Scalable and Asymptotically Safe Opponent Exploitation

Anonymous Author(s)
Submission Id: 2518

Abstract

The “Risk What You’ve Won in Expectation" (RWYWE) algorithm achieves opponent exploitation while guaranteeing safety in the sense of not being exploitable in turn. However, this algorithm scales poorly because it requires pessimistic and computationally expensive correction for hidden information whose conservatism compounds as the game tree grows. We show that the simpler “Risk What You’ve Won" (RWYW) approach which forgoes that pessimistic correction can still achieve a novel asymptotic safety guarantee. We provide a scalable implementation which replaces exact game-value computation and full opponent-strategy assessment with Monte-Carlo CFR and a neural network opponent model. Empirically, our approach scales to larger games, like Connect 4 and five-card Goofspiel, achieving strong exploitation of weak opponents while playing near-optimally against strong opponents and maintaining safety against attempts to exploit it.

Keywords

Safety, Exploitation, Game Theory, OpenSpiel

ACM Reference Format:
Anonymous Author(s). 2027. Scalable and Asymptotically Safe Opponent Exploitation. In *Proc. of the 26th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2027), Hanoi, Vietnam, 3–7 May 2027*, IFAAMAS, 9 pages.

1 Introduction

Most game-playing algorithms are designed to perform well against an optimal opponent, approximating a Nash equilibrium. This guarantees the value of the game in two-player zero-sum settings. Yet most human opponents, and many algorithmic ones, play suboptimally and can be exploited for a greater payoff than equilibrium play earns. Exploitation carries a risk: deviating from equilibrium to exploit an opponent’s weaknesses opens the agent to counter-exploitation if the opponent model is wrong or the opponent adapts [17]. This paper proposes a method for exploiting suboptimal opponents that controls this risk and remains feasible to run in larger settings than prior work.

Ganzfried and Sandholm [3] showed that exploitation can be made *safe*: an agent that risks only the “gifts” an opponent has given, in expectation, never falls below the game value (in the sense that the cumulative expected payoff is guaranteed to exceed the cumulative game value). Their algorithm, Risk What You’ve Won in Expectation (RWYWE), has two limitations in large games. First, it requires computing your expected utility given the opponent’s strategy, which is computationally expensive beyond games of moderate size. Second, in extensive-form games, its safety rests on a pessimistic assumption about what the opponent would have done in counterfactual scenarios whose conservatism compounds as the game tree grows. We show that this conservatism can severely delay exploitation, even against obviously exploitable opponents. Verifying complete safety is thus incompatible with meaningful exploitation in large extensive-form games.

We therefore relax the safety requirement. We introduce *asymptotic safety*, which requires that the agent’s average shortfall below the game value vanish almost surely as play continues, rather than holding in every round. We show that the simpler Risk What You’ve Won (RWYW) algorithm, which risks realized rather than expected winnings and forgoes the pessimistic correction, is asymptotically safe. This requires no additional computation and can begin to exploit significantly more rapidly.

Building on this, we develop a scalable implementation that replaces exact equilibrium computation with Monte Carlo CFR [10], uses a neural network as a scalable opponent model, and replaces exact search over safe strategies with sampled rollouts guarded by a conservative bound.

Empirically, our variant matches the performance of RWYWE in Kuhn Poker, where both can run, and scales to games where full-tree methods cannot. Against a random opponent, it earns the highest average return in all but one game tested (Table 3), well beyond standard error. Against MC-CFR opponents in larger games it earns substantial positive expected value per game. It continues to win against MC-CFR models with the same amount or more iterations than its baseline. In smaller examples, its exploitability remains comparable to its MC-CFR baseline.

2 Related Work

The general approach we build on was introduced by McCracken and Bowling [13], who defined the notion of ε-safe strategies against opponents who are modeled but not fully understood and proposed updating ε based on expected performance. Ganzfried and Sandholm [3] built on this by defining the overall safety guarantee we relax. Jeary and Turrini [8] observe the same framework can, and in practice will, be applied even to agents whose base strategy is imperfect, an approach we adopt in our scalable implementation. All these works adopt an approach similar to RWYWE, in contrast to our focus on the advantages of RWYW.

Restricted Nash Response (RNR) [9] models the opponent as alternating between fixed and equilibrium (or arbitrary) play and tries to exploit such an opponent, with the assumed probability implicitly serving as a way to trade off between safety and exploitation. More recent work has made this type of approach more scalable by combining it with search and ideas from subgame solving [4, 12, 14, 15]. Essentially, in large games we expect the base strategy we learn will be imperfect. As a result, subgame solving techniques can be used to improve it in the specific scenarios that occur during play. These improvements create a new form of budget that can be spent



This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 26th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2027), M. Baldoni, F. Fang, W. Yeoh, N. Yorke-Smith (eds.), 3–7 May 2027, Hanoi, Vietnam. © 2027 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

for safe opponent exploitation: the improved performance can be risked to exploit the opponent while still guaranteeing performance at least as good as the baseline. This line is orthogonal to our work and could be used as a complement to our approach in the case where RWYW chooses not to exploit.

Our method accomplishes opponent modeling through a neural network, which allows for quick adaptability and some inference, but does require some amount of training before it can be effective. One solution to this, not explored in this paper as it is orthogonal to our contributions, is to train across a wide population, adapting to a particular opponent on some instance. This idea has been explored in deep reinforcement learning. DRON [6] learns a shared policy network as well as an opponent-embedding network, trained against a population, with an ultimate policy tailored to observed opponent behavior without gradient updates. This concept is also vital for policy-space response oracles [11], which iteratively build best-responses to mixtures of previously learned policies. The goal of PSROs is to generalize across strategies, the resulting policies can form a great foundation against most real-world strategies. There is also a family of methods which treat fast per-opponent adaptation as a meta-learning objective. Model-agnostic meta-learning (MAML) [1] optimizes networks to need only a small number of gradient steps on a new task, and has been applied to similar settings of opponent modeling. Some frameworks like RL^2 [7] save adaptation for the hidden state of the recurrent policy, being particularly trained to adapt, meaning the modeling happens only in the forward pass. Again these are orthogonal to our focus which treats the details of the opponent model largely as a black box. Nashed and Zilberstein [16] provide a survey of opponent modeling approaches.

