Combining Deep Reinforcement Learning and Search for Imperfect-Information Games

Noam Brown* **Anton Bakhtin*** **Adam Lerer** **Qucheng Gong**
Facebook AI Research
{noambrown,yolo,alerer,qucheng}@fb.com

Abstract

The combination of deep reinforcement learning and search at both training and test time is a powerful paradigm that has led to a number of successes in single-agent settings and perfect-information games, best exemplified by AlphaZero. However, prior algorithms of this form cannot cope with imperfect-information games. This paper presents ReBeL, a general framework for self-play reinforcement learning and search that provably converges to a Nash equilibrium in any two-player zero-sum game. In the simpler setting of perfect-information games, ReBeL reduces to an algorithm similar to AlphaZero. Results in two different imperfect-information games show ReBeL converges to an approximate Nash equilibrium. We also show ReBeL achieves superhuman performance in heads-up no-limit Texas hold'em poker, while using far less domain knowledge than any prior poker AI.

Introduction

Combining reinforcement learning with search at both training and test time **(RL+Search)** has led to a number of major successes in AI in recent years. For example, the AlphaZero algorithm achieves state-of-the-art performance in the perfect-information games of Go, chess, and shogi (Silver et al. 2018).

However, prior RL+Search algorithms do not work in imperfect-information games because they make a number of assumptions that no longer hold in these settings. An example of this is illustrated in Figure 1, which shows a modified form of Rock-Paper-Scissors in which the winner receives two points (and the loser loses two points) when either player chooses Scissors (Brown, Sandholm, and Amos 2018). The figure shows the game in a sequential form in which player 2 acts after player 1 but does not observe player 1's action.

The optimal policy for both players in this modified version of the game is to choose Rock and Paper with 40% probability, and Scissors with 20%. In that case, each action results in an expected value of zero. However, as shown in Figure 2, if player 1 were to conduct one-ply lookahead search as is done in perfect-information games (in which the equilibrium value of a state is substituted at a leaf node), then there would not be enough information for player 1 to arrive at this optimal policy.

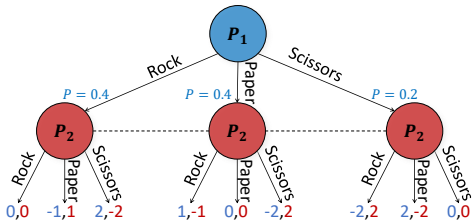

Figure 1: Variant of Rock-Paper-Scissors in which the optimal player 1 policy is (R=0.4, P=0.4, S=0.2). Terminal values are color-coded. The dotted lines mean player 2 does not know which node they are in.

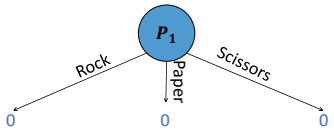

Figure 2: The player 1 subgame when using perfect-information one-ply search. Leaf values are determined by the full-game equilibrium. There is insufficient information for finding (R=0.4, P=0.4, S=0.2).

This illustrates a critical challenge of imperfect-information games: unlike perfect-information games and single-agent settings, the value of an action may depend on the probability it is chosen. Thus, a state defined only by the sequence of actions and observations does not have a unique value and therefore existing RL+Search algorithms such as AlphaZero are not sound in imperfect-information games. Recent AI breakthroughs in imperfect-information games have highlighted the importance of search at test time (Moravčík et al. 2017; Brown and Sandholm 2017b, 2019b; Lerer et al. 2020), but combining RL and search during training in imperfect-information games has been an open problem.

This paper introduces ReBeL (Recursive Belief-based Learning), a general RL+Search framework that converges to a Nash equilibrium in two-player zero-sum games. ReBeL builds on prior work in which the notion of "state" is expanded to include the probabilistic belief distribution of all

*Equal contribution

agents about what state they may be in, based on common knowledge observations and policies for all agents. Our algorithm trains a value network and a policy network for these expanded states through self-play reinforcement learning. Additionally, the algorithm uses the value and policy network for search during self play.

