Continual Depth-limited Responses for Computing Counter-strategies in Extensive-form Games

David Milec, Viliam Lisý
AI Center, FEE, Czech Technical University in Prague, Czech Republic
milecdav@fel.cvut.cz, viliam.lisy@agents.fel.cvut.cz

Abstract

In real-world applications, game-theoretic algorithms often interact with imperfect opponents, and incorporating opponent models into the algorithms can significantly improve performance. Opponent exploitation approaches often use the best response or robust response to compute counter-strategy to an opponent model updated during the game-play or to build a portfolio of exploitative strategies beforehand. However, in massive games with imperfect information, computing exact responses is intractable. Existing approaches for best response approximation are either domain-specific or require an extensive computation for every opponent model. Furthermore, there is no approach that can compute robust responses in massive games. We propose using depth-limited solving with optimal value function to approximate the best response and restricted Nash response. Both approaches require computing the value function beforehand, but then allow computing the responses quickly even to previously unseen opponents. Furthermore, we provide a utility lower bound for both approaches and a safety guarantee for the robust response. Our best response approach can also be used for evaluating the quality of strategies computed by novel algorithms through approximating exploitability. We empirically evaluate the approaches in terms of gain and exploitability, compare the depth-limited responses with the poker-specific local best response, and show the robust response indeed has an upper bound on exploitability.

1 Introduction

More and more computer-generated strategies are deployed in the real world An *et al.* [2013]; Fang *et al.* [2017], and with the deployment to the real world, we can not avoid the interaction of AI systems with humans. This interaction might be cooperative, but in many cases, for example, in both network security and physical security, it is usually competitive. Therefore, there has been a significant amount of work towards computing strategies to use against imperfect opponents such as humans Bard *et al.* [2013]; Wu *et al.* [2021]; Southey *et al.* [2012]; Mealing and Shapiro [2015]; Korb *et al.* [2013]; Milec *et al.* [2021]; Johanson and Bowling [2009]. We will focus on extensive-form games, a powerful model able to describe recreational games, such as poker, as well as real-world situations from physical or network security. In extensive-form games, many exploitative approaches use the best response or robust response, either to respond to an explicit model of the opponent or to create a portfolio of exploitative strategies beforehand Bard *et al.* [2013]; Ganzfried and Sandholm [2011]. Best responses are also used to evaluate the robustness of strategies.

However, real-world games are often massive, and in such games, computing the best response or robust response is intractable. There are approaches to approximate best response using frequentist best response with abstraction Johanson *et al.* [2008] or by learning the response using deep reinforcement learning Heinrich and Silver [2016]; Mnih *et al.* [2013]. Nevertheless, none of those approaches have theoretical guarantees, and the responses using deep reinforcement learning can not be quickly recomputed for different opponent models. We want to guarantee that the response will achieve at least the value of the game against the strategy we are responding to. Furthermore, we want the robust response to have a guaranteed upper bound on the exploitability of the resulting strategy. This will ensure that we will always be gaining against the model, and what we lose when the model is incorrect will be bounded, unlike in the case of approximate responses. There is also previous work in the evaluation route, and here it is a fast poker-specific local best response Lisy and Bowling [2017] and computationally expensive but general approximate best response Timbers *et al.* [2020]. We want to have a response that can be quickly computed against different models, and we only allow precomputed statistics for the game domain, not for a specific opponent model.

A Breakthrough in solving massive imperfect-information extensive-form games was the depth-limited solving, which builds the game until a depth-limit and evaluates the rest using a value function[1] Moravčík *et al.* [2017]; Brown and Sandholm [2018]. We use depth-limited solving with a fixed opponent strategy to compute the responses, which already changes the computation to iterative instead of one pass in the

[1]Value function takes player reaches at the depth limit and returns the evaluation of the information sets for both players.

normal best response. However, when computing response against a model straightforwardly, we would need a specific value function learned for that model. This would strongly reduce the algorithm's applicability because learning a value function is expensive, and in many cases, we do not even know the opponent's models in advance. Consequently, we explore the effects of using only one value function across all models. The value function assumes full rationality of both players after the depth-limit. We analyze the resulting depth-limited best response and robust response and show that we have guarantees for both exploitation of the opponent model and also, for the robust response, the safety of our strategy. Conversely, we can also use the response in a scenario where we need to approximate the exploitability of a strategy. Our approach can create a diverse portfolio of responses by exchanging the value function, and it can be added to the portfolio with other approaches as local best response Lisy and Bowling [2017] and approximate best response Timbers *et al.* [2020].

Apart from the best response, we focus on the robust restricted Nash response Johanson *et al.* [2008]. Solving a game adjusted like this creates the best possible epsilon-safe responses McCracken and Bowling [2004], where epsilon varies based on the initial coin probabilities. To guarantee the robustness of the response, we need an upper bound on exploitability even in the depth-limited scheme. Resolving gadgets Burch *et al.* [2014]; Moravcik *et al.* [2016]; Brown and Sandholm [2017] are tools used to guarantee no increase in exploitability when we are resolving a subgame. However, when we have a setup where we need to measure the change in exploitability in the gadget compared to what we can gain somewhere else in the game, the existing gadgets can provide wrong values at the top. We explain and illustrate the problem and propose a novel gadget modification that solves the issue.

Our contributions are: **1)** We analyze the properties of the best response and restricted Nash response computed using the continual depth-limited scheme with Nash equilibrium value function, and we prove the exploitation properties and the safety guarantees. **2)** We formulate the algorithms to find the responses given opponent strategy and an evaluation function. **3)** We show the problem with gadget values and propose a new gadget that solves the problem. **4)** We investigate the practical performance of the continual depth-limited responses compared to best responses, restricted Nash response, and local best response Lisy and Bowling [2017].

2 Background

A two-player extensive-form game (EFG) consist of a set of players $N = \{\triangle, \triangledown, c\}$, where c denotes the chance, $\triangle$ is the maximizer and $\triangledown$ is the minimizer. A is a finite set of all actions available in the game. $H \subset \{a_1 a_2 \cdots a_n \mid a_j \in A, n \in \mathbb{N}\}$ is the set 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 $\varnothing$. 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 $H = H_\triangle \cup H_\triangledown \cup H_c \cup Z$. σ_c is the *chance strategy* defined on H_c. For each $h \in H_c, \sigma_c(h)$ is a probability distribution over $A(h)$. Utility functions assign each player utility for each leaf node, $u_i : Z \to \mathbb{R}$. The game is of *imperfect information* if some actions or chance events are not fully observed by all players. The information structure is described by *information sets* for each player i, which form 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* $\sigma = (\sigma_\triangle, \sigma_\triangledown)$ 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 $u_i(\sigma) = \sum_{z \in Z} u_i(z)\pi^\sigma(z)$. BR, NE and SSE are defined as in NFGs.

Define $u_i(\sigma, h)$ as an expected utility given that the history h is reached and all players play according to σ. A *counterfactual value* $v_i(\sigma, I)$ is the expected utility given that the information set I is reached and all players play according to strategy σ except player i, which plays to reach I. Formally, $v_i(\sigma, I) = \sum_{h \in I, z \in Z} \pi_{-i}^\sigma(h)\pi^\sigma(h, z)u_i(z)$. And similarly counterfactual value for playing action a in information set I is $v_i(\sigma, I, a) = \sum_{h \in I, z \in Z, ha \sqsubset z} \pi_{-i}^\sigma(ha)\pi^\sigma(ha, z)u_i(z)$.

Metrics for Evaluating Quality of Strategy

In a two-player zero-sum game, the *exploitability* of a strategy is defined as 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 $\mathcal{E}(\sigma_i) = u_{-i}(\sigma_i, \sigma_{-i}) - u_{-i}(\sigma^{NE}), \quad \sigma_{-i} \in BR_{-i}(\sigma_i)$.