3 Background

A representation of a game must show who the players are, their available actions, the values they assign to each action, and the available per-player knowledge. A normal form game is the simplest possible, using a matrix of tuples to show the outcome of each possible pair of moves. These can only represent a single decision for each player, which makes for unwieldy representations of multi-stage games. Extensive Form Games are represented as a tree, allowing successive actions to be informed by the opponent's prior actions, certain game elements to be hidden from some players, and chance nodes to make random traversals through layers of the tree. Our algorithm functions on both representations.

Some behaviors in a game can be represented as a *pure strategy*, which is a deterministic prescription of which move to make for every node on the game tree. A *mixed strategy* is a probability distribution over possible moves, or pure strategies. Player i's *best response* to opponent strategy σ_{-i} in the set of mixed strategies Σ_i is any strategy in

$$\arg\max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}).$$

A *Nash Equilibrium* is a strategy profile σ where every player's strategy σ_i is a best response to σ_{-i} for every i. In zero sum games, an equilibrium strategy is also referred to as the *Game-Theoretic Optimum*, or GTO. The game value v captures the resulting optimal expected utility for each player and is defined as

$$v = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2)$$

3.1 Safety

A game value provides a benchmark for safety. A strategy's *exploitability* is a measure of how susceptible it is to being exploited, which gets higher as it veers further from equilibrium. Formally, it can be expressed as the difference of the value of the game and the performance of the strategy (σ_i) against the best response against it

$$\text{expl}(\sigma_i) = v_i - \min_{\sigma'_{-i}} u_i(\sigma_i, \sigma'_{-i}).$$

This definition of exploitability of a single player's strategy, rather than a joint strategy, is less common, but Ganzfried and Sandholm [3] use it and we follow them because a notion of exploitability that can be calculated without access to the opponent's strategy profile is essential in our setting.

From this, *safety* can be formally defined. An algorithm is safe when at least the value of the game is guaranteed in expectation regardless of the strategy used by the opponent. Clearly this is possible to achieve by simply always playing the GTO, since by definition it has exploitability zero.

More interestingly, Ganzfried and Sandholm [3] showed that, when playing the game repeatedly, strategies other than equilibrium can be safe. Specifically suppose the opponent has played suboptimally in the past to such an extent that the player has been 'gifted' an expected payoff higher than the value of the game (excluding Connect 4, in which it almost always wins). Then as long as the 'gifts' outweigh the exploitability of the current strategy the result will be safe. This leaves room to choose an exploitable strategy in an attempt to exploit weak opponents while maintaining safety if the opponent turns out to be sophisticated.

3.2 Risk What You've Won

The simplest algorithm built on the idea of attempting to maintain safety even when playing non-GTO strategies is *Risk What You've Won* (RWYW) [3]. This is a basic but unsafe algorithm for exploitation of suboptimal strategies. It consists of constructing a model $M(\pi)$ of the opponent's strategy π, and a constant k representing the current possible exploitability. Specifically, at round t, k^t is the cumulative payoff minus $(t-1)v$. From this, a collection of strategies SAFE_{k^t} is defined from all strategies with exploitability less than k^t, and of those the best-response is found to the opponent model $M(\pi)$ among strategies in SAFE_{k^t}.

This method is not safe, but a slight alteration can make it so. The value of k^t must be calculated using the expected payoff under the current mixed strategy, not the realized payoff at that point. Using only the realized payoff makes RWYW sensitive to noise, which can allow k^t to be arbitrarily higher than the actual amount that can be safely wagered. See Proposition 6.4 of Ganzfried and Sandholm [3]. Essentially, RWYW mistakes luck for a gift. The algorithm with this "corrected" version of k^t is called Risk What You've Won in Expectation, or RWYWE. Pseudocode is provided as Algorithm 1.

RWYWE can safely exploit opponents in normal form games, but some further tweaks are needed to make it work in extensive-form games, particularly those with imperfect information [3]. In extensive-form games, we do not know what the opponent would have done in counterfactual situations, so guaranteeing safety requires a pessimistic assumption that the opponent is playing a best

response to your strategy everywhere off the path of play (and in imperfect information settings, the best possible infoset), while playing as observed on the path of play. We denote this assumed strategy τ^t_{-i}. It is assumed that the setting allows for the opponent's private information to be revealed at the end of the game, letting k be updated according to the actual information. This substitution is what supplies RWYWE's safety guarantee in these settings, but its conservatism compounds with the size and depth of the game tree, making it a major obstacle to running the algorithm at scale. We provide the pseudocode for this generalization in Algorithm 2.

4 Asymptotic Safety

The safety guarantee of Ganzfried and Sandholm [3] is a finite-horizon guarantee in expectation: at every horizon T and against every opponent, the agent's expected cumulative payoff is at least Tv_i. This is a strong requirement, and it is what forces RWYWE's pessimistic accounting. An agent may only spend winnings it can attribute to the opponent's mistakes in expectation, and in imperfect-information games it must credit those mistakes as though they were made with every private type the opponent could hold. We show the cost of this pessimism in Sec. 4.1: against an opponent that is obviously exploitable, RWYWE needs a number of rounds linear in the size of the opponent's private information before it can play the best response, and the credit it gains per round shrinks as the game grows.

