Continual Depth-limited Responses for Computing Counter-strategies in Sequential Games

David Milec
AI Center, FEE, CTU in Prague
Czech Republic
milecdav@fel.cvut.cz

Ondřej Kubíček
AI Center, FEE, CTU in Prague
Czech Republic
kubicon3@fel.cvut.cz

Viliam Lisý
AI Center, FEE, CTU in Prague
Czech Republic
viliam.lisy@agents.fel.cvut.cz

ABSTRACT

In zero-sum games, the optimal strategy is well-defined by the Nash equilibrium. However, it is overly conservative when playing against suboptimal opponents and it can not exploit their weaknesses. Limited look-ahead game solving in imperfect-information games allows superhuman play in massive real-world games such as Poker, Liar's Dice, and Scotland Yard. However, since they approximate Nash equilibrium, they tend to only win slightly against weak opponents. We propose theoretically sound methods combining limited look-ahead solving with an opponent model, in order to 1) approximate a best response in large games or 2) compute a robust response with control over the robustness of the response. Both methods can compute the response in real time to previously unseen strategies. We present theoretical guarantees of our methods. We show that existing robust response methods do not work combined with limited look-ahead solving of the shelf, and we propose a novel solution for the issue. Our algorithm performs significantly better than multiple baselines in smaller games and outperforms state-of-the-art methods against SlumBot.

KEYWORDS

large games, approximating best response, robust response, opponent exploitation, imperfect information, depth limited solving, gadgets

1 INTRODUCTION

We can not enumerate all the decision points in large games, which makes computing optimal strategy, a Nash equilibrium (NE) in two-player zero-sum games, infeasible. A breakthrough that allowed approximating the NE and defeating human experts in several large imperfect-information games is limited look-ahead solving or search, which adapts the well-known approach from perfect-information games to games with imperfect-information [3, 18, 19]. Limited look-ahead solving takes advantage of decomposition. It iteratively builds the game to some depth and solves a small part of the game while summarising the required values from the rest of the game by a value function. The value function is commonly learned using neural networks. When the algorithms solve the game step by step, it is called continual depth-limited solving or continual resolving.

The vast majority of theoretically sound, continual depth-limited solving algorithms assume perfect rationality of the opponent and do not allow explicit modeling of an opponent and exploitation of the opponent's mistakes. As a result, even very weak opponents exploitable by the heuristic local best response (LBR) [11] can tie or lose very slowly against these methods [24]. Therefore, there has been a significant amount of work towards computing strategies to use against imperfect opponents to create AI systems that would perform well in the real world, for example, against humans [1, 8, 10, 15, 16, 20, 22].

The opponent modeling and exploitation process consists of two steps: opponent modeling and model exploitation. Opponent modeling requires building a model from previous data or actions observed during an online play. Model exploitation is finding a good strategy against the given model and is the main focus of this paper. In smaller games, we can trivially compute a best response to exploit the opponent maximally, or we can use methods to compute robust responses [8, 9] if there is uncertainty in the model and we want to be safer, meaning we want to limit the possible loss when facing the worst-case adversary. However, even the best response (BR) computation in large games is non-trivial, and currently, no approach can compute it while interacting in real-time.

This work explores the full model exploitation and proposes continual depth-limited best response (CDBR). CDBR relies on the value function used in the standard limited look-ahead solving, and we prove theoretical guarantees on the performance. A drawback of using the same value function is decreased performance, and we could improve CDBR by training a specific value function for a particular opponent model. However, it would be impractical since the training is expensive. Furthermore, in cases where we learn the opponent model in real-time interaction and update it after each step, it would be impossible.

The best response and CDBR are useful, e.g., for evaluating the quality of strategies, but they are brittle in game play. We can lose significantly when facing an opponent different from the expected model. In the real world, we will never have exact models, which makes BR and CDBR impractical for game play. To address the issue, robust responses are used [6, 8, 9]. They introduce a notion of safety, and the safety criterion requires the response to stay close to the NE. In other words, only to lose a limited amount to the worst-case adversary. Trivially, we can compute both BR and NE and create a linear combination where we can control the safety by a parameter. However, previous work shows that we can perform significantly better and recover the whole Pareto set of maximally exploiting strategies with maximal safety [9]. We adapt the method to limited look-ahead solving, creating a continual depth-limited restricted Nash response (CDRNR). Similarly to the full robust response, CDRNR significantly outperforms the linear combination. However, it comes with drawbacks in the limited look-ahead solving. Namely, we need to keep the previously solved subgames as a path to the root to ensure theoretical soundness, which linearly increases the size of the game solved each step,

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Milec, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.

making it scalable to games with low depth like Poker or Goofspiel but impractical in games with high depth.

Our contributions are: **1)** We formulate the algorithms to find the responses given the opponent strategy and an evaluation function. This results in the best performing theoretically sound robust response applicable to large games. **2)** We prove the soundness of the proposed algorithms. **3)** We provide an analysis of problems that arise when using opponent models in limited look-ahead solving and propose a solution we call a full gadget. **4)** We empirically evaluate the algorithms on poker and goofspiel variants and compare them to multiple baselines. We show that our responses exploit the opponents, and CDBR outperforms domain-specific local best response [11] on poker. We also compare CDBR with the approximate best response (ABR) on smaller games and on full Heads-up No-Limit Texas Hold'em (HUNL), where we exploit SlumBot significantly more than ABR.

2 BACKGROUND

A two-player extensive-form game (EFG) consists of a set of players $N = \{1, 2, c\}$, where c denotes the chance, 1 is the maximizer and 2 is the minimizer, a finite set A of all actions available in the game, a set $H \subset \{a_1 a_2 \cdots a_n \mid a_j \in A, n \in \mathbb{N}\}$ of histories in the game. We assume that H forms a non-empty finite prefix tree. We use $g \sqsubset h$ to denote that h extends g. The *root* of H is the empty sequence $\emptyset$. The set of leaves of H is denoted Z, and its elements z are called *terminal histories*. The histories not in Z are *non-terminal histories*. By $A(h) = \{a \in A \mid ha \in H\}$, we denote the set of actions available at h. $P : H \setminus Z \to N$ is the *player function* which returns who acts in a given history. Denoting $H_i = \{h \in H \setminus Z \mid P(h) = i\}$, we partition the histories as $H = H_1 \cup H_2 \cup H_c \cup Z$. σ_c is the *chance strategy* defined on H_c. For each $h \in H_c, \sigma_c(h)$ is a fixed probability distribution over $A(h)$. Utility functions assign each player utility for each leaf node, $u_i : Z \to \mathbb{R}$. The game is zero-sum if $\forall z \in Z : \quad u_1(z) + u_2(z) = 0$. In the paper, we assume all the games are zero-sum. The game is of *imperfect information* if all players do not fully observe some actions or chance events. The information structure is described by *information sets* for each player i, which forms a partition $\mathcal{I}_i$ of H_i. For any information set $I_i \in \mathcal{I}_i$, any two histories $h, h' \in I_i$ are indistinguishable to player i. Therefore $A(h) = A(h')$ whenever $h, h' \in I_i$. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$.