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

Depth-limited Solving

We will denote $H_i(h)$ the sequence of player i's information states 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* $PS_i \subset H$, 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 parent nodes and public states. *Subgame* $S \subset H$ is a forest of trees closed under descendant relationship and public states.

Range of a player i is a probability distribution over his information sets in some public state PS_i, given we reached the public state PS_i. *Value function* is a function that takes the public state and both players' ranges as input and outputs counterfactual values for each information set in the public state for both players. In the paper, we assume using an *optimal value function*, which is a value function returning the same values as we would get using some Nash equilibrium 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). $u_i(\sigma)^T_V$ is utility for player i if we use strategy σ in trunk T and compute values at the dept-limit using value function V. Subgame partitioning can be created differently, and factored-observation stochastic games give it naturally and can be easily converted to EFGs. Therefore, we use EFGs but assume we can always easily create some subgame partitioning Kovařík *et al.* [2019].

Gadgets

When resolving a subgame we can add a chance node at the beginning that will aggregate the probabilities of all the players that the root states are reached. However, there is no guarantee on the resulting exploitability of the strategy in the full game and the exploitability can raise significantly Burch *et al.* [2014]. To address the issue *gadgets* are used to limit the increase in exploitability.

Resolving Gadget

The resolving gadget Burch *et al.* [2014] is the gadget that creates an initial chance node by aggregating only reaches of chance and the resolving player. The chance node can be normalized, and we would then multiply terminals by the normalization factor, which is equivalent. Instead of the opponent's strategy, it adds a node after each initial chance action, where the opponent can follow and play the game as it is normally played or terminate. In that case, it will receive the counterfactual best response value in that information set, saved from the previous step.

Max-margin Gadget

The gadget used to improve the resolved strategy is the max-margin gadget Moravcik *et al.* [2016]. It starts with a node for player $\triangledown$, which allows him to choose his beliefs freely. Then after each action from the initial node, the chance node follows, which assigns probabilities to nodes in the chosen information set using the aggregated reaches of both chance and resolving players.

3 Continual Depth-limited Responses

This section describes the theoretical concepts we are trying to compute, and we provide formal definitions. For each concept, we also explain a practical algorithm to compute the strategy defined by the concept. The pseudo-code of the algorithms is in the appendix.

Using Value Function

We want to solve massive games, and the problem with massive games is that even with the best hardware possible, they will not fit in the memory. Therefore, recently most promising option is depth limited solving, which requires a value function to evaluate public states at the edge of the trunk. To use depth-limited solving with an opponent model, we need different value functions for each model to capture the opponent's future behavior. This is infeasible for a large number of opponent models, and we instead explore the effects of using only the optimal value function.

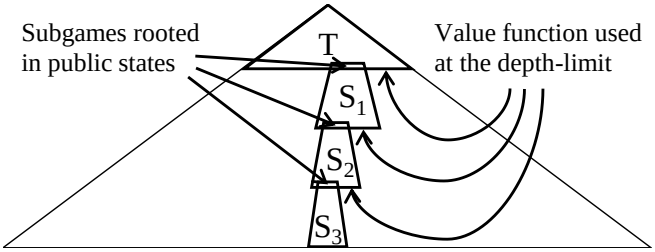

Figure 1: Illustration of the depth-limited solving.

We start from the top of the game and construct a trunk with a depth-limit. We solve the trunk, using the value function to evaluate public states at the depth-limit. Then we move forward to the next public state reached in the game, and we construct a subgame starting from the public state with some depth-limit. Based on the type of response, we solve the depth-limited subgame, and we continue with the same steps until we reach the end of the game.

Continual Depth-limited Best Response

Given any extensive-form game G with perfect recall, opponent's fixed strategy $\sigma^F_\triangledown$ 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 $\triangle$ with value function V is defined as $\sigma^{\mathcal{B}}_\triangle(\sigma^F_\triangledown)^{T_1}_V = \arg\max_{\sigma_\triangle} u_\triangle(\sigma_\triangle, \sigma^F_\triangledown)^{T_1}_V$. In other words we maximize the utility over the strategy in the trunk where after the depth-limit we return values from the value function. In each step afterwards 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 $\triangle$ in the T_{i-1} and maximize over strategy in the subgame. $\sigma^{\mathcal{B}}_\triangle(\sigma^F_\triangledown)^{T_i}_V = \arg\max_{\sigma^{S_i}_\triangle} u_\triangle(\sigma^{S_i}_\triangle \cup \sigma^{T_{i-1}}_\triangle, \sigma^F_\triangledown)^{T_i}_V$. We continue like that for each step and we always create 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 $\sigma^{\mathcal{B}}_\triangle(\sigma^F_\triangledown)^{\mathcal{P}}_V$.

Computing CDBR

We need the definition of the game and a possibility to query the opponent's strategy. We have a fixed solving depth, such that the created subgames can fit in the memory, and it can be solved in a reasonable time. First, we get the trunk from the subgame partitioning. We need to compute best response in the trunk given the value function and since value function changes when strategy in trunk changes, we can no longer compute it in one pass as a normal best response. We need to instead use some version of CFR Tammelin *et al.* [2015]; Brown and Sandholm [2019]; Farina *et al.* [2020] and con-

struct the strategy iteratively. Lemma 1 shows that it converges to the best response in the trunk even in current iterations.

After we solve the trunk, we move into the subgame, and we use value resolving without any gadgets because the opponent can not change his strategy in the trunk. It means that the initial reaches in the subgame are set using both player strategies and chance. We then solve the resulting subgame using some CFR with value function. It is necessary to use CFR instead of just computing the best response since the values returned by the value function differ based on the changing strategy in the trunk. Using the best response repeatedly would likely lead to the strategy never converging.

Continual Depth-limited Restricted Nash Response

Given any extensive-form game G with perfect recall, restricted Nash response can be computed by solving a modified game: We add initial chance node with two actions that player $\triangle$ does not observe. We copy the whole game G after both outcomes of the chance node. In one tree the opponent plays the fixed strategy and we denote it G^F. The other tree is the same as the original game and we mark it G'. We denote the full modified game G^M. Parameter p is the initial chance node probability of picking G^F. Given opponent's fixed strategy $\sigma^F_\triangledown$ and some subgame partitioning $\mathcal{P}$ of G^M we will define continual depth-limited restricted Nash response (CDRNR) recursively from the top. First we have trunk T^M_1 using $\mathcal{P}$ and value function V. Note that the number and semantic of infosets at the value function remain unchanged. CDRNR for player $\triangle$ in the trunk T^M_1 using value function V is $\sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{T^M_1}_V = \arg\max_{\sigma_\triangle} u_\triangle(\sigma_\triangle, BR(\sigma_\triangle))^{T^M_1}_V$. And then in every following step we create the new subgame S^M_i and enlarge the trunk to incorporate this subgame, creating trunk $T^M_i = T^M_{i-1} \cup S^M_i$. Next, we fix player $\triangle$'s strategy $\sigma^{T^M_{i-1}}_\triangle$ in the previous trunk T^M_{i-1} so it can not be changed anymore and the CDRNR is $\sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{T_i}_V = \arg\max_{\sigma^{S^M_i}_\triangle} u_\triangle(\sigma'_\triangle, BR(\sigma'_\triangle))^{T^M_i}_V$ where $\sigma'_\triangle$ is a combination of the strategy we optimize over and the fixed strategy from previous step, formally $\sigma'_\triangle = \sigma^{S^M_i}_\triangle \cup \sigma^{T^M_{i-1}}_\triangle$.

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 $\triangle$, and we give the opponent a chance to best respond in the whole T'_i. We denote the full CDRNR strategy $\sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{\mathcal{P}}_V$.

Computing CDRNR

CDRNR also needs a definition of the game, opponent strategy, value function with the same inputs and outputs as CDBR, and a modified game. We can create a modified game G^M explicitly, or we can use the original game tree store everything in it because the trees are identical. The trunk is solved using some CFR with the value function, where both players act, but the opponent changes his strategy only in G'. We play following the strategy in the trunk, and when we need to play outside of the trunk, we need to create a new subgame.

