OPTIMAL ACTION ABSTRACTION FOR IMPERFECT INFORMATION EXTENSIVE-FORM GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Action abstraction is critical for solving imperfect information extensive-form games (IIEFGs) with large action spaces. However, due to the large number of states and high computational complexity in IIEFGs, existing methods often focus on using a fixed abstraction, which can result in sub-optimal performance. To tackle this issue, we propose a novel Markov Decision Process (MDP) formulation for finding the optimal (and possibly state-dependent) action abstraction. Specifically, the state of the MDP is defined as the public information of the game, each action is a feature vector representing a particular action abstraction, and the reward is defined as the expected value difference between the selected action abstraction and a default fixed action abstraction. Based on this MDP, we build a game tree with the action abstraction selected by reinforcement learning (RL), and solve for the optimal strategy based on counterfactual regret minimization (CFR). This two-phase framework, named RL-CFR, effectively trades off computational complexity (due to CFR) and performance improvement (due to RL) for IIEFGs, and offers a novel RL-guided action abstraction selection in CFR. To demonstrate the effectiveness of RL-CFR, we apply the method to solve Heads-up No-limit (HUNL) Texas Hold'em, a popular representative benchmark for IIEFGs. Our results show that RL-CFR defeats ReBeL, one of the best fixed action abstraction-based HUNL algorithms, and a strong HUNL agent Slumbot by significant win-rate margins 64 ± 11 and 84 ± 17 mbb/hand, respectively.

1 INTRODUCTION

The imperfect information extensive-form game (IIEFG) model (Streufert, 2021) is a general formulation for studying multi-player turn-taking games in a tree representation, including Poker (Moravcík et al., 2017), MahJong (Wang, 2023) and Scotland Yard (Schmid et al., 2021). Solving IIEFGs requires finding the Nash equilibrium (Nash, 2002) of the game, especially under two-person zero-sum conditions. In recent years, the most popular approach for solving large IIEFGs has been counterfactual regret minimization (CFR) or its variants (Burch & Bowling, 2013; Tammelin, 2014; Brown & Sandholm, 2019b; Brown et al., 2019), which gives a mixed strategy with low exploitability for IIEFG.

However, many IIEFGs have myriad actions. As a result, the size of the game tree increases exponentially with the number of actions (Schnizlein et al., 2009), and directly applying CFR-based solutions encounters tremendous computational complexity. To mitigate this problem, action abstraction (Aceto, 1991), which selects limited number of actions from all available actions, has thus been extensively applied to greatly reduce the size of the game tree so that CFR can still be solved efficiently. Nevertheless, due to the large number of different states and high computational complexity in IIEFGs, existing results mostly focus on fixed action abstractions (Moravcík et al., 2017; Brown et al., 2020; Zarick et al., 2020). Doing so inevitably lead to sub-optimality, and it remains an unsolved challenge to design strategies that can find optimal dynamic action abstractions with manageable computational complexity (Brown, 2020).

Reinforcement learning (Humphreys, 1997; Sutton & Barto, 1998) (RL) has been shown to be a revolutionary method in many games (Madeira et al., 2006), e.g., Go (Silver et al., 2016), StarCraft II (Lee et al., 2018) and Dota 2 (Berner et al., 2019). However, applying RL methods to IIEFGs is challenging due to two important features of IIEFG: (i) the optimal strategy for an IIEFG is most

likely a mixed strategy on its support (Chen & Ankenman, 2007; Neyman, 2008), and (ii) the value of an information set may depends on the strategy that it is chosen (Brown et al., 2020). To see this, consider the simplified poker example (Burch, 2017) in Figure 1. In this case, player 1 has equal chance of being dealt J or K and player 2 is always dealt Q. There are 2 chips in the pot (both player put 1 chips in the pot), and both players have 2 chips left behind, with player 1 acts first. The Nash equilibrium strategy for player 1 is all-in all of K and 50% of J, and checking the other 50% of J. If player 1 declares all-in, then the Nash equilibrium strategy for player 2 is calling and folding with equal probabilities. If a Nash equilibrium strategy is adopted, player 1's K expects to win 2 chips, while J expects to lose 1 chips, and player 2 expects to lose 0.5 chips. If player 1 all-in with 100% probability, and player 2's best response strategy is call with 100% probability, player 1's K expects to win 3 chips, while J expects to lose 3 chips, and player 2 expects to win 0 chips.

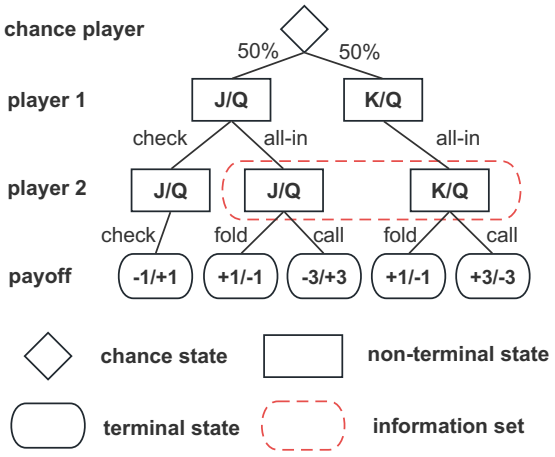

Figure 1: The game starts with a chance state where player 1 have equal chance of being dealt J or K, and player 2 is always dealt Q. Player 1 will always all-in with K and have to decide whether to check or all-in with J. If player 1 declares all-in, player 2 has to decide whether to fold or call. Player 2 does not know player 1's cards, the information set contains two states player 2 incapable of distinguishing. In the terminal state, we assign payoffs to both players based on the assignment rule.

To tackle the above challenge, we propose a two-phase framework, named RL-CFR, which ingeniously combines deep reinforcement learning (DRL) and CFR. Specifically, we first formulate a novel Markov Decision Process (MDP) (van Otterlo & Wiering, 2012) for determining the action-abstraction, where the state of the MDP is the *public information* of the game. For this MDP, each control action is a feature vector representing a particular action abstraction, and the action rewards are set to be the value differences calculated by CFR between selected action abstractions and default fixed action abstraction. Based on this MDP, we then build a game tree according to action abstraction selected by the actor-critic DRL method (Konda & Tsitsiklis, 1999), and eventually solve the strategy for selected action abstraction based on CFR. Our RL-CFR framework offers a principled way to reaps benefits from both RL and CFR, and handles the aforementioned mixed-strategy and probability-dependent reward issues. It also effectively trades off computational complexity (due to CFR) and performance improvement (due to RL) for IIEFGs. As we will see in the experiments, *RL-CFR can be trained from scratch* given only the rules of the IIEFG. Compared to other methods for choosing action abstractions (Hawkin et al., 2011; 2012; Zarick et al., 2020), RL-CFR has a wider range of applicability and faster convergence.

To demonstrate the effectiveness of RL-CFR on large IIEFGs, we evaluate its performance on the challenging Heads-up No-limit Texas Hold'em (HUNL) poker game.[1] Our results show that RL-CFR defeats the fixed action abstraction-based HUNL algorithm ReBeL (Brown et al., 2020) by 64 mbb/hand win-rate in a test of over $600,000$ hands, and beats the popular strong HUNL agent Slumbot (Jackson, 2013) by 84 mbb/hand win-rate in a test of over $250,000$ hands. These significant win-rate margins clearly show the power of our novel RL-CFR solution.

The main contributions of our work are summarized as follows.

- We introduce a novel MDP formulation for IIEFGs, whose states are carefully defined based on public information, actions are feature vectors representing action abstractions, and rewards are

[1]Heads-up No-limit Texas Hold'em is a two-player form of Texas Hold'em, and is an important version of Texas Hold'em for investigating mixed strategy two-player zero-sum IIEFGs (Bard et al., 2013) due to its complex nature (Rubin & Watson, 2011) and extremely large decision space (Johanson, 2013).

value differences between selected action abstractions and default fixed action abstractions. The MDP formulation allows us to dynamically adjust the action abstraction at different states.

- Based on our novel MDP, we propose a novel framework RL-CFR, which effectively combines DRL with CFR to achieve a good balance between computation and optimism, and can be trained from scratch given only the rules of the IIEFG. RL-CFR effectively handles the large decision space and computational complexity of IIEFGs, and enables one to tradeoff computational complexity (due to CFR) and performance improvement (due to RL).
- We evaluate RL-CFR on the popular HUNL game. Our results show that RL-CFR defeats ReBeL (one of the best fixed action abstraction-based HUNL algorithms) and Slumbot (the strongest publicly available HUNL AI provides online comparisons) by significantly win-rate margins, i.e., by margins 64 ± 11 and 84 ± 17 mbb/hand, respectively.

