR where $\forall z \in \mathcal{Z}$ either $\pi_{-i}^{\sigma}(z) = 0$ or $q(z) \ge \delta > 0$ at every time step, with probability $1 - p$, average overall regret is bounded by $R_i^T \le (1 + \frac{\sqrt{2}}{\sqrt{p}})(\frac{1}{\delta})\Delta_{u,i}M_i\sqrt{|A_i|}/\sqrt{T}$, where $M_i = \sum_{\overrightarrow{a}_i \in \overrightarrow{A}_i}\sqrt{|\mathcal{I}_i(\overrightarrow{a})|}$*

4 Semi-OS

In Outcome-Sampling MCCFR, while $\widetilde{v}_i(\sigma, I|j)$ is an unbiased estimator of $v_i(\sigma, I)$, only one action is sampled at each iteration and values of other actions at the information set are set to zeros, which introduces great variance. Intuitively, if we can obtain a more accurate estimation of those values, then the variance in Outcome-Sampling MCCFR is tended to be reduced.

Here we introduce Semi-OS, a fast-convergence method developed from OS. Semi-OS is similar to OS in the following way: at each iteration, one single trajectory is sampled and only regrets at information sets that this trajectory goes through are updated, which is why this method is called Semi-OS. The key idea of Semi-OS is that utilities of actions that are not sampled will not be set to zeros. Rather, we store all histories that go through the information set and their values, respectively. Then we calculate the utility of information set by its definition in CFR, which yields better-estimated values. Thus, here we define $sampled\ counterfactual\ value$ of information set I on the iteration t given that terminal history z is sampled as:

$$\widetilde{v}_i^t(\sigma^t, I) = \sum_{z \in Z_I} \pi_{-i}^{\sigma^t}(z[I])\pi^{\sigma^t}(z[I], z)u_i(z) \tag{17}$$

During the back-propagation, we compute the regrets at each information set which the history goes through as follows:

$$\widetilde{r}^t(I, a) = \widetilde{v}_i^t(\sigma^t_{(I \to a)}, I) - \widetilde{v}_i^t(\sigma^t, I) \tag{18}$$

Another key part in Semi-OS is that after each time we update the strategy, Semi-OS requires a full game-tree traversal to update those values mentioned above to keep $\widetilde{v}_i(\sigma, I)$ an unbiased estimator of $v_i(\sigma, I)$.

The pseudo-code of the Semi-OS algorithm is given on the next page. The function $parent(v)$ returns the parent node of v and the function $child(v, a)$ returns the child of node v after choosing the action a. The function $player(v)$ returns the player that acts at node v. The function $CalculateUtility(h)$ returns the utilities of the player 1 and player 2 according to the terminal history that is sampled. The key part of Semi-OS is the function $TreeTraversal(v_0)$ on line 21 which updates values of all histories at each information set. It enables us to obtain an unbiased estimation of the regret in the following iterations. The function $PolicyUpdate(I)$ updates the strategy on information set I after each update interval which is needed to be carefully set beforehand. Empirically, large update interval surely yields a better evaluation of the current strategy but a slow update rate. Small update interval yields fast update rate but too many times of full game-tree traversals which harms its efficiency and inadequate evaluation of current strategy. The function $DiscountRegret(I, DiscountRate)$ discounts the regrets on information set I after each discount interval which is also set beforehand.

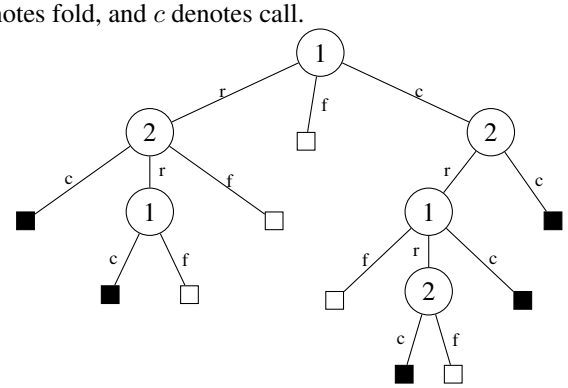
Algorithm 1 the Semi-OS algorithm.

```
1: initiate UpdateInterval
2: initiate DiscountInterval
3: initiate DiscountRate
4: function Semi-OS(h)
5:     create root node v_0 with history h_0
6:     while within computational budgets do
7:         v_current ← v_0
8:         while v_current is nonterminal do
9:             v_next ← TreePolicy(h, v_current)
10:            v_current ← v_next
11:            if v_current is an action node then
12:                alternate player
13:            end if
14:        end while
15:        CalculateUtility(h)
16:        BackUp(v_terminal)
17:        if UpdateInterval then
18:            for each information set I do
19:                PolicyUpdate(I)
20:            end for
21:            TreeTraversal(v_0)
22:        end if
23:        if DiscountInterval then
24:            for each information set I do
25:                DiscountRegret(I, DiscountRate)
26:            end for
27:        end if
28:    end while
29: end function
30:
31: function TreePolicy(h, v)
32:    while h is nonterminal do
33:        choose a ∈ A(h) uniformly at random
34:        return Expand(v, a)
35:    end while
36: end function
37:
38: function Expand(v, a)
39:    if a is untried then
40:        add a new child v' to v
41:        return v'
42:    else
43:        return child(v, a)
44:    end if
45: end function
46:
47: function BackUp(v_terminal)
48:    while v is not root node do
49:        if v is an action node then
50:            i ← player(v)
51:            I ← information set
52:            ṽ_i(σ, I) ← Σ_{z∈Z_I} π^σ_{-i}(z[I]) π^σ(z[I], z) u_i(z)
53:            for a ∈ A do
54:                r̃(I, a) ← ṽ_i(σ^t_{(I→a)}, I) − ṽ_i(σ^t, I)
55:                add r̃(I, a) to Regret(I, a)
56:            end for
57:        end if
58:        v ← parent(v)
59:    end while
60: end function
```

