RL-CFR: Improving Action Abstraction for Imperfect Information Extensive-Form Games with Reinforcement Learning

Boning Li [1] **Zhixuan Fang** [1 2] **Longbo Huang** [1]

Abstract

Effective action abstraction is crucial in tackling challenges associated with large action spaces in Imperfect Information Extensive-Form Games (IIEFGs). However, due to the vast state space and computational complexity in IIEFGs, existing methods often rely on fixed abstractions, resulting in sub-optimal performance. In response, we introduce RL-CFR, a novel reinforcement learning (RL) approach for dynamic action abstraction. RL-CFR builds upon our innovative Markov Decision Process (MDP) formulation, with states corresponding to public information and actions represented as feature vectors indicating specific action abstractions. The reward is defined as the expected payoff difference between the selected and default action abstractions. RL-CFR constructs a game tree with RL-guided action abstractions and utilizes counterfactual regret minimization (CFR) for strategy derivation. Impressively, it can be trained from scratch, achieving higher expected payoff without increased CFR solving time. In experiments on Heads-up No-limit Texas Hold'em, RL-CFR outperforms ReBeL's replication and Slumbot, demonstrating significant win-rate margins of 64 ± 11 and 84 ± 17 mbb/hand, respectively.

1. Introduction

The Imperfect Information Extensive-Form Game (IIEFG) model provides a comprehensive framework for analyzing multi-player turn-taking games represented in tree structures (Burch, 2017). This model encompasses diverse games such as Poker (Brown & Sandholm, 2019a), Mahjong (Li et al., 2020), and Scotland Yard (Schmid et al., 2023). Resolving IIEFGs involves determining Nash equilibrium (Nash Jr, 1950), particularly in scenarios featuring two-person zero-sum conditions. The popular method for tackling large IIEFGs is to approximate Nash equilibrium using iterative algorithms (Nesterov, 2005; Zinkevich et al., 2007; Chaudhuri et al., 2009). Among these iterative algorithms, Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2007) and its variants (Lanctot et al., 2009; Tammelin, 2014; Brown et al., 2019; Brown & Sandholm, 2019b; Xu et al., 2024) have been the predominant methods for addressing large IIEFGs, yielding low-exploitability mixed strategies.

However, numerous IIEFGs exhibit a vast array of actions, leading to an exponential growth in the size of the game tree with the increasing number of actions (Schnizlein et al., 2009). This poses a significant computational challenge when applying CFR-based solutions directly. To address this, action abstraction, involving the selection of a limited number of actions from the available set (Aceto, 1991), has been widely employed to substantially reduce the size of the game tree, facilitating more efficient CFR solving.

Nevertheless, in the realm of IIEFGs, existing results predominantly focus on fixed action abstractions (Moravčík et al., 2017; Brown et al., 2020). The adoption of fixed action abstractions unavoidably leads to sub-optimality (Waugh et al., 2009). While methods for dynamic action abstraction exist (Hawkin et al., 2011; 2012; Brown & Sandholm, 2014), these approaches often suffer from poor convergence and limited applicability (Brown, 2020). Consequently, identifying strategies capable of achieving dynamic action abstractions with manageable computational complexity remains an outstanding challenge (Sandholm, 2015).

Reinforcement Learning (RL) (Humphreys, 1997; Sutton & Barto, 1998) has emerged as a revolutionary method in various games such as Go (Silver et al., 2017), StarCraft II (Lee et al., 2018), and Dota 2 (Berner et al., 2019). However, its application to IIEFGs presents distinctive challenges, primarily arising from two critical features. Firstly, the optimal strategy for an IIEFG typically involves a mixed strategy on its support (Neyman, 2008). RL algorithms, however, are primarily designed for learning deterministic policies (Lillicrap et al., 2016), making them less suitable for directly handling the probabilistic nature of mixed strategies

[1]Tsinghua University, IIIS, Beijing, China [2]Shanghai Qi Zhi Institute, Shanghai, China. Correspondence to: Longbo Huang <longbohuang@tsinghua.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

in IIEFGs. Secondly, the value of an information set in IIEFGs may depend on the chosen strategy (Brown et al., 2018). However, in RL, the value function is generally assumed to be independent of the agent's policy (Watkins & Dayan, 1992), posing a challenge in modeling scenarios where the value of an information set is contingent on the specific strategy being employed. Addressing these challenges requires specialized adaptations of RL techniques or exploration of alternative approaches tailored to the unique characteristics of IIEFGs.

To further clarify these challenges and the motivation for employing RL, consider the simplified poker example (Burch, 2017) depicted in Figure 1. In this scenario, Player 1 has an equal chance of being dealt J or K, while Player 2 is always dealt Q. Both players contribute 1 chip to the pot, resulting in a total of 2 chips, and each player has 2 chips remaining. Player 1 acts first. The Nash equilibrium strategy for Player 1 involves going all-in with K and committing 50% of J, while checking the other 50% of J. If Player 1 declares all-in, the Nash equilibrium strategy for Player 2 is to call and fold with equal probabilities, ensuring that, regardless of Player 1's strategy, Player 2's expected payoff is not reduced. In this equilibrium, Player 1's K expects to win 2 chips, J expects to lose 1 chip, and Player 2 expects to lose 0.5 chips. In contrast, if Player 1 goes all-in with 100% probability, and Player 2's best response strategy is to call with 100% probability, Player 1's K expects to win 3 chips, J expects to lose 3 chips, and player 2 expects to win 0 chips. This example vividly demonstrates the intricate nature of strategies in IIEFGs, where the strategy is likely a mixed strategy, and the chosen strategy impacts the expected values for all players involved.

To surmount the aforementioned challenges and harness the power of RL in sequential decision making, we introduce RL-CFR, a two-phase framework that ingeniously integrates Deep Reinforcement Learning (DRL) (Arulkumaran et al., 2017) with CFR. In the initial phase, we formulate a novel Markov Decision Process (MDP) (van Otterlo & Wiering, 2012) to identify the action abstraction with the highest expected payoff. Within this MDP, the state encapsulates the *public information* of the game, each control action is represented as a feature vector denoting a specific action abstraction, and action rewards are determined as the payoff differences calculated by CFR between the selected action abstractions and a default fixed action abstraction.

Expanding upon this MDP, we construct a game tree based on the action abstraction chosen by the actor-critic DRL method (Konda & Tsitsiklis, 1999), ultimately solving the strategy for the selected action abstraction through CFR. RL-CFR offers a principled approach to harness the strengths of both RL and CFR, adeptly addressing challenges related to mixed-strategy and probability-dependent reward scenarios.

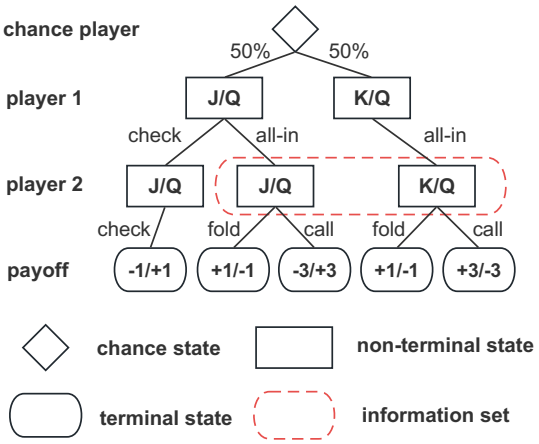

Figure 1. The game starts with a *chance state* where Player 1 faces an equal probability of receiving either J or K, and Player 2 is consistently dealt Q. If Player 1 is dealt K, an automatic all-in is initiated. When Player 1 holds J, a pivotal decision arises between opting for a cautious check or committing to an all-in strategy. Subsequently, if Player 1 opts for an all-in action, Player 2 is confronted with the dilemma of whether to fold or call. Importantly, Player 2 remains uninformed about the specific cards held by Player 1, resulting in an information set that encompasses two states. Upon reaching the terminal state, payoffs are assigned to both players in accordance with a predefined assignment rule.

Importantly, for every state in IIEFG, RL-CFR enhances the expected payoff by selecting a superior dynamic action abstraction compared to a fixed action abstraction without an increase in the CFR solving time or a decrease in convergence. Furthermore, *RL-CFR is capable of being trained from scratch* with only the rules of the IIEFG. This capability is crucial as it allows the algorithm to dynamically adapt to the unique dynamics and intricacies inherent in IIEFGs, mitigating the risk of biases introduced by existing knowledge or inherent biases in the initial model.

To demonstrate the effectiveness of RL-CFR in addressing large IIEFGs, we conducted an evaluation using the challenging Heads-up No-limit Texas Hold'em (HUNL) poker game.[1] Our results show that RL-CFR outperforms the fixed action abstraction-based HUNL agent ReBeL's replication (Brown et al., 2020) with a win-rate margin of 64 milli-big blinds per hand (mbb/hand) in a test of over $600,000$ hands. Furthermore, RL-CFR surpasses the well-known HUNL agent Slumbot (Jackson, 2013) by a substantial win-rate margin of 84 mbb/hand in a test of over $250,000$ hands. These substantial win-rate margins underscore the effectiveness of our innovative RL-CFR solution in tackling the

[1]Heads-up No-limit Texas Hold'em is a two-player variant of Texas Hold'em and serves as a crucial version for exploring mixed strategy two-player zero-sum IIEFGs (Bard et al., 2013). It is chosen for its intricate nature (Rubin & Watson, 2011) and an exceedingly vast decision space (Johanson, 2013).

challenges posed by large IIEFGs. To extend the assessment of RL-CFR beyond HUNL, we devised a new poker variant named PREFLOP43. In the PREFLOP43 poker game, RL-CFR significantly outperformed the fixed action abstraction method in action abstraction evaluation, exploitability evaluation, and heads-up evaluation, with statistical significance.

The main contributions of our work are as follows.