A *strategy* $\sigma_i \in \Sigma_i$ of player i is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathcal{I}_i$. A *strategy profile* $\sigma = (\sigma_1, \sigma_2)$ consists of strategies for both players. $\pi^\sigma(h)$ is the probability of reaching h if all players play according to σ. We can decompose $\pi^\sigma(h) = \prod_{i \in N} \pi_i^\sigma(h)$ into each player's contribution. Let π_{-i}^σ be the product of all players' contributions except that of player i (including chance). For $I_i \in \mathcal{I}_i$ define $\pi^\sigma(I_i) = \sum_{h \in I_i} \pi^\sigma(h)$, as the probability of reaching information set I_i given all players play according to σ. $\pi_i^\sigma(I_i)$ and $\pi_{-i}^\sigma(I_i)$ are defined similarly. Finally, let $\pi^\sigma(h, z) = \frac{\pi^\sigma(z)}{\pi^\sigma(h)}$ if $h \sqsubset z$, and zero otherwise. $\pi_i^\sigma(h, z)$ and $\pi_{-i}^\sigma(h, z)$ are defined similarly. Using this notation, *expected payoff* for player i is $u_i(\sigma) = \sum_{z \in Z} u_i(z)\pi^\sigma(z)$. A *best response* (BR) of player i to the opponent's strategy σ_{-i} is a strategy $\sigma_i^{BR} \in BR_i(\sigma_{-i})$, where $u_i(\sigma_i^{BR}, \sigma_{-i}) \geq u_i(\sigma_i', \sigma_{-i})$ for all $\sigma_i' \in \Sigma_i$. A tuple of strategies $(\sigma_i^{NE}, \sigma_{-i}^{NE})$, $\sigma_i^{NE} \in \Sigma_i, \sigma_{-i}^{NE} \in \Sigma_{-i}$ is a *Nash Equilibrium* (NE) if σ_i^{NE} is an optimal strategy of player i against strategy σ_{-i}^{NE}. Formally: $\sigma_i^{NE} \in BR(\sigma_{-i}^{NE}) \quad \forall i \in \{1, 2\}$.

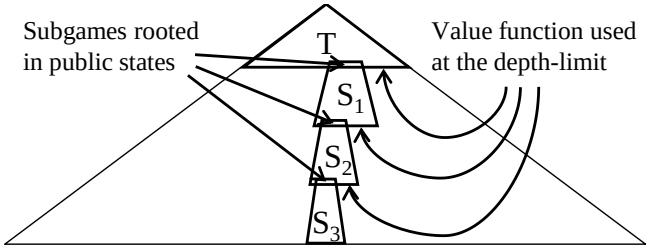

Figure 1: Illustration of the depth-limited solving.

In a two-player zero-sum game, the **exploitability** of a strategy is the expected utility a fully rational opponent can achieve above the value of the game. Formally, exploitability $\mathcal{E}(\sigma_i)$ of strategy $\sigma_i \in \Sigma_i$ is $\mathcal{E}(\sigma_i) = u_{-i}(\sigma_i, \sigma_{-i}) - u_{-i}(\sigma^{NE}), \quad \sigma_{-i} \in BR_{-i}(\sigma_i)$.

Safety is defined based on exploitability and ϵ-safe strategy is a strategy which has exploitability at most ϵ.

We define **gain** of a strategy against a model as the expected utility we receive above the value of the game. We formally define the gain $\mathcal{G}(\sigma_i, \sigma_{-i})$ of the strategy σ_i against a strategy σ_{-i} as $\mathcal{G}(\sigma_i, \sigma_{-i}) = u_i(\sigma_i, \sigma_{-i}) - u_i(\sigma^{NE})$.

Depth-limited Solving - Figure 1. We denote $H_i(h)$ the sequence of player i's information sets and actions on the path to a history h. Two histories h, h' where player i does not act are in the same *augmented information set* I_i if $H_i(h) = H_i(h')$. We partition the game histories into **public states** $PS \subset H$, which are closed under the membership within the augmented information sets of all players. **Trunk** is a set of histories $T \subset H$, closed under prefixes and public states. **Subgame** $S \subset H$ is a forest of trees with all the roots starting in one public state. It is closed under public states, and the trees can end in terminal public states or often end after a number of moves or rounds in the game. **Range** of a player i is a probability distribution over his information sets in some public state PS_i, given we reached the PS_i. **Value function** is a function that takes the public state and both players' ranges as input and outputs values for each information set in the public state for both players. We assume using an approximation of an **optimal value function**, which is a value function returning the values of using some NE after the depth-limit. **Subgame partitioning** $\mathcal{P}$ is a partitioning that splits the game into trunk and subgames into multiple different levels based on some depth-limit or other factors (domain knowledge). Subgame partitioning can be naturally created using the formalism of factored-observation stochastic games [**?**]. By $u_i(\sigma)_V^T$, we denote the utility for player i if we use strategy profile σ in trunk T and compute values at the depth-limit using value function V. When resolving a subgame with just the ranges, there are no guarantees on the resulting exploitability of the strategy in the full game, and the exploitability can rise significantly [4]. To address the issue, artificially constructed games called **gadgets** are used to limit the increase in exploitability. They do it by adding nodes to the top of the subgame, which simulates that the opponent is allowed to deviate from its strategy in an already solved game.