2 Related Work on Extensive-Form Games

Methods of solving IIEFGs. CFR-based algorithms (Burch & Bowling, 2013; Tammelin, 2014; Brown & Sandholm, 2019b; Brown et al., 2019) are are commonly used to solve large IIEFGs, because the regret of CFR is bounded linearly with the game size (a more detailed description of CFR is presented in Appendix C). There are methods such as Hedge (Cesa-Bianchi & Lugosi, 2006) or excessive gap technique (EGT) (Hoda et al., 2010) that theoretically converge faster than CFR.

Faster convergence and better efficiency for solving large IIEFGs. (Habara et al., 2023) combines EGT with CFR for accelerating the solving of large IIEFGs. (Liu et al., 2023) investigates RL regularization techniques in solving IIEFGs and proposes a regularization-based payoff function. (Meng et al., 2023) proposes an efficient deep reinforcement learning method to solve the problem of inaccurate state value estimation in large IIEFGs.

Action abstraction in IIEFGs. In IIEFGs with myriad actions, one can gain more by choosing the right action abstraction (Chen & Ankenman, 2007). A parametric method (Hawkin et al., 2011) has been proposed to find the optimal action abstraction in IIEFGs, and an iterative algorithm (Hawkin et al., 2012) has been introduced to adjust the action abstraction during iteration. These methods of automatically selecting action abstraction for IIEFGs were also used in the early states of Libratus (Brown & Sandholm, 2017a) (one of the strongest HUNL AI). However, these methods change the action abstraction of each node in the game tree at each iteration, and therefore converge slower compared to fixed action abstraction methods (Brown, 2020; Zarick et al., 2020). As a result, these methods require a significant amount of time for pre-computation, and can generally only be used to a small number of early states in large IIEFGs.

3 Background and Notation

Imperfect Information Extensive-Form Games We first provide the necessary notations for Imperfect Information Extensive-Form Games (IIEFGs) based on notations from (Streufert, 2021; Osborne & Rubinstein, 1994; Burch, 2017; Brown, 2020; Kovaík & Lis, 2019). Specifically, an IIEFG describes an imperfect information games in the form of a tree, and can be represented by $G = \langle \mathcal{H}, \mathcal{Z}, \mathcal{A}, \mathcal{N}, \mathcal{P}, \sigma_c, u, \mathcal{I} \rangle$, where each notation is explain below.

- $\mathcal{H}$ is the set of states (histories/nodes). A state $h \in \mathcal{H}$ is described by all history actions from the initial game state $\emptyset$. We use $\cdot$ to indicate concatenation, and $h \cdot a$ means taking an action a at state h. $h \sqsubseteq h'$ means h is an ancestor of h', and $h \sqsubset h'$ means h is a strict ancestor of h'.
- $\mathcal{Z} \subset H$ is the set of terminal states. A terminal state $z \in \mathcal{Z}$ has no available action.
- $\mathcal{A}(h) := \{a | h \cdot a \in \mathcal{H}\}$ is the set of available actions at a non-terminal state $h \in \mathcal{H} \backslash \mathcal{Z}$. $\mathcal{AA}(h) \subseteq \mathcal{A}(h)$ is an action abstraction for $\mathcal{A}(h)$.
- $\mathcal{N} = \{1, \cdots, N\}$ is the set of players. There is a "player" not in player set $\mathcal{N}$, defined as c, called chance decisions, which represents random events players can not control.
- A function $\mathcal{P} : \mathcal{H} \backslash \mathcal{Z} \to \mathcal{N} \cup \{c\}$ determines the acting player at a non-terminal state h. $\mathcal{H}_p$ is the set of all states h such that $\mathcal{P}(h) = p$, and $\mathcal{H}_c$ is the set of chance states.
- The chance strategy $\sigma_c(h, a)$ is a probability that chance will act $a \in \mathcal{A}(h)$ at a state $h \in \mathcal{H}_c$.
- $u = (u_p)_{p \in \mathcal{N}}$ is the value function for each terminal state z.

- The information-partition $\mathcal{I} = (\mathcal{I}_p)_{p\in\mathcal{N}}$ describes the imperfect information of G where $\mathcal{I}_p$ is a partition of $\mathcal{H}_p$ for each player p. A set $I \in \mathcal{I}_p$ is called an information set, and all states in I are indistinguishable for player p. We denote $I(h)$ as the unique information set that contains h. There is a constraint that $\forall I \in \mathcal{I}_p, \forall h \in I$, we have same acting player p, same available actions $\mathcal{A}(h) := \mathcal{A}(I(h))$ and same action abstraction $\mathcal{AA}(h) := \mathcal{AA}(I(h))$.

A behaviour strategy $\sigma_p \in \Sigma_p$ is a function $\sigma_p(I, a) \in \mathbb{R}$ that determines a probability distribution over available actions $a \in \mathcal{A}(I)$ for every information set $I \in \mathcal{I}_p$. We denote $\sigma(I, a) = \sigma_{\mathcal{P}(I)}(I, a)$. $\sigma = (\sigma_p)_{p\in\mathcal{N}}$ is a strategy profile. σ_{-p} is the strategy other than player p's. $\pi^\sigma(h)$ is the probability of reaching state h if players follow σ, calculated as $\pi^\sigma(h) = \prod_{h'\cdot a\sqsubseteq h} \sigma(h', a)$. $\pi^\sigma_p(h)$ is the probability of reaching state h if players except p take actions to h and player p follows σ. $\pi^\sigma_{-p}(h)$ is the probability of reaching state h if player p takes actions to reach h and other players follow σ.

Under a strategy profile σ, player p's expected value (EV) of a non-terminal state h is $u^\sigma_p(h) = \sum_{z\in\mathcal{Z},h\sqsubset z} \pi^\sigma(z)u_p(z)$, and the EV of an information set $I \in \mathcal{I}_p$ is $u^\sigma_p(I) = \sum_{h\in I} u^\sigma_p(h)$. The counterfactual value (CFV) (Zinkevich et al., 2007) for player p of state h is $v^\sigma_p(h) = \sum_{z\in\mathcal{Z},h\sqsubset z} \pi^\sigma_{-p}(h)\pi^\sigma(z|h)u_p(z)$. We have $u^\sigma_p(h) = \pi^\sigma_p(h)v^\sigma_p(h)$. The CFV of an information set $I \in \mathcal{I}_p$ is $v^\sigma_p(I) = \sum_{h\in I}(\pi^\sigma_{-p}(h) \sum_{z\in\mathcal{Z},h\sqsubseteq z}(\pi^\sigma(z|h)u_p(z)))$ and the CFV of taking an action a at $I \in \mathcal{I}_p$ is $v^\sigma_p(I, a) = \sum_{h\in I}(\pi^\sigma_{-p}(h) \sum_{z\in\mathcal{Z},h\sqsubseteq z}(\pi^\sigma(z|(h \cdot a))u_p(z)))$.

Public Belief State Intuitively, the common knowledge of an IIEFG should include the player's strategies (Brown et al., 2020). Public belief state (PBS) is an assumption that treats players' strategies as common knowledge for reducing the state of large IIEFGs significantly, e.g., (Burch et al., 2014; Sustr et al., 2019; Kovarík & Lisý, 2019; Brown et al., 2020). Specifically, we define player p's observation-action history (infostate) as $O_p = (I_1, a_1, I_2, a_2, \cdots)$,[2] which includes the information sets visited and actions taken by p. The unique infostate corresponding to a state $h \in \mathcal{H}_p$ for player p is $O_p(h)$. The set of states that correspond to O_p is denoted $\mathcal{H}(O_p)$. We use $\sim$ to denote states indistinguishable by some player, i.e., $g \sim h$ means $\bigvee_{i=1}^N O_i(g) = O_i(h)$ ($\bigvee$ is the OR operation on all expressions). A public partition is any partition $\mathcal{PS}$ of $\mathcal{H}\backslash\mathcal{Z}$ whose elements are closed under $\sim$ and form a tree Johanson et al. (2011). An element $PS \in \mathcal{PS}$ is called a public state that include the public information that each player knows. The unique public state of a state h and an infostate O_p is denoted by $PS(h)$ and $PS(O_p)$, respectively. The set of states that match the public information of PS is denoted as $\mathcal{H}(PS)$.