- *Innovative MDP Formulation for IIEFGs*: We present a pioneering Markov Decision Process (MDP) formulation designed specifically for Imperfect Information Extensive-Form Games (IIEFGs). This formulation meticulously defines states based on public information, represents actions as feature vectors denoting action abstractions, and establishes rewards as value differences between selected action abstractions and default fixed action abstractions. The dynamic adjustment of action abstractions at different states is facilitated by our MDP formulation, enhancing adaptability.
- *RL-CFR Framework Integration*: Building upon our novel MDP, we introduce the RL-CFR framework—a novel fusion of Deep Reinforcement Learning (DRL) with Counterfactual Regret Minimization (CFR). This framework achieves a harmonious balance between computation and optimism, and can be trained from scratch with only the rules of the IIEFG. RL-CFR adeptly addresses the challenges posed by the large decision space and computational complexity inherent in IIEFGs, offering a customizable tradeoff mechanism between CFR-related computational complexity and RL-driven performance improvement.
- *Evaluation on HUNL Poker Game*: We evaluate the performance of RL-CFR on the widely recognized Heads-up No-limit Texas Hold'em (HUNL) poker game. Our results showcase the superiority of RL-CFR by achieving significant win-rate advantages over ReBeL's replication, one of the leading fixed action abstraction-based HUNL algorithms, and Slumbot, a robust publicly available HUNL AI for online comparisons. Specifically, RL-CFR outperforms ReBeL's replication and Slumbot by substantial win-rate margins of 64 ± 11 and 84 ± 17 mbb/hand, respectively.

2. Related Work on Extensive-Form Games

Action Abstraction in IIEFGs. The action abstraction technique expedites strategy computation in IIEFGs, providing solutions with theoretical bounds (Kroer & Sandholm, 2014; 2015; 2018). In IIEFGs with a multitude of actions, such as poker games, the impact of action abstraction on strategy quality can be surprisingly significant (Chen & Ankenman, 2007; Waugh et al., 2009). Parametric methods proposed by Hawkin et al. (2011; 2012) aim to find optimal action abstraction for early states of IIEFGs, while an iterative algorithm (Brown & Sandholm, 2014) has been introduced to adjust action abstraction during iteration. However, it is worth noting that these methods, altering action abstraction during CFR iteration, tend to converge more slowly compared to fixed action abstraction methods (Brown, 2020).

Reinforcement Learning (RL) Approaches for IIEFGs. Various methodologies inspired by RL have emerged to solve IIEFGs. Noteworthy contributions include regression counterfactual regret minimization (Waugh et al., 2015; D'Orazio et al., 2020), neural fictitious self-play (Heinrich & Silver, 2016), and ReBeL (Brown et al., 2020). Furthermore, Pérolat et al. (2021) introduced a regularization-based reward adaptation technique, ensuring robust convergence guarantees when addressing two-player zero-sum IIEFGs. In addition, Liu et al. (2023) delved into RL regularization techniques for IIEFGs, proposing a regularization-based payoff function. To tackle the challenge of inaccurate state value estimation in large IIEFGs, Meng et al. (2023) presented an efficient deep reinforcement learning method. Additionally, Xu et al. (2024) enhanced the DCFR algorithm (Brown & Sandholm, 2019b) by replacing the fixed discount parameter with a dynamic discount parameter using RL selection, resulting in improved outcomes. These approaches collectively underscore the diverse applications of RL methodologies in addressing various challenges within the domain of IIEFGs.

3. Background and Notation

Imperfect Information Extensive-Form Games. An Imperfect Information Extensive-Form Game (IIEFG) models the sequential interaction of one or more players (Burch, 2017), and can be represented by $G = \langle \mathcal{H}, \mathcal{Z}, \mathcal{A}, \mathcal{N}, \mathcal{P}, \sigma_c, u, \mathcal{I} \rangle$. Let $\mathcal{N} = \{1, \cdots, N\}$ be the set of players, and $\mathcal{H}$ be the set of states (histories). A state $h \in \mathcal{H}$ is defined by all historical actions from the initial game state $\emptyset$. The state $h \cdot a \in \mathcal{H}$ is a child of the state h, and $h \sqsubseteq h'$ implies that h is an ancestor of h'. $\mathcal{Z}$ is the set of terminal states. For each non-terminal state h, $\mathcal{A}(h)$ is the set of available actions, $\mathcal{AA}(h) \subseteq \mathcal{A}(h)$ is an *action abstraction* for $\mathcal{A}(h)$, and $\mathcal{P}(h) \in \mathcal{N} \cup \{c\}$ determines the acting player, where c denotes the "chance player", representing random events players in $\mathcal{N}$ cannot control. $\sigma_c(h, a)$ is the probability that chance player will act a at state h. $\mathcal{H}_p$ is the set of all states h where $\mathcal{P}(h) = p$. For every terminal state $z \in \mathcal{Z}$, $u(z) = \langle u_p(z) \rangle_{p \in \mathcal{N}}$ gives the payoff for each players. While the work presented in this paper focuses on two-player zero-sum games, where $\mathcal{N} = \{1, 2\}$ and $u_1(z) = -u_2(z)$ for all terminal states z, many of the ideas and techniques developed can be extended to the general-sum and multi-player scenarios as well.

The information-partition $\mathcal{I} = (\mathcal{I}_p)_{p\in\mathcal{N}}$ describes the imperfect information of the IIEFG, 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. A behaviour strategy $\sigma_p \in \Sigma_p$ is a function where $\sigma_p(I, a) \in \mathbb{R}$ determines the probability distribution over available actions $a \in \mathcal{A}(I)$ for every information set $I \in \mathcal{I}_p$. We denote $\sigma(I, a) = \sigma_{\mathcal{P}(I)}(I, a)$. $\sigma = \langle\sigma_p\rangle_{p\in\mathcal{N}}$ is a strategy profile. $\pi^\sigma(h)$ 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 player p takes actions to reach h and other players follow σ.

The counterfactual value (CFV) (Zinkevich et al., 2007) of an information set $I \in \mathcal{I}_p$ is the expected utility for player p given that I has been reached, calculated as $v^\sigma_p(I) = \sum_{h\in I}(\pi^\sigma_{-p}(h)\sum_{z\in\mathcal{Z},h\sqsubseteq z}(\pi^\sigma(z|h)u_p(z)))$. The exploitability $e_p(\sigma)$ of a strategy profile σ and a player p in a two-player zero-sum game describes how much worse the strategy does versus a best response strategy. Formally, $e_p(\sigma) = u^\sigma_p - \min_{\sigma^*_{-p}\in\Sigma_{-p}} u_p^{\langle\sigma_p,\sigma^*_{-p}\rangle}$ (Cesa-Bianchi & Lugosi, 2006). If no player has exploitability higher than ε under σ, then σ is an ε-Nash equilibrium strategy.

Counterfactual Regret Minimization. Counterfactual Regret Minimization (CFR) is an algorithm tailored for large IIEFGs, aimed at minimizing regret independently within each information set (Zinkevich et al., 2007). It is capable of finding ε-Nash equilibrium in two-player zero-sum IIEFGs.

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

For an information set I, on each iteration $t + 1$, an action $a \in \mathcal{AA}(I)$ is chosen based on 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 $\sum_{a'\in\mathcal{AA}(I)} R^t_+(I, a') = 0$, an arbitrary strategy can be chosen. Generally, the upper bound on regret values for CFR or its variants (Lanctot et al., 2009; Tammelin, 2014; Brown et al., 2019; Brown & Sandholm, 2019b) is $O(L\sqrt{|\mathcal{AA}(I)|}\sqrt{T})$, where L is the payoff range, $|\mathcal{AA}(I)|$ 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) stands out as a prominent equilibrium-finding algorithm for large IIEFGs (Brown, 2020). DCFR is a variant of CFR with parameters α, β, γ (DCFR$_{\alpha,\beta,\gamma}$). Specifically, accumulated positive regrets are multiplied 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. In our experiments, we use the DCFR algorithm and set $\alpha = \frac{3}{2}$, $\beta = 0$, and $\gamma = 2$, denoted as DCFR$_{\frac{3}{2},0,2}$.

Public Belief State. A Public Belief State (PBS) is an extended notion of history for IIEFGs based on the common knowledge belief distribution over histories (Burch et al., 2014; Sustr et al., 2019). Specifically, we define player p's observation-action history (infostate) (Burch et al., 2014) as $O_p = (I_1, a_1, I_2, a_2, \cdots)$, which includes the information sets visited and actions taken by player 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 includes the public information that each player knows. The unique public state for a state h and an infostate O_p are 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., 2013), and can shed extraneous history to refine information (Brown et al., 2020). 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 set of 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 with well-defined values (Brown et al., 2020; Kovarík et al., 2023). At the beginning of a subgame, a history is sampled from the 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 defined as $v^\sigma_p(\beta) = \sum_{h\in\mathcal{H}(PS(\beta))}(\pi^\sigma(h|\beta)v^\sigma_p(h))$. The value for an infostate $O_p \in \beta$ when all players play according to σ is defined as $v^\sigma_p(O_p|\beta) = \sum_{h\in\mathcal{H}(O_p)}(\pi^\sigma(h|O_p, \beta_{-p})v^\sigma_p(h))$ 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 probability distribution over infostates for players other than p is β_{-p}.

4. A Novel MDP Formulation for IIEFGs

In this section, we present our novel MDP formulation tailored for IIEFGs. It is important to emphasize that our formulation serves as an abstract MDP model, strategically designed to determine the action abstraction of IIEFGs. This action abstraction, once determined, becomes the basis for executing a CFR algorithm, allowing us to solve for the mixed strategy. In this MDP, states correspond to public information, and actions are represented as feature vectors indicating specific action abstractions. The reward is defined as the expected payoff difference between the selected and default action abstractions. Below, we delve into the specific details of our MDP.

In general, a Markov Decision Process (MDP) (van Otterlo & Wiering, 2012) comprises 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 primary objective is to find an optimal control policy π^*, which determines $\mathbf{a}_t = \pi^*(\mathbf{s}_t)$ at each time, aiming to maximize the expected cumulative reward $R = \mathbb{E}\{\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\}$.

Action Abstraction MDP for IIEFGs.