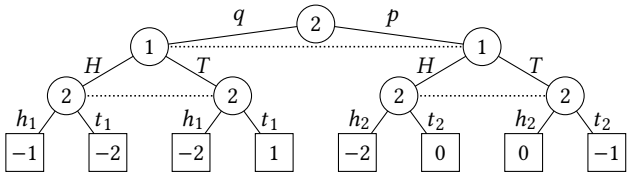

Figure 2: Simple zero-sum imperfect-information game. Nodes denote the decisions of the players, dotted lines mark information sets, and the leaf shows for player 1

Figure 2 shows a simple game illustrating depth-limited solving. The game starts with player 2 choosing to either play standard biased matching pennies (p) or playing his own version of the game (q). In the next round, player 1 does not know which game player 2 chose, and he chooses head (H) or tail (T). Then player 2 guesses head (h_i) or tail(t_i), and if he chooses to play the standard version, he receives 2 when correctly guessing head and 1 when correctly guessing tails. Otherwise, the reward is 0. In the modified version, guessing incorrectly gives 2 to the player 2, and guessing correctly gives 1 for heads and -1 for tails.

In the Nash equilibrium of this game, player 1 plays heads with probability $\frac{2}{3}$ and player 2 chooses his own version of the game with probability $\frac{2}{3}$ and follows with only heads. Public states in this game are always the whole levels (rows) since the actions are never observable by both players. When we start depth-limited solving, we create a trunk, which we select as just the root with the choice to play q or p. We start solving the trunk using an iterative algorithm, e.g., counterfactual regret minimization (CFR) [25].

We initialize strategy to uniform, which gives us *range* in the next public state $(\frac{1}{2}, \frac{1}{2})$. We give the *range* to the value function, which returns values as if we played equilibrium in the rest of the game. Value function gives us values in the information sets, which translates to the utility of $-\frac{5}{3}$ for playing heads and $-\frac{2}{3}$ for playing tails. We use the values to update regrets in the CFR and perform the next iteration similarly. When we solve the trunk and recover the equilibrium strategy for the first node, we move to a subgame, for example, a game starting in the information set of player 1 and ending after his action. We need to reconstruct what happened earlier. If we replace the already computed strategy with a chance node, which is called unsafe resolving, we are not guaranteed to recover the equilibrium for player 1. Unsafe resolving can produce solutions ranging from heads with probability $\frac{3}{4}$ to $\frac{1}{3}$ but the only equilibrium is heads with probability $\frac{2}{3}$. The situation is fixed using the mentioned gadgets, which allow the opponent to modify their range above the subgame, forcing the other player to play robustly against all the possible ranges and recover the equilibrium.

3 FULLY EXPLOITING THE OPPONENT

Fully exploiting opponent models in small games boils down to computing a best response. This is infeasible in games with an intractable number of information sets for which we use the continual depth-limited solving algorithms. The depth-limited setting does not allow computing BR in one pass anymore. The game we already saw in Figure 2 can be an example of that. Suppose we know the player 2 always makes a mistake in the first move and plays only to the standard biased matching pennies. If we knew his strategy of guessing heads or tails, we could compute a best response. However, our trunk will end before the choice, and we need to use a value function. Since the value function in this simple case is just a best response of the opponent, the problem is reduced to finding the optimal strategy against a best response, which corresponds to finding NE, and it can not be solved in one pass. In this section, we propose an algorithm for continual depth-limited best response (CDBR), which generalizes a best response to be used with a value function for depth-limited solving.

Continual Depth-limited Best Response

Given any extensive-form game G with perfect recall, opponent's fixed strategy σ_2^F and some subgame partitioning $\mathcal{P}$, we define continual depth-limited best response (CDBR) recursively from the top, see Figure 1. First, we have trunk $T_1 = T$ and value function V. CDBR in the trunk T_1 for player 1 with value function V is defined as $\mathcal{B}(\sigma_2^F)_V^{T_1} = \arg\max_{\sigma_1} u_1(\sigma_1, \sigma_2^F)_V^{T_1}$. In other words, we maximize the utility over the strategy in the trunk, where we return values from the value function after the depth limit. In each step afterward, we create a new subgame S_i and create new trunk by joining the old one with the subgame, creating $T_i = T_{i-1} \cup S_i$. We fix the strategy of player 1 in the T_{i-1} and maximize over the strategy in the subgame. $\mathcal{B}(\sigma_2^F)_V^{T_i} = \arg\max_{(\sigma_1^{S_i})} u_1(\sigma_1^{S_i} \cup \sigma_1^{T_{i-1}}, \sigma_2^F)_V^{T_i}$. We continue like that for each step, and we always create a new trunk T_i using the strategy from step T_{i-1} until we reach the end of the game. We denote the full CDBR strategy created by joining strategies from all possible branches $\mathcal{B}(\sigma_2^F)_V^{\mathcal{P}}$.

Intuitively, we always solve the game until the depth limit. The opponent is fixed everywhere above the depth limit, and the rational player is fixed in the already solved parts, and she can play in the part that was added last. Looking at Figure 1 CDBR in S_2 would allow player 1 to play in S_2, it would replace anything bellow S_2 with a value function, player 1 would be fixed in T and S_1 and player 2 would be fixed in T, S_1 and S_2.

Computing CDBR and the complexity. In practice, we will compute CDBR similarly to depth-limited solving with a few key changes. First, we fix the opponent's strategy in the currently resolved part of the game to allow the player to respond to it, which corresponds to the argmax from the definition. Another key change that simplifies the algorithm is that we no longer need a gadget since the opponent is fixed in the parts we already played through, so we do not need to be robust against different ranges than the one taken from the opponent model.

The difference from the standard depth-limited solving is that we fix the opponent's strategy in the resolved part of the game, and we do not use a gadget. Hence, there is less computation required compared to the standard depth-limited solving.

Convergence in current iterations. CFR is an algorithm that needs to track average strategies since the current strategy does not converge to an equilibrium. CFR against best response or a fixed strategy is known to converge in the current strategy [5, 13]. The next lemma says that CDBR also converges in the current strategy even when a value function is used after the depth-limit.

Lemma 3.1. *Let G be a zero-sum imperfect-information extensive-form game. Let σ_2^F be the fixed opponent's strategy, and let T be some trunk of the game. If we perform CFR with t iterations in the trunk for player 1, then for the strategy $\hat{\sigma}_1$ from the iteration with highest expected utility* $\max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, \sigma_2^F)_V^T - u_1(\hat{\sigma}_1, \sigma_2^F)_V^T \leq \Delta\sqrt{\frac{A}{t}|\mathcal{I}_{TR}|} + tN_S\epsilon_S$ *where Δ is a span of leaf utilities, $\Delta = \max_{z \in Z} u_i(z) - \min_{z \in Z} u_i(z)$, A is an upper bound on the number of actions, $|\mathcal{I}_{TR}|$ is a number of information sets in the trunk, N_S is the number of information sets at the root of any subgame, and value function error is at most ϵ_S.*

4 SAFE MODEL EXPLOITATION

While CDBR maximizes the exploitation of the fixed opponent model, it allows a player to be exploited. When we face an opponent unsure if our model is perfect we must limit our exploitability. For example, when we gradually build a model during play, we must limit our exploitability in the initial game rounds when the model is still very inaccurate.