We are using continual resolving, and robust CDBR needs a method that limits the exploitability increase. In the definition, it is done by the opponent being able to best respond in the whole T' trunk. However, in the practical setting, we start the resolving at the leaf public state, the trunk is discarded, and we need to use the resolving gadget, to measure the exploitability in the subgame correctly as the best response did. We construct the subgame with the gadget at the start of the subgame in G', and the subgame in G^F is still constructed the same way as in CDBR. Since CDRNR dynamically compares possible exploitability by a perfect opponent in G' with possible gain against the model in G^F through the initial chance node, we need the gadget to return an accurate increase of exploitability. Although commonly used gadgets are good at resolving the correct strategy, they are unable to accurately measure the increase in exploitability when the strategy changes. Therefore, we use a new gadget we call a sequential gadget that returns the values accurately, and we discuss the gadgets in detail in Section 4.

Then we resolve the subgame using some CFR with the value function, and we play according to the resolved strategy while we are in the subgame. When we leave the subgame, we construct a new subgame, and we continue until the game ends. In the G_F we continue as in CDBR, while in G' we have replaced BR with a gadget. BR is in the concept to measure the exploitability of the strategy and, using the new gadget that can also measure the exploitability, the solution will be the desired concept.

4 Using Gadgets

We need to use a gadget in our CDRNR to ensure robustness, since the RNR algorithm implicitly compares the subgame value of a part with the gadget on G' and the subgame value of G^F. We face a problem that previous applications of the gadgets did not need correct subgame value, and the gadgets are constructed to not increase exploitability, but in the process, they distort the subgame value. This section shows the distortion problem with commonly used resolving gadgets, and we fix the problem using a new sequential gadget.

Restricted Nash Response with Gadget

When using the algorithm to compute the CDRNR described in Section 3, we are using a gadget to ensure the exploitability of the resulting strategy is bounded, and the opponent does not deviate. However, fully rational player $\triangle$ will likely change strategy to exploit mistakes in G^F. In that case, we need the gadget to also measure the change in subgame value because CDRNR relies on the implicit comparison of subgame values in G' and G^F, which corresponds to a comparison of gain and exploitability.

In other words, when we do the trade-off and switch strategy in the gadget for some exploitable strategy to exploit the opponent model, the expected utility of the gadget for the opponent will increase. We need this increase to be equal to the exploitability increase in the original subgame. Next, we show that commonly used resolving gadgets are either overestimating or underestimating this value on an example game in Figure 2. Note that player $\triangledown$ can play anything in those games because they would correspond to G' part of the

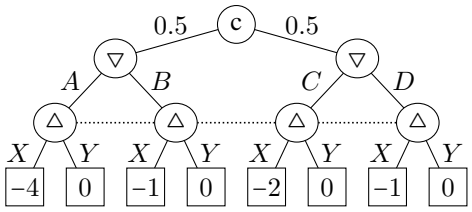

Figure 2: Example game to show problems with gadgets.

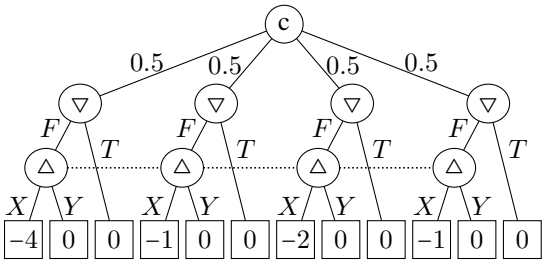

Figure 3: Resolving gadget for game in Figure 2.

CDRNR where $\triangledown$ can play. In equilibrium, player $\triangle$ plays action Y, and player $\triangledown$ can play anything. This gives the value of the game 0, and counterfactual values in all inner nodes are also 0. If the opponent model in the other part of the tree makes it worth for player $\triangle$ to play action X, player $\triangledown$ will play (A, C) with utility 3. We will use gadgets to resolve the game beginning in the player $\triangle$ information set.

Resolving Gadget

Resolving gadget on the game in our example has all utilities after *terminate* actions 0. When we resolve the gadget, the utility is 0. However, when player $\triangle$ deviates to action X, player $\triangledown$ plays *follow* action in every node, and his utility will be 4. Therefore, common resolving gadget may overestimate the real exploitability of the strategy in the subgame. It might seem that normalization of the chance node could solve the problem, but it would only halve the value to 2, which is still incorrect.

Max-margin Gadget

All the counterfactual best response values are 0 and we do not need to offset any node in the max-margin gadget. We simply add the initial decision node and the chance nodes (since there is only one state in each infoset the nodes have only one action). When we solve the gadget, player $\triangle$ will pick action Y, and the gadget value will be 0. However, when player $\triangle$ deviates to action X, player $\triangledown$ now has a choice between terminal utilities and simply picks action E to receive the highest one. This will result in utility 2, and we see that max-margin gadgets can underestimate the real exploitability. Similar to the previous gadget, the normalization of the chance nodes would lead to double the utility, which is still incorrect.

Sequential Gadget

We analyzed the problem, and the cause is the propagation through the player $\triangledown$ actions to the root. In a normal game, it is propagated through player $\triangledown$'s strategy, which is constrained to follow the sequence form. However, in shown gadgets, those constraints change, and in resolving gadget player $\triangle$ can reach all the nodes with probability 1, hence the overestimation. In the max-margin gadget, the player $\triangledown$'s strategy is constrained to sum to one over all the nodes. However, in the real game, it can sum to 2 because we have two decision sets, hence the underestimation.

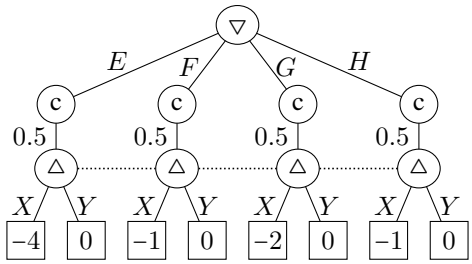

Figure 4: Max-margin gadget for game in Figure 2

To solve the issue, we will create a gadget that incorporates the correct constraints in itself. We will use the resolving gadget as the basis. Instead of using the initial chance node, we will build a tree above the gadget to encode the sequence form constraints of the player $\triangledown$ to ensure the correct estimate. This construction will enlarge the gadget, but since we are building a tree above the initial nodes, we will add at most $n - 1$ additional nodes, where n is the number of follow/terminate nodes. The gadget correctly measures exploitability in the subgame because if terminate action is played, we know exploitability in that part did not increase, and when follow is played (in the CDRNR case, the follow will be often played everywhere), the sequence structure when solved is exactly the best response. It is also possible to use the max-margin gadget as the basis and apply the sequence form constraints on it.

When we solve the sequential gadget subgame, player $\triangle$ will pick action Y, and the gadget value will be 0. However, when player $\triangle$ plays action X, player $\triangledown$ will follow everywhere, but because of the sequence form constraints above the follow/terminate nodes, she will only pick A' and C', which results in correct utility 3. We picked this simple game to showcase the problems of the other gadgets nicely, and unfortunately, the sequential gadget actually rebuilds the whole game. This is caused by the example game having only one public state, and in real games with multiple public states, this will not happen.

When using the gadget in full CDRNR, different gadgets create different strategies for one value of p. We also provide an example game in the appendix, where player $\triangle$ resolves the correct RNR strategy using a sequential gadget with a particular value of p. Still, for other gadgets, there exists no p such that they would resolve the same strategy.

5 Theory

This section provides the theoretical guarantees of performance for both CDBR and CDRNR, and provides an example why the guarantee can not be even better. Furthermore, we provide an upper bound on the exploitability of CDRNR, linking it to the gain by the parameter p. Formal proofs of all the theorems are in the appendix.