ReBeL provably converges to a Nash equilibrium in all two-player zero-sum games. In perfect-information games, ReBeL simplifies to an algorithm similar to AlphaZero, with the major difference being in the type of search algorithm used. Experimental results show that ReBeL is effective in large-scale games and defeats a top human professional with statistical significance in the benchmark game of heads-up no-limit Texas hold'em poker while using far less expert domain knowledge than any previous poker AI. We also show that ReBeL approximates a Nash equilibrium in Liar's Dice, another benchmark imperfect-information game, and open source our implementation of it.[1]

Related Work

At a high level, ReBeL resembles past RL+Search algorithms used in perfect-information games (Tesauro 1994; Silver et al. 2017; Anthony, Tian, and Barber 2017; Silver et al. 2018; Schrittwieser et al. 2019). These algorithms train a value network through self play. During training, a search algorithm is used in which the values of leaf nodes are determined via the value function. Additionally, a policy network may be used to guide search. These forms of RL+Search have been critical to achieving superhuman performance in benchmark perfect-information games. For example, so far no AI agent has achieved superhuman performance in Go without using search at both training and test time. However, these RL+Search algorithms are not theoretically sound in imperfect-information games and have not been shown to be successful in such settings.

A critical element of our imperfect-information RL+Search framework is to use an expanded notion of "state", which we refer to as a **public belief state (PBS)**. PBSs are defined by a common-knowledge belief distribution over states, determined by the public observations shared by all agents and the policies of all agents. PBSs can be viewed as a multi-agent generalization of belief states used in partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998). The concept of PBSs originated in work on decentralized multi-agent POMDPs (Nayyar, Mahajan, and Teneketzis 2013; Oliehoek 2013; Dibangoye et al. 2016) and has been widely used since then in imperfect-information games more broadly (Moravčík et al. 2017; Foerster et al. 2019; Serrino et al. 2019; Horák and Bošanskỳ 2019).

ReBeL builds upon the idea of using a PBS value function during search, which was previously used in the poker AI DeepStack (Moravčík et al. 2017). However, DeepStack's value function was trained not through self-play RL, but rather by generating random PBSs, including random probability distributions, and estimating their values using search. This would be like learning a value function for Go by randomly placing stones on the board. This is not an efficient way of learning a value function because the vast majority of randomly generated situations would not be relevant in actual play. DeepStack coped with this by using handcrafted features to reduce the dimensionality of the public belief state space, by sampling PBSs from a distribution based on expert domain knowledge, and by using domain-specific abstractions to circumvent the need for a value network when close to the end of the game.

An alternative approach for depth-limited search in imperfect-information games that does not use a value function for PBSs was used in the Pluribus poker AI to defeat elite humans in multiplayer poker (Brown, Sandholm, and Amos 2018; Brown and Sandholm 2019b). This approach trains a population of "blueprint" policies without using search. At test time, the approach conducts depth-limited search by allowing each agent to choose a blueprint policy from the population at leaf nodes. The value of the leaf node is the expected value of each agent playing their chosen blueprint policy against all the other agents' choice for the rest of the game. While this approach has been successful in poker, it does not use search during training and therefore requires strong blueprint policies to be computed without search. Also, the computational cost of the search algorithm grows linearly with the number of blueprint policies.

Notation and Background

We assume that the rules of the game and the agents' policies (including search algorithms) are **common knowledge** (Aumann 1976).[2] That is, they are known by all agents, all agents know they are known by all agents, etc. However, the outcome of stochastic algorithms (i.e., the random seeds) are not known. We later show how to remove the assumption that we know another player's policy.

Our notation is based on that of factored observation games (Kovařík et al. 2019) which is a modification of partially observable stochastic games (Hansen, Bernstein, and Zilberstein 2004) that distinguishes between private and public observations. We consider a game with $\mathcal{N} = \{1, 2, ..., N\}$ agents.