Combination of CDBR and Nash Equilibrium

The combination of CDBR and Nash equilibrium (CDBR-NE) is the first approach to limit exploitability. We can simultaneously compute both strategies using depth-limited solving and do a linear combination in every decision node. Let p be the linear combination parameter and σ_2^F be the opponent model. The gain and exploitability are limited accordingly.

$$\sigma_1^{LC} = p\sigma_1^{NE} + (1-p)\mathcal{B}(\sigma_2^F)_V^{\mathcal{P}}$$
$$\mathcal{E}(\sigma_1^{LC}) = p\mathcal{E}(\sigma_1^{NE}) + (1-p)\mathcal{E}(\mathcal{B}(\sigma_2^F)_V^{\mathcal{P}})$$
$$\mathcal{G}(\sigma_1^{LC}, \sigma_2^F) = p\mathcal{G}(\sigma_1^{LC}, \sigma_2^F) + (1-p)\mathcal{G}(\mathcal{B}(\sigma_2^F)_V^{\mathcal{P}}, \sigma_2^F)$$

Desired exploitability or gain may be achieved by tuning the parameter p while being only two times slower than the CDBR since we need to find the Nash equilibrium separately and perform CDBR. The required value function is the same for both parts and is still the same as in standard depth-limited solving.

Required computation is exactly running standard depth-limited solving and CDBR in parallel. Since CDBR computation has standard depth-limited solving as an upper bound, the required computation is at most twice as much as standard depth-limited solving.

Continual Depth-limited RNR

CDBR-NE is safe, but [9] shows we can get a much better trade-off between gain and exploitability using RNR as it recovers the optimal Pareto set of ϵ-safe best responses [14]. It also gives us better control of safety as it links the allowed exploitability to the achieved gain. We combine depth-limited solving with RNR to create CDRNR.

Description of Restricted Nash Response. For CDRNR, we first need to explain the RNR method briefly [9]. RNR is solved by computing a modified game, adding an initial chance node with two outcomes that player 1 does not observe. We copy the whole game tree under both chance node outcomes, and in one tree, the opponent plays the fixed strategy, and we denote it G^F. In the other tree, the opponent can play as he wants, resulting in a best response to the strategy of player 1. We denote the other tree G'. Since player 1 does not observe the initial chance node, his information sets span over G' and G^F, and we denote the full modified game with both trees G^M. Parameter p is the method to control the safety and is the initial probability of picking G^F.

Definition. Given the opponent's fixed strategy σ_2^F and some subgame partitioning $\mathcal{P}$ of G^M, we define continual depth-limited restricted Nash response (CDRNR) recursively from the top. First, we have trunk T_1^M using $\mathcal{P}$ and value function V. CDRNR for player 1 in the trunk T_1^M using value function V is $\mathcal{R}(\sigma_2^F, p)_V^{T_1^M} = \arg\max_{\sigma_1} u_1(\sigma_1, BR(\sigma_1))_V^{T_1^M}$. And then, in every following step, we create the new subgame S_i^M and enlarge the trunk to incorporate this subgame, creating trunk $T_i^M = T_{i-1}^M \cup S_i^M$. Next, we fix strategy $\sigma_1^{T_{i-1}^M}$ of player 1 in the previous trunk T_{i-1}^M and the CDRNR is $\mathcal{R}(\sigma_2^F, p)_V^{T_i} = \arg\max_{\sigma_1^{S_i^M}} u_1(\sigma_1', BR(\sigma_1'))_V^{T_i^M}$ where σ_1' is a combination of the strategy we optimize over and the fixed strategy from the previous step, formally $\sigma_1' = \sigma_1^{S_i^M} \cup \sigma_1^{T_{i-1}^M}$.

To summarize, we optimize only over the strategy in the subgame used in the current step while the strategy in the previous parts of the game is fixed for player 1. The strategy of the opponent is fixed in G^F and free in G'. We denote the full CDRNR strategy $\mathcal{R}(\sigma_2^F, p)_V^{\mathcal{P}}$.

Computing CDRNR. In practice, we want to avoid duplicating the tree, and we also want to use the exact same value function as in the standard depth-limited solving. We explain why the RNR does not need the duplicated trees in practice. It only needs the reaches of the fixed strategy injected to the terminal nodes in the ratio defined by the parameter p. This allows us to precompute the reaches, run CFR as in standard depth-limited solving, and then modify the computed reaches from the iteration using the precomputed fixed reaches. However, we also need to query the value function, which differs from the previous one in the theoretical definition as it spans over the modified public state. However, since the reaches of p_1 are the same for G' and G_F we can compute it only once by joining the reaches together as in the previous example and querying the standard value function.

So far, we described exactly the standard depth-limited solving with only one modification: modifying the reaches using the fixed strategy. We also use the gadget since now the opponent can deviate in the G'. However, standard gadgets will fail due to the addition of imperfect parts of the opponent, and we discuss details along with a solution in the next section.

5 GADGETS AND MODEL EXPLOITATION

When we exploit an opponent model, we need to worsen the strategy in terms of exploitability. We must limit how much the strategy worsens if we want a safe response. Gadgets are used to ensure exploitability does not increase [2, 4, 17], and all the common gadgets work in scenarios where we do not expect our strategy to worsen. However, we need to worsen our strategy to exploit the opponent. We try to gain as much as possible in RNR in G^F. As soon as the strategy gets worse and the exploitability increases, the common gadgets fail to quantify this increase, which is crucial

in applications doing a delicate trade-off. The requirement for the gadget which would work in CDRNR is in Definition 5.1

Definition 5.1. For each information set $I \in \mathcal{I}_1$ we need the value of its part in G', formally $\sum_{h \in I, h \in G', z \in Z, h \sqsubset z} \pi^{\sigma}(z) u_1(z)$, to be the same as the value we would get if we let player 2 play BR in full G'.