(**State**) State $\mathbf{s} = PS(\beta)$, where $PS(\beta)$ is the public state of the current PBS β. Our design offers two notable advantages compared to using PBS β directly as a state:

1. *Dimension Reduction.* Our state effectively reduces the dimensionality. In typical IIEFGs, the PBS typically has a large dimensionality, as it needs to record the distribution of all possible infostates. In contrast, the public state that we use has a more moderate dimensionality, as it only captures public information known to all players. For example, in HUNL, a public state includes only essential information like previous actions of the players, the public cards, the chips in the pot, the remaining chips and the acting player. In contrast, a PBS in HUNL would need to encompass $1,326$ different private hands for both players, resulting in a significantly higher-dimensional representation.

2. *Stability during CFR iterations.* The public states of the non-root nodes remain fixed during CFR iterations. In the iterative process of CFR, PBS of the non-root nodes may change at each iteration, while the public state remains constant. Since the selection of the action abstraction is based on the state, maintaining a fixed state during CFR iterations is crucial. If the state changes at each iteration, the action abstraction will also change, potentially leading to poor convergence of CFR (Brown, 2020).

(**Action**) Action $\mathbf{a} = (x_1, y_1, \cdots, x_K, y_K)$ is a $2K$-dimensional vector, where K corresponds to the number of actions that can be selected in the chosen action abstraction. Each x_i, y_i have values between -1 and 1. In our MDP, this action $\mathbf{a}$ is utilized to select an action abstraction $\mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$ at PBS β. The subsequent game tree for CFR solving is constructed based on this action abstraction. The specifics are detailed below.

In an IIEFG, there are actions that are common and are added to the action abstraction regardless of the PBS β. We denote this set of actions as always-selected action set $\mathcal{AA}_{\text{always}}$, which may include some of the most common actions available. Additionally, we define a *default* fixed action abstraction $\mathcal{AA}_{\text{base}}(\beta)$ at PBS β. Here, $\mathcal{AA}_{\text{base}}(\beta) \subseteq \mathcal{A}(\beta)$ is a set of actions solely related to PBS β, and we have $\mathcal{AA}_{\text{always}} \subseteq \mathcal{AA}_{\text{base}}(\beta)$. Typically, $\mathcal{AA}_{\text{base}}(\beta)$ is pre-specified to a set of available actions related to crucial information of PBS β.

The choices for $\mathcal{AA}_{\text{always}}$ and $\mathcal{AA}_{\text{base}}(\beta)$ can be arbitrary in any IIEFG. However, different choices can impact the win-rate and running time, as demonstrated in Waugh et al. (2009) and Moravčík et al. (2017). For examples, in HUNL experiments, when $\mathcal{AA}_{\text{always}} = \{F, C, A\}$ (F, C, A refer to fold, check/call and all-in respectively), Moravčík et al. (2017) shows that the action abstraction $\mathcal{AA}_{\text{base}}(\beta) = \mathcal{AA}_{\text{always}} \cup \{0.5 \times \text{pot}, 1 \times \text{pot}, 2 \times \text{pot}\}$ ($\times$pot means the fraction of the size of the pot being bet) achieves a win-rate of 96 mbb/hand compared to action abstraction $\mathcal{AA}_{\text{base}}(\beta) = \mathcal{AA}_{\text{always}} \cup \{1 \times \text{pot}\}$.

Formally at a PBS β, the action abstraction chosen by $\mathbf{a} = (x_1, y_1, \cdots, x_K, y_K)$ is

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

Here, the optional action set $\mathcal{AA}_{\text{optional}}(\beta, \mathbf{a})$ is the set of actions generated from PBS β and the chosen action vector $\mathbf{a}$. Since the size of the game tree increases exponentially with the number of available actions, we limit the optional action set $\mathcal{AA}_{\text{optional}}(\beta, \mathbf{a})$ to have most K actions. Precisely, $\mathcal{AA}_{\text{optional}}(\beta, \mathbf{a}) = \bigcup_{i=1}^{K} f(x_i, y_i, \beta)$, where $f(x_i, y_i, \beta)$ is a function that generates an available action from all available actions except those in $\mathcal{AA}_{\text{always}}$. Notably, if $f(x_i, y_i, \beta) = \emptyset$, it means that there is no chosen action in this dimension of the action abstraction. Since the set of available actions of IIEFGs with numerous actions tends to be continuous, we define the function $f(x_i, y_i, \beta)$ by correspondingly mapping continuous parameters x_i and y_i to the set of available actions $\mathcal{A}(\beta)$ of PBS β.

Below, using HUNL as an example, we describe how to choose K and define $f(x_i, y_i, \beta)$. We set $K = 3$, which means we can select up to 3 raising scales other than all-in. We let $\mathcal{AA}_{\text{always}} = \{F, C, A\}$ and $\mathcal{AA}_{\text{base}}(\beta) = \{F, C, A, 0.5 \times \text{pot}, 1 \times \text{pot}, 2 \times \text{pot}\}$ (consistent with Moravčík et al. (2017)). Based on human experience and inspired by prior studies (Hawkin et al., 2011; 2012), a reason-

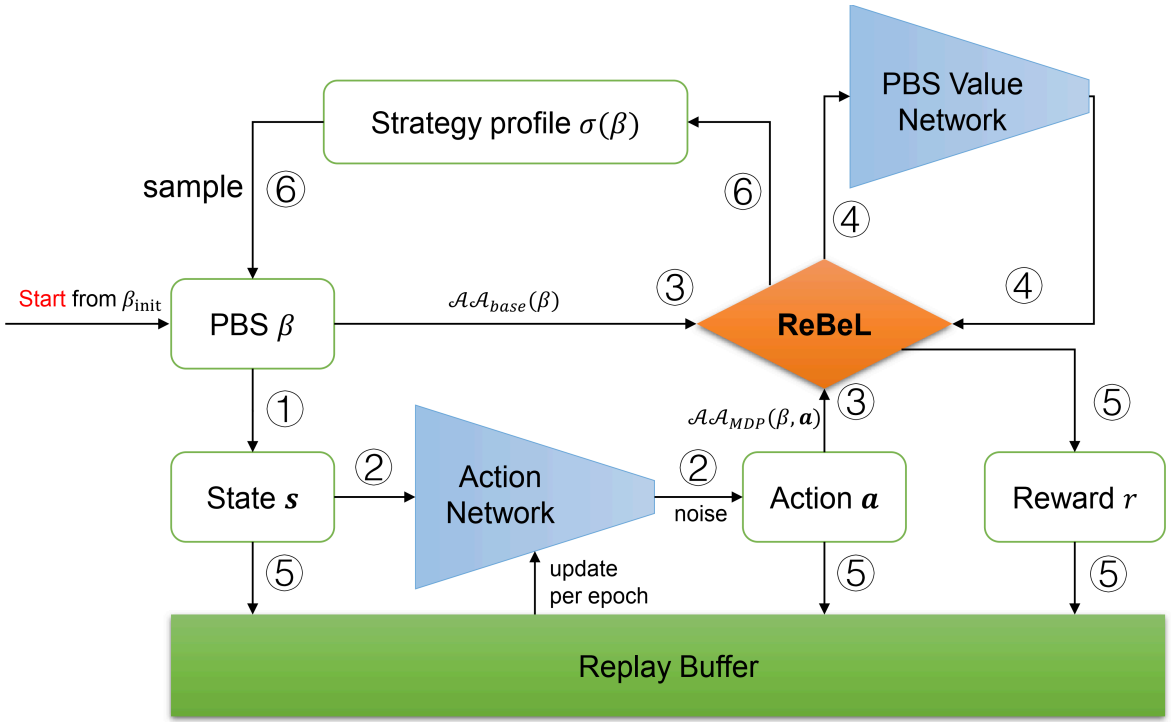

Figure 2. Training procedure for the RL-CFR framework. The labels in the figure correspond to the sampling steps for RL-CFR framework. A sampling epoch starts from the initial PBS β_{init}.

able 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

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

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

(**Reward**) The reward function $r(\mathbf{s}, \mathbf{a})$ in our MDP, is defined as the difference in PBS values between the chosen action abstraction $\mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$ and the default action abstraction $\mathcal{AA}_{\text{base}}(\beta)$.

For our abstract MDP, each action's reward needs to be obtained by solving two independently depth-limited subgames (Brown et al., 2018) using the CFR-based ReBeL algorithm (Brown et al., 2020), as described in Appendix D. Here is how we compute the reward r based on the PBS β, the state vector $\mathbf{s}$, and the action vector $\mathbf{a}$:

1. *Build Game Tree.* Construct a game tree rooted at PBS β with selected action abstraction $\mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$. Note that this selected action abstraction is used only for the root.

2. *Compute Strategy Profile σ_{MDP}.* Utilize depth-limited subgame solving method ReBeL to obtain a strategy profile σ_{MDP} for the game tree. This profile provides state transfers for all infostates corresponding to non-leaf nodes in the subgame.

3. *Calculate PBS Value for Chosen Action Abstraction.* With the calculated strategy profile σ_{MDP}, compute the PBS value $v^{\sigma_{\text{MDP}}}_{\mathcal{P}(\beta)}(\beta)$ for the acting player. This represents the expected payoff calculated for the acting player on PBS β (details of the PBS value calculation are in the last paragraph of Section 3).

4. *Build Another Game Tree.* Construct another game tree rooted at PBS β, but this time using the default fixed action abstraction $\mathcal{AA}_{\text{base}}(\beta)$ at the root. For the non-root nodes, both game trees use either the default action abstraction or the MDP-based action abstraction, depending on the training stage. Further details about the choice of action abstraction for non-root nodes can be found in Sections 5 and 6.

5. *Compute Strategy Profile σ_{base}.* Similarly, obtain the strategy profile σ_{base} for this game tree based on ReBeL.

6. *Calculate PBS Value for Default Action Abstraction.* Using the calculated strategy profile σ_{base}, compute the PBS value $v^{\sigma_{\text{base}}}_{\mathcal{P}(\beta)}(\beta)$ for the acting player.

7. *Compute Reward.* Finally, compute the reward $r(\mathbf{s}, \mathbf{a}) = v^{\sigma_{\text{MDP}}}_{\mathcal{P}(\beta)}(\beta) - v^{\sigma_{\text{base}}}_{\mathcal{P}(\beta)}(\beta)$.