In general, a PBS β is described by the joint probability distribution of the possible infostates of the players (Nayyar et al., 2013; Oliehoek, 2013; Dibangoye et al., 2016). Formally, given a public state PS, $\mathcal{O}_p(PS)$ is the set of infostates that player p may be in, and $\triangle\mathcal{O}_p(PS)$ is a probability distribution over the elements of $\mathcal{O}_p(PS)$. Then, PBS $\beta = (\triangle\mathcal{O}_1(PS), \cdots, \triangle\mathcal{O}_N(PS))$. The public state of PBS β is denoted as $PS(\beta)$. The acting player at PBS β is denoted $\mathcal{P}(\beta)$. The available actions for acting player at PBS β is denoted $\mathcal{A}(\beta)$, and the action abstraction at PBS β is denoted $\mathcal{AA}(\beta)$.

A subgame can be rooted at a PBS because PBS is a state of the perfect-information belief-representation game (Brown et al., 2020). At the beginning of a subgame, a state is sampled from the joint probability distribution of the PBS, and then the game plays as if it is the original game. The value for player p of PBS β (PBS value) when all players play according to σ is $v^\sigma_p(\beta) = \sum_{h\in\mathcal{H}(PS(\beta))}(\pi^\sigma(h|\beta)v^\sigma_p(h))$. The value for an infostate $O_p \in \beta$ when all players play according to σ is $v^\sigma_p(O_p|\beta) = \sum_{h\in\mathcal{H}(O_p)}(\pi^\sigma(h|O_p, \beta_{-p})v^\sigma_p(h))$ where $\pi^\sigma(h|O_p, \beta_{-p})$ is the probability of reaching state h according to σ assuming O_p is reached and the joint probability distribution over infostates for player other than p is β_{-p}.

4 A NOVEL MDP FORMULATION FOR IIEFGS

Below, we present our novel MDP formulation for IIEFGs. It is important to note that our formulation is an abstract MDP model, designed to determine the action abstraction of IIEFGs, based on which we perform a CFR algorithm to solve for the optimal strategy. Thus, it does not correspond exactly to the actual game dynamics in IIEFGs.

[2]Observation-action history is a kind of information set introduced in (Burch et al., 2014).

In general, a Markov Decision Process (van Otterlo & Wiering, 2012) (MDP) consists of the tuple $\langle \mathbf{S}, \mathbf{A}, P, r, \gamma \rangle$, where $\mathbf{S}$ is the set of states, $\mathbf{A}$ is the set of actions, $r : \mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ is the reward function, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the state transfer function, and γ is the discount factor. The objective is to find an optimal control policy π^*, which determines $\mathbf{a}_t = \pi^*(\mathbf{s}_t)$ at each time, to maximize the expected cumulative reward $R = \mathbb{E}\{\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\}$.

New state, action and reward function for IIEFGs. We now specify the state $\mathbf{s}$, the action $\mathbf{a}$ and the value r of our MDP formulation. Our design is inspired by (Brown et al., 2019), which transforms high-dimensional public belief states into low-dimensional public states.

(**State**) The dimension of the PBS is generally large because it needs to record the distribution of infostates. To reduce the dimension of the state, we use the public state as the state in MDP for a PBS β, denoted as $\mathbf{s} = PS(\beta)$. The public state needs to record public information known to both players, and the dimensionality is generally not very large.[3] The selection of public states has the additional advantage that the public states of the non-root nodes are fixed during the CFR iterations, while the PBS of the non-root nodes can change during the CFR iterations.

(**Action**) We design novel control actions for our abstract MDP to represent different action abstractions in the IIEFGs, based on which we will build a game tree for the CFR solving.

In IIEFG, there are some actions that are common and are added to the action abstraction no matter what the PBS β is. We define such a set of actions as always-selected action set $\mathcal{AA}_{always}(\beta)$, and $\mathcal{AA}_{always}(\beta)$ can consist of a few of the most common actions in the set of available actions $\mathcal{A}(\beta)$. $\mathcal{AA}_{always}(\beta)$. Meanwhile, we also define a *default* fixed action abstraction $\mathcal{AA}_{base}(\beta)$ at PBS β. $\mathcal{AA}_{base}(\beta) \subseteq \mathcal{A}(\beta)$ is a set of actions related only to PBS β, and we have $\mathcal{AA}_{always}(\beta) \subseteq \mathcal{AA}_{base}(\beta)$. In general, $\mathcal{AA}_{base}(\beta)$ will be pre-specified to a set of available actions related to the important information of PBS β.

$\mathcal{AA}_{always}(\beta)$ and $\mathcal{AA}_{base}(\beta)$ can be chosen arbitrarily in any IIEFG, although different choices can affect the win-rate and running time, as mentioned in (Moravcík et al., 2017). For example, in HUNL experiments, we can choose $\mathcal{AA}_{always}(\beta) = \{F, C, A\}$ and $\mathcal{AA}_{base}(\beta) = \{F, C, A, 0.5 \times pot, 1 \times pot, 2 \times pot\}$ (same setting as (Moravcík et al., 2017; Brown et al., 2020)), where F, C, A refer to fold, check/call and all-in respectively and $\times pot$ means the fraction of the size of the pot being bet. (Moravcík et al., 2017) shows the win-rate decreases by 96 mbb/h after $10,000$ hands if we make $\mathcal{AA}_{base}(\beta) = \{F, C, A, 1 \times pot\}$, although the running time improves by 6 times.

Action $\mathbf{a}$ in our MDP is used to select an action abstract $\mathcal{AA}_{MDP}(\beta, \mathbf{a})$ at PBS β, and we describe the specifics next. Formally at PBS β, the action abstraction chosen by $\mathbf{a}$ is

$$\mathcal{AA}_{MDP}(\beta, \mathbf{a}) = \mathcal{AA}_{always}(\beta) \cup \mathcal{AA}_{optional}(\beta, \mathbf{a}) \tag{1}$$

where optional action set $\mathcal{AA}_{optional}(\beta, \mathbf{a})$ is the set of actions generated from the PBS β and chosen action vector $\mathbf{a}$. Since the size of the game tree increases exponentially with the number of available actions, we can choose to have K actions to the optional action set $\mathcal{AA}_{optional}(\beta, \mathbf{a})$. The action we design is a $2K$-dimensional vector $\mathbf{a} = (x_1, y_1, \cdots, x_K, y_K)$, with each dimension having a value between -1 and 1. Precisely, $\mathcal{AA}_{optional}(\beta, \mathbf{a}) = \bigcup_{i=1}^{K} f(x_i, y_i, \beta)$, where $f(x_i, y_i, \beta)$ is a function to generate an available action from all available actions except actions in $\mathcal{AA}_{always}$ (specially, if $f(x_i, y_i, \beta) = \emptyset$, there is no action abstraction in this dimension) according to variable x_i, y_i and PBS β. Since the set of available actions of IIEFGs with myriad actions tends to be continuous, to define the function $f(x_i, y_i, \beta)$, we can correspond the continuous parameters x_i, y_i one by one to the set of available actions $\mathcal{A}(\beta)$ of PBS β.

Below, we use HUNL as a concrete example to describe how to choose K and $f(x_i, y_i, \beta)$. For HUNL, we set $K = 3$, which means we can select up to 3 raising scales other than all-in. Based on human experience and inspired by prior studies (Hawkin et al., 2011; 2012), a reasonable range for a raising scale other than all-in is $[0, 5]\times$pot. Thus, we define the $f(x_i, y_i, \beta)$ function to be

$$f(x_i, y_i, \beta) = \begin{cases} CLIP(2.5(x_i + 1) \times pot), & y_i \geq 0; \\ \emptyset, & y_i < 0. \end{cases} \tag{2}$$

[3]For example, in HUNL, a public state only includes the previous actions of the two players, the public cards, chips in the pot, remaining chips and acting player. On the other hand, a public belief state includes $1,326$ different private hands for both two players, and requires $2,652$ more dimensions than a public state.

where $CLIP$ is a function that corresponds the nearest available raising scale.

(**Reward**) For our abstract MDP, the reward of each action needs to be obtained by solving two depth-limited subgames (Brown et al., 2018) (a technique for limiting the size of IIEFG, described in Appendix D) according to CFR-based algorithm ReBeL (Brown et al., 2020), as described in the Appendix E. We now describe how to compute the reward r from the PBS β, the state vector $\mathbf{s}$ and the action vector $\mathbf{a}$.