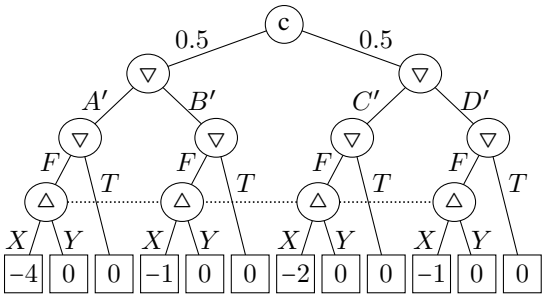

Figure 5: Sequential gadget for game in Figure 2.

Response Performance Against Model

The first two theorems show that CDBR and CDRNR will achieve at least the value of the game against the model. However, as we show in an example in appendix, CDBR can perform worse than a Nash equilibrium against the fixed opponent because of the perfect opponent assumption at the depth-limit.

Lemma 1. *Let G be zero-sum imperfect-information extensive-form game with perfect recall. Let $\sigma^F_\triangledown$ be fixed opponent's strategy, let T be some trunk of the game. If we perform CFR iterations in the trunk for player $\triangle$ then for the best iterate*

$$\max_{\sigma^*_\triangle \in \Sigma_\triangle} u_\triangle(\sigma^*_\triangle, \sigma^F_\triangledown)^T_V - u_\triangle(\hat{\sigma}_\triangle, \sigma^F_\triangledown)^T_V \leq \Delta\sqrt{\frac{A}{T}}|\mathcal{I}_{TR}|$$

where Δ is variance in leaf utility, A is an upper bound on the number of actions and $|\mathcal{I}_{TR}|$ is number of information sets in the trunk.

Previous lemma states that the algorithm converges to the best response in the trunk and not only using average strategy but also using current strategy.

Theorem 2. *Let G be zero-sum extensive-form game with perfect recall. Let $\sigma^F_\triangledown$ be fixed opponent's strategy, let $\mathcal{P}$ be any subgame partitioning of the game G and let $\sigma^{\mathcal{B}}_\triangle$ be a CDBR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma^F_\triangledown$, formally $\sigma^{\mathcal{B}}_\triangle = \sigma^{\mathcal{B}}_\triangle(\sigma^F_\triangledown)^{\mathcal{P}}_V$. Let σ^{NE} be any Nash equilibrium. Then $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^F_\triangledown) \geq u_\triangle(\sigma^{NE})$.*

Theorem 3. *Let G be any zero-sum extensive-form game with perfect recall and let $\sigma^F_\triangledown$ be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using $\sigma^F_\triangledown$. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma^{\mathcal{R}}_\triangle$ be a CDRNR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma^F_\triangledown$, formally $\sigma^{\mathcal{R}}_\triangle = \sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{\mathcal{P}}_V$. Let σ^{NE} be any Nash equilibrium in G. Then $u_\triangle(\sigma^{\mathcal{R}}_\triangle, \sigma^F_\triangledown) \geq u_\triangle(\sigma^{NE})$.*

Previous theorems state that our approaches have the first property that we wanted and when we respond to the model we will receive at least the value of the game. It is not as strong as being better than any Nash equilibrium but as we show in the Observation 6 we do not have that property. It is caused by the algorithm trying to exploit the opponent in the current step while thinking the opponent will be rational in the rest of the game. However, as we show in the experiments in practice the algorithms work very well.

Exploitability Guarantee of Continual Depth Limited Restricted Nash Response

Theorem 4. *Let G be any zero-sum extensive-form game with perfect recall and let $\sigma^F_\triangledown$ be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using $\sigma^F_\triangledown$. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma^{\mathcal{R}}_\triangle$ be a CDRNR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma^F_\triangledown$, formally $\sigma^{\mathcal{R}}_\triangle = \sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{\mathcal{P}}_V$. Then exploitability has a bound $\mathcal{E}(\sigma^{\mathcal{R}}_\triangle) \leq \mathcal{G}(\sigma^{\mathcal{R}}_\triangle, \sigma^F_\triangledown)\frac{p}{1-p}$, where $\mathcal{E}$ and $\mathcal{G}$ are exploitability and gain defined in Section 2.*

Last theorem is more complex and it bounds the exploitability by the gain of the strategy against the model. This is because of the internal working of RNR which guarantees to find an epsilon safe best response, but the epsilon is not known beforehand and it depends both on the parameter p and on the opponent and that is why we have to reflect both in the theorem, one explicitly and second hidden in the gain. Dependence on opponent can cause problems when in the game is some extremely bad action with substantially higher payoff then other actions and the opponent actually plays that action. Then even with small p the algorithm can be very exploitable. This is extreme case that will not happen in most games and we just point to it if anyone would use the algorithm in a game with very disproportionate utilities.

6 Experiments

We show a comparison between continual depth-limited best response (CDBR) and local best response (LBR) Lisy and Bowling [2017]. We also show empirically that the continual depth-limited restricted Nash response (CDRNR) exploitability is bounded, and we explore the tradeoff between exploitability and gain provided by CDRNR. Hardware setup, domain description, and algorithm details will be in the appendix. We use two types of opponent strategies, strategies generated by a low amount of CFR iterations and random strategies with different seeds.

Local Best Response vs. CDRNR

We compare LBR and CDBR in Leduc Holde'm. We only use the poker domain because even though CDBR is general, LBR is poker-specific. We show that with smaller steps CDBR and LBR are very similar, and as we increase the step size of CDBR, it starts outperforming LBR. The behavior differs on every strategy because LBR uses only call as the value function while CDBR uses the perfectly rational extension. Furthermore, it is possible to exchange the value function of CDBR, and both concepts would be almost the same. However, we would lose the guarantee that CDBR will never perform worse than the value of the game.

When we look at the results in Figure 6, we can see that both concepts are good at approximating the best response, with CDBR being better against both types of strategies. LBR looks at one next action, so in terms of comparability, it is best compared to the CDBR1. Moreover, we can use CDBR to complement LBR when trying to find exploitability lower

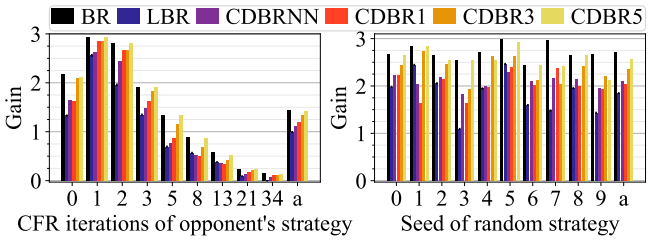

Figure 6: Gain comparison of best response (BR), local best response (LBR), and continual depth-limited best response (CDBR) against strategies from CFR using a small number of iterations (left) and against random strategies (right). The **a** stands for the average of the other values in the plot. The number after CDBR stands for the amount of action CDBR was allowed to look in the future, and CDBRNN is one step CDBR with a neural network as a value function.

bounds of algorithms playing in large games where BR is impossible. We can use CDBR with different value functions to build a portfolio that would make the approximation better and possible on any game where continual resolving is usable. Next, we observe a lack of monotonicity in step increase, which is linked to the counterexample in Section 5, because when we increase the depth-limit, the algorithm can exploit some early mistake which causes it to miss a sub-tree where the opponent makes a much bigger mistake in the future. Finally, we can see the difference between the algorithm with guarantees and LBR without them nicely. Against strategy from 34 CFR iterations, LBR can no longer achieve positive gain, and it only gets worse with larger iterations, while CDBR can always achieve at least zero gain (assuming we have a perfect value function).

Exploitability of Robust Responses

We report both gain and exploitability for CDRNR on the imperfect information version of Goofspiel with 5 cards. Results in Figure 7 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 significantly exploit the opponent without significantly raising its exploitability.