The state transition of the MDP depends on the mixed strategy calculated by the CFR-based ReBeL algorithm, as detailed in Section 5. The discount factor γ is set to 1 during training.

5. RL-CFR Framework

In this section, we introduce the RL-CFR framework, an extension of ReBeL algorithm (Brown et al., 2020). The ReBeL algorithm is known for its efficiency in solving depth-limited subgames (Brown et al., 2018), allowing it to obtain effective fixed action abstraction strategies for large IIEFGs through a combination of self-play RL and CFR (see Appendix D). In contrast to ReBeL, which employs a fixed action abstraction, RL-CFR takes a novel approach by dynamically selecting its action abstraction through RL, based on our novel MDP.

As demonstrated in our experiments, this dynamic selection allows for the discovery of superior action abstractions, resulting in significant performance improvements. It is noteworthy that applying the DRL approach to IIEFGs is a highly nontrivial task. The main challenge lies in determining a mixed strategy for all information sets (Burch, 2017; Brown, 2020), a computation that is intricate when approached directly through RL methods. Despite this challenge, RL-CFR tackles the complexity by incorporating dynamic action abstraction, showcasing its capability to navigate the intricacies of IIEFGs and deliver enhanced performance outcomes.

The RL-CFR framework constitutes an end-to-end self-training reinforcement learning process, as illustrated in Figure 4. We now elucidate the sampling steps for RL-CFR framework:[2]

Step ① Compressing PBS. Starting from the initial PBS of the game, each handling of a PBS β involves several stages. If we encounter a chance PBS where the acting player is a chance player, we allow the chance player to act randomly and update the PBS. If we encounter a terminal PBS, the epoch of sampling ends. Moving on to a non-chance and non-terminal PBS β, we proceed to compress the high-dimensional PBS β into a low-dimensional public state $\mathbf{s}$ using the method described in Section 4.

Step ② Action Abstraction Selection. Passing through the action network and adding a Gaussian noise (for increased exploration) yields an action vector $\mathbf{a}$, which is then mapped to a specified action abstraction $\mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$.

Step ③ Building Depth-Limited Subgames. Constructing two depth-limited subgames rooted at β according to the default action abstraction $\mathcal{AA}_{\text{base}}(\beta)$ and selected action abstraction $\mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$, respectively.

Step ④ ReBeL Algorithm. Utilizing the ReBeL algorithm to solve the strategies and values of the two subgames.

[2]Here, "Sampling" refers to the process of selecting and generating instances or experiences that are used for training and updating the RL model.

Step ⑤ Calculating Reward and Updating RL Data. Calculating the PBS value difference as a reward r, and adding RL data $\{\mathbf{s}, \mathbf{a}, r\}$ to the training data (denoted as $Data^{\text{RL}}$) for the action and critic network.

Step ⑥ State Transition. Randomly choosing a subgame and following the corresponding strategy $\sigma(\beta)$ for state transition to a child PBS β' next. Setting $\beta = \beta'$, and repeating Step ①.

Algorithm 1 shows the formal procedure of the sampling process. These steps collectively form the foundation of the RL-CFR sampling process, driving the iterative learning and optimization within the framework.

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

Input: θ_α, noise, η, ϵ ▷ noise $= 0.15, \eta = 0.33, \varepsilon = 0.25$ during training
  $\beta \leftarrow \beta_{\text{init}}$ ▷ A sampling epoch starts from the initial PBS
  $Data^{\text{RL}} \leftarrow \{\}$
  while !IsTerminal(β) **do**
    while $P(\beta) = c$ **do**
      $\beta \leftarrow$ TakeChance(β) ▷ Random chance event
    end while
    $\sigma_{\text{base}}(\beta), v^{\sigma_{\text{base}}}_{\mathcal{P}(\beta)}(\beta) \leftarrow$ReBeL$(\beta, \mathcal{AA}_{\text{base}}(\beta))$ ▷ Compute the strategy and value for default action abstraction
    $s \leftarrow PS(\beta)$ ▷ Use public state as the state in MDP
    $\mathbf{a} \leftarrow$ActionNetwork$(s, \theta_\alpha) + \mathcal{N}(0, \text{noise})$ ▷ Sample an action abstracion
    $\sigma_{\text{MDP}}(\beta), v^{\sigma_{\text{MDP}}}_{\mathcal{P}(\beta)}(\beta) \leftarrow$ReBeL$(\beta, \mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a}))$ ▷ Compute the strategy and value for selected action abstraction
    $r \leftarrow v^{\sigma_{\text{MDP}}}_{\mathcal{P}(\beta)}(\beta) - v^{\sigma_{\text{base}}}_{\mathcal{P}(\beta)}(\beta)$ ▷ Calculating the reward
    Add $\{\mathbf{s}, \mathbf{a}, r\}$ to $Data^{\text{RL}}$
    $c \sim$ unif$[0, 1]$
    $d \sim$ unif$[0, 1]$
    if $c < \eta$ **then** ▷ State transition
      if $d < \epsilon$ **then**
        $a_{\text{next}} \sim \mathcal{AA}_{\text{base}}(\beta)$
      else
        $a_{\text{next}} \sim \sigma_{\text{base}}(\beta)$
      end if
      $\beta \leftarrow$ NextPBS$(\beta, \sigma_{\text{base}}(\beta), a_{\text{next}})$
    else
      if $d < \epsilon$ **then**
        $a_{\text{next}} \sim \mathcal{AA}_{\text{MDP}}(\beta, \mathbf{a})$
      else
        $a_{\text{next}} \sim \sigma_{\text{MDP}}(\beta)$
      end if
      $\beta \leftarrow$ NextPBS$(\beta, \sigma_{\text{MDP}}(\beta), a_{\text{next}})$
    end if
  end while
Output: $Data^{\text{RL}}$

After each epoch, a trajectory $(\mathbf{s}_1, \mathbf{a}_1, r_1, \cdots, \mathbf{s}_t, \mathbf{a}_t, r_t)$ is sampled based on the current action network, where t is the length of the game, contingent on the player actions. Upon collecting RL data $Data^{\text{RL}} = \{(\mathbf{s}, \mathbf{a}, r)\}$ over several epochs, the Actor-Critic algorithm (Konda & Tsitsiklis, 1999) is employed, along with Mean Squared Error (MSE) Loss to train both the action network and critic network (refer to Section 6 for network structures). The loss functions are defined as follows:

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

$$\mathcal{L}(\theta_a) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r)\sim Data^{\text{RL}}}[-r^{\theta_c}(\mathbf{s}, \mathbf{a}^{\theta_a}(s))], \tag{4}$$

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

During the initial epochs of training, the action network may tend to select sub-optimal action abstractions compared to the default fixed action abstraction $\mathcal{AA}_{\text{base}}(\beta)$. To address this, we initiate training by utilizing the default action abstraction for non-root nodes when constructing the depth-limited subgame. As training progresses, the action network becomes more adept at selecting superior action abstractions. Consequently, we transition to choosing the action abstraction for non-root nodes based on the output of the action network when building the depth-limited subgame. Simultaneously, to enhance the accuracy of PBS values, we can retrain the PBS value network according to the action abstraction selected by the action network. The iterative process of updating the PBS value network and the action network can theoretically be repeated for ongoing training, allowing the framework to adapt and improve its decision-making capabilities over time.

6. Experiment

To demonstrate the effectiveness of our RL-CFR framework in handling large IIEFGs with numerous actions, we conducted experiments on Heads-up No-limit Texas Hold'em (HUNL) and PREFLOP43 (see Appendix A for specific rules). Similar to prior studies on large IIEFGs (Brown et al., 2019; 2020; Zarick et al., 2020), we chose HUNL due to its representative nature and inherent complexity. During HUNL evaluation, players start with 200 big blinds, switching positions every two hands, akin to the annual computer poker competition (ACPC) (Bard et al., 2013). During PREFLOP43 evaluation, players start with 10 to 100 big blinds, switching positions every two hands.

Experiments were executed on a server with 8 NVIDIA PH402 SKU 200 GPUs and an 80-core Intel(R) Xeon(R) Gold 6145 2.00GHz CPU. Neural networks in our implementation, including the PBS value network, action network, and critic network, are Multi-Layer Perceptrons (MLPs) with ReLU activation functions. The networks are trained with the Adam optimizer (Kingma & Ba, 2015). In the CFR iteration to solve a PBS, we use the discounted CFR (DCFR) algorithm (Brown & Sandholm, 2019b), with the number of iterations $T = 250$ during training and evaluation.[3]

We initiated our experiments by training a PBS value network, comprising approximately 18 million parameters (as detailed in Appendix D). It is worth noting that all PBS value networks used in our experiments, including those employed in RL-CFR, were trained based on the default action abstraction. This deliberate setting aims to emphasize that any performance enhancements achieved by RL-CFR are solely attributed to the action abstraction chosen by the action network.

The action network and the critic network both have 3 layers and 2×10^4 parameters, with hidden layers of dimensions 128 and 96. The training process has 2×10^6 epochs, each sampling approximately 10 RL data.[4] Random sampling from the entire RL data set was conducted, utilizing a learning rate of 1×10^{-5} and a batch size of $1,024$ during training. After 5×10^5 epochs, we generated PBS data by constructing the game tree precisely according to the action abstraction provided by the action network. Notably, the training cost of action network and critic network is approximately 30% of the training cost of PBS value network.

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)	-	18 ± 16
ReBeL (Brown et al., 2020)	-	45 ± 5
RL-CFR (Ours)	64 ± 11	84 ± 17

We assessed the head-to-head performance of RL-CFR, ReBeL's replication (Brown et al., 2020) and the open source AI Slumbot (Jackson, 2013) (the winner of the 2018 ACPC).[5] We play RL-CFR for over $600,000$ hands against ReBeL's replication and RL-CFR achieves 64 mbb/hand win-rate against ReBeL's replication. We play RL-CFR and ReBeL's replication for over $250,000$ hands against

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

[4] An RL data instance $(\mathbf{s}, \mathbf{a}, r)$ comprises a 64-dimensional state $\mathbf{s}$ (recording public cards, chips, positions and previous actions in HUNL), a 6-dimensional action $\mathbf{a}$ and a scalar reward r. Since the number of rounds in a HUNL game is not deterministic, a single sample to the terminate state generally yields no more than 10 pieces of RL data.