We first build a game tree rooted at PBS β with selected action abstraction $\mathcal{AA}_{MDP}(\beta, \mathbf{a})$. Based on ReBeL, we then obtain a strategy profile σ_{MDP} for PBS β that gives state transfers for all infostates corresponding to all non-leaf nodes of the subgame. Based on the calculated strategy profile σ_{MDP}, we can calculate PBS value $v_{\mathcal{P}(\beta)}^{\sigma_{MDP}}(\beta)$ for the acting player, which is the expected value calculated for the acting player on PBS β (details of the PBS value calculation are in the last paragraph of Section 3). We then build another game tree rooted at PBS β with default fixed action abstraction $\mathcal{AA}_{base}(\beta)$. Similarly, we obtain the strategy profile σ_{base} for this game tree based on ReBeL, and compute the PBS value $v_{\mathcal{P}(\beta)}^{\sigma_{base}}(\beta)$ for the acting player.

Finally, we define the reward as the PBS value difference between the chosen action abstraction and the default action abstraction, denoted $r(\mathbf{s}, \mathbf{a}) = v_{\mathcal{P}(\beta)}^{\sigma_{MDP}}(\beta) - v_{\mathcal{P}(\beta)}^{\sigma_{base}}(\beta)$.

5 RL-CFR Framework

In this section, we present the RL-CFR framework, which builds upon the ReBeL algorithm (Brown et al., 2020), an efficient method for solving depth-limited subgame (Brown et al., 2018) mentioned in Appendix E.[4] In contrast to ReBeL, which selects a fixed action abstraction, RL-CFR selects a dynamic action abstraction via RL. As we will see in the experiments, doing so allows us to optimize over the set of action abstraction and achieve better performance. It is important to note that applying the DRL approach to IIEFGs is highly nontrivial. The key challenge comes from the fact that one has to decide the action probability distributions for all information sets (Burch, 2017; Brown, 2020), which imposes enormous difficulty in the computation required in training.

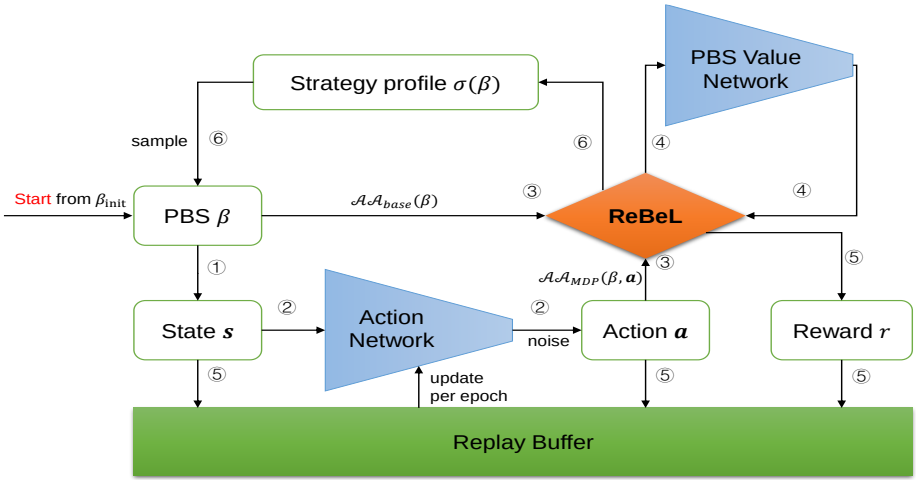

Figure 2: Training procedure for the RL-CFR framework. The labels in the figure correspond to the sampling steps for RL-CFR framework. A sampling epoch starts from the initial PBS β_{init}.

RL-CFR framework is an end-to-end self-training reinforcement learning process, with the procedure shown in Figure 2. We now describe the sampling steps for RL-CFR framework: ① Starting from the initial PBS of the game, each time we handle a non-chance and non-terminate PBS β [5], and we transform a high-dimensional PBS β into a low-dimensional public state $\mathbf{s}$ by the method in

[4] ReBeL can be used to obtain efficient solutions for large IIEFGs by self-play RL and CFR.

[5] If we encounter a chance PBS where acting player is chance player, we let the chance player randomly act and update the PBS. If we encounter a terminate PBS, the epoch of sampling ends.

Section 4. ② Passing through the action network and add a Gaussian noise to obtain action vector $\mathbf{a}$, and this action vector will be mapped to a specified action abstraction $\mathcal{AA}_{MDP}(\beta, \mathbf{a})$. ③ Building two depth-limited subgames rooted at β according to the default action abstraction $\mathcal{AA}_{base}(\beta)$ and selected action abstraction $\mathcal{AA}_{MDP}(\beta, \mathbf{a})$ respectively. ④ Using ReBeL algorithm to solve the strategies and values of the two subgames. ⑤ Calculating PBS value difference as reward r, and adding RL data $\{\mathbf{s}, \mathbf{a}, r\}$ to the training data (denoted as $Data^{RL}$) for action and critic network. ⑥ Randomly choosing a subgame and following the corresponding strategy $\sigma(\beta)$ for state transition to a child PBS β' next. Let $\beta = \beta'$, and repeat step ①. Algorithm 1 shows the formal procedure of sampling process.

Algorithm 1: RL-CFR framework: Sampling (s, a, r) data

Input: $\theta_\alpha, noise, \eta, \epsilon$// $\eta = 0.33, \varepsilon = 0.25$ during training
$\beta \leftarrow \beta_{init}$
$Data^{RL} \leftarrow \{\}$
while *!IsTerminal*(β) **do**
  while $P(\beta) = c$ **do**
    $\beta \leftarrow$ TakeChance(β)// Random chance event
  $\sigma_{base}(\beta), v^{\sigma_{base}}_{\mathcal{P}(\beta)}(\beta) \leftarrow$ReBeL$(\beta, \mathcal{AA}_{base}(\beta))$ // calculate the strategy and value for fixed default action abstraction
  $s \leftarrow PS(\beta)$ // use public state as the state in MDP
  $\mathbf{a} \leftarrow$ActionNetwork$(s, \theta_\alpha) + \mathcal{N}(0, noise)$ // $noise = 0.15$ during training
  $\sigma_{MDP}(\beta), v^{\sigma_{MDP}}_{\mathcal{P}(\beta)}(\beta) \leftarrow$ReBeL$(\beta, \mathcal{AA}_{MDP}(\beta, \mathbf{a}))$// calculate the strategy and value for selected action abstraction
  $r \leftarrow v^{\sigma_{MDP}}_{\mathcal{P}(\beta)}(\beta) - v^{\sigma_{base}}_{\mathcal{P}(\beta)}(\beta)$// reward function
  Add $\{\mathbf{s}, \mathbf{a}, r\}$ to $Data^{RL}$
  $c \sim unif[0, 1]$
  $d \sim unif[0, 1]$
  if $c < \eta$ **then**
    if $d < \epsilon$ **then**
      $nextaction \sim \mathcal{AA}_{base}(\beta)$
    else
      $nextaction \sim \sigma_{base}(\beta)$
    $\beta \leftarrow$NEXTPBS$(\beta, \sigma_{base}(\beta), nextaction)$
  else
    if $d < \epsilon$ **then**
      $nextaction \sim \mathcal{AA}_{limit}(\beta, \mathbf{a})$
    else
      $nextaction \sim \sigma_{MDP}(\beta)$
    $\beta \leftarrow$NEXTPBS$(\beta, \sigma_{MDP}(\beta), nextaction)$
Output: $Data^{RL}$

Thus, after each epoch, we sample a trajectory of $(\mathbf{s}_1, \mathbf{a}_1, r_1, \mathbf{s}_2, \mathbf{a}_2, r_2, \cdots, \mathbf{s}_t, \mathbf{a}_t, r_t)$ based on the current action network, where t is the length of the game and depends on the player actions. After collecting a number of RL data $Data^{RL} = \{(\mathbf{s}, \mathbf{a}, r)\}$ in several epochs, we use the Actor-Critic algorithm (Konda & Tsitsiklis, 1999) and MSE Loss to train the action network and critic network (These network structures are described in the Section 6). The loss function is as follows:

$$\mathcal{L}(\theta_c) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r)\sim Data^{RL}}[(r^{\theta_c}(\mathbf{s}, \mathbf{a}) - r)^2], \quad \mathcal{L}(\theta_a) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r)\sim Data^{RL}}[-r^{\theta_c}(\mathbf{s}, \mathbf{a}^{\theta_a}(s))], \tag{3}$$

where θ_c, θ_a are the parameters of critic network and action network.