In most cases, both 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 $p = 0.9$ against strategy generated by 5 iterations of CFR. The depth-limited scheme causes that, and because the algorithm, unlike RNR, has not full information when making individual decisions, it can choose different paths to exploit the opponent, raising its exploitability more than it has to. However, the bound will always hold. We provide results on different domains and for more values of parameter p in the appendix.

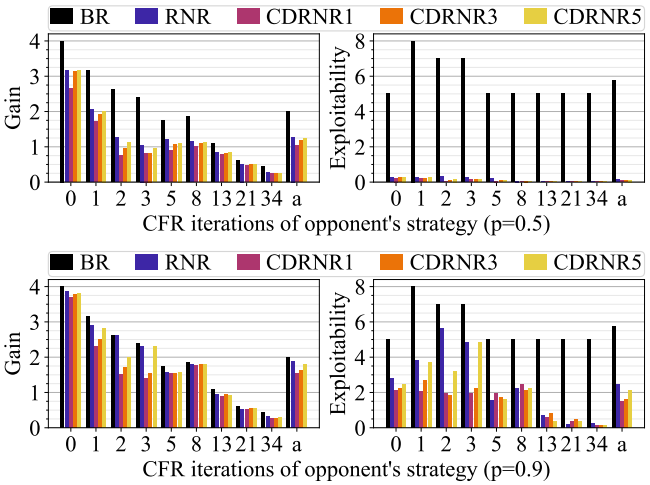

Figure 7: Gain and exploitability comparison of BR, RNR and CDRNR against strategies from CFR using small number of iterations with different p values. The **a** stands for the average of the other values. Number after CDRNR stands for the depth-limit.

7 Conclusion

Opponent exploitation is an essential topic in computational game theory. A significant amount of approaches exist to exploit opponents, but scalability of the approaches to massive games is a problem. In widely used zero-sum two-player sequential imperfect information games, we focus on two previously unsolved problems in games that can no be fit into the computer memory. How to approximate the best response fast with changing opponent model and how to compute robust response. Our approach is able to do both if we first train a value function for the game domain we will be working with, and on top of that, we can use it to approximate the lower bound on exploitability for other approaches similarly to approximate the best response. We use depth-limited solving with optimal value function, analyze the resulting concepts, and prove guarantees on gain and safety guarantee on exploitability for the CDRNR. We formulate algorithms that can be practically used to compute the concepts online. Furthermore, we show that common resolving gadgets can not correctly compute subgame values, and we propose a new gadget, which solves the issue. Finally, we investigate practical performance compared to the best response and local best response, and we show that CDRNR achieves high exploitation of its opponent with much lower exploitability than suggested by the theoretical bound.

Acknowledgements

This research is supported by Czech Science Foundation (grant no. 18-27483Y). 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.

References

Bo An, Fernando Ordóñez, Milind Tambe, Eric Shieh, Rong Yang, Craig Baldwin, Joseph DiRenzo III, Kathryn Moretti, Ben Maule, and Garrett Meyer. A deployed quantal response-based patrol planning system for the us coast guard. *Interfaces*, 43(5):400–420, 2013.

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

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

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

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1829–1836, 2019.

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

Fei Fang, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Brian C Schwedock, Milind Tambe, and Andrew Lemieux. Paws-a deployed game-theoretic application to combat poaching. *AI Magazine*, 38(1):23–36, 2017.

Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Faster game solving via predictive blackwell approachability: Connecting regret matching and mirror descent. *arXiv preprint arXiv:2007.14358*, 2020.

Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 533–540. Citeseer, 2011.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Michael Johanson and Michael Bowling. Data biased robust counter strategies. In *Artificial Intelligence and Statistics*, pages 264–271, 2009.

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

Kevin B Korb, Ann Nicholson, and Nathalie Jitnah. Bayesian poker. *arXiv preprint arXiv:1301.6711*, 2013.

Vojtěch Kovařík, Dominik Seitz, Viliam Lisý, Jan Rudolf, Shuo Sun, and Karel Ha. Value functions for depth-limited solving in imperfect-information games. *arXiv preprint arXiv:1906.06412*, 2019.

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

Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr TImbers, and Karl Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 464–470, 7 2019.

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

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

David Milec, Jakub Černý, Viliam Lisý, and Bo An. Complexity and algorithms for exploiting quantal opponents in large two-player games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5575–5583, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

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

Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

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

Zhe Wu, Kai Li, Enmin Zhao, Hang Xu, Meng Zhang, Haobo Fu, Bo An, and Junliang Xing. L2e: Learning to exploit your opponent. *arXiv preprint arXiv:2102.09381*, 2021.

A Algorithm Pseudo-code

Algorithm 1 shows the pseudo-code of the depth-limited response algorithms.

Algorithm 1: Computing CDBR (CDRNR)

Require: game G, opponent strategy $\sigma^F_\triangledown$, value function V
  create (virtually) modified game G^M (only CDRNR)
  create subgame partitioning $\mathcal{P}$ from G (G^M)
  $\sigma^{\mathcal{B}}_\triangle$ = empty strategy ready to be filled
  I = initial information set in which we act
  S = current constructed subgame
  while I not null **do**
    if I not in S **then**
      S = construct new S from $\mathcal{P}$ using previous S
      $\sigma^{\mathcal{B}}_\triangle$ += solution of S using CFR+ with V
    else
      pick action A according to $\sigma^{\mathcal{B}}_\triangle$ in I
      get new I using A (or null if the game ends)
    end if
  end while

B Counterexample Gadget Game

In this section, we describe the more profound problem with the gadget. Both resolving and max-margin gadgets might be unable to find the correct strategy in the RNR depth limited scenario. We constructed an example game in Figure 8 such that action b is always 0 for player $\triangle$ and other actions are dominated by b unless the opponent makes a mistake. We construct the RNR modified game with the opponent's strategy playing purely actions (W, Z, M, O, Q, S) action b still dominates for p slightly smaller than 0.5. When we have $p = 0.5$ the best action is c and when we slightly increase p action a dominates all the way up to $p = 1$.

With the gadgets, we take such depth-limit that the game is first solved without player $\triangle$ playing, and then the next subgame starts in his information set. The previous game is solved using the perfect value function, so counterfactual values everywhere are 0, and the resolving and sequence form gadget will have 0 at the terminate actions, and the max-margin gadget will not offset anything. Then when we solve the subgame using the sequence form gadget, it is the same as the original RNR. for $p = 0.5$ we resolve c, and for smaller p we have b, and for higher p we have a. First, we thought that even though other gadgets overestimate/underestimate the values, we can re-scale p and resolve the same strategies. However, as we show in Figures 10 and 11, there is no p such that resolving or max-margin gadget resolve the subgame to create strategy with pure c.

C Experiment Details

Experimental Setup

For all experiments, we use Python 3.8 and C++. We solved linear programs using gurobi 9.0.3, and experiments were done on Intel i7 1.8GHz CPU with 8GB RAM. We used Leduc Holde'm for CDBR experiments with the torch library

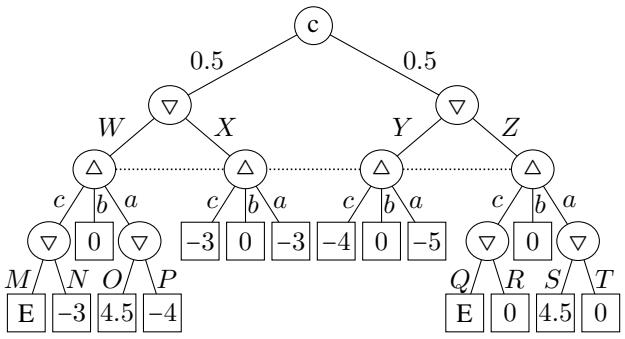

Figure 8: Example game where gadgets can not resolve the strategy in CDRNR. $E = 3.500001$

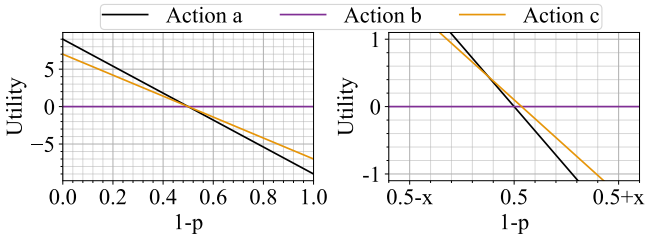

Figure 9: Expected utility of different actions from Figure 8 using sequence form gadget, where $x = 0.000001$ and utility on the right is times 10^{-5}.