In many settings where opponent exploitation is desirable, such as long sessions of poker, repeated negotiation or adversarial bots, the quantity of interest is the long-run average payoff rather than the expected payoff at a fixed horizon. We therefore propose a different guarantee: the agent may fall below the game value temporarily, but it may not do so at a non-vanishing rate.

We now define a new notion of safety, called *Asymptotic Safety*. For a strategy to be Asymptotically Safe, the following limit must hold almost surely with respect to its chosen sequence of play sequences a^t_{-i}.

$$\liminf_{T\to\infty} \frac{1}{T}\left(\sum_{t=1}^{T} u_i(a^t_i, a^t_{-i}) - v_i\right) \geq 0$$

While RWYW is not safe, it is *asymptotically safe*. Note that the definition only requires non-negativity to account exploitation of the opponent, which can be positive on average.

Although prior work has recognized that RWYWE is not scalable and thus sought alterative approaches[4, 12, 14, 15], they have kept its approach of framing safety as a property that should hold in expectation, based at least in part by the belief that RWYW lacked safety guarantees. Our results provide a basis for rethinking this *even setting aside computational issues*. They show that RWYW can exploit more aggressively in large games while providing an appropriate form of safety. Of course, RWYW also has attractive computational advantages, which we explore further in Sec. 5.

4.1 Slow exploitation under RWYWE

RWYWE maintains safety by being extremely conservative, which leads to its ability to diverge from equilibrium taking more time as the complexity of the played game increases. The following example demonstrates that this tendency leads it to suboptimal play for arbitrary lengths of time.

Consider a variant of Rock-Paper-Scissors in which there exists a fourth option called "Trap" which is only available to player 2. In each round, a private number θ is given to the opponent (P2) in $\{1..n\}$. If $\theta = n$ then T has value -1 for P1 regardless of strategy. On the other hand, if $\theta < n$ then it returns 2 for playing R, and 1 for P and S. Equilibrium in this game is for P1 to always play uniform random, and for P2 to play uniform random when $\theta < n$, but play T when $\theta = n$.

Suppose an opponent always plays T, regardless of the value of n. The best response to this opponent is to always play R, with expected return $2 - 3/n$ per round. The GTO strategy of uniform random, by comparison, only has expected value of $\frac{4n-7}{3n}$, which is less than the pure R strategy for all $n > 1$. Thus the always-play-T opponent is highly exploitable relative to equilibrium play.

RWYWE, when playing against a repeated version of this game with the extensive form k-update, will begin shifting toward pure R as soon as k is positive, but only gradually. This is because k will update pessimistically, only updating k for the particular θ value revealed that round, assuming GTO play otherwise. With game value $v = -1/n$, this adds $(1 + r)/n$ per round when $\theta < n$, where r is its current probability of R This quantity starts at $4/(3n)$ as $r = 1/3$ at equilibrium and reaches a maximum value of $2/n$ when $r = 1$. Pure R has exploitability $1 - 1/n$, so RWYWE only reaches it after between $n/2$ and $3n/4$ rounds. RWYW, on the other hand, will typically begin fully exploiting in round 2 once it has attained surplus of at least 1 with probability $(n - 1)/n$.

For $n = 100$, played over ten rounds, RWYW almost always switches after one round and earns about 1.90 per round, while RWYWE has only moved its probability on R from 1/3 to about 0.47 by round ten and earns about 1.38, barely above GTO's 1.31 and far from the best response's 1.97; it needs nearly 60 rounds on average to reach pure R. As n grows with the number of rounds fixed, RWYWE's gain over equilibrium vanishes while RWYW's does not.

While this example is for a particular game, it illustrates a general problem of RWYWE. The pessimistic τ^t_{-i} correction is what gives RWYWE its safety guarantee in hidden-information settings, but it does so by assuming the opponent best-responds to every yet-unseen infoset, a correction whose conservatism compounds with tree depth and width. As a result, the size of the "gift" it recognizes from an opponent's mistake shrinks as the game tree grows.

4.2 Asymptotic Safety of RWYW

We now present our main theoretical result, which demonstrates that, contrary to prior assumptions, RWYW does actually provide a safety guarantee. The concern raised in previous work is that it might mistake luck for gifts, leading it to get repeatedly exploited. Our key observation is that, while this is true, this is partially canceled out by the times it mistakes bad luck for having been exploited. So to get repeatedly exploited to a large extent requires substantially more good luck than bad luck. Such events become unlikely as the the number of rounds grows (much as the law of large numbers guarantees that the fraction of heads becomes quite

unlikely to be far from half when tossing a fair coin a large number of times. This provides the basis for our safety guarantee.

Theorem 4.1. *RWYW is asymptotically safe.*

Our analysis proceeds by analyzing the evolution of the value of k the algorithm uses to determine whether it is safe to exploit. This is a complicated stochastic process because it depends not only on chance but also the history-dependent rule for RWYW and arbitrary behavior by the opponent. However, using the standard Doob decomposition we can separate it into a predictable component and a Martingale one. Importantly, all the "luck" ends up in the Martingale component which allows us to use the law of the iterated logarithm to provide a bound on how much RWYW can be exploited that grows sublinearly.

Proof. Let a game and an arbitrary opponent be given. Without loss of generality normalize the game so that $v = 0$ and let Δ denote a bound the absolute value of the possible payoffs of the normalized game. With this normalization, the current value of k is exactly the extent to which safety has been violated. Let G_t be a random variable for the value of k at time t.

Decompose the process G_t at time t into its predictable trend and its fluctuation (the Doob decomposition): $G_t = A_t + M_t$, where M_t is a martingale with increments bounded by Δ. This portion carries the conditionally mean-zero part of every increment, while predictable drift is absorbed into A_t, which moves only on decisions which exploit or gift.