In the early epochs of training, the action network selects an action abstraction that is often not as good as the default fixed action abstraction $\mathcal{AA}_{base}(\beta)$. Thus, we begin the training by choosing the default action abstraction when building the depth-limited subgame except for the root node. After a period of training the action abstraction chosen by the action network will outperform the default action abstraction, at which point, when building the depth-limited subgame, we choose the action abstraction for the child nodes based on the action network. Meanwhile, in order to get a more accurate PBS value, we can retrain the PBS value network according to the action abstraction

selected by the action network. Theoretically, the PBS value network and action network can be repeatedly updated for training.

6 EXPERIMENT

To illustrate the effectiveness of our RL-CFR framework on large IIEFGs with myriad actions, our conduct experiments on Heads-up No-limit Texas Hold'em (HUNL), similar to many prior studies on large IIEFGs (Brown et al., 2019; 2020; Zarick et al., 2020). The rules of HUNL are provided in Appendix A. During the evaluation, both players start with 200 big blinds, and the two players will switch their positions and private hands in every two hands, as similarly done in annual computer poker competition (ACPC) (Bard et al., 2013).

Our experiments were run on a compute server with 4 NVIDIA PH402 SKU 200 GPUs and an 80-core Intel(R) Xeon(R) Gold 6145 2.00GHz CPU. All neural networks in our implementation consist of MLPs (size specified below) with ReLU (Glorot et al., 2011) activation functions, and are trained with Adam (Kingma & Ba, 2015). In the CFR iteration to solve a PBS, we let the number of iterations $T = 250$ during training and evaluation. [6]

A PBS value network has 6 layers and 18 million parameters, of which the input layer has $2,678$ dimensions (corresponding to all possible private hands of the two players and the public state information). Each hidden layer has $1,536$ dimensions and the output layer has $2,652$ dimensions (corresponding to all possible private hands of the two players). During the training process, we sample 4.8×10^7 PBS data in total. We randomly sampled data from the last 1×10^7 PBS data and set a learning rate of 1×10^{-5} and a batch size of 512 during training. The training process and the data sampling process are performed simultaneously. Specifically, the data generation process is run in parallel in 60 threads, and the training process is run continuously on a single GPU. After the training of PBS value networks, we obtained a replication version of ReBeL as a baseline. In addition, the PBS value networks used for all our experiments (including the PBS value network used for RL-CFR) are trained based on the default action abstraction.[7]

The action network and the critic network both have 3 layers and 2×10^4 parameters, with hidden layers of 128 and 96 dimensions. The training process has 2×10^6 epochs, each sampling approximately 10 RL data. [8] We randomly sample data from all RL data, and set a learning rate of 1×10^{-5} and a batch size of $1,024$ in training. After 5×10^5 epochs, we generated PBS Data by building the game tree exactly according to the action abstraction given by the action network. We generate data in parallel on 60 threads while training on a single GPU.

We evaluate the performance of RL-CFR and ReBeL under the common knowledge in HUNL. In detail, the common knowledge in HUNL is that the agent know each other's hand ranges and historical actions (such an evaluation procedure has also been adopted in Burch et al. (2018); Li et al. (2020); Kovarík & Lisý (2019)). Hence, we can avoid actions that are not in the action abstraction. As shown in Table 1, after performing $600,000$ hands, RL-CFR achieves 64 mbb/hand win-rate compared to the replication of ReBeL.

We further compare RL-CFR against the open source AI Slumbot (Jackson, 2013), which was the winner of the 2018 ACPC and is the only HUNL AI we know that offers online competition testing. Since the opponent may select actions that deviate from the game tree, we perform nested subgame solving technique (Billings et al., 1998; Brown & Sandholm, 2017b; Moravcík et al., 2017) mentioned in Appendix D. We play RL-CFR for $250,000$ hands against Slumbot, and the test results are shown in Table 1, which illustrate that the replication of ReBeL beat Slumbot with a win-rate of 16 mbb/hand, while RL-CFR beat Slumbot with a win-rate of 84 mbb/hand, and the win-rate of RL-CFR against Slumbot improved by 68 mbb/hand relative to ReBeL. Note that a win-rate of over

[6] Since HUNL evaluations are generally time-limited and need to be solved within a few seconds, common poker AIs typically take between 100 to 1000 CFR iterations (Brown et al., 2015; Bowling et al., 2017; Brown & Sandholm, 2017a; Moravcík et al., 2017; Brown et al., 2020).

[7] This setting is to illustrate that the performance improvement achieved by RL-CFR is entirely due to the action abstraction chosen by the action network.

[8] A RL Data $(\mathbf{s}, \mathbf{a}, r)$ consists of a 64-dimensional state $\mathbf{s}$ (record public cards, chips, positions and previous actions in HUNL), a 6-dimensional action $\mathbf{a}$ and a scalar r. Since the number of rounds in a HUNL game is not deterministic, a single sample to the terminate state will generally yield around 10 pieces of RL data.

50 mbb/hand in poker is called a significant win-rate (Bowling et al., 2017), and RL-CFR clearly achieves significant win-rate against ReBeL and Slumbot.

Table 1: Competition results of the HUNL AIs against each other, measured in mbb/hand (variance was reduced by AIVAT technique (Burch et al., 2018)).

AI name	ReBeL (replication)	Slumbot
ReBeL (replication)	-	16 ± 16
ReBeL (Brown et al., 2020)	-	45 ± 5
RL-CFR	64 ± 11	84 ± 17

We also conducted an exploitability evaluation in over $10,000$ random river stage states.[9] The exploitability of a strategy σ and a player p is calculated by $expl_p(\sigma) = u_p^{\sigma} - \min_{\sigma^* \in \Sigma_{-p}} u_p^{\langle \sigma_p, \sigma^* \rangle}$ Cesa-Bianchi & Lugosi (2006). RL-CFR's exploitability is 17 mbb/hand and ReBeL's exploitability is 20 mbb/hand. The results indicate that RL-CFR generates action abstractions that are also less likely to be exploited in the context of generating more win-rate.

We perform additional experiments for RL-CFR, the results are shown in Table 2 Here we play against the method of choosing an optimal action abstraction among multiple fixed action abstractions (MUL-ACTION). MUL-ACTION works by choosing an action abstraction that has the greatest value to the root PBS β_r among three action abstractions $\mathcal{AA}_{base1}(\beta_r), \mathcal{AA}_{base2}(\beta_r), \mathcal{AA}_{base3}(\beta_r)$[10] , and compute the strategy based on chosen action abstraction. We see that RL-CFR beats MUL-ACTION by 21 ± 26 mbb/hand win-rate after $100,000$ hands and only requires 1/3 of running time.

We also comapre RL-CFR against choosing a finer-grained action abstraction (FINE-GRAIN) $\mathcal{AA}_{base}(\beta_r) = \{F, C, A, 0.33 \times pot, 0.5 \times pot, 0.75 \times pot, 1.0 \times pot, 1.25 \times pot, 2.0 \times pot\}$ at root PBS β_r (the same setting as in (Zarick et al., 2020)). RL-CFR beats FINE-GRAIN by 23 ± 28 mbb/hand win-rate after $100,000$ hands and only requires approximately 4/7 of running time.[11]

Table 2: Win-rate and solving time of the method relative to RL-CFR.

Method	Win-rate	Running time
ReBeL	-64 ± 11	$1\times$
MUL-ACTION	-21 ± 26	$3\times$
FINE-GRAIN	-23 ± 28	$1.75\times$

7 CONCLUSIONS

In this work, we propose a novel algorithmic solution, named RL-CFR, for solving large-scale IIEFGs. RL-CFR builds on a novel abstract MDP formulation, which uses public information as states, action abstraction features as actions, and a carefully defined reward function. RL-CFR ingeniously combines reinforcement learning for action abstraction selection with CFR, to enable dynamic action abstraction selection in IIEFGs. Through extensive experiments in HUNL benchmark, we show that RL-CFR achieves a significant performance improvement compared to fixed action abstraction methods.

[9] We simulate RL-CFR versus ReBeL until reaching river, i.e., the two agents choose their respective action abstractions, and the performance of the previously chosen action abstraction has no effect on the test results.

[10] We set $\mathcal{AA}_{base1}(\beta_r) = \{F, C, A, 0.5\times pot, 1\times pot, 2\times pot\}$, $\mathcal{AA}_{base2}(\beta_r) = \{F, C, A, 0.25\times pot, 0.5\times pot, 1 \times pot\}$, $\mathcal{AA}_{base3}(\beta_r) = \{F, C, A, 0.33 \times pot, 0.7 \times pot, 1.5 \times pot\}$, and the action abstractions other than the root are the same as in ReBeL.