5 Experimental Results

5.1 Leduc Poker

Leduc Poker is a two-player hold'em with only two betting rounds. It is widely used as one of the benchmarks in poker research. There are two suits of three identically-ranked cards, a total of six cards. At the beginning of one game, each player antes one chip to play. Before the first betting round, each player is dealt with one private card. If none of the players fold, then the game moves on to the second betting round where one public card will be uncovered. There is a two-bet maximum in a round and the raise is fixed at two chips in the first round, four chips in the second round. The game tree of *Leduc Poker* is given below. The number in each circle denotes the player to act. Closed boxes represent the game moves to the next round and open boxes represent the end of the game. Here, r denotes raise, f denotes fold, and c denotes call.



5.2 Convergence

We compare Outcome-Sampling MCCFR with Semi-OS on Leduc Poker. Fig. 1 shows the upper bound of exploitability for Outcome-Sampling MCCFR using RM (OS) and RM+ (OS+) and Semi-OS using RM with a discount rate of $\frac{T}{T+1}$. Here, we compute the upper bound by playing against the local best response (LBR) player and calculate the 95 percentile over 100 runs. It shows that Semi-OS significantly speeds the convergence rate.

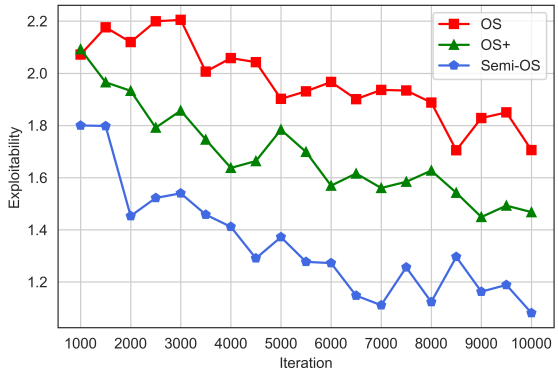


Fig. 1: Upper Bound of Exploitability

5.3 Head-to-head Performance

We also evaluate their performance in head-to-head matches of Leduc Poker involving 200,000 hands. We show that Semi-OS can statistically beat OS and OS+ in 1,000 games after they are both trained for equal iterations. The result is computed over 20 runs and is presented in Fig. 2

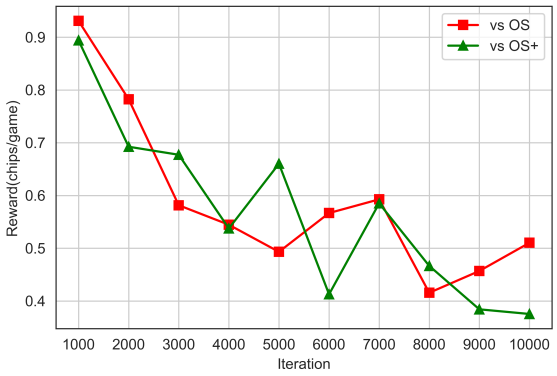


Fig. 2: Mean Reward of Semi-OS

6 Conclusions

In this paper, we introduce Semi-OS, a fast-convergence method of counterfactual regret minimization for imperfect information dynamic games. Semi-OS is similar to Outcome-Sampling MCCFR in that both methods sample a single trajectory in each iteration. But unlike OS, the utilities of actions that are not sampled will not be set to zeros. Rather, Semi-OS calculates the utility of information set by all histories that go through it. We evaluate the performance of Semi-OS on Leduc Poker. We show that by selecting an appropriate discount rate, Semi-OS significantly speeds the convergence of Outcome-Sampling MCCFR. Moreover, Semi-OS can beat OS in head-to-head matches of Leduc Poker. Semi-OS requires to store all histories and a full game-tree traversal after each time we update the strategy. Therefore, if the game is relatively small, Semi-OS may serve as a strong algorithm to solve them. In future work, we hope to apply this technique to larger games and evaluate its performance in the online setting of regret minimization.