When GTO is played by both players at time t, $A_{t+1} - A_t = 0$, because by definition the expected change is zero. Thus M_t captures all of GTO as well as fair gambles within rounds where there is some amount of exploitation or gift. Similarly A_t will capture both pure exploitation and gifts and mixed behavior between GTO and exploitation.

The predictable drift A_t is the object that must be bounded, since it carries exploitation. In rounds when the opponent exploits RWYW, so $A_t < A_{t-1}$, RWYW did not play GTO so $k_{t-1} > 0$. In terms of our process, this means that at such times $G_{t-1} = M_{t-1} + A_{t-1} > 0 \Rightarrow M_{t-1} > -A_{t-1}$. That is, M_{t-1} is an upper bound on the amount of net exploitation to date, meaning that taking the maximum of the M_t (plus Δ for the effect of the exploitation that occurred at time t) gives a universal upper bound on the amount of exploitation.

Because M_t is a martingale with bounded increments, the law of the iterated logarithm (in the martingale form of [2]) gives $\max_{t \le T} |M_t| = O(\sqrt{T \log\log T})$ almost surely. This requires linear accumulation of conditional variance, which holds because $|M_t - M_{t-1}| \le \Delta \Rightarrow \langle M \rangle_T \le (\Delta)^2 T$ (where $\langle M \rangle_T = \Sigma_{t \le T} \mathbf{E}[(M_t - M_{t-1})^2]$ is the predictable quadratic variation of the martingale M), so conditional variance grows at most linearly. We use Theorem 4.1 from [2], with $S_n = M_T$ and $T_n = \langle M \rangle_T$. Thus, M_T is sublinearly bounded, and because M_{t-1} is an upper bound on net exploitation, G_T is also sublinear, and RWYW is asymptotically safe. □

5 Methodology

Thus far we have discussed two key obstacles that make RWYWE scale poorly: pessimism and the need for an expensive computation of expected utility in the full game. RWYW inherently avoids these: it simply never needs to compute the expected utility it would have earned, pessimistic or otherwise.

We now discuss two other design aspects that contribute to a scalable implementation. First, RWYW requires computing the v and the equilibrium strategy that realizes it, which is impractical to do exactly in any game of reasonable size. Jeary and Turrini [8] observe that we can replace this with any arbitrary baseline and achieve a safety guarantee relative to that baseline. Our implementation uses MC-CFR for this purpose. Second, RWYW is silent on how the opponent model should operate. We adopt a simple and highly scalable approach that uses a neural network representation and selects exploitation strategies with a sample, rollout-based search.

Algorithm 1 Risk What You've Won (in Expectation) in Normal Form, Perfect Information Contexts

```
v ← game value to player
k^1 ← 0
for t = 1 to T do
  π^t ← argmax_{π∈SAFE(k^t)} M(π)
  Play action a_i^t according to π^t
  Opponent plays action a_{-i}^t according to unobserved distribution π_{-i}^t
  Update M with opponent's actions, a_{-i}^t
  (In Expectation) k^{t+1} ← k^t + u_i(π_i^t, π_{-i}^t) − v
  (Not in Expectation) k^{t+1} ← k^t + u_i(a_i^t, a_{-i}^t) − v
end for
```

Algorithm 2 Risk What You've Won (in Expectation) in Extensive Form Games with imperfect information

```
v ← game value to player
k^1 ← 0
for t = 1 to T do
  π^t ← argmax_{π∈SAFE(k^t)} M(π)
  Play actions a_i^t according to π^t
  Opponent plays actions a_{-i}^t according to unobserved distribution π_{-i}^t
  Update M with opponent's actions, a_{-i}^t, and their private information θ_{-i}^t
  τ_{-i}^t ← strategy for opponent who plays best response to π^t provided they play a_{-i}^t when they have private information θ_{-i}^t.
  (In Expectation) k^{t+1} ← k^t + u_i(π_i^t, τ_{-i}^t) − v
  (Not In Expectation) k^{t+1} ← k^t + u_i(a_i^t, a_{-i}^t) − v
end for
```

5.1 Algorithm Overview

This algorithm is broadly the same as Risk What You've Won, with changes to the opponent model and game value so that it can run on large games, with a malleable tradeoff between efficiency and efficacy. It starts by learning a baseline strategy, which is now not equilibrium, but approximates one using Monte Carlo CFR. A policy is trained to be an opponent during rollouts, which are used to estimate the value of nodes along the game tree. This policy is a

Algorithm 3 Risk What You've Won (RWYW): Faster Version

v ← game value to player
mn ← minimum return possible in game
$k^1 \leftarrow 0$
$M \leftarrow$ neural network to approximate opponent behavior
for $t = 1$ **to** T **do**
  $\pi^t \leftarrow$ MCCFR-chosen baseline move.
  $\pi^t \leftarrow \pi^t$ **if** $k \leq t*v+(mn-v)$ **else** Approximate Best Response
  Play action a_i^t according to π^t
  Opponent plays action a_{-i}^t according to unobserved distribution π_{-i}^t
  Update M with opponent's actions, a_{-i}^t
  $k^{t+1} \leftarrow k^t + u_i(\pi_i^t, a_{-i}^t) - v$
end for

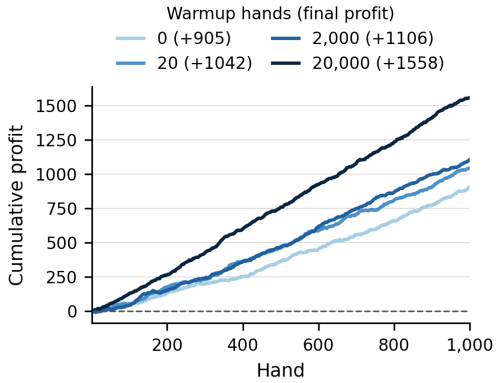


Figure 1: Model returns by number of training rounds

neural network, which maps information states to the opponent's moves. The training occurs during play, so many warmup iterations are useful, although not strictly necessary (see Figure 1). After this, a candidate move is evaluated for potential gain, and if the difference between accumulated gain and potential exploitability of the move allows the risk to be taken, the chosen move is played. Otherwise the method defaults to the CFR baseline. After each opponent action, the opponent model training data is updated with their latest move, and k is updated at the end of each hand. The pseudocode for this method is given as Algorithm 3.