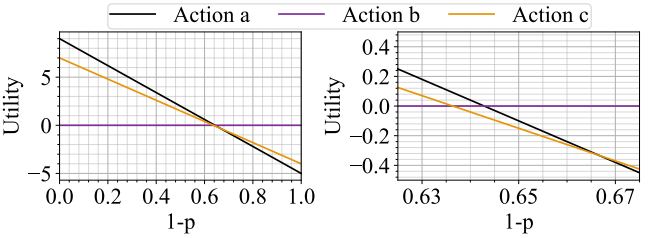

Figure 10: Expected utility of different actions from Figure 8 using max-margin gadget.

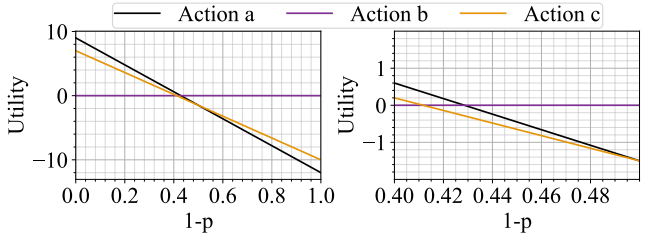

Figure 11: Expected utility of different actions from Figure 8 using resolving gadget.

for the neural network experiment. For CDRNR experiments, we used an imperfect information version of Goofspiel 5. For most of the experiments, we wanted to solve the concepts perfectly with perfect value function, so we used LP and fixed the parts of the game that needed to be fixed. For the neural network experiment with CDBR, we used CFR+ to solve the subgame and the neural network as a value function.

Domain Definition

Leduc Hold'em is a medium-sized poker game. Both players give one chip at the beginning of the match and receive one card from a deck with six cards of 2 suits and three ranks. Then players switch and can call or bet. After a bet, the opponent can also fold, which finishes the game, and he forfeits all the staked money. After both players call or after at most two bets public card is revealed, and another betting round begins. In the first round, the bet size is two, and in the second, it is 4. If the game ends without anyone, folding cards are compared, and the player with pair always wins, and if there is no pair, the player with the higher card wins. If both have the same card, the money is split. ***Goofspiel*** is a bidding card game where players are trying to obtain the most points. Cards are shuffled and set face-down. Each turn, the top point card is revealed, and players simultaneously play a bid card; the point card is given to the highest bidder or discarded if the bids are equal. In this implementation, we use a fixed deck with K = 5.

Liar's Dice is a game where players have some number of dice and secretly roll. Then first player bids amount of numbers rolled and the other player can bid more or disbelieve the first player. When bidding ends by disbelief action both players show dice and if the bid is right the caller loses one die and when the bidder is wrong the bidder loses one die. Then the game continues but for out computation we use game which ends by the loss of a die and we use one die with four sides for each player.

D Proofs

Lemma 2. *Let G be zero-sum imperfect-information extensive-form game with perfect recall. Let $\sigma^F_\triangledown$ be fixed opponent's strategy, let T be some trunk of the game. If we perform CFR iterations in the trunk for player $\triangle$ then for the best iterate*

$$\max_{\sigma^*_\triangle \epsilon \Sigma_\triangle} u_\triangle(\sigma^*_\triangle, \sigma^F_\triangledown)^T_V - u_\triangle(\hat{\sigma}_\triangle, \sigma^F_\triangledown)^T_V \leq \Delta\sqrt{\frac{A}{T}}|\mathcal{I}_{TR}|$$

where Δ is variance in leaf utility, A is an upper bound on the number of actions and $|\mathcal{I}_{TR}|$ is number of information sets in the trunk.

Proof. Using Theorem 2 from Burch *et al.* [2014] we know that regret for player $\triangle$ is bounded $R^T_\triangle = \frac{1}{T}\max_{\sigma^*_\triangle \epsilon \Sigma_\triangle} \sum^T_{t=1}(u_\triangle(\sigma^*_\triangle, \sigma^F_\triangledown)^T_V - u_\triangle(\sigma^t_\triangle, \sigma^F_\triangledown))^T_V \leq \Delta\sqrt{AT}|\mathcal{I}_{TR}|$. Then we can directly map the regret to regret using time-independent loss function $l(\sigma_\triangle) = -u_\triangle(\sigma_\triangle, \sigma^F_\triangledown)^T_V$. We can then use Lemma 2 from Lockhart *et al.* [2019] and we get $l(\hat{\sigma}_\triangle) - \min_{\sigma^*_\triangle \epsilon \Sigma_\triangle} l(\sigma^*_\triangle) \leq \frac{R^T_\triangle}{T}$. Substituting l and $R^T_\triangle$ back we get

$$\max_{\sigma^*_\triangle \epsilon \Sigma_\triangle} u_\triangle(\sigma^*_\triangle, \sigma^F_\triangledown)^T_V - u_\triangle(\hat{\sigma}_\triangle, \sigma^F_\triangledown)^T_V \leq \Delta\sqrt{\frac{A}{T}}|\mathcal{I}_{TR}|$$

□

Theorem 3. *Let G be zero-sum extensive-form game with perfect recall. Let $\sigma^F_\triangledown$ be fixed opponent's strategy, let $\mathcal{P}$ be any subgame partitioning of the game G and let $\sigma^{\mathcal{B}}_\triangle$ be a CDBR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma^F_\triangledown$, formally $\sigma^{\mathcal{B}}_\triangle = \sigma^{\mathcal{B}}_\triangle(\sigma^F_\triangledown)^{\mathcal{P}}_V$. Let σ^{NE} be any Nash equilibrium. Then $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^F_\triangledown) \geq u_\triangle(\sigma^{NE})$.*

Proof. Using subgame partitioning P, let T_1 be the trunk of the game. From the properties of a Nash equilibrium $u_\triangle(\sigma^{NE}) \leq u_\triangle(\sigma^{NE}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_1}_V$. By definition of CDBR we can write $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_1}_V$ as $\max_{\sigma_\triangle} u_\triangle(\sigma_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_1}_V$ and $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_1}_V \geq u_\triangle(\sigma^{NE}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_1}_V$. We continue using induction over steps with induction assumption that in step i, $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_i}_V \geq u_\triangle(\sigma^{NE})$. We already know it holds for T_1. Now we assume we have trunk T_{i-1} for which the induction step holds and trunk T_i which is T_{i-1} joined with new subgame S_i. From the definition of CDBR we know that in each step the strategy is computed as $\sigma^{\mathcal{B}}_\triangle(\sigma^F_2)^{T_i}_V = \arg\max_{\sigma^{S_i}_\triangle} u_\triangle(\sigma^{S_i}_\triangle \cup \sigma^{T_{i-1}}_\triangle, \sigma^F_\triangledown)^{T_i}_V$ which means that $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^F_\triangledown)^{T_i}_V \geq u_\triangle(\sigma^{NE_{S_i}}_\triangle \cup \sigma^{\mathcal{B}_{T_{i-1}}}_\triangle, \sigma^F_\triangledown)^{T_i}_V \geq u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_{i-1}}_V$ last step is because we replace subgame part of opponent's strategy by equilibrium and then we can replace equilibrium in the subgame by V. Therefore, $u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_{i-1}}_V \leq u_\triangle(\sigma^{\mathcal{B}}_\triangle, \sigma^{\mathrm{F}}_\triangledown)^{T_i}_V$ which concludes the induction step and hence the whole proof. □