[11] Since the numbers of non-terminal nodes extended by the root node in the game tree built by FINE-GRAIN and RL-CFR are 7 and 4, respectively, the running time of FINE-GRAIN is approximately 1.75 times of that under RL-CFR.

REFERENCES

Luca Aceto. *Action refinement in process algebras*. PhD thesis, University of Sussex, Falmer, East Sussex, UK, 1991.

Nolan Bard, John Alexander Hawkin, Jonathan Rubin, and Martin Zinkevich. The annual computer poker competition. *AI Mag.*, 34(2):112, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker. In Jack Mostow and Chuck Rich (eds.), *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA*, pp. 493–499. AAAI Press / The MIT Press, 1998.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Commun. ACM*, 60(11):81–88, 2017.

N. Brown and T. Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):eaay2400, 2019a.

Noam Brown. *Equilibrium Finding for Large Adversarial Imperfect-Information Games*. PhD thesis, Carnegie Mellon University, 2020.

Noam Brown and Tuomas Sandholm. Strategy-based warm starting for regret minimization in games. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 432–438. AAAI Press, 2016a.

Noam Brown and Tuomas Sandholm. Baby tartanian8: Winning agent from the 2016 annual computer poker competition. In Subbarao Kambhampati (ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 4238–4239. IJCAI/AAAI Press, 2016b.

Noam Brown and Tuomas Sandholm. Libratus: The superhuman AI for no-limit poker. In Carles Sierra (ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 5226–5228. ijcai.org, 2017a.

Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 689–699, 2017b.

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 1829–1836. AAAI Press, 2019b. doi: 10.1609/aaai.v33i01.33011829.

Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold'em agent. In Sam Ganzfried (ed.), *Computer Poker and Imperfect Information, Papers from the 2015 AAAI Workshop, Austin, Texas, USA, January 26, 2015*, volume WS-15-07 of *AAAI Technical Report*. AAAI Press, 2015.

Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 7674–7685, 2018.

Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR, 2019.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Neil Burch. *Time and Space: Why Imperfect Information Games are Hard*. PhD thesis, University of Alberta, 2017.

Neil Burch and Michael Bowling. CFR-D: solving imperfect information games using decomposition. *CoRR*, abs/1303.4441, 2013.

Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In Carla E. Brodley and Peter Stone (eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 602–608. AAAI Press, 2014.

Neil Burch, Martin Schmid, Matej Moravcik, Dustin Morrill, and Michael Bowling. AIVAT: A new variance reduction technique for agent evaluation in imperfect information games. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 949–956. AAAI Press, 2018.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006. ISBN 978-0-521-84108-5. doi: 10.1017/CBO9780511546921.

B. Chen and J. Ankenman. The mathematics of poker. 2007.

Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving dec-pomdps as continuous-state mdps. *J. Artif. Intell. Res.*, 55:443–497, 2016. doi: 10.1613/jair.4623.

Sam Ganzfried and Tuomas Sandholm. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In Francesca Rossi (ed.), *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pp. 120–128. IJCAI/AAAI, 2013.

Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In Carla E. Brodley and Peter Stone (eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 682–690. AAAI Press, 2014.

Sam Ganzfried and Tuomas Sandholm. Endgame solving in large imperfect-information games. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind (eds.), *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pp. 37–45. ACM, 2015.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pp. 315–323. JMLR.org, 2011.

Keigo Habara, Ellen Hidemi Fukuda, and Nobuo Yamashita. Convergence analysis and acceleration of the smoothing methods for solving extensive-form games. *CoRR*, abs/2303.11046, 2023. doi: 10.48550/arXiv.2303.11046.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Game Theory and Information*, 1997.

John Alexander Hawkin, Robert Holte, and Duane Szafron. Automated action abstraction of imperfect information extensive-form games. In Wolfram Burgard and Dan Roth (eds.), *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

John Alexander Hawkin, Robert Holte, and Duane Szafron. Using sliding windows to generate action abstractions in extensive-form games. In Jörg Hoffmann and Bart Selman (eds.), *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.

Johannes Heinrich. *Reinforcement learning from self-play in imperfect-information games*. PhD thesis, University College London, UK, 2017.

Samid Hoda, Andrew Gilpin, Javier Peña, and Tuomas Sandholm. Smoothing techniques for computing nash equilibria of sequential games. *Math. Oper. Res.*, 35(2):494–512, 2010. doi: 10.1287/moor.1100.0452.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

Mark Humphreys. *Action selection methods using reinforcement learning*. PhD thesis, University of Cambridge, UK, 1997.

Eric Jackson. Slumbot nl: Solving large games with counterfactual regret minimization using sampling and distributed processing. In *AAAI Workshop on Computer Poker and Imperfect Information*, 2013.

Michael Johanson. Measuring the size of large no-limit poker games. *CoRR*, abs/1302.7008, 2013.

Michael Johanson, Kevin Waugh, Michael H. Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In Toby Walsh (ed.), *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pp. 258–265. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-054.

Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive-form games. In Jörg Hoffmann and Bart Selman (eds.), *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.

Michael Johanson, Neil Burch, Richard Anthony Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker (eds.), *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pp. 271–278. IFAAMAS, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1008–1014. The MIT Press, 1999.

Vojtech Kovarík and Viliam Lisý. Value functions for depth-limited solving in zero-sum imperfect-information games. *CoRR*, abs/1906.06412, 2019.

Vojtěch Kovaík and Viliam Lis. Value functions for depth-limited solving in zero-sum imperfect-information games. 2019.

Dennis Lee, Haoran Tang, Jeffrey O. Zhang, Huazhe Xu, Trevor Darrell, and Pieter Abbeel. Modular architecture for starcraft II with deep reinforcement learning. In Jonathan P. Rowe and Gillian Smith (eds.), *Proceedings of the Fourteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2018, November 13-17, 2018, Edmonton, Canada*, pp. 187–193. AAAI Press, 2018.

Kai Li, Hang Xu, Meng Zhang, Enmin Zhao, Zhe Wu, Junliang Xing, and Kaiqi Huang. Openholdem: An open toolkit for large-scale imperfect-information game research. *CoRR*, abs/2012.06168, 2020.

Mingyang Liu, Asuman E. Ozdaglar, Tiancheng Yu, and Kaiqing Zhang. The power of regularization in solving extensive-form games. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Charles A. G. Madeira, Vincent Corruble, and Geber L. Ramalho. Designing a reinforcement learning-based adaptive AI for large-scale strategy games. In John E. Laird and Jonathan Schaeffer (eds.), *Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference, June 20-23, 2006, Marina del Rey, California, USA*, pp. 121–123. The AAAI Press, 2006.

Linjian Meng, Zhenxing Ge, Pinzhuo Tian, Bo An, and Yang Gao. An efficient deep reinforcement learning algorithm for solving imperfect information extensive-form games. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 5823–5831. AAAI Press, 2023. doi: 10.1609/aaai.v37i5.25722.

Matej Moravcík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael H. Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR*, abs/1701.01724, 2017.

John Nash. *5. Equilibrium Points in n-Person Games*. 01 2002. ISBN 9781400884087.

A. Nayyar, A. Mahajan, and D. Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7): 1644–1658, 2013.

Abraham Neyman. Existence of optimal strategies in markov games with incomplete information. *Int. J. Game Theory*, 37(4):581–596, 2008.

Frans Adriaan Oliehoek. Sufficient plan-time statistics for decentralized pomdps. In Francesca Rossi (ed.), *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pp. 302–308. IJCAI/AAAI, 2013.

M. J. Osborne and A. Rubinstein. A course in game theory. *MIT Press Books*, 1, 1994.

Jonathan Rubin and Ian D. Watson. Computer poker: A review. *Artif. Intell.*, 175(5-6):958–987, 2011.

Martin Schmid, Matej Moravcik, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, and Zach Holland. Player of games. 2021.

David Schnizlein, Michael H. Bowling, and Duane Szafron. Probabilistic state translation in extensive games with large action sets. In Craig Boutilier (ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pp. 278–284, 2009.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

Peter A. Streufert. A category for extensive-form games. *CoRR*, abs/2105.11398, 2021.

Michal Sustr, Vojtech Kovarík, and Viliam Lisý. Monte carlo continual resolving for online strategy computation in imperfect information games. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (eds.), *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pp. 224–232. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054, 1998.

Oskari Tammelin. Solving large imperfect information games using CFR+. *CoRR*, abs/1407.5042, 2014.