The following examples show that the requirement is not satisfied for common resolving gadgets. We tried to construct a gadget that would satisfy the condition, but in the end, we kept the previously resolved parts of the game G' followed by the value function which the game does not follow. We call the construction the full gadget, it satisfies the condition, and still only increases the size of the solved part linearly. Constant-size gadget fulfilling the Definition 5.1 is an open problem.

Restricted Nash Response with Gadget

We show that commonly used resolving gadgets are either overestimating or underestimating the values from Definition 5.1 on an example game in Figure 3. In the game, we first randomly pick a red or green coin. Player 2 observes this and decides to place the coin heads up (RH, GH) or tails up (RT, GT). Player 1 cannot observe anything and ultimately chooses whether he wants to play the game (P) or quit (Q).

In equilibrium, player 1 plays action Q, and player 2 can mix actions up to the point where the utility for P is at most 0. This gives the value of the game 0, and counterfactual values in all inner nodes are also 0. Assuming the modified RNR game G^M with an opponent model playing GT, that makes it worth for player 1 to play (P) in the game, 2 will play (RH, GH) in G' with utility -3 for player 1 in G'. We will use gadgets to resolve the game from the player 1 information set.

Resolving Gadget. [4] Resolving gadget constructs a game that allows the opponent to choose whether he wants to play in the subgame we created or terminate. It is done by inserting nodes above the roots of the subgame, and the opponent has two actions before each root, either to follow and play the game or to terminate and receive a reward they would get by playing the previously resolved equilibrium. Those nodes are grouped into information sets based on the opponent's augmented information sets at the subgame's roots.

Resolving gadget on the game in Figure 3 has all utilities after *terminate* actions 0. When we resolve the gadget, the utility is 0. However, when player 1 deviates to action P, player 2 plays *follow* action in all but the rightmost node, and the utility of player 1 will be -3.5. Therefore, the common resolving gadget may overestimate the real exploitability of the strategy in the subgame. Overestimating may lead to not exploiting as much as we can and makes it impossible to prove Theorem 5.2 about the minimal gain of our algorithm. Normalization of the chance node might seem to solve the problem, but it would only halve the value to -1.75, which is still incorrect.

(Reach) Max-margin Gadget. [2, 17] Both reach max-margin and max-margin gadgets allow the opponent to choose any information set at the start of the subgame. This is done by inserting a single node to the top, where the opponent has an action for each of his augmented information sets in the root of the subgame. After the action is a chance node to split the information set to the histories, with correct reaches by the resolving player and chance. Furthermore, all the terminal values are adjusted by the same value, which is in the terminate action in the resolving gadget. In the reach max-margin gadget, this value is further modified by an approximation of opponent mistakes.

All the counterfactual best response values are 0, and we assume both players played perfectly before the depth limit. Hence, we do not need to offset any node in the (reach) max-margin gadget. Then, both gadget constructions are identical. We add the initial decision node and the chance nodes (since there is only one state in each information set, the nodes have only one action). When we solve the gadget, player 1 will pick action Q, and the gadget value will be 0. However, when player 1 deviates to action P, player 2 now has a choice between terminal utilities and picks action E to receive the highest one. This will result in utility -2, and we see that (reach) max-margin gadgets can underestimate the real exploitability. It can lead to our algorithm being more exploitable than we want using some p, and it makes Theorem 5.3 impossible to prove. Similar to the previous gadget, normalizing the chance nodes would lead to double the utility, which is still incorrect.

Full Gadget. The only construction fulfilling the requirements we found is to keep all the previously explored parts of the game in a path to the root and use a value function when we leave. Using the optimal value function, the construction simulates the best response, which measures exploitability.

Formally, when we reach subgame S_i we construct a composite game by joining S_i, the trunk T, and all the previous subgames S_j, $j \in 1, ..., i-1$. It corresponds to the illustration in Figure 1, and the value function will evaluate every public state PS from which the actions lead outside of the game.

We show that other gadgets can overestimate or underestimate exploitability, which could shift the distribution of the parameter p, and we could still compute the same solutions. However, in Figure 4, we show the results of games created to break the other gadgets. We have two games in which the full gadget behaves correctly, and each breaks the other gadget. Figure 4 shows the results joined together. Both games have five actions for the exploiter, and each one is crucial in reconstructing the full RNR set. The full gadget can recover all five actions using different p, but other gadgets can only compute two actions regardless of the choice of p.

Complexity of CDRNR. We can use other gadgets in CDRNR to obtain fast algorithms without any theoretical guarantees and with the same bound on computation as we have for the combination of CDBR and Nash equilibrium. The soundness of the algorithm relies on using the full gadget, which requires solving increasingly larger parts of the game as the depth increases. This increase in size is linear with the resolving steps, so the full algorithm complexity is quadratic in the depth of the game compared with vanilla continual resolving or CDBR. It makes the algorithm applicable to shorter games like Poker or Goofspiel but infeasible for long games like Stratego.

Soundness of CDRNR. For the following theorems, we denote $\mathcal{S}$ as the set consisting of the trunk and all the subgames explored when

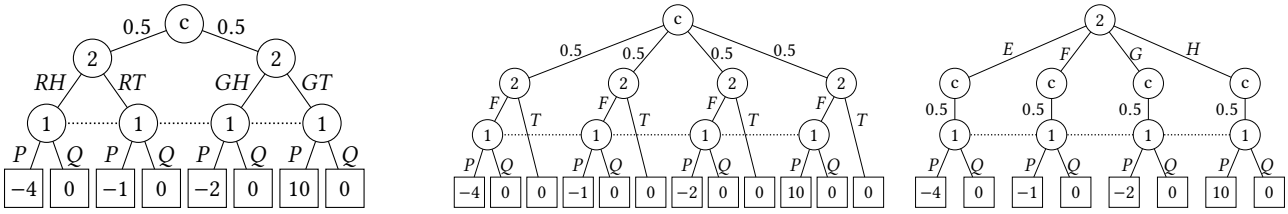

Figure 3: (left) A game to show problems with gadgets. (middle) Resolving gadget for the left game. (right) Max-margin and Reach max-margin gadget.

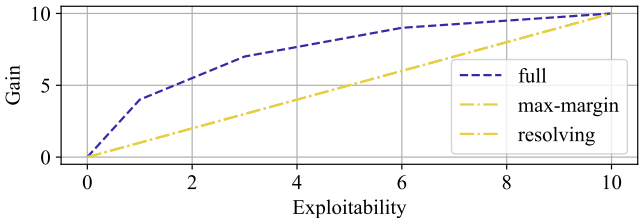

Figure 4: Comparison of Gain and Exploitability of the solutions using Full gadget compared to other gadgets. Max-margin and resolving gadget lines are overlapping.

computing the response, and $\mathcal{S}'$ is the same but without the last subgame. S^B denotes a border of the subgame. We also denote S^O, the set of all the states where we leave the trunk not going in the currently resolving subgame.