Theorem 4. *Let G be any zero-sum extensive-form game with perfect recall and let $\sigma^F_\triangledown$ be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using $\sigma^F_\triangledown$. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma^{\mathcal{R}}_\triangle$ be a CDRNR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma^F_\triangledown$, formally $\sigma^{\mathcal{R}}_\triangle = \sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{\mathcal{P}}_V$. Let σ^{NE} be any Nash equilibrium in G. Then $u_\triangle(\sigma^{\mathcal{R}}_\triangle, \sigma^F_\triangledown) \geq u_\triangle(\sigma^{NE})$.*

Proof. Let T_1 be a trunk of a modified game G^M using partitioning $\mathcal{P}$. We will use $u^G(\sigma)$ as utility in G. Utility of player $\triangle$ for playing Nash equilibrium of the G in trunk T_1 will be higher or the same as game value of G, formally $u^G_\triangle(\sigma^{NE}) \leq u^{G^M}_\triangle(\sigma^{NE})^{T_1}_V$. We continue using induction over steps with induction assumption that in step i, $u^{G^M}_\triangle(\sigma^{\mathcal{R}}_\triangle, BR(\sigma^{\mathcal{R}}_\triangle))^{T_i}_V \geq u^G_\triangle(\sigma^{NE})$. By definition of CDRNR we know it holds for T_1. Now we assume we have trunk T_{i-1} for which the induction step holds and trunk T_i which is T_{i-1} joined with new subgame S_i. From the definition of CDRNR we know that in each step the strategy is computed as $\sigma^{\mathcal{R}}_\triangle(\sigma^F_\triangledown, p)^{T_i}_V =$

$\arg\max_{\sigma_\triangle^{S_i}} u_\triangle^{G^M}(\sigma'_\triangle, BR(\sigma'_\triangle))_V^{T_i}$, where $\sigma'_\triangle$ is a combination of the strategy we optimize over and the fixed strategy from previous step, formally $\sigma'_\triangle = \sigma_\triangle^{S_i} \cup \sigma_\triangle^{T_{i-1}}$. Since we are maximizing against best response we will find a Nash equilibrium of the subgame we are optimizing in, therefore the utility will be higher than $u_\triangle^{G^M}(\sigma_\triangle^{\mathcal{R}}, BR(\sigma_\triangle^{\mathcal{R}}))_V^{T_{i-1}}$ and using the induction assumption we have that $u_\triangle^{G^M}(\sigma_\triangle^{\mathcal{R}}, BR(\sigma_\triangle^{\mathcal{R}}))_V^{T_i} \geq u_\triangle^G(\sigma^{NE})$. Using this with terminal subgames we know that $u_\triangle^{G^M}(\sigma_\triangle^{\mathcal{R}}, BR(\sigma_\triangle^{\mathcal{R}})) \geq u_\triangle^G(\sigma^{NE})$. However, we still need to show it works for $u_\triangle^G(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)$. We can do it by replacing strategy of player $\triangledown$ in the G' by $\sigma_\triangledown^F$ which will effectively transform G^M game back to G with player $\triangledown$ playing $\sigma_\triangledown^F$. Since we did this transformation by changing the strategy that was a best response the utility can only increase and $u_\triangle^G(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F) \geq u_\triangle^{G^M}(\sigma_\triangle^{\mathcal{R}}, BR(\sigma_\triangle^{\mathcal{R}})) \geq u_\triangle^G(\sigma^{NE})$. □

Theorem 5. *Let G be any zero-sum extensive-form game with perfect recall and let $\sigma_\triangledown^F$ be any fixed opponent's strategy in G. Then we set G^M as restricted Nash response modification of G using $\sigma_\triangledown^F$. Let $\mathcal{P}$ be any subgame partitioning of the game G^M and using some $p \in \langle 0, 1\rangle$, let $\sigma_\triangle^{\mathcal{R}}$ be a CDRNR given optimal value function V, partitioning $\mathcal{P}$ and opponent strategy $\sigma_\triangledown^F$, formally $\sigma_\triangle^{\mathcal{R}} = \sigma_\triangle^{\mathcal{R}}(\sigma_\triangledown^F, p)_V^{\mathcal{P}}$. Then exploitability has a bound $\mathcal{E}(\sigma_\triangle^{\mathcal{R}}) \leq \mathcal{G}(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)\frac{p}{1-p}$, where $\mathcal{E}$ and $\mathcal{G}$ are exploitability and gain defined in Section 2.*

Proof. We will examine the exploitability increase in each step. First, we define gain in a single step as $\mathcal{G}(\sigma_\triangle, \sigma_\triangledown)_V^{T_i} = u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_i} - u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_{i-1}}$ for $i > 0$ and $\mathcal{G}(\sigma_\triangle, \sigma_\triangledown)_V^{T_0} = u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_0} - u_\triangle(\sigma^{NE})$. This is consistent with full definition of gain because sum of gains over all steps will results in $u_\triangle(\sigma_\triangle, \sigma_\triangledown)^G - u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_n} + u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_n} - \ldots - u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_0} + u_\triangle(\sigma_\triangle, \sigma_\triangledown)_V^{T_0} - u_\triangle(\sigma^{NE}) = u_\triangle(\sigma_\triangle, \sigma_\triangledown)^G - u_\triangle(\sigma^{NE}) = \mathcal{G}(\sigma_\triangle, \sigma_\triangledown)$. We define exploitability in a single step similarly as $\mathcal{E}(\sigma_\triangle)_V^{T_i} = u_\triangledown(\sigma_\triangle, BR(\sigma_\triangle))_V^{T_i} - u_\triangledown(\sigma_\triangle, BR(\sigma_\triangle))_V^{T_{i-1}}$ for $i > 0$ and $\mathcal{E}(\sigma_\triangle)_V^{T_0} = u_\triangledown(\sigma_\triangle, BR(\sigma_\triangle))_V^{T_0} - u_\triangledown(\sigma^{NE})$ and it also sums to full exploitability. By definition of CDRNR at step i the strategy is computed as $\sigma_\triangle^{\mathcal{R}}(\sigma_\triangledown^F, p)_V^{T_i} = \arg\max_{\sigma_\triangle^{S_i}} u_\triangle^{G^M}(\sigma'_\triangle, BR(\sigma'_\triangle))_V^{T_i}$, where $\sigma'_\triangle$ is a combination of the strategy we optimize over and the fixed strategy from previous step, formally $\sigma'_\triangle = \sigma_\triangle^{S_i} \cup \sigma_\triangle^{T_{i-1}}$. If we use Nash equilibrium strategy in the step i exploitability will be 0 and gain will be non-negative. The full utility of the step can be written as $\mathcal{G}(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)_V^{T_i} p - \mathcal{E}(\sigma_\triangle^{\mathcal{R}})_V^{T_i}(1-p)$ and for Nash equilibrium strategy it is bigger than 0. Since we are maximizing over the strategy it will be at least as good as Nash strategy ans we can write $\mathcal{G}(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)_V^{T_i} p - \mathcal{E}(\sigma_\triangle^{\mathcal{R}})_V^{T_i}(1-p) \geq 0$ reorganizing the equation gives us $\mathcal{G}(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)_V^{T_i}\frac{p}{1-p} \geq \mathcal{E}(\sigma_\triangle^{\mathcal{R}})_V^{T_i}$. Summing over all the steps gives us $\mathcal{E}(\sigma_\triangle^{\mathcal{R}}) \leq \mathcal{G}(\sigma_\triangle^{\mathcal{R}}, \sigma_\triangledown^F)\frac{p}{1-p}$. □

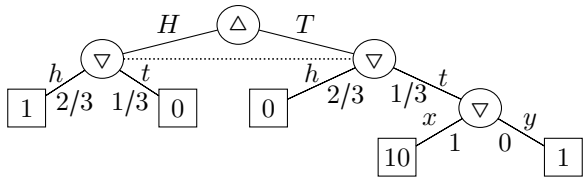

Figure 12: Example of game where step best response is worse than NE against fixed strategy $\sigma(h) = \frac{2}{3}, \sigma(x) = 1$.