A **world state** $w \in \mathcal{W}$ is a state in the game. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_N$ is the space of joint actions. $\mathcal{A}_i(w)$ denotes the legal actions for agent i at w and $a = (a_1, a_2, ..., a_N) \in \mathcal{A}$ denotes a joint action. After a joint action a is chosen, a transition function $\mathcal{T}$ determines the next world state w' drawn from the probability distribution $\mathcal{T}(w, a) \in \Delta\mathcal{W}$. After joint action a, agent i receives a reward $\mathcal{R}_i(w, a)$.

Upon transition from world state w to w' via joint action a, agent i receives a **private observation** from a function $\mathcal{O}_{\text{priv}(i)}(w, a, w')$. Additionally, all agents receive a **public observation** from a function $\mathcal{O}_{\text{pub}}(w, a, w')$. Public observations may include observations of publicly taken actions by agents. For example, in many recreational games, including poker, all betting actions are public.

[1] https://github.com/facebookresearch/rebel

[2] This is a common assumption in game theory. One argument for it is that in repeated play an adversary would eventually determine an agent's policy.

A **history** (also called a trajectory) is a finite sequence of legal actions and world states, denoted $h = (w^0, a^0, w^1, a^1, ..., w^t)$. An **infostate** (also called an **action-observation history (AOH)**) for agent i is a sequence of an agent's observations and actions $s_i = (O_i^0, a_i^0, O_i^1, a_i^1, ..., O_i^t)$ where $O_i^k = \big(\mathcal{O}_{\text{priv}(i)}(w^{k-1}, a^{k-1}, w^k), \mathcal{O}_{\text{pub}}(w^{k-1}, a^{k-1}, w^k)\big)$. The unique infostate corresponding to a history h for agent i is denoted $s_i(h)$. The set of histories that correspond to s_i is denoted $\mathcal{H}(s_i)$.

A **public state** is a sequence $s_{\text{pub}} = (O_{\text{pub}}^0, O_{\text{pub}}^1, ..., O_{\text{pub}}^t)$ of public observations. The unique public state corresponding to a history h and an infostate s_i is denoted $s_{\text{pub}}(h)$ and $s_{\text{pub}}(s_i)$, respectively. The set of histories that match the sequence of public observation of s_{pub} is denoted $\mathcal{H}(s_{\text{pub}})$.

For example, consider a game where two players roll two six-sided dice each. One die of each player is publicly visible; the other die is only observed by the player who rolled it. Suppose player 1 rolls a 3 and a 4 (with 3 being the hidden die), and player 2 rolls a 5 and a 6 (with 5 being the hidden die). The history (and world state) is $h = \big((3,4),(5,6)\big)$. The set of histories corresponding to player 2's infostate is $\mathcal{H}(s_2) = \big\{\big((x,4),(5,6)\big) \mid x \in \{1,2,3,4,5,6\}\big\}$, so $|\mathcal{H}(s_2)| = 6$. The set of histories corresponding to s_{pub} is $\mathcal{H}(s_{\text{pub}}) = \big\{\big((x,4),(y,6)\big) \mid x,y \in \{1,2,3,4,5,6\}\big\}$, so $|\mathcal{H}(s_{\text{pub}})| = 36$.

Public states provide an easy way to reason about common knowledge in a game. All agents observe the same public sequence s_{pub}, and therefore it is common knowledge among all agents that the true history is some $h \in \mathcal{H}(s_{\text{pub}})$.[3]

An agent's **policy** π_i is a function mapping from an infostate to a probability distribution over actions. A **policy profile** π is a tuple of policies $(\pi_1, \pi_2, ..., \pi_N)$. The expected sum of future rewards (also called the **expected value (EV)**) for agent i in history h when all agents play policy profile π is denoted $v_i^\pi(h)$. The EV for the entire game is denoted $v_i(\pi)$. A **Nash equilibrium** is a policy profile such that no agent can achieve a higher EV by switching to a different policy (Nash 1951). Formally, π^* is a Nash equilibrium if for every agent i, $v_i(\pi^*) = \max_{\pi_i} v_i(\pi_i, \pi^*_{-i})$ where π_{-i} denotes the policy of all agents other than i. A **Nash equilibrium policy** is a policy π_i^* that is part of some Nash equilibrium π^*.