[5] Since the opponent may select actions that deviate from the game tree, we perform nested subgame solving technique (Brown & Sandholm, 2017b) mentioned in Appendix C.

Slumbot, and the results illustrate that the replication of ReBeL beat Slumbot with a win-rate of 18 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 66 mbb/hand relative to ReBeL's replication. The performance of ReBeL's replication is worse than the original ReBeL version on the HUNL benchmark. This is because the number of training samples of ReBeL's replication being less than those of the original ReBeL implementation due to the limited computational resources. We emphasize that ReBeL is a building block of our novel RL-CFR scheme, and our results show that RL-CFR achieves significant win-rate against ReBeL's replication and Slumbot just by optimizing the action abstraction.[6]

We also performed an exploitability evaluation in over $10,000$ random river stage states. We simulate RL-CFR versus ReBeL's replication until reaching the river, meaning that the two players choose their respective action abstractions, and the performance of the previously chosen action abstraction has no effect on the test results. The exploitability of RL-CFR is 17 ± 0 mbb/hand, and the exploitability of ReBeL's replication is 20 ± 0 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.

Table 2. Competition results of fixed action abstraction methods against RL-CFR.

Method	Win-rate	Running time
ReBeL(Replication)	-64 ± 11	$1\times$
MUL-ACTION	-21 ± 26	$3\times$
FINE-GRAIN	-23 ± 28	$1.5\times$

We conduct additional experiments for RL-CFR, and the results are presented in Table 2. In these experiments, RL-CFR is compared against the method of choosing an optimal action abstraction among multiple fixed action abstractions (MUL-ACTION). MUL-ACTION operates by selecting an action abstraction with the highest value for the root PBS β_r among three action abstractions $\mathcal{AA}_1(\beta_r), \mathcal{AA}_2(\beta_r), \mathcal{AA}_3(\beta_r)$.[7] RL-CFR outperforms MUL-ACTION by 21 mbb/hand in win-rate after $100,000$ hands, requiring only 1/3 of the running time.

[6]A win-rate of over 50 mbb/hand in poker is a commonly cited benchmark for what a professional poker player seeks to win from a weaker opponent (Bowling et al., 2017).

[7]We set $\mathcal{AA}_1(\beta_r) = \{F, C, A, 0.5 \times \text{pot}, 1 \times \text{pot}, 2 \times \text{pot}\}, \mathcal{AA}_2(\beta_r) = \{F, C, A, 0.25 \times \text{pot}, 0.5 \times \text{pot}, 1 \times \text{pot}\}, \mathcal{AA}_3(\beta_r) = \{F, C, A, 0.33 \times \text{pot}, 0.7 \times \text{pot}, 1.5 \times \text{pot}\}$. Here the action abstractions other than the root are the same as those used in ReBeL.

Additionally, we compare RL-CFR against choosing a finer-grained action abstraction (FINE-GRAIN) $\mathcal{AA}(\beta_r) = \{F, C, A, 0.25 \times \text{pot}, 0.5 \times \text{pot}, 0.75 \times \text{pot}, 1.0 \times \text{pot}, 1.25 \times \text{pot}, 2.0 \times \text{pot}\}$ at the root PBS β_r.[8] RL-CFR surpasses FINE-GRAIN by 23 mbb/hand in win-rate after $100,000$ hands, requiring approximately $2/3$ of the running time.[9] The results of RL-CFR versus fixed action abstraction methods illustrate that the action abstraction chosen by RL-CFR can effectively increase the win-rate without increasing the running time of the CFR algorithm.

Beyond just HUNL, we conducted experiments on a poker variant of the game PREFLOP43. The game tree of PREFLOP43 is constructed directly to terminal nodes to ensure precise value estimation, obviating the necessity for the depth-limited subgame solving algorithm like ReBeL (Brown et al., 2020). RL-CFR undergoes training for $1,200,000$ epochs. Since the game tree can be built until the end of the game, we performed action abstraction evaluation. RL-CFR's action abstraction is anticipated to achieve an average increase of 4 mbb/action against fixed action abstraction at a random PBS. We also performed an exploitability test, and RL-CFR reduced exploitability from 23 mbb/hand to 21 mbb/hand after 250 CFR iterations. Finally, we assessed the head-to-head performance of RL-CFR and the fixed abstraction method, and RL-CFR surpassed the fixed abstraction method with a win-rate advantage of 5 ± 5 mbb/hand in a test of over $3,500,000$ hands. These results further demonstrate the effectiveness of RL-CFR in addressing the challenges posed by large IIEFGs beyond the HUNL poker game.

7. Conclusions

In this study, we introduce RL-CFR, a pioneering algorithmic solution designed for the resolution of large-scale IIEFGs. Anchored in a unique abstract MDP formulation, RL-CFR employs public information as states, leverages action abstraction features as actions, and adopts a meticulously crafted reward function. The ingenuity of RL-CFR lies in its fusion of reinforcement learning for action abstraction selection with CFR, facilitating dynamic action abstraction selection in IIEFGs. Our extensive experiments conducted in HUNL demonstrate that RL-CFR attains a substantial performance improvement when juxtaposed with fixed action abstraction HUNL methods. These findings underscore the efficacy and promise of RL-CFR in tackling the complexities inherent in large-scale IIEFGs.

[8]Following the same setting as in Zarick et al. (2020), and the action abstractions other than the root are the same as those used in ReBeL.

[9]Since the numbers of non-terminal nodes extended by the root node in the game tree built by FINE-GRAIN and RL-CFR are 9 and 6, respectively.

Impact Statement

This paper presents work whose goal is to advance the field of Imperfect Information Extensive-Form Games. There are many potential societal consequences of our work, and the most immediate risk posed by out work is its potential for cheating in poker games. RL-CFR can compute a strategy for arbitrary situations in seconds. For this reason, we have decided not to release the RL-CFR code for HUNL. We instead open source RL-CFR's implementation of PREFLOP43, a poker game with huge action space used to help researchers conduct IIEFG and RL-CFR studies.

Acknowledgements

The work of Boning Li and Longbo Huang was supported by the Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108400 and 2020AAA0108403. The work of Zhixuan Fang was supported by Tsinghua University Dushi Program.

References

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

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Bard, N., Hawkin, J. A., Rubin, J., and Zinkevich, M. The annual computer poker competition. *AI Mag.*, 34(2):112, 2013.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

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

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

Brown, N. and Sandholm, T. Regret transfer and parameter optimization. In *AAAI*, pp. 594–601. AAAI Press, 2014.

Brown, N. and Sandholm, T. Strategy-based warm starting for regret minimization in games. In *AAAI*, pp. 432–438. AAAI Press, 2016.

Brown, N. and Sandholm, T. Libratus: The superhuman AI for no-limit poker. In *IJCAI*, pp. 5226–5228. ijcai.org, 2017a.

Brown, N. and Sandholm, T. Safe and nested subgame solving for imperfect-information games. In *NIPS*, pp. 689–699, 2017b.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

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

Brown, N. and Sandholm, T. Solving imperfect-information games via discounted regret minimization. In *AAAI*, pp. 1829–1836. AAAI Press, 2019b.

Brown, N., Ganzfried, S., and Sandholm, T. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold'em agent. In *AAAI Workshop: Computer Poker and Imperfect Information*, volume WS-15-07 of *AAAI Technical Report*. AAAI Press, 2015.

Brown, N., Sandholm, T., and Amos, B. Depth-limited solving for imperfect-information games. In *NeurIPS*, pp. 7674–7685, 2018.

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep counterfactual regret minimization. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR, 2019.

Brown, N., Bakhtin, A., Lerer, A., and Gong, Q. Combining deep reinforcement learning and search for imperfect-information games. In *NeurIPS*, 2020.

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

Burch, N., Johanson, M., and Bowling, M. Solving imperfect information games using decomposition. In *AAAI*, pp. 602–608. AAAI Press, 2014.

Burch, N., Schmid, M., Moravcik, M., Morrill, D., and Bowling, M. AIVAT: A new variance reduction technique for agent evaluation in imperfect information games. In *AAAI*, pp. 949–956. AAAI Press, 2018.

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

Chaudhuri, K., Freund, Y., and Hsu, D. J. A parameter-free hedging algorithm. In *NIPS*, pp. 297–305. Curran Associates, Inc., 2009.

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

Dibangoye, J. S., Amato, C., Buffet, O., and Charpillet, F. Optimally solving dec-pomdps as continuous-state mdps. In *IJCAI*, pp. 90–96. IJCAI/AAAI, 2013.

D'Orazio, R., Morrill, D., Wright, J. R., and Bowling, M. Alternative function approximation parameterizations for solving games: An analysis of f-regression counterfactual regret minimization. In *AAMAS*, pp. 339–347. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

Ganzfried, S. and Sandholm, T. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In *IJCAI*, pp. 120–128. IJCAI/AAAI, 2013.

Ganzfried, S. and Sandholm, T. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI*, pp. 682–690. AAAI Press, 2014.

Ganzfried, S. and Sandholm, T. Endgame solving in large imperfect-information games. In *AAMAS*, pp. 37–45. ACM, 2015.

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

Hawkin, J. A., Holte, R., and Szafron, D. Automated action abstraction of imperfect information extensive-form games. In *AAAI*, pp. 681–687. AAAI Press, 2011.

Hawkin, J. A., Holte, R., and Szafron, D. Using sliding windows to generate action abstractions in extensive-form games. In *AAAI*, pp. 1924–1930. AAAI Press, 2012.

Heinrich, J. and Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016.

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

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

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

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

Johanson, M., Waugh, K., Bowling, M. H., and Zinkevich, M. Accelerating best response calculation in large extensive games. In *IJCAI*, pp. 258–265. IJCAI/AAAI, 2011.