Theorem 5.2 (Gain of CDRNR). *Let G be any zero-sum extensive-form game and let σ_2^F be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using σ_2^F. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma_1^{\mathcal{R}}$ be a CDRNR given approximation $\bar{V}$ of value function V with error at most ϵ_V and opponent strategy σ_2^F approximated in each step with regret at most ϵ_R, formally $\sigma_1^{\mathcal{R}} = \mathcal{R}(\sigma_2^F, p)_V^{\mathcal{P}}$. Let σ^{NE} be any Nash equilibrium in G. Then $u_1(\sigma_1^{\mathcal{R}}, \sigma_2^F) + \sum_{S \in \mathcal{S}'} |I_{S^O}|(1 - p)\epsilon_V + |\mathcal{S}|\epsilon_R + \sum_{S \in \mathcal{S}'} |I_{S^B}|\epsilon_V \geq u_1(\sigma^{NE})$.*

The previous theorem states that our approaches will receive at least the value of the game when responding to the model. All the proofs are in the appendix.

Theorem 5.3 (Safety of CDRNR). *Let G be any zero-sum extensive-form game and let σ_2^F be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using σ_2^F. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma_1^{\mathcal{R}}$ be a CDRNR given approximation $\bar{V}$ of the optimal value function V with error at most ϵ_V, partitioning $\mathcal{P}$ and opponent strategy σ_2^F, which is approximated in each step with regret at most ϵ_R, formally $\sigma_1^{\mathcal{R}} = \sigma_1^{\mathcal{R}}(\sigma_2^F, p)_V^{\mathcal{P}}$. Then exploitability has a bound $\mathcal{E}(\sigma_1^{\mathcal{R}}) \leq \mathcal{G}(\sigma_1^{\mathcal{R}}, \sigma_2^F)\frac{p}{1-p} + \sum_{S \in \mathcal{S}'} |I_{S^O}|(1 - p)\epsilon_V + |\mathcal{S}|\epsilon_R + \sum_{S \in \mathcal{S}'} |I_{S^B}|\epsilon_V$, $\mathcal{E}$ and $\mathcal{G}$ are defined in Section 2.*

The last theorem is more complex, and it bounds the exploitability by the gain of the strategy against the model. With $p = 0$, it is reduced to the continual resolving, and with $p = 1$ to CDBR with unbounded exploitability. The theorem shows the parameter p directly links allowed exploitability to the gain we receive. The same works in RNR without the resolving and value errors; as far as we know, the authors do not explicitly mention it.

SES has bound relies on opponent estimation being close to an equilibrium strategy. When the estimation is more different, the bound is infinity for a large portion of the parameter alpha. We give a detailed explanation in the appendix.

More intuitively, Theorem 5.2 says that by playing the proposed algorithm, we have at least the same safety guarantees we would get by playing a NE against the opponent we modeled correctly. Theorem 5.3 allows us to choose a trade-off between the exploitation of the opponent behaving according to the model and safety against an opponent who would deviate arbitrarily from the model.

6 EXPERIMENTS

We compared CDBR and local best response (LBR) [11]. We empirically show the performance of CDRNR and explore the trade-off between exploitability and gain in CDRNR. The appendix contains hardware setup, domain description, algorithm details, and experiments on more domains. We use two types of opponent strategies: strategies generated by few CFR iterations and random strategies with different seeds.

SES explanation

Safe exploitation search (SES) [12] is a similar method to the one we propose. However, there are two significant differences. First, the method uses a max-margin gadget without the analysis we did. Hence, the bound of exploitability is very loose, and for a wide range of inputs, the exploitability can be unbounded. Second, the method does not fix the opponent's strategy at all and only uses opponent reaches when resolving the subgame. As a result, SES exploitation is very limited, and as we show in experiments, it is often worse than using the best Nash equilibrium. On top of that, in some games, it fundamentally cannot exploit the opponent, notably in any perfect information game, even with simultaneous moves.

Exploitability of Robust Responses

We report both gain and exploitability for CDRNR on Leduc Hold'em. Results in Figure 5 show that the proven bound on exploitability works in practice, and we see that the bound is very loose in practice. For example, with $p = 0.5$, the bound on the exploitability is the gain itself, but the algorithm rarely reaches even a tenth of the gain in exploitability. This shows that the CDRNR is similar to the restricted Nash response because, with a well-set p, it can

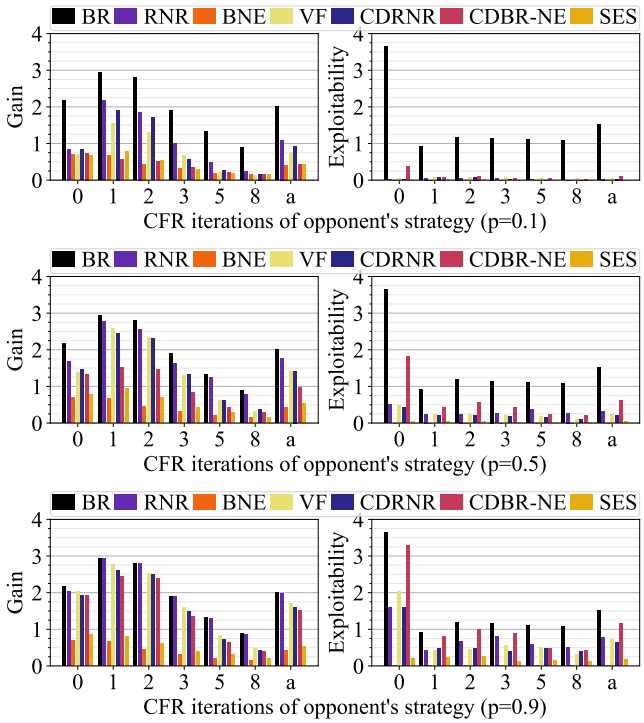

Figure 5: Gain and exploitability comparison of BR, RNR, best Nash equilibrium (BNE), CDBR-NE, SES, and CDRNR in Leduc Hold'em against strategies from CFR using a small number of iterations with different p values. The a stands for the average of the other values. VF is CDRNR using an imperfect value function.

significantly exploit the opponent without significantly raising its exploitability. The only exception is the CDRNR with a value function, which shows the added constants from the value function's imprecision. When the opponent is close to optimal, we see that the exploitability can rise above the gain.