The original RWYWE algorithm updates the safety budget k by adding $u_i(\pi_i^t, \tau_{-i}^t) - v^*$ to k each hand, where τ_{-i}^t is a pessimistic opponent strategy that plays the observed actions a_{-i}^t, and v^* is the game value to player i. We instead use the RWYW update $k^{t+1} \leftarrow k^t + u_i(a_i^t, a_{-i}^t) - v^*$, where $u_i(a_i^t, a_{-i}^t)$ is the realized payoff of hand t and v^* is the estimated game value.

5.2 Opponent Model

In principle any framework for an opponent model could be deployed, but the experiments in the original RWYWE paper uses one which "assumes the opponent plays according to his observed frequencies so far, where we assume that we observe his hand at the end of each game iteration," [3] using a prior of five hands played at equilibrium. This works in the examples of the paper, but previous data becomes less relevant against an online algorithm, assuming the opponent also has learning capabilities and changes its strategy to work better against the agent. Additionally, in large games and a limited number of plays risks leaving most infosets unreached and simply set to the prior.

For our algorithm, we use a neural network to approximate a mixed strategy, instead of assessing all possible pure strategies as in the original RWYWE paper [3]. The input is an information state, and the output is a probability distribution over the possible actions.

The implementation is a straightforward three-layer MLP, via pytorch. It trains with the cross-entropy loss between the observed opponent's moves at a given infostate and the model's prediction of what it will be. The intermediate layers are sized 128 and 64, and the output layer is normalized via softmax. The learning rate is variable, set with an Adam optimizer.

The opponent model is trained online during play. Before each opponent action, the pair (o, a) of the opponent's observation at that point and the action they took is stored in memory. As in [3], we assume the opponent's private information is revealed after each hand, even when this is not automatic, so o includes the opponent's private card. Only a constant number of recent actions are included in the training data, forming a replay buffer of maximum size 10,000. After each hand, the model takes one gradient step on a batch of 32 pairs sampled from this buffer. This strikes a balance between prioritizing more recent data and maintaining a steady strategy in the face of random variance inherent to mixed strategies.

The need for opponent data is a limitation of this algorithm. In very simple games like Kuhn Poker the opponent model can be reasonably accurate after only hundreds of iterations, but it can benefit from thousands in larger examples studied here. We tested the extent to which training was necessary, with the results shown in Figure 1. Here RWYW is trained for a variable amount of rounds against a static, exploitable opponent (MC-CFR trained for 500 iterations) in Leduc Poker. Its profit steadily increases as amount of training rounds before play starts increase, but not by a large amount. The session with zero training drew a total profit of +905 over 1,000 rounds, which shows that the model is able to adjust to an opponent quickly even without training. In our main experiments (Tables 2 and 3) no separate warmup is used: the model trains purely during the measured games.

Given the scale needed, training a strategy on a particular human opponent seems unlikely to be feasible in most practical situations. It could be useful as an adversary against a non-equilibrium algorithmic opponent, to demonstrate weaknesses, or to collect a composite strategy against a wide human playerbase, where training data is collected from an initial MC-CFR based algorithm, in the spirit of DRON [6] and PSRO [11]. In any case, this challenge is orthogonal to our work.

5.3 Other Optimizations

To approximate the game value, this algorithm uses MC-CFR [5] to find the move it will deem "optimal," and compare its best candidate for exploitation to. The MC-CFR implementation used takes a variable number iterations i to train, which is the bulk of the time complexity of the algorithm for most amounts. This is fairly underpowered at low values of i, but the buffer provided by the variance calculation makes it a reasonable baseline.

Finding an optimal move in a tractable amount of time is difficult, and for large games we use several relaxations and approximations at various stages. To accomplish this, an ε-safe best response must be found to the opponent model. We assure safety by only veering from equilibrium when k is greater than the lowest negative payoff possible in the game. This guarantees that every possible strategy is safe and so we can choose whatever is found as the best response to the opponent.

To begin, we use our pre-computed MCCFR to find our approximate GTO move for the particular infoset and game state. Then, we assess a random subset of m candidate moves. The assessment consists of r rollouts, which are conducted between the opponent model and MCCFR baseline. Each rollout starts from a history sampled from our information set, weighted by chance probabilities and the opponent model, so rollouts never use the opponent's hidden information, and all candidates share the same sampled histories. This does not give the true or expected value, as it is possible that optimal play is nothing like the opponent model, or that RWYW will make more exploitative moves were the gamestate to arise naturally. This process is detailed in pseudocode in Algorithm 4.

This approach is substantially simpler and more lightweight than approaches that compute some form of best response to the opponent model, either globally or via subgame solving. Nevertheless, it suffices to demonstrate the effectiveness of a RWYW-based approach in our experiments. As the details of the exploitation strategy computation are orthogonal to our goal of providing a new, scalable form of safety guarantee, this is suitable for our purposes.