Johanson, M., Bard, N., Burch, N., and Bowling, M. Finding optimal abstract strategies in extensive-form games. In *AAAI*, pp. 1371–1379. AAAI Press, 2012.

Johanson, M., Burch, N., Valenzano, R. A., and Bowling, M. Evaluating state-space abstractions in extensive-form games. In *AAMAS*, pp. 271–278. IFAAMAS, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In Solla, S. A., Leen, T. K., and Müller, K. (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.

Kovarík, V., Seitz, D., Lisý, V., Rudolf, J., Sun, S., and Ha, K. Value functions for depth-limited solving in zero-sum imperfect-information games. *Artif. Intell.*, 314:103805, 2023.

Kroer, C. and Sandholm, T. Extensive-form game abstraction with bounds. In *EC*, pp. 621–638. ACM, 2014.

Kroer, C. and Sandholm, T. Discretization of continuous action spaces in extensive-form games. In *AAMAS*, pp. 47–56. ACM, 2015.

Kroer, C. and Sandholm, T. A unified framework for extensive-form game abstraction with bounds. In *NeurIPS*, pp. 613–624, 2018.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. H. Monte carlo sampling for regret minimization in extensive games. In *NIPS*, pp. 1078–1086. Curran Associates, Inc., 2009.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lee, D., Tang, H., Zhang, J. O., Xu, H., Darrell, T., and Abbeel, P. Modular architecture for starcraft II with deep reinforcement learning. In Rowe, J. P. and Smith, G. (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.

Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Zhao, L., Qin, T., Liu, T., and Hon, H. Suphx: Mastering mahjong with deep reinforcement learning. *CoRR*, abs/2003.13590, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Liu, M., Ozdaglar, A. E., Yu, T., and Zhang, K. The power of regularization in solving extensive-form games. In *ICLR*. OpenReview.net, 2023.

Meng, L., Ge, Z., Tian, P., An, B., and Gao, Y. An efficient deep reinforcement learning algorithm for solving imperfect information extensive-form games. In *AAAI*, pp. 5823–5831. AAAI Press, 2023.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. doi: 10.1126/science.aam6960.

Nash Jr, J. F. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

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

Nesterov, Y. E. Excessive gap technique in nonsmooth convex minimization. *SIAM J. Optim.*, 16(1):235–249, 2005.

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

Oliehoek, F. A. Sufficient plan-time statistics for decentralized pomdps. In *IJCAI*, pp. 302–308. IJCAI/AAAI, 2013.

Pérolat, J., Munos, R., Lespiau, J., Omidshafiei, S., Rowland, M., Ortega, P. A., Burch, N., Anthony, T. W., Balduzzi, D., Vylder, B. D., Piliouras, G., Lanctot, M., and Tuyls, K. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8525–8535. PMLR, 2021.

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

Sandholm, T. Abstraction for solving large incomplete-information games. In *AAAI*, pp. 4127–4131. AAAI Press, 2015.

Schmid, M., Moravčík, M., Burch, N., Kadlec, R., Davidson, J., Waugh, K., Bard, N., Timbers, F., Lanctot, M., Holland, G. Z., Davoodi, E., Christianson, A., and Bowling, M. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023. doi: 10.1126/sciadv.adg3256.

Schnizlein, D., Bowling, M. H., and Szafron, D. Probabilistic state translation in extensive games with large action sets. In *IJCAI*, pp. 278–284, 2009.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017.

Sustr, M., Kovarík, V., and Lisý, V. Monte carlo continual resolving for online strategy computation in imperfect information games. In *AAMAS*, pp. 224–232. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

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

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

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

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

Waugh, K., Schnizlein, D., Bowling, M. H., and Szafron, D. Abstraction pathologies in extensive games. In *AAMAS (2)*, pp. 781–788. IFAAMAS, 2009.

Waugh, K., Morrill, D., Bagnell, J. A., and Bowling, M. H. Solving games with functional regret estimation. In *AAAI*, pp. 2138–2145. AAAI Press, 2015.

Xu, H., Li, K., Fu, H., FU, Q., Xing, J., and Cheng, J. Dynamic discounted counterfactual regret minimization. In *The Twelfth International Conference on Learning Representations*, 2024.

Zarick, R., Pellegrino, B., Brown, N., and Banister, C. Unlocking the potential of deep counterfactual value networks. *CoRR*, abs/2007.10442, 2020.

Zhao, E., Yan, R., Li, J., Li, K., and Xing, J. Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In *AAAI*, pp. 4689–4697. AAAI Press, 2022.

Zinkevich, M., Johanson, M., Bowling, M. H., and Piccione, C. Regret minimization in games with incomplete information. In *NIPS*, pp. 1729–1736. Curran Associates, Inc., 2007.

A. Description of Games used for Evaluation

A.1. Heads-up No-limit Texas Hold'em Poker (HUNL)

The game begins with each player receiving two cards, referred to as their "private hand". The gameplay unfolds through four stages: pre-flop, flop, turn, and river. During these stages, a total of five public cards are revealed – three at the start of the flop, one at the start of the turn, and one at the start of the river.

Before the game commences, several players are required to contribute a pre-specified number of chips, known as the "small blind" and "big blind". Typically, the small blind is set at half the value of the big blind (the small blind is 50 chips and the big blind is 100 chips in our experiments). In the pre-flop stage of HUNL, the small blind player takes the first action, while in other stages, the big blind player initiates the action. The permissible actions include fold, check/call, and bet/raise. Players can bet or raise any number of chips between the last bet/raise in the stage (at least 1 big blind) and their remaining chips, even opting for an "all-in".

At the conclusion of a game, players who did not fold by the end of all stages select the best five cards from their private hand and the five public cards for comparison. The player, or players, with the best hand win the pot. The win-rate in poker is often expressed as the average number of big blinds won per game. Alternatively, it can be measured in more granular units such as mbb (Bowling et al., 2017), equivalent to one thousandth of a big blind. For instance, a win-rate of 0.01 big blind per hand can also be stated as 10 mbb/hand (10 mbb per hand).

A.2. PREFLOP43

PREFLOP43 a poker game that we designed manually. The relatively small size of the PREFLOP43 game allows the game tree to be built directly to the terminal nodes, thus allowing for better evaluation of the performance of the RL-CFR without the using of PBS value networks.

At the start of the game, each of the two players receives two private cards, and then 43 public cards are revealed to both players. The two players then move as in the pre-flop stage of HUNL. In PREFLOP43, the size of the raise may not be a whole number, but at least 0.5 times the size of the pot. Once the pre-flop stage is over, if neither player folds, the remaining five cards are dealt together and the player, or players, with the best hand win the pot.

During training and evaluation, we let the fixed default action abstraction $\mathcal{AA}_{\text{base}}(\beta) = \{F, C, A, 0.5 \times \text{pot}, 1 \times \text{pot}, 2 \times \text{pot}, 4 \times \text{pot}, 8 \times \text{pot}\}$. For the chosen dynamic action abstraction, we let $K = 5$ and

$$f(x_i, y_i, \beta) = \begin{cases} (\text{pot}_A - 0.5 \times \text{pot}) * (x_i + 1) * 0.5 + 0.5 \times \text{pot}, & y_i \geq 0; \\ \emptyset, & y_i < 0. \end{cases} \tag{5}$$

where pot_A is the amount of chips needed for an all-in.

B. Related Work of Texas Hold'em AIs

In the realm of Texas Hold'em AI, substantial progress has been made, with notable achievements by powerful agents like Libratus (Brown & Sandholm, 2018) and Pluribus (Brown & Sandholm, 2019a), demonstrating supremacy over top human players in both two-player and multi-player poker. These AI models employed intricate abstractions (Johanson et al., 2012; Ganzfried & Sandholm, 2014; Brown et al., 2015) to handle the vast decision space of Texas Hold'em, requiring extensive computational resources for pre-calculating a blueprint strategy (Brown & Sandholm, 2016; 2017a) through CFR under an extensive game tree.

DeepStack (Moravčík et al., 2017) introduced deep learning to estimate the values of private hands within the game tree, effectively reducing its size (Johanson, 2013). ReBeL (Brown et al., 2020) leveraged self-play reinforcement learning to generate realistic training data, providing an alternative approach to training. Notably, both Libratus and Pluribus did not incorporate reinforcement learning, while DeepStack and ReBeL did not employ reinforcement learning in action abstraction selection, specifically raising scales in HUNL.

A recent contribution by Zhao et al. (2022) presented a HUNL AI based on reinforcement learning, showcasing the potential for creating excellent AIs with minimal computational resources. However, the absence of the widely used CFR algorithm in HUNL limits its theoretical guarantees and raises concerns about exploitability.

In summary, the landscape of Texas Hold'em AI is marked by diverse approaches, encompassing complex abstractions, deep learning, CFR, and reinforcement learning. Ongoing research aims to strike a balance between computational efficiency and strategic sophistication in order to further advance the capabilities of AI agents in this challenging poker variant.

C. Abstraction

The huge solution complexity of IIEFGs is characterized by three dimensions: the depth of the game D, the size of the information set $|I|$ and the number of available actions $|\mathcal{A}(I)|$.The original space complexity is $O(|\mathcal{A}(I)|^D \cdot |I|)$, reaching 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, it is common practice not to compute the strategy until the end. Instead, a depth-limited subgame (Brown et al., 2018) is generated, extending only a limited number of states into the future. Strategies or expected values are estimated for leaf states, which are non-terminal states in the full game but terminal states in the depth-limited subgame. DeepStack (Moravčík et al., 2017) and ReBeL (Brown et al., 2020) employ deep learning to estimate counterfactual values of leaf states, thereby avoiding solving until the end of the game. Another approach to limit depth involves consuming substantial computing resources to pre-calculate a blueprint strategy (Brown & Sandholm, 2016; 2017a), avoiding extensive solving for deep game instances.