E CDBR Against Nash Strategy

Observation 6. *An example in Figure 12 shows that CDBR can perform worse than a Nash equilibrium against the fixed opponent because of the perfect opponent assumption after the depth-limit. An example is a game of matching pennies with a twist. Player $\triangledown$ can choose in the case of the tails whether he wants to give the opponent 10 instead of only 1. A rational player will never do it, and the equilibrium is a uniform strategy as in normal matching pennies.*

Now we have an opponent model that plays h with probability $\frac{2}{3}$ and always plays x. The best response to the model will always play T and get payoff $\frac{10}{3}$. Nash equilibrium strategy will get payoff 2, and CDBR with depth-limit 2 will cut the game before the x/y choice. Assuming the opponent plays perfectly after the depth-limit and chooses y, $\triangle$ will always play H. Playing H will result in receiving payoff $\frac{2}{3}$, which is higher than the value of the game ($\frac{1}{2}$) but lower than what Nash equilibrium can get against the model.

F Aditional CDBR Results

We show CDBR results on more domains that we were not able to fit in the main paper. Interestingly on small liar's dice the CDBR is almost perfect even with only depth one. On Goofspiel the result resembles more the Leduc poker result. We use random strategies and strategies generated by CFR.

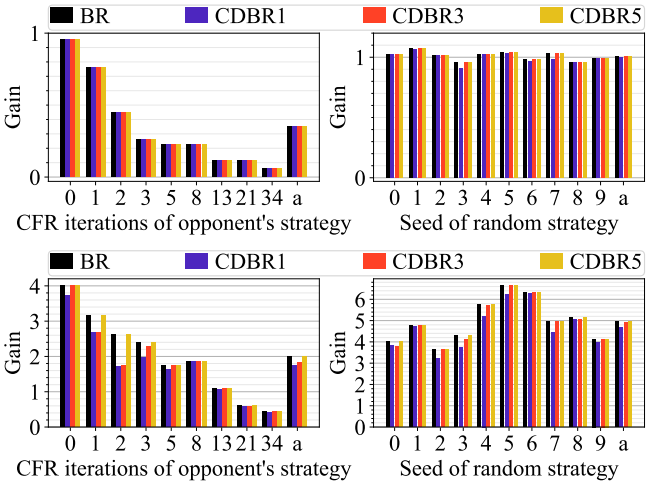

Figure 13: More CDBR results on Liar's dice (top) and imperfect-information Goofspiel (bottom).

G Aditional CDRNR Results

We show more results for Goofspiel with different values of p and also Leduc Hold'em and Liar's dice.

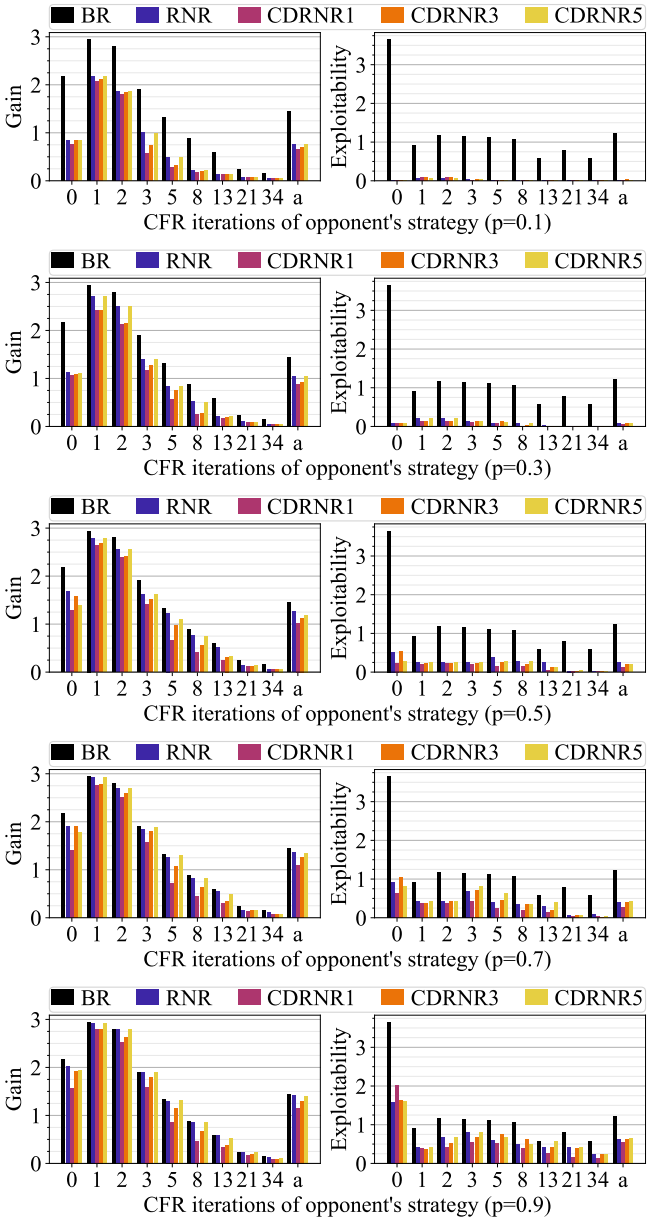

Figure 14: Aditional results for CDRNR with different values of p. Generated on Leduc Hold'em.

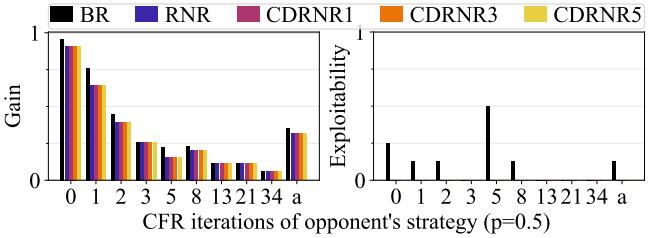

Figure 15: Aditional result generated on Liar's dice. For every p it exactly mimics the RNR so we only show one value.

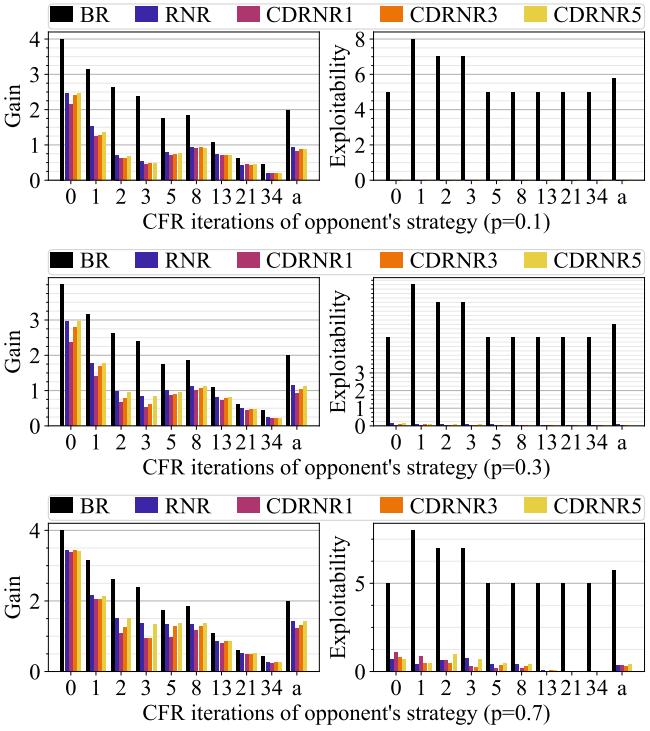

Figure 16: Aditional results for CDRNR with different values of p. Generated on Goofspiel.