In most cases, the gain and exploitability of CDRNR are lower than that of RNR. Gain must be lower because of the different value function, but the exploitability can be higher, as seen with $p = 0.1$ against low iteration strategies, due to the depth-limited nature. We provide results on other domains and for more parameter values p in the appendix.

We also compare the algorithm against the best possible Nash equilibrium. We compute the best NE by a linear program, and it serves as the theoretical limit of maximal gain, which does not allow exploitability. It would be impossible to compute for larger games. We can see we can gain more than twice as much, with exploitability still being almost zero.

In our results on smaller games, we use the optimal value function computed by a linear program. To show the performance of imperfect value function we included results where the value function is approximated by CFR with 500 iterations. As expected, the imperfect value function slightly decreases the performance but is comparable to the algorithms using the optimal value function.

	ABR	CDBR1	CDBR3	CDBR5	BNE
Leduc	98%	74.4%	96%	97.1%	32.6%
IIGS4	97%	98.5%	100%	100%	77.1%
IIGS5	95%	93.5%	100%	100%	50.4%
IIGS6	97%	90.7%	100%	100%	45.9%

Table 1: Comparison of ABR and CDBR with BNE baseline on different games against uniform random. The values are the percentage of gain achieved by the best response.

The last comparison is with SES, which performs poorly, and its gain is only slightly above the best Nash equilibrium. Conversely, it is almost not exploitable. Our results are consistent with results in the paper [12] and are a direct consequence of using the information from the opponent model only to set the reaches to the subgame. Only reaches are not enough to do meaningful exploitation, and SES produces strategies that are very close to Nash equilibrium.

ABR vs CDBR

We compared CDBR with ABR [21] on Leduc and different imperfect information Goofspiel. The results are in Table 1, showing that our method is slightly behind in Leduc, even with the highest search depth. In Goofspiel, CDBR1, which looks only one action into the future, is already pretty good, and as soon as we allow CDBR to look three turns in the future, it fully exploits the opponents. In IIGS4, that is half of the game, but in IIGS6, it is less than $\frac{1}{3}$ of the game, and it is still enough.

Local Best Response vs. CDBR

We compare LBR and CDBR in Leduc Hold'em. We also compare CDBR with the BR in imperfect information Goofspiel 5, but without LBR, which is poker-specific. We show that CDBR and LBR are very similar with smaller steps, and as we increase the depth limit of CDBR, it starts outperforming LBR. The behavior differs slightly based on the specific strategy because LBR assumes the player continues the game by only calling till the end, while CDBR uses the perfectly rational extension.

The results in Figure 6 show that both concepts are good at approximating the best response, with CDBR being better against both strategies. LBR looks at one following action, so it is best compared to the CDBR1 in terms of comparability. Next, we observe a lack of monotonicity in step increase, which is explained with an example in the appendix. When we increase the depth limit, the algorithm can exploit some early mistake that causes it to miss a sub-tree where the opponent makes a much bigger mistake in the future. We can clearly see the difference between the algorithm with guarantees and LBR without them. Against strategy from 34 CFR iterations, LBR can no longer achieve positive gain and only worsens with more iterations. In contrast, CDBR can always achieve at least zero gain, assuming we have an optimal value function.

Playing SlumBot

We tested our method in HUNL against SlumBot [7], which is a publicly available abstraction-based bot commonly used for benchmarking. We used a fold, call, pot, and all-in (FCPA) abstraction

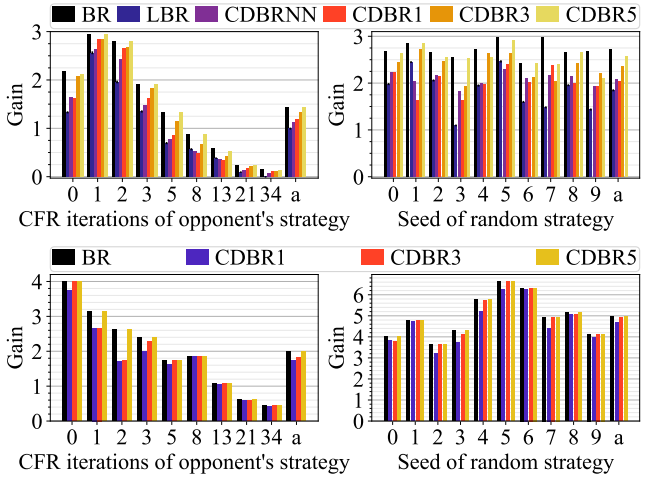

Figure 6: Gain comparison of best response (BR), local best response (LBR - only poker), and continual depth-limited best response (CDBR) in Leduc Hold'em (top) and IIGoofspiel 5 (bottom) against strategies from CFR using a small number of iterations (left) and random strategies (right). The a stands for the average of the other values in the plot. The number after CDBR stands for the number of actions CDBR was allowed to look at in the future, and CDBRNN is a one-step CDBR with a neural network as a value function.

	ABR	LBR	CDBR
Win-rate [mbb/h]	1259 ± ?	1388 ± 150	1774 ± 137

Table 2: Comparison of CDBR with LBR and ABR against SlumBot. Results are reported in milibigblinds per hand (mbb/h) with 95% confidence intervals. (Authors of ABR did not provide confidence intervals.)

for CDBR, and we also rerun the results of LBR against the new SlumBot (since the author of SlumBot confirmed that the strategy we were provided is different from the one LBR used but the same as the one provided to authors of ABR). CDBR significantly outperforms both ABR and LBR and we report the results in Table 2. We tried to run the LBR restricted to only call in the first two rounds but found it no longer helps against the new SlumBot, and we reported results for LBR with FCPA. Authors in [21] also use FCPA for their method but did not report a confidence interval for the results. However, the difference is large enough to have statistical significance if we assume they played over 50 thousand hands.

7 RELATED WORK

This section describes the related work focusing more on distinguishing our novel contributions.

Restricted Nash response (RNR) [9] is an opponent-exploiting scheme. It solves the entire game and allows changing the trade-off between exploitability and gain. Essentially it always produces ϵ safe best response [14]. It accomplishes the goal by copying the whole game and then fixing the opponent in one part while having a chance node at the top decide which game we play.