To limit the size of the information set, similar states can be grouped into the same bucket (state-space abstraction) (Johanson et al., 2012; 2013; Brown et al., 2015) or represented in a high-dimensional feature abstraction (Brown et al., 2019). State-space abstractions require careful design tailored to the specific game. To demonstrate the generality of our method for general IIEFGs, our experiments refrain from using any state-space abstractions.

To restrict the number of available actions, it is common to utilize action abstraction in IIEFGs. Formally, $\mathcal{AA}(I)$ represents 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 not in the action abstraction $\mathcal{AA}(I)$, rounding off-tree action to a nearby in-abstraction action (Schnizlein et al., 2009; Ganzfried & Sandholm, 2013) or resolving the strategy based on the new action abstraction $\mathcal{AA}(I) \cup a$ (nested subgame solving (Ganzfried & Sandholm, 2015; Brown & Sandholm, 2017b)) is commonly employed.

D. Solving the strategy and PBS value for large IIEFGs

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

In each epoch, training commences from the initial state of the game, and the PBS corresponding to the initial state is denoted as β_{init}. During training, we handle a PBS β_r and its corresponding action abstraction $\mathcal{AA}(\beta_r)$. The task involves computing the PBS value and sampling to a leaf PBS. Algorithm 2 details these steps (we restate the algorithms with our notations to facilitate understanding), and we provide a description of the training process below.

Algorithm 2 ReBeL algorithm: 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$(\beta_r, \mathcal{AA}(\beta_r))$ ▷ Construct a subgame rooted with β_r
  $\overline{\sigma}, \sigma^0 \leftarrow$UniformPolicy$(\beta_r, \mathcal{AA}(\beta_r))$
  $\mathbf{v}(\beta_r) \leftarrow \mathbf{0}$
  $t_{\text{sample}} \sim$ 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 $DCFR_{\frac{3}{2},0,2}$
    $\mathbf{v}(\beta_r) \leftarrow \frac{t-1}{t+1}\mathbf{v}(\beta_r) + \frac{2}{t+1}\mathbf{v}^{\sigma^t}(\beta_r)$ ▷ Update PBS value for all infostates at β_r
    if $t = t_{\text{sample}}$ **then**
      $\beta_{\text{next}} \leftarrow$SampleLeaf$(G, \sigma^t)$ ▷ Sample a leaf PBS
    end if
  end for
  Add $\{\beta_r, \mathbf{v}(\beta_r)\}$ to $Data^{\text{PBS}}$ ▷ 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 $\overline{\sigma}, v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}(\beta_r), \beta_{\text{next}}$
end function

At the onset of training, we construct a depth-limited subgame rooted with β_r.[10] During the construction of the game tree, for non-terminal and non-leaf node β', we expand the child nodes downwards according to the action abstraction $\mathcal{AA}(\beta')$.[11]

Once the game tree is built, the subgame is solved by running T iterations of the CFR algorithm. The value of leaf nodes is estimated using a learned value network $\hat{v}$ at each iteration based on their PBS. On each iteration t, CFR is employed to determine a strategy profile σ^t in the subgame. Subsequently, 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.

Due to the non-zero-sum nature of estimates from the neural network, adjustments are made to the infostate values at each PBS to ensure the game satisfies the zero-sum property. Additionally, for infostates that should have the same value, their value estimates are averaged. Since PBSs may change every iteration, the leaf node values may also change. Given σ^t and leaf node values, each infostate in each node has a calculated PBS value, as explained in Section 3. This information is then used to update the regret and average strategy $\overline{\sigma}$ for the CFR algorithm.

After completing T iterations, we obtain the solved average strategy $\overline{\sigma}$. Using this strategy, we calculate the PBS values for all infostates $v_p^{\overline{\sigma}}(O_p|\beta_r)$ for the root PBS β_r. We denote this vector of PBS values as $\mathbf{v}(\beta_r)$. Subsequently, we add the PBS data $\beta_r, \mathbf{v}(\beta_r)$ to the training data (denoted $Data^{\text{PBS}}$) for $\hat{v}(\beta_r)$. Meanwhile, we calculate the PBS value $v^{\overline{\sigma}}_{\mathcal{P}(\beta_r)}$ of β_r according to the calculated value vector $\mathbf{v}(\beta_r)$. This PBS value is required as part of the reward function for our RL-CFR framework, although it is not needed in the ReBeL algorithm itself.

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. To ensure more exploration, we can sample a random leaf PBS with probability ε. Additionally, we can modify some public information in the sampled PBS for more exploration.[12] We repeat the above processes until the game ends.

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

$$\begin{aligned}\mathcal{L}(\theta, \delta) =& \mathbb{E}_{\{O_p, v_p(O_p)\} \sim \{\beta_r, \mathbf{v}(\beta_r)\}, \{\beta_r, \mathbf{v}(\beta_r)\} \sim Data^{\text{PBS}}} \\ & [\min\{\frac{1}{2}(v_p(O_p) - \hat{v}^\theta(O_p|\beta_r))^2, \delta|v_p(O_p) - \hat{v}^\theta(O_p|\beta_r)| - \frac{1}{2}\delta^2\}]\end{aligned} \tag{6}$$

[10] In the HUNL experiments, the subgame is built up to the end of a stage or the start of the chance player's action. This implies that an epoch has up to 7 phases: start of pre-flop, end of pre-flop, start of flop, end of flop, start of turn, end of turn, and start of river.

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

[12] For HUNL agent training, we set $\varepsilon = 25\%$. 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 to a random number between 50 and 250 big blinds.

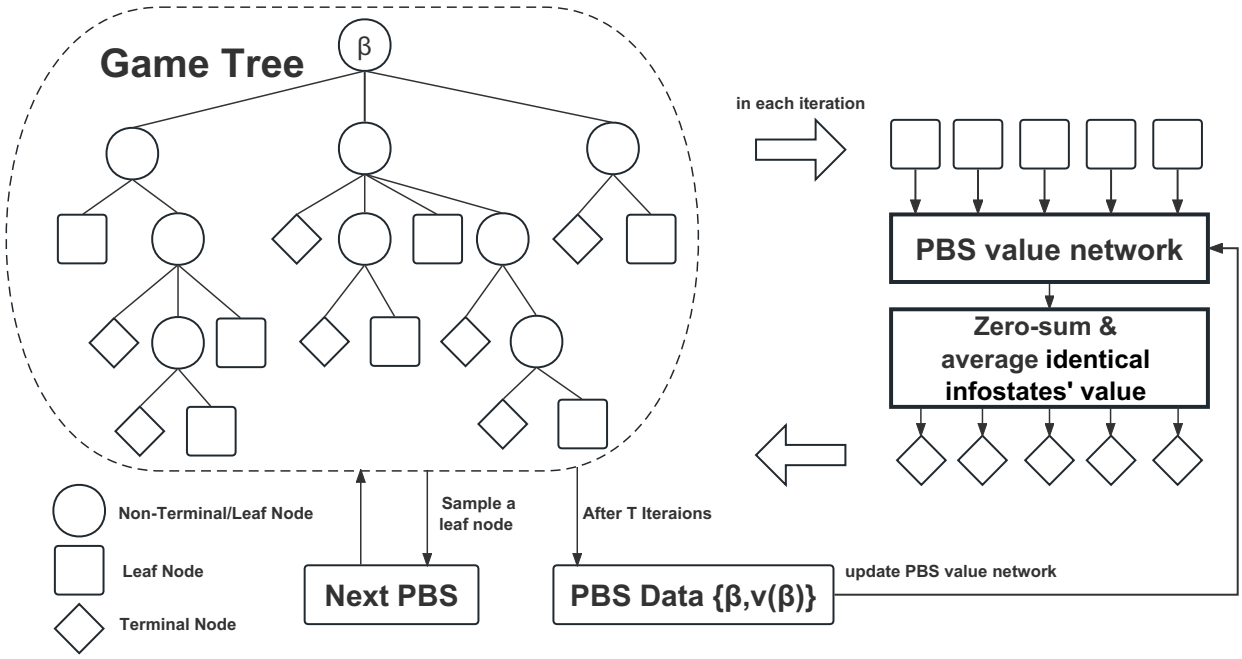

Figure 3. This figure illustrates the process of generating PBS data and training the PBS value network. For a given PBS β, we construct a depth-limited subgame rooted with β. Non-terminal and non-leaf nodes are depicted as circles, and during the construction of the game tree, we expand child nodes based on the action abstraction of the PBS associated with the node. Terminal nodes, denoted by diamonds, allow for direct calculation of the PBS value. Leaf nodes, represented by rectangles, require the estimation of PBS values in every iteration of CFR, where the PBS value network is employed to estimate the PBS values for these leaf nodes.

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

In our HUNL experiments, the PBS value network consists of 6 layers and 18 million parameters. 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 comprises $1,536$ dimensions, and the output layer has $2,652$ dimensions, representing all possible private hands of the two players. We sampled 6×10^7 epochs during training. Random sampling was performed from the last 1×10^7 epochs, and the training employed a learning rate of 1×10^{-5} with a batch size of 512. The training process and data sampling occurred simultaneously.

In summary, ReBeL is a self-play RL framework capable of continuously generating data from scratch for training. Figure D illustrates the training process of the ReBeL algorithm. Following the training process, we obtained a replication version of ReBeL as a baseline.

E. Examples of RL-CFR strategies

We use several examples from HUNL to illustrate how RL-CFR selects action abstractions. We show examples of heads-up evaluation between ReBeL's replication (Brown et al., 2020) and RL-CFR. Both players start with 200 big blinds (BB) and $20,000$ chips (100 chips for 1 BB) in all examples.

Example 1.