Martijn van Otterlo and Marco A. Wiering. Reinforcement learning and markov decision processes. In Marco A. Wiering and Martijn van Otterlo (eds.), *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pp. 3–42. Springer, 2012.

Shiheng Wang. Cfr-p: Counterfactual regret minimization with hierarchical policy abstraction, and its application to two-player mahjong. *CoRR*, abs/2307.12087, 2023. doi: 10.48550/arXiv.2307.12087.

Ryan Zarick, Bryan Pellegrino, Noam Brown, and Caleb Banister. Unlocking the potential of deep counterfactual value networks. *CoRR*, abs/2007.10442, 2020.

Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing. Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 4689–4697. AAAI Press, 2022.

Martin Zinkevich, Michael Johanson, Michael H. Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (eds.), *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 1729–1736. Curran Associates, Inc., 2007.

A TEXAS HOLD'EM RULES

At the start of the game (hand), each player is given two cards, which we call "private hand". There are four stages in a game, called pre-flop, flop, turn and river, respectively. There are five public cards in a game, three cards are dealt[12] at the start of the flop and one card is dealt at the start of the turn and the river. Several players have to put in a pre-specified number of chips before the game starts, known as "small blind" and "big blind", and a "small blind" is usually half of a "big blind". In HUNL, the small blind player acts first in pre-flop stage, and the big blind player acts first in other stages. The legal actions are fold, check/call and bet/raise. In No-limit Texas Hold'em, players can bet/raise any number of chips between the last bet/raise in the stage (at least 1 big blind) and their remaining chips (all-in).

At the end of a game, each player who did not fold by the end of all stages chooses the best five cards out of two cards from private hand and the five public cards to compare, and the player (or several players) who has the best hand wins the pot. The win-rate in Texas Hold'em can be expressed as the average number of big blinds won per game, or in more granular units of mbb (Bowling et al., 2017) (one thousandth of a big blind). For example, we can say a win-rate of 0.01 big blind per hand, or 10 mbb/hand (10 mbb per hand).

B RELATED WORK OF TEXAS HOLD'EM AIS

The strongest Texas Hold'em AIs up to date, represented by Libratus (Brown & Sandholm, 2017a) and Pluribus (Brown & Sandholm, 2019a), have beaten top human players in both two-player and multi-player poker. They use complex abstraction (Johanson et al., 2012; Ganzfried & Sandholm, 2014; Brown et al., 2015) to reduce the huge decision space in Texas Hold'em and consume enormous computing resources to pre-calculate a blueprint strategy (Brown & Sandholm, 2016a) by CFR under a huge game tree.

DeepStack (Moravcík et al., 2017) employs deep learning to estimate values of private hands in game tree, thus reducing the size of the game tree and speeding up the CFR (Johanson, 2013). ReBeL (Brown et al., 2020) is an upgraded version of DeepStack that uses self-play reinforcement learning (Heinrich, 2017) to generate more realistic training data. However, Libratus and Pluribus do not use reinforcement learning, and DeepStack and ReBeL do not use reinforcement learning in action selection (raising scales) in HUNL.

(Zhao et al., 2022) designs a HUNL AI based on reinforcement learning, which allows for an excellent AI with very few computational resources. However, since it does not use the widely used CFR algorithm in HUNL at all, the method is fairly exploitable and has no theoretical guarantees at all.

C COUNTERFACTUAL REGRET MINIMIZATION

Counterfactual Regret Minimization (CFR) is an algorithm for large IIEFGs that minimizes regret in each information set independently (Zinkevich et al., 2007), and can find ε-Nash equilibrium in two-player zero-sum IIEFGs.

[12]Poker term, means to open a new card.

Let σ^t be the strategy profile of iteration t. The instantaneous regret for taking an action a at information set $I \in \mathcal{I}_p$ on iteration t is $r^t(I,a) = v_p^{\sigma^t}(I,a) - v_p^{\sigma^t}(I)$. The counterfactual regret for taking an action a at I on iteration T is $R^T(I,a) = \sum_{t=1}^{T} r^t(I,a)$. The counterfactual regret is used for regret matching (RM) (Hart & Mas-Colell, 1997), a no-regret learning algorithm for solving imperfect-information game.

For an information set I, on each iteration $t+1$, an action $a \in \mathcal{AA}(I)$ is selected according to probabilities $\sigma^{t+1}(I,a) = \frac{R_+^t(I,a)}{\sum_{a' \in \mathcal{AA}(I)} R_+^t(I,a')}$ where $R_+^t(I,a) = \max\{0, R^t(I,a)\}$. If $\sum_{a' \in \mathcal{AA}(I)} R_+^t(I,a') = 0$, then we can chose an arbitrary strategy. In general, the upper bound on the regret value of the CFR or its variants (Burch & Bowling, 2013; Brown & Sandholm, 2017a; 2019b) is $O(L\sqrt{|\mathcal{AA}(I)|}\sqrt{T})$, where L is the range of payoffs, $|\mathcal{AA}(I)|$ is the size of action abstraction for information set I and T is the number of iterations (Cesa-Bianchi & Lugosi, 2006).

Discounted CFR (DCFR) (Brown & Sandholm, 2019b) is the leading equilibrium-finding algorithm for large IIEFGs (Brown, 2020). DCFR is an variant of CFR with parameters α, β, γ (DCFR$_{\alpha,\beta,\gamma}$), defined by multiplying accumulated positive regrets by $\frac{t^\alpha}{t^\alpha+1}$, negative regrets by $\frac{t^\beta}{t^\beta+1}$ and contributions to the average strategy $\overline{\sigma}$ by $(\frac{t}{t+1})^\gamma$ on each iteration t. Our experiment setting is $\alpha = \frac{3}{2}, \beta = \frac{1}{2}$ and $\gamma = 2$, denoted $DCFR_{\frac{3}{2},\frac{1}{2},2}$.

D ABSTRACTION

The huge solution complexity of IIEFGs is reflected in 3 dimensions: the depth of the game D, the size of the information set $|I|$ and the number of available actions $|\mathcal{A}(I)|$. In fact, the origin space complexity is $O(|\mathcal{A}(I)|^D \cdot |I|)$, which is over the order of 10^{160} for HUNL with stacks of 200 big blinds and $20,000$ chips (Johanson, 2013). The time complexity of CFR to solve an IIEFG is $O(T \cdot |\mathcal{A}(I)|^D \cdot |I|)$ where T is the number of iterations.

To limit the depth of the game, we generally do not compute the strategy to the end of the game, but instead generate a depth-limited subgame (Brown et al., 2018) that extends only a limited number of states into the future. We estimate the strategy or expected value of leaf states, which are non-terminal states in the full game but terminal states in the depth-limited subgame. In HUNL, Libratus Brown & Sandholm (2017a) consumes enormous computing resources to pre-calculate a blueprint strategy Brown & Sandholm (2016a), thus avoiding solving the first two stages when the game is deep. DeepStack (Moravcík et al., 2017) and ReBeL (Brown et al., 2020) employs deep learning to estimate counterfactual values of leaf states, thus avoiding solving until the end of the game.

To limit the size of the information set, we can put similar states into the same bucket (state-space abstraction) (Johanson et al., 2012; 2013; Brown et al., 2015) or represent the states in a high-dimensional feature abstraction (Brown et al., 2019).[13] State-space abstractions need to be carefully designed to the specific game, and in order to illustrate the generality of our method to general EFGs, our experiments do not use any state-space abstractions.

To limit the number of available actions, it is common to use action abstraction in IIEFGs (Brown & Sandholm, 2016b). Formally, $\mathcal{AA}(I)$ is the set of available actions at information set I, and $\mathcal{AA}(I) \subseteq \mathcal{A}(I)$ is an action abstraction for $\mathcal{A}(I)$. If the opponent chooses an off-tree action a that is not in the action abstraction $\mathcal{AA}(I)$, we can round off-tree action to a nearby in-abstraction action (Schnizlein et al., 2009; Ganzfried & Sandholm, 2013) or resolve the strategy based on new action abstraction $\mathcal{AA}(I) \cup \{a\}$ (nested subgame solving (Ganzfried & Sandholm, 2015; Brown & Sandholm, 2017b)).

E SOLVING THE STRATEGY AND PBS VALUE FOR LARGE EFGS

In this section, we introduce the training process of ReBeL algorithm (Brown et al., 2020), a self-play RL method for solving the strategy and PBS values for large IIEFGs. Meanwhile, we take HUNL as an example to describe the setting of specific parameters.