However, it is impossible to compute RNR in huge games, and we fused the RNR approach with depth-limited solving creating a novel algorithm we call CDRNR. CDRNR is the best performing theoretically sound robust response calculation that can be done in huge games, enabling new opponent exploiting approaches.

Local best response [11] is an evaluation tool for poker. It uses a given abstraction in its action space. It picks the best action in each decision set, looking at the fold probability for the opponent in the next decision node and then assuming the game is called until the end. Our algorithm CDBR is a generalization of the LBR because we can use it on any game solvable by depth-limited solving. In the algorithm, we have explicitly defined value function, which we can exchange for different heuristics.

Approximate best response (ABR) [21] is also a generalization of the LBR and showed promising results in evaluating strategies. However, our approach focuses on model exploitation, which requires crucial differences, such as quick re-computation against unseen models. ABR needs to independently learn the response for every combination of opponent and game, making it unusable in the opponent modeling scenario. Our algorithms learn a single domain-specific value function and can subsequently compute strategies against any opponent in the run-time. Furthermore, ABR and even CDBR are extremely brittle, making it a bad choice if we are unsure about the opponent, which we often are in a game against an unknown opponent. On the other hand, CDRNR tackles exactly this issue and provides powerful exploitation with very limited exploitability.

Another reinforcement-learning (RL) method uses neuroevolution with RL, they show a significant increase in performance over DQN and evaluate their method on HUNL. However, they do not share the details of the baseline opponents they played against. We tried to contact the authors without any response and we could not compare the performance with our methods. [23]

8 CONCLUSION

Opponent modeling and exploitation is an essential topic in computational game theory, with many approaches attempting to model and exploit opponents in various games. However, exploiting opponents in very large games is not trivial, and only recently was an algorithm created to exploit models in depth-limited solving. We explain the problem arising from the inability of gadgets to measure exploitability and we propose a full gadget that solves the issue. We propose a new algorithm to quickly compute depth-limited best response and depth-limited restricted Nash response once we have a value function, creating the best performing theoretically sound robust response applicable to large games. Finally, we empirically evaluate the algorithms on multiple games. We show that CDBR outperforms LBR in both Leduc and HUNL and we show that CDBR performs significantly better against SlumBot than any other previous method. Finally, we show that CDRNR outperforms SES in any game and can achieve over half of the possible gain without almost any exploitability.

ACKNOWLEDGEMENTS

This research was supported by Czech Science Foundation (grant no. GA22-26655S) and and the Grant Agency of the Czech Technical University in Prague, grant No. SGS22/168/OHK3/3T/13. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and also the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” which are both gratefully acknowledged. We also greatly appreciate the help of Eric Jackson, who provided the SlumBot strategy and helped with the experiments.

REFERENCES

[1] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. 2013. Online implicit agent modelling. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. 255–262.

[2] Noam Brown and Tuomas Sandholm. 2017. Safe and nested endgame solving for imperfect-information games. In *Workshops at the thirty-first AAAI conference on artificial intelligence*.

[3] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.

[4] Neil Burch, Michael Johanson, and Michael Bowling. 2014. Solving imperfect information games using decomposition. In *Twenty-eighth AAAI conference on artificial intelligence*.

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

[6] Sam Ganzfried and Tuomas Sandholm. 2015. Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)* 3, 2 (2015), 1–28.

[7] Eric Griffin Jackson. 2017. Targeted cfr. In *Workshops at the thirty-first AAAI conference on artificial intelligence*.

[8] Michael Johanson and Michael Bowling. 2009. Data biased robust counter strategies. In *Artificial Intelligence and Statistics*. 264–271.

[9] Michael Johanson, Martin Zinkevich, and Michael Bowling. 2008. Computing robust counter-strategies. In *Advances in neural information processing systems*. 721–728.

[10] Kevin B Korb, Ann Nicholson, and Nathalie Jitnah. 2013. Bayesian poker. *arXiv preprint arXiv:1301.6711* (2013).

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

[12] Mingyang Liu, Chengjie Wu, Qihan Liu, Yansen Jing, Jun Yang, Pingzhong Tang, and Chongjie Zhang. 2022. Safe Opponent-Exploitation Subgame Refinement. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=YpHb0IVJu92

[13] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr TImbers, and Karl Tuyls. 2019. Computing Approximate Equilibria in Sequential Adversarial Games by Exploitability Descent. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 464–470.

[14] Peter McCracken and Michael Bowling. 2004. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*. 103–110.

[15] Richard Mealing and Jonathan L Shapiro. 2015. Opponent modeling by expectation–maximization and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games* 9, 1 (2015), 11–24.

[16] David Milec, Jakub Černý, Viliam Lisý, and Bo An. 2021. Complexity and Algorithms for Exploiting Quantal Opponents in Large Two-Player Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5575–5583.

[17] Matej Moravcik, Martin Schmid, Karel Ha, Milan Hladik, and Stephen Gaukrodger. 2016. Refining subgames in large imperfect information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[18] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. 2017. Deepstack: Expert-level artificial intelligence in Heads-Up No-Limit Poker. *Science* 356, 6337 (2017), 508–513.

[19] Martin Schmid, Matej Moravcik, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, Zach Holland, et al. 2021. Player of Games. *arXiv preprint arXiv:2112.03178* (2021).

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

[21] Finbarr Timbers, Edward Lockhart, Marc Lanctot, Martin Schmid, Julian Schrittwieser, Thomas Hubert, and Michael Bowling. 2020. Approximate exploitability: Learning a best response in large games. *arXiv preprint arXiv:2004.09677* (2020).

[22] Zhe Wu, Kai Li, Enmin Zhao, Hang Xu, Meng Zhang, Haobo Fu, Bo An, and Junliang Xing. 2021. L2E: Learning to Exploit Your Opponent. *arXiv preprint arXiv:2102.09381* (2021).

[23] Jiahui Xu, Jing Chen, and Shaofei Chen. 2021. Efficient opponent exploitation in no-limit Texas hold'em poker: A neuroevolutionary method combined with reinforcement learning. *Electronics* 10, 17 (2021), 2087.

[24] Ryan Zarick, Bryan Pellegrino, Noam Brown, and Caleb Banister. 2020. Unlocking the potential of deep counterfactual value networks. *arXiv preprint arXiv:2007.10442* (2020).

[25] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2008. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*. 1729–1736.