References

[1] Shi, Jiefu and Littman, Michael L, Abstraction methods for game theoretic poker, in *International Conference on Computers and Games*, 2000: 333-345.

[2] Billings, Darse and Davidson, Aaron and Schaeffer, Jonathan and Szafron, Duane, The challenge of poker, in *Artificial Intelligence 134(1-2)*, 2002: 201-240.

[3] Sandholm, Tuomas, The state of solving large incomplete-information games, and application to poker, in *Ai Magazine 31(4)*, 2010: 13-32.

[4] Gilpin, Andrew, Algorithms for abstracting and solving imperfect information games, PhD thesis, Carnegie Mellon University, 2009.

[5] Johanson, Michael and Burch, Neil and Valenzano, Richard and Bowling, Michael, Evaluating state-space abstractions in extensive-form games, in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013: 271-278.

[6] Kuhn, Harold W, A simplified two-person poker, in *Contributions to the Theory of Games 1*, 1950: 97-103.

[7] Bowling, Michael and Burch, Neil and Johanson, Michael and Tammelin, Oskari, Heads-up limit hold?em poker is solved, in *Science 347(6218)*, 2015: 145-149.

[8] Moravčík, Matej and Schmid, Martin and Burch, Neil and Lisỳ, Viliam and Morrill, Dustin and Bard, Nolan and Davis, Trevor and Waugh, Kevin and Johanson, Michael and Bowling, Michael, Deepstack: Expert-level artificial intelligence in heads-up, in *Science(356)6337*, 2017: 508-513.

[9] Brown, Noam and Sandholm, Tuomas, Superhuman AI for heads-up no-limit poker: Libratus beats top professionals, in *Science(359)6374*, 2018: 418-424.

[10] Brown, Noam and Sandholm, Tuomas, Superhuman AI for multiplayer poker, in *Science(365)6456*, 2019: 885-890.

[11] Zinkevich, Martin and Johanson, Michael and Bowling, Michael and Piccione, Carmelo, Regret minimization in games with incomplete information, in *Advances in neural information processing systems*, 2008: 1729-1736.

[12] Lanctot, Marc and Waugh, Kevin and Zinkevich, Martin and Bowling, Michael, Monte Carlo sampling for regret minimization in extensive games, in *Advances in neural information processing systems*, 2009: 1078-1086.

[13] Gibson, Richard and Lanctot, Marc and Burch, Neil and Szafron, Duane and Bowling, Michael, Generalized sampling and variance in counterfactual regret minimization, in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[14] Lisý, Viliam and Lanctot, Marc and Bowling, Michael, Online monte carlo counterfactual regret minimization for search in imperfect information games, in *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, 2015: 27-36.

[15] Schmid, Martin and Burch, Neil and Lanctot, Marc and Moravcik, Matej and Kadlec, Rudolf and Bowling, Michael, 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, 33*, 2019: 2157-2164.

[16] Brown, Noam and Sandholm, Tuomas, Solving imperfect-information games via discounted regret minimization, in *Proceedings of the AAAI Conference on Artificial Intelligence, 33*, 2019: 1829-1836.

[17] Southey, Finnegan and Bowling, Michael P and Larson, Bryce and Piccione, Carmelo and Burch, Neil and Billings, Darse and Rayner, Chris, Bayes' bluff: Opponent modelling in poker, in *arXiv preprint arXiv:1207.1411*, 2012.

[18] Nash, John F and others, Equilibrium points in n-person games, in *Proceedings of the national academy of sciences, 36*, pp: 48-49.

[19] Davis, Trevor and Burch, Neil and Bowling, Michael, Using response functions to measure strategy strength, in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[20] Lisý, Viliam and Bowling, Michael, Equilibrium approximation quality of current no-limit poker bots, in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] Waugh, Kevin, Abstraction in large extensive games, 2009.

[22] Cesa-Bianchi, Nicolo and Lugosi, Gábor, Prediction, learning, and games, 2006.

[23] Tammelin, Oskari, Solving large imperfect information games using CFR+, arXiv preprint arXiv: 1407.5042, 2014.