Algorithm 4 Finding Approximate Best Response

```
mc ← maximum number of candidates chosen
num_rollouts ← rollouts per value estimate
π_base ← MCCFR baseline policy
legal ← LegalActions(state)
candidates ← RandomSubset(legal, min(|legal|, mc))

values ← ∅
for a ∈ candidates do
  values[a] ← Mean_{i∈1..num_rollouts}(Rollout(state))
end for
new_action ← argmax(values)
return new_action

function Rollout(state)
while state is not terminal and depth < depth_cap do
  cp ← CurrentPlayer(state)
  if cp == Chance then
    state ← ApplyChanceOutcome(state)
    continue
  end if
  b ← opp_model_π(state) if cp == opponent else π_base(state)
  apply SampleAction(state, cp, b) to state
end while
return Payoff(state) if terminal else 0
```

6 Experiments

The central algorithm built in this paper has two central claims that must be tested. First, removing the "expectation" part from the base of the algorithm has negligible impact on its overall payoff against a strong opponent and positive impact on its ability to exploit weak opponents in complex games. Second, our changes to the game value calculation, opponent model, k-update, and finding the best move have made the algorithm scalable to larger-scale games than the original can handle.

Throughout this section, the optimized version of RWYW, with MC-CFR game value calculations and MLP opponent models, will be referred to as RWYW_2. Their MCCFR instances use 1500 iterations of regret accumulation, and other hyperparameters are given in Sec 5. Experiments that take more than one hour to run on a given game and algorithm will be skipped, with a hyphen in their place. Where applicable, values are given with standard error to the right.

6.1 RWYW vs RWYWE

To test our first claim, we reran an experiment of Ganzfried and Sandholm [3] with our new model, simply testing the performance against four classes of opponent in Kuhn Poker (results in Table 1). This allows for both ensuring our implementation of RWYWE aligns to the one described in the original paper, and comparing the new model to it on existing data. For each opponent we report the mean payoff per-round from 1000 rounds, averaged across fifty runs. The opponents span the range of exploitability:

- Uniform Random - Probabilities set at 0.5 for all binary choices in Kuhn Poker
- Sophisticated Static - A random probability is selected for each infoset within 0.2 of GTO.
- Dynamic - Uniform random for the first 100 rounds to pretend to be a weak opponent, then switches to best response to maximally exploit RWYW(E).
- Equilibrium - The GTO strategy.

The ambition here is to have opponents vary through the spectrum of possible exploitability, with random being very exploitable, and SS being only somewhat so. Dynamic tests how well either method adapts to a sophisticated opponent attempting to exploit it, and equilibrium ensures it still plays well against an unexploitable opponent. The agent plays first, so the game value $v = -1/18$. This means some negative numbers are above expectation.

We hypothesized that this experiment would have very similar results to the original paper's experiments against RWYWE, corollary to our broad hypothesis that the two algorithms will generally behave the same way on a long enough time scale. The results verify both claims. The two algorithms had very similar results against all four opponents, with neither clearly performing better in an example this small. The results against a random opponent were fairly close to the expected value of a best response (0.5 for player

Agent	Uniform Random	Sophisticated Static	Dynamic	Equilibrium
RWYWE	0.3814 ± 0.0332	-0.0235 ± 0.0102	-0.0879 ± 0.0042	-0.0687 ± 0.0065
RWYW	0.3877 ± 0.0330	-0.01638 ± 0.0079	-0.0846 ± 0.0035	-0.0712 ± 0.0066

Table 1: Performance against four opponents in Kuhn Poker

Agent	Uniform Random	Sophisticated Static	Dynamic	Equilibrium
RWYWE	1.5157 ± 0.0778	0.13752 ± 0.02619	-0.92804 ± 0.03044	-0.07036 ± 0.0198
RWYW	1.4596 ± 0.0501	0.09528 ± 0.03310	-0.19156 ± 0.01904	-0.07588 ± 0.02597

Table 2: Performance against four opponents in Leduc Poker

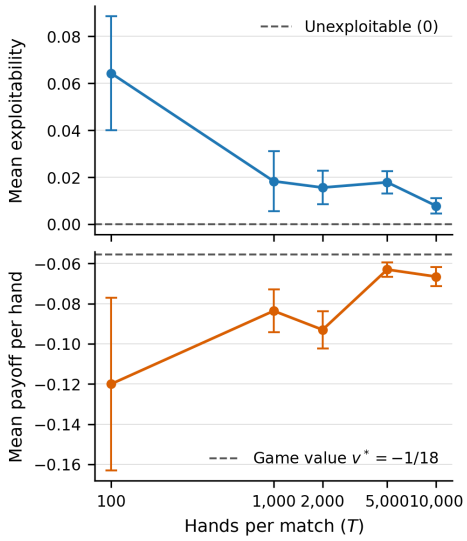


Figure 2: Average exploitability and payoff of RWYW in Kuhn Poker against pure best response opponent.

1), but somewhat held back by the equilibrium prior and safety constraints. This is not inconsistent with our example in Sec. 4.1 of why RWYWE can exploits slowly: the opponent in Kuhn Poker has only three private types, so RWYWE's pessimistic correction costs little, and the difference between the updates is expected to appear only in larger games.

Another important measure of an algorithm's performance against strong opponents is exploitability. RWYW has variable exploitability depending on its opponent, but ideally against a pure best response it will tend toward a low number. The mean exploitability of pure RWYW stays fairly low against a best response, as can be seen in Figure 2. Net payoff approaches equilibrium ($-1/18$) as the number of rounds increase. For this experiment, we ran five sessions each of 100, 1,000, 2,000, 5,000, and 10,000 rounds of Kuhn Poker against a pure best response opponent, to verify that RWYW converges to equilibrium on a large enough timescale, when the expectation approaches reality. This comes moderately quickly for both metrics, although the exploitability remains above 0.01 on average through 10,000 rounds.