Pre-flop stage. ReBeL sits in small blind position with hand 4♠3♠ and RL-CFR sits in big blind position with hand $J\heartsuit 8\diamondsuit$. ReBeL acts first with action abstraction $\{F, C, 2, 3, 5, A\}$ (F, C, A refer to fold, check/call and all-in respectively and the numbers represent raising scales in BB). The strategy calculated by ReBeL is: call with 3.21%, raise to 2 BB ($0.5\times$pot) with 52.10%, raise to 3 BB ($1\times$pot) with 44.11% and raise to 4 BB ($2\times$pot) with 0.58%. ReBeL raises to 2 BB in this example. In this situation, RL-CFR selects an action abstraction $\{F, C, 3, 8.8, 16.21, A\}$. The strategy is: call with 76.08%, raise to 8.8 BB ($1.7\times$pot) with 23.67% and raise to 16.21 BB ($3.5525\times$pot) with 0.25%. ReBeL calls in this example. When

RL-CFR is faced with a 2 BB raise in the big blind position, RL-CFR will use the three raising scales of 3 BB, 8.8 BB, 16.21 BB and expect to win 10 mbb/hand compared to the default raising scales (4 BB, 6 BB, 10 BB).

Flop stage. Flop is $J\diamondsuit 6\heartsuit 3\diamondsuit$. There are 4 BB in the pot and RL-CFR acts first. RL-CFR selects an action abstraction $\{F, C, A\}$. In this case, RL-CFR will check all hands, which is a common strategy that human professional players will employ. Now turn to ReBeL and the strategy is: check with 47.94%, bet 2 BB ($0.5\times$pot) with 51.63% and bet 4 BB ($1\times$pot) with 0.42%. ReBeL bets 2 BB in this example. In this situation, RL-CFR selects an action abstraction $\{F, C, 4, 25.36, A\}$. The strategy is: call with 72.09% and raise to 4 BB ($0.25\times$pot) with 27.91%. RL-CFR calls in this example. It's an interesting strategy, with RL-CFR opting for a minimum raising scale (mini-raise) and a very large raising scale, gaining an additional 6 mbb/hand win-rate compared to the default action abstraction.

Turn stage. Turn is $4\clubsuit$. There are 8 BB in the pot and RL-CFR acts first. RL-CFR selects an action abstraction $\{F, C, 1, A\}$. The strategy is: check with 20.64% and bet 1 BB with 79.36%. The turn card is very favourable to RL-CFR's calling range, so RL-CFR had a high frequency of betting (donk). Choosing a 1 BB raising scale (minimum betting) gives RL-CFR an additional win-rate of 42 mbb/hand compared to the default action abstraction, which is very impressive. RL-CFR bets 1 BB in this example. Now turn to ReBeL and the strategy is: call with 99.63% and raise to 6 BB ($0.5\times$pot) with 0.37%. ReBeL has two-pairs now, however the strategy calculated by CFR is calling most hands since the turn card is unfavorable to small blind player's hand range.

River stage. River is $2\heartsuit$. There are 10 BB in the pot and RL-CFR acts first. RL-CFR selects an action abstraction $\{F, C, 8.42, 14.88, 46.22, A\}$ with 4 mbb/hand extra win-rate. The strategy is: check with 99.93% and bet 8.42 BB ($0.842\times$pot) with 0.07%. Now turn to ReBeL and the strategy is: check with 0.37%, bet 5 BB ($0.5\times$pot) with 43.76%, bet 10 BB ($1\times$pot) with 55.66% and bet 20 BB ($2\times$pot) with 0.19%. ReBeL bets 5 BB in the example. Now turn to RL-CFR and the selected action abstraction is $\{F, C, 17.81, 55.31, 98.77, A\}$ with 6 mbb/hand extra win-rate. The strategy of RL-CFR is: fold with 42.04%, call with 56.52%, raise to 17.81 BB ($0.6405\times$pot) with 0.55% and raise to 93.77 BB ($4.4385\times$pot) with 0.87%. RL-CFR folds and loses the 10 BB pot in the example.

Example 2. It is a symmetrical example of Example 1.

Pre-flop stage. RL-CFR sits in small blind position with hand $4\spadesuit 3\spadesuit$ and ReBeL sits in big blind position with hand $J\heartsuit 8\diamondsuit$. RL-CFR acts first and selects an action abstraction $\{F, C, 2.9, 4.56, 7.64, A\}$ with 24 mbb/hand extra win-rate. The strategy of RL-CFR is: call with 18.16%, raise to 2.9 BB ($0.95\times$pot) with 78.54% and raise to 4.56 BB ($1.78\times$pot) with 3.29%. RL-CFR raises to 2.9 BB and ReBeL calls in this example.

Flop stage. Flop is $J\diamondsuit 6\heartsuit 3\diamondsuit$. There are 5.8 BB in the pot and ReBeL checks first. The action abstraction selected by RL-CFR is $\{F, C, 8.6, 28.1, A\}$. It is an interesting strategy generated by RL-CFR with overbet (bet more than a pot) only and gain an additional 46 mbb/hand win-rate compared to the default action abstraction. The strategy is: check with 99.65%, bet 8.6 BB with 0.22% and bet 28.1 BB with 0.13%. RL-CFR checks in the example.

Turn stage. Turn is $4\clubsuit$. There are 5.8 BB in the pot and ReBeL acts first. The strategy of ReBeL is: check with 60.17%, bet 2.9 BB with 26.06%, bet 5.8 BB with 13.66% and bet 11.6 BB with 0.12%. ReBeL checks in the example and turn to RL-CFR. The action abstraction calculated by RL-CFR is $\{F, C, 1, 1.85, A\}$. However, after evaluation by the policy network, RL-CFR considers this action abstraction to be inferior to the default action abstraction, so the default action abstraction will be chosen this time.[13] The strategy of RL-CFR is: check with 0.03%, bet 2.9 BB with 98.88%, bet 5.8 BB with 0.14% and bet 11.6 BB with 0.94%. In this example, RL-CFR bets 2.9 BB and ReBeL calls.

River stage. River is $2\heartsuit$. There are 11.6 BB in the pot and ReBeL checks first. The action abstraction selected by RL-CFR is $\{F, C, 2.3, 44.41, 46.43\}$ and the strategy of RL-CFR is: check with 0.30%, bet 2.3 BB with 99.64% and bet $44.41, 44.63$ BB with 0.06%. In this example, RL-CFR bets 2.3 BB and ReBeL calls. RL-CFR wins the 16.2 BB pot with two pairs at showdown.

Example 3. In this example RL-CFR performed a bluff (betting with a weaker hand) and successfully bluffing with a suitable action abstraction.

Pre-flop stage. ReBeL sits in small blind position with hand $Q\heartsuit 9\heartsuit$ and RL-CFR sits in big blind position with hand $9\clubsuit 8\heartsuit$. ReBeL raises 2 BB first and RL-CFR calls in this example.

[13] After selecting an action abstraction through the action network, we evaluate it using the policy network and if the evaluation value is negative, we use the default action abstraction.

Flop stage. Flop is $K\clubsuit 6\spadesuit 2\diamondsuit$. There are 4 BB in the pot. RL-CFR and ReBeL check in this example.

Turn stage. Turn is $7\clubsuit$. There are 4 BB in the pot and RL-CFR acts first. The action abstraction selected by RL-CFR is $\{F, C, 1, 2.25, A\}$ with 10 mbb/hand extra win-rate. The strategy of RL-CFR is: check with 34.31%, bet 1 BB with 11.06% and bet 2.25 BB with 54.63%. RL-CFR bets 1 BB in this example. The stragety of ReBeL is: fold with 48.32%, call with 51.04%, raises to 4 BB with 0.61% and raises to 7 BB with 0.03%. ReBeL calls in this example.

River stage. River is $4\spadesuit$. There are 6 BB in the pot and RL-CFR acts first. The action abstraction selected by RL-CFR is $\{F, C, 4.56, 5.71, 29.23, A\}$ with 3 mbb/hand extra win-rate. The strategy of RL-CFR is: check with 12.48%, bet 4.56 BB with 41.63%, bet 5.71 BB with 33.11% and bet 29.23 BB with 12.78%. RL-CFR bets 5.71 BB in this example and the strategy of ReBeL is: fold with 99.14% and call with 0.85%. ReBeL folds and RL-CFR wins the 6 BB pot.

Example 4. In this example RL-CFR calls the 3-bet (re-raise at pre-flop stage) from ReBeL.

Pre-flop stage. RL-CFR sits in small blind position with hand $A\diamondsuit 4\diamondsuit$ and ReBeL sits in big blind position with hand $K\spadesuit J\spadesuit$. RL-CFR raises to 2.9 BB. The strategy of ReBeL is: call with 2.37%, raises to 5.8 BB with 36.31%, raises to 8.7 BB with 42.01% and raises to 14.5 BB with 19.31%. ReBeL raises to 8.7 BB in this example. The action abstraction selected by RL-CFR is $\{F, C, 14.5, 16.22, 27.96, A\}$ with 13 mbb/hand extra win-rate. The strategy of RL-CFR is: call with 70.90%, raise to 14.5 BB with 0.03%, raise to 16.22 BB with 0.04%, and raise to 27.96 BB with 29.03%. RL-CFR calls in the example.

Flop stage. Flop is $Q\clubsuit 5\diamondsuit 3\diamondsuit$. There are 17.4 BB in the pot and ReBeL acts first. The strategy of ReBeL is: check with 80.33%, bet 8.7 BB with 14.94%, bet 17.4 BB with 4.46% and bet 34.8 BB with 0.26%. ReBeL bets 17.4 BB in the example. The action abstraction selected by RL-CFR is $\{F, C, 34.8, A\}$ with 142 mbb/hand extra win-rate, which means that if there is no mini-raise in the action abstraction there will be a huge loss in this situation. The strategy of RL-CFR is: call with 97.69%, raise to 34.8 BB with 1.73% and all-in with 0.58%. RL-CFR calls in the example.

Turn stage. Turn is $6\diamondsuit$ and RL-CFR has the nuts (strongest hand). There are 52.2 BB in the pot and ReBeL checks first. The action abstraction selected by RL-CFR is $\{F, C, 15.11, 64.73, 97.22, A\}$ with 46 mbb/hand extra win-rate. The strategy of RL-CFR is: check with 0.43%, bet 15.11 BB with 88.13% and bet 64.73 BB with 11.43%. RL-CFR bets 15.11 BB in the example and ReBeL folds. RL-CFR wins the 52.2 BB pot. The action abstraction of RL-CFR in this situation is very reasonable, and a 15.11 BB bet can put many of opponent's hands in an embarrassing situation.