[13] These methods also reduce the number of nodes in the game tree by putting similar nodes into same bucket.

In each epoch, we start training from the initial state of the game, and the PBS corresponding to the initial state is denoted as β_{init}. During training, we will deal with a PBS β_r and corresponding action abstraction $\mathcal{AA}(\beta_r)$. We need to compute the PBS value $v(\beta_r)$ and sample to the a leaf PBS β_z. Algorithm 2 shows the details and we describe the training process next.

Algorithm 2: ReBeL (Brown et al., 2020) algorithm [14]: Solving the strategy and PBS value for PBS β_r with action abstraction $\mathcal{AA}(\beta_r)$

Function *ReBeL*$(\beta_r, \mathcal{AA}(\beta_r))$**:**
- $G \leftarrow$ConstructSubgame$(\beta_r, \mathcal{AA}limit(\beta_r))$
- $\overline{\sigma}, \sigma^0 \leftarrow$UniformPolicy$(\beta_r, \mathcal{AA}(\beta_r))$
- $\mathbf{v}(\beta_r) \leftarrow \mathbf{0}$
- $t_{sample} \sim unif\{1, T\}$// Sample next iteration
- **for** $t = 1 \cdots T$ **do**
  - $G \leftarrow$LeafValueEstimate$(G, \sigma^{t-1}, \theta)$// θ is the parameters of PBS value network
  - $\sigma^t \leftarrow$UpdatePolicy(G, σ^{t-1})
  - $\overline{\sigma} \leftarrow \frac{t-1}{t+1}\overline{\sigma} + \frac{2}{t+1}\sigma^t$// Update average strategy based on $DCFR_{\frac{3}{2},\frac{1}{2},2}$
  - $\mathbf{v}(\beta_r) \leftarrow \frac{t-1}{t+1}\mathbf{v}(\beta_r) + \frac{2}{t+1}\mathbf{v}^{\sigma^t}(\beta_r)$// Update PBS value for all infostates at β_r
  - **if** $t = t_{sample}$ **then**
    - $\beta_{next} \leftarrow$SampleLeaf(G, σ^t)// Sample a leaf PBS
- Add $\{\beta_r, \mathbf{v}(\beta_r)\}$ to $Data^{PBS}$// Add PBS data for training
- $v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}(\beta_r) \leftarrow$ComputeValue$(\mathbf{v}(\beta_r))$// Compute PBS value for acting player at β_r
- **return** $\overline{\sigma}, v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}(\beta_r), \beta_{next}$;

At the beginning of the training, we build a depth-limited subgame rooted with β_r.[15] In the process of building the game tree, when we deal with a non-ternimal and non-leaf node β', we expand the child nodes downwards according to the action abstraction $\mathcal{AA}(\beta')$.[16]

After building the game tree, this subgame is solved by running T iterations of CFR algorithm, and estimating the value of leaf nodes by a learned value network $\hat{v}$ at each iteration based on their PBS. On each iteration t, we first use CFR to determine a strategy profile σ^t in the subgame. Next, the infostate value of a leaf node z is set to $\hat{v}(O_p(z)|\beta_z^{\sigma^t})$, where $\beta_z^{\sigma^t}$ is the PBS at z when players play according to σ^t. Since the estimates of the neural network lead to a non-zero-sum game, we adjust the infostate values at each PBS so that the game satisfies zero-sum property. Also, for some infostates that shall have the same value under the rules of the game, we average their value estimates. PBSs may change every iteration, so the leaf node values may change every iteration. Given σ^t and leaf node values, each infostate in each node has a calculated PBS value,[17] so that we can update the regret and average strategy $\overline{\sigma}$ for CFR algorithm.

After T iterations, we solved the average strategy $\overline{\sigma}$. Based on this strategy, we calculate the PBS values for all infostates $v_p^{\overline{\sigma}}(O_p|\beta_r)$ for root PBS β_r, and denote this vector of PBS values as $\mathbf{v}(\beta_r)$. We then add the PBS data $\{\beta_r, \mathbf{v}(\beta_r)\}$ to the training data (denoted $Data^{PBS}$) for $\hat{v}(\beta_r)$. Meanwhile, we calculate the PBS value $v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}$ of β_r according to calculated value vector $\mathbf{v}(\beta_r)$.[18]

Next, we sample a leaf PBS β_z according to σ^t on a random iteration $t \sim unif\{1, T\}$ where T is the number of iterations, and to ensure more exploration, we can sample random leaf PBS with

[15] In the HUNL experiments, we build the subgame up to the end of the two players' actions in a stage or the end of the chance player's action. This means that an epoch has up to 7 phases, i.e., start of pre-flop, end of pre-flop, start of flop, end of flop, start of turn, end of turn and start of river.

[16] In the HUNL experiments, in order to reduce the size of the game tree, for the non-terminal PBS β other than the root and root's sons, we set $\mathcal{AA}(\beta) = \{F, C, A, 0.8 \times pot\}$.

[17] The details of calculating the PBS value are explained in Section 3.

[18] The ReBeL algorithm itself does not need to compute the PBS value $v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}$, but our RL-CFR framework requires this PBS value as part of the reward function.

probability ε, and modify some public information in sampled PBS for more exploration. [19]. We repeat above processes until the game ends.

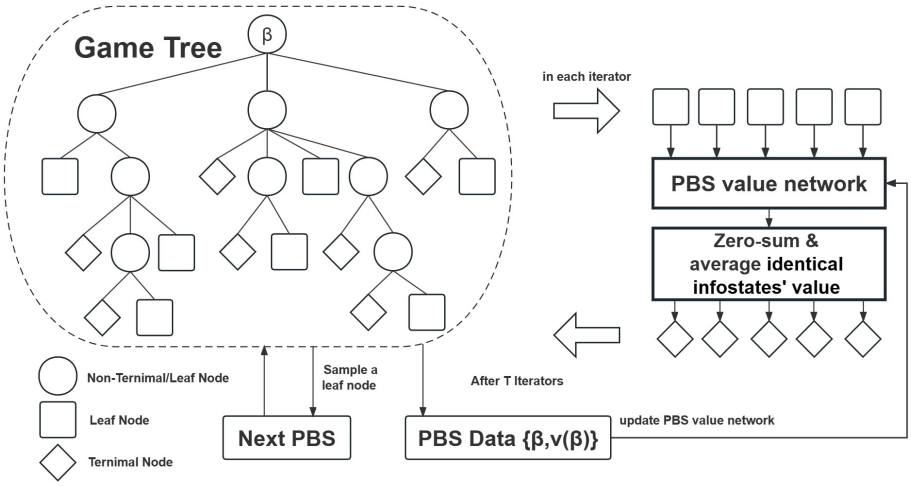

Figure 3: This figure shows how to generate PBS data and train PBS value network. For a PBS β, we build a depth-limit subgame rooted with β. A non-terminal and non-leaf node if represented by a circle, we expand the child nodes according to the action abstraction of the PBS of the node when we are building the game tree. A terminal node is represented by a diamond, we can directly calculate the PBS value for a terminal node. A leaf nodes is represented by a rectangle, and in each iteration of CFR we will use the PBS value network to estimate the PBS values of these leaf nodes (PBS values are re-estimated each iteration since they will be different each time we iterate to these nodes), then we can regard these leaf nodes as terminal nodes in this iteration.

We use Huber Loss (Huber, 1964) as the loss function for the PBS value network:

$$\begin{aligned}\mathcal{L}(\theta,\delta) =& \mathbb{E}_{\{O_p,v_p(O_p)\}\sim\{\beta_r,\mathbf{v}(\beta_r)\},\{\beta_r,\mathbf{v}(\beta_r)\}\sim Data^{PBS}} \\ & [\min\{\frac{1}{2}(v_p(O_p)-\hat{v}^\theta(O_p|\beta_r))^2,\delta|v_p(O_p)-\hat{v}^\theta(O_p|\beta_r)|-\frac{1}{2}\delta^2\}]\end{aligned} \tag{4}$$

where θ is the parameters of PBS value network, O_p is an infostate in PBS β_r, and δ is a hyperparameter of Huber Loss.

In summary, ReBeL is a self-play RL framework capable of continuously generating data from scratch for training, and Figure 3 shows the training process of ReBeL algorithm.

[19]For HUNL agent training, we set $\varepsilon = 25\%$ for HUNL agent training, and for a sampled PBS, we multiply the chips in the pot by a random number between 0.9 and 1.1. For the PBS corresponding to the initial state, we set the chips of all players by a random number between 50 big blinds and 250 big blinds.