Game	MCCFR(1000) Prep	$RWYW_2$	RWYWE	minimax	MCCFR (1000)
Kuhn Poker	214	4.81	2.78	0.047	0.025
Leduc Poker	466	6.89	12.32	0.500	0.023
Liar's Dice	356	5.94	802	8.94	0.027
Tic Tac Toe	505	25.61	1041	232	0.022
Goofspiel(5)	720	29.14	-	375	0.035
Connect Four	1635	111	-	-	0.020

Table 3: Average runtime per round in milliseconds by game and method, out of 5000 games (500 for RWYWE).

Game	$RWYW_2$	minimax	MCCFR (100)	MCCFR (1000)
Kuhn Poker	0.489 ± 0.033	0.245 ± 0.029	0.251 ± 0.035	0.158 ± 0.032
Leduc Poker	1.636 ± 0.124	0.851 ± 0.069	0.201 ± 0.110	0.677 ± 0.109
Liar's Dice	0.592 ± 0.018	0.739 ± 0.015	0.077 ± 0.022	0.173 ± 0.022
Tic-Tac-Toe	0.969 ± 0.005	0.996 ± 0.001	0.394 ± 0.019	0.484 ± 0.018
Goofspiel(5)	0.642 ± 0.016	0.248 ± 0.021	0.035 ± 0.021	0.181 ± 0.021
Connect Four	0.979 ± 0.010	-	0.122 ± 0.022	0.139 ± 0.022

Table 4: Average return by game and method against random player, modeling exploitation of a weak opponent. 5000 games per cell.

6.2 Scalability

We compared the new algorithm to three others: RWYWE, minimax, and standard Monte Carlo CFR. We chose minimax as a baseline in terms of speed, because as basic tree-search method it is very slow on complex games, especially with the maximum search depth of 100 we used. It provides a natural comparison point given that our strategy selection relies on a (more scalable) form of search. In imperfect-information games, minimax searches a sample of histories consistent with its information set rather than the true state, so it cannot see hidden information. Its returns should be close to GTO, although as it doesn't have an opponent model it will not find a best response to weak opponents. MC-CFR is fast, but requires the additional time of building the model, which is also given in Table 2. Because it is used in $RWYW_2$, that algorithm also needs warmup time which is roughly the same as that of MC-CFR, but with more time needed per-iteration for finding the best response and updating the opponent model. Thus, it is overall slower than MC-CFR, though much faster than minimax in large games. We included standard RWYWE, but because it requires enumerating all information sets as well as all pure strategies (the number of which is exponential in the number of information sets), it was not capable of running on anything past Kuhn Poker. We considered a sequence-form solution to the linear program, which was faster, but still unable to finish the necessary iterations in an hour for Leduc Poker and other larger games. We do not include other opponent exploitation baselines because, as discussed in Sec. 2, RWYWE has been dismissed as non-scalable and subsequent work has pursued orthogonal approaches. As such a comparison to them is not necessary to demonstrate our primary claim that RWYW provides a path to safe, scalable implementations. Many of the ideas from this literature our complementary to our approach; we leave exploring how best to exploit such opportunities to future research.

$RWYW_2$'s changes make the algorithm much more scalable with only modest dropoff in both exploitation and equilibrium convergence. Against minimax, it takes longer to run on Kuhn Poker,

Game	MCCFR (100)	MCCFR (500)	MCCFR (2000)	MCCFR (5000)	Best Response
Kuhn Poker	-0.0260 ± 0.0194	-0.0322 ± 0.0195	-0.0244 ± 0.0190	-0.0700 ± 0.0193	-0.1168 ± 0.0181
Leduc Poker	0.4734 ± 0.0661	0.5384 ± 0.0647	0.5102 ± 0.0632	0.4862 ± 0.0644	0.0266 ± 0.0685
Tic-Tac-Toe	0.9286 ± 0.0044	0.9356 ± 0.0039	0.9354 ± 0.0042	0.9436 ± 0.0039	0.2752 ± 0.0095
Liar's Dice	0.3532 ± 0.0132	0.4000 ± 0.0130	0.3924 ± 0.0130	0.4064 ± 0.0129	-
Goofspiel(5)	0.5478 ± 0.0110	0.5506 ± 0.0109	0.5414 ± 0.0110	0.5874 ± 0.0106	-
Connect 4	0.9540 ± 0.0042	0.9684 ± 0.0035	0.9696 ± 0.0035	0.9457 ± 0.0037	-

Table 5: Average return by game of $RWYW_2$ against MC-CFR players of varying iterations, modelling performance against weak and strong opponents. 5000 games per cell

and presumably other extremely simple games, but takes a similar length of time for Liar's Dice (See Table 2). For games beyond that, it rapidly outscale minimax. The MC-CFR methods took a similar amount of time (to be expected, since it is called by $RWYW_2$). This experiment gives MC-CFR's regret accumulation time in the first column, which is a step in the algorithm that occurs before any games are played, analogous to training in machine learning. $RWYW_2$ is quick in actual gameplay, but a thorough enough regret accumulation can be slow. However, essentially all approaches in the literature require some form of initial training to form a baseline strategy.

Speed allows the algorithm to be applied more widely, but it only matters if the algorithm accomplishes its main goal of exploiting weak opponents more thoroughly than equilibrium strategies do. We therefore test against a range of opponent strengths, beginning with the weakest. Against a random opponent, $RWYW_2$ performs far better than algorithms which stick closer to equilibrium (see Table 3). It wins most of the time, whereas competing algorithms either win far less, or are so computationally demanding that they could not be run. The exception is Liar's Dice, where minimax outperforms $RWYW_2$; with up to thirteen legal bids per decision, $RWYW_2$'s random sampled subset of five candidates often omits the strongest bid.

Exploitability and performance against random demonstrate the algorithm's proficiency against very strong and very weak opponents, but ideally it would maintain good performance against those lying between those two extremes. Table 4 shows an experiment running $RWYW_2$ against MC-CFR opponents with a variety of iteration counts, as well as a Best Response opponent in games where that computation is feasible. It plays first in all games, and maintains expected value higher than equilibrium against all non-Best Response opponents, except for Kuhn Poker against MCCFR(5000), against which its distance from GTO was within standard error. This is sometimes by a commanding margin, including against those with more regret accumulation steps than its baseline (1500 iterations). Together, these results demonstrate that $RWYW_2$'s approximations allow substantial scalability for only a modest cost in exploitation and equilibrium play.

7 Conclusion

The proposed algorithm allows for opponent exploitation at or beyond that of an equilibrium or near-equilibrium solution, while playing competitively against a suite of benchmark algorithms, including instances of MC-CFR with more regret accumulation than that used by it as a baseline. This algorithm performs strongly even in large games like Connect 4, which is a strong improvement on what is possible with RWYWE. It also progresses existing work on safe opponent exploitation by showing that meaningful, asymptotically-safe exploitation is achievable without the pessimistic worst-case corrections that make fully safe variants like RWYWE ineffective to run at scale even without considering computational limitations.

These gains come with limitations. Asymptotic safety allows for sublinear loss to accumulate on a long enough timescale, which might make the given algorithm insufficiently safe for certain applications. Additionally, our experiments assume the opponent's private information is revealed at the end of the game, which is not always true.

There is much work still to be done on this subject. Our notion of asymptotic safety chooses a particular way of quantifying the asymptotic performance. If others are of interest, Theorem 1 could potentially be extended to capture them, perhaps by making use of other martingale related results such as the BDG and Azuma inequalities. Some of our suggested alternative routes for opponent modeling have potential use in more accurately modeling the opponent with less warmup time. More broadly there is room to explore how RWYW can be combined with other types of opponent exploitation. Finally, scaling to even larger domains, like no-limit poker and human opponents could show that these guarantees hold in even more complex settings.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *CoRR* abs/1703.03400 (2017). arXiv:1703.03400 http://arxiv.org/abs/1703.03400

[2] David A. Freedman. 1975. On Tail Probabilities for Martingales. *The Annals of Probability* 3, 1 (1975), 100–118. http://www.jstor.org/stable/2959268

[3] Sam Ganzfried and Tuomas Sandholm. 2015. Safe Opponent Exploitation. *ACM Trans. Econ. Comput.* 3, 2, Article 8 (April 2015), 28 pages. doi:10.1145/2716322

[4] Zhenxing Ge, Zheng Xu, Tianyu Ding, Linjian Meng, Bo An, Wenbin Li, and Yang Gao. 2024. Safe and robust subgame exploitation in imperfect information games. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 611, 16 pages.

[5] Richard Gibson, Neil Burch, Marc Lanctot, and Duane Szafron. 2012. Efficient Monte Carlo Counterfactual Regret Minimization in games with many player actions. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'12)*. Curran Associates Inc., Red Hook, NY, USA, 1880–1888.

[6] He He, Jordan L. Boyd-Graber, Kevin Kwok, and Hal Daumé III. 2016. Opponent Modeling in Deep Reinforcement Learning. *CoRR* abs/1609.05559 (2016). arXiv:1609.05559 http://arxiv.org/abs/1609.05559

[7] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. 2017. A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *CoRR* abs/1707.09183 (2017). arXiv:1707.09183 http://arxiv.org/abs/1707.09183

[8] Linus Jeary and Paolo Turrini. 2023. Safe Opponent Exploitation For Epsilon Equilibrium Strategies. *arXiv preprint arXiv:2307.12338* (2023).

[9] Michael Johanson, Martin Zinkevich, and Michael Bowling. 2007. Computing Robust Counter-Strategies. In *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), Vol. 20. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/6e7b33fdea3adc80ebd648fffb665bb8-Paper.pdf

[10] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. 2009. Monte Carlo Sampling for Regret Minimization in Extensive Games. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/00411460f7c92d2124a67ea0f4cb5f85-Paper.pdf

[11] Marc Lanctot, Vinícius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. *CoRR* abs/1711.00832 (2017). arXiv:1711.00832 http://arxiv.org/abs/1711.00832

[12] Mingyang Liu, Chengjie Wu, Qihan Liu, Yansen Jing, Jun Yang, Pingzhong Tang, and Chongjie Zhang. 2022. Safe opponent-exploitation subgame refinement. *Advances in Neural Information Processing Systems* 35 (2022), 27610–27622.

[13] Peter McCracken and Michael Bowling. 2004. Safe strategies for agent modelling in games. (01 2004).

[14] David Milec, Vojtěch Kovařík, and Viliam Lisý. 2025. Adapting Beyond the Depth Limit: Counter Strategies in Large Imperfect Information Games. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems* (Detroit, MI, USA) *(AAMAS '25)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2675–2677.

[15] David Milec, Ondřej Kubíček, and Viliam Lisý. 2024. Continual Depth-limited Responses for Computing Counter-strategies in Sequential Games. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems* (Auckland, New Zealand) *(AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2393–2395.

[16] Samer Nashed and Shlomo Zilberstein. 2022. A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research* 73 (2022), 277–327.

[17] Tuomas Sandholm. 2007. Perspectives on multiagent learning. *Artif. Intell.* 171, 7 (May 2007), 382–391. doi:10.1016/j.artint.2007.02.004