A **subgame** is defined by a root history h in a perfect-information game and all histories that can be reached going forward. In other words, it is identical to the original game except it starts at h. A **depth-limited subgame** is a subgame that extends only for a limited number of actions into the future. Histories at the bottom of a depth-limited subgame (i.e., histories that have no legal actions in the depth-limited subgame) but that have at least one legal action in the full game are called **leaf nodes**. In this paper, we assume for simplicity that search is performed over fixed-size depth-limited subgame (as opposed to Monte Carlo Tree Search, which grows the subgame over time (Gelly and Silver 2007)).

A game is **two-player zero-sum (2p0s)** if there are exactly two players and $\mathcal{R}_1(w,a) = -\mathcal{R}_2(w,a)$ for every world state w and action a. In 2p0s perfect-information games, there always exists a Nash equilibrium that depends only on the current world state w rather than the entire history h. Thus, in 2p0s perfect-information games a policy can be defined for world states and a subgame can be defined as rooted at a world state. Additionally, in 2p0s perfect-information games every world state w has a unique value $v_i(w)$ for each agent i, where $v_1(w) = -v_2(w)$, defined by both agents playing a Nash equilibrium in any subgame rooted at that world state. Our theoretical and empirical results are limited to 2p0s games, though related techniques have been empirically successful in some settings with more players (Brown and Sandholm 2019b). A typical goal for RL in 2p0s perfect-information games is to learn v_i. With that value function, an agent can compute its optimal next move by solving a depth-limited subgame that is rooted at its current world state and where the value of every leaf node z is set to $v_i(z)$ (Shannon 1950; Samuel 1959).

From World States to Public Belief States

In this section we describe a mechanism for converting any imperfect-information game into a continuous state (and action) space perfect-information game where the state description contains the probabilistic belief distribution of all agents. In this way, techniques that have been applied to perfect-information games can also be applied to imperfect-information games (with some modifications).

For intuition, consider a game in which one of 52 cards is privately dealt to each player. On each turn, a player chooses between three actions: fold, call, and raise. Eventually the game ends and players receive a reward. Now consider a modification of this game in which the players cannot see their private cards; instead, their cards are seen by a "referee". On a player's turn, they announce the probability they would take each action with each possible private card. The referee then samples an action on the player's behalf from the announced probability distribution for the player's true private card. When this game starts, each player's belief distribution about their private card is uniform random. However, after each action by the referee, players can update their belief distribution about which card they are holding via Bayes' Rule. Likewise, players can update their belief distribution about the *opponent's* private card through the same operation. Thus, the probability that each player is holding each private card is common knowledge among all players at all times in this game.

A critical insight is that *these two games are strategically identical*, but the latter contains no private information and is instead a continuous state (and action) space perfect-information game. While players do not announce their action probabilities for each possible card in the first game, we assume (as stated earlier) that all players' policies are common knowledge, and therefore the probability that a player would choose each action for each possible card is indeed known by all players. Of course, at test time (e.g., when our

[3] As explained in (Kovařík et al. 2019), it may be possible for agents to infer common knowledge beyond just public observations. However, doing this additional reasoning is inefficient both theoretically and practically.

agent actually plays against a human opponent) the opponent does not actually announce their entire policy and therefore our agent does not know the true probability distribution over opponent cards. We later address this problem.

We refer to the first game as the **discrete representation** and the second game as the **belief representation**. In the example above, a history in the belief representation, which we refer to as a **public belief state (PBS)**, is described by the sequence of public observations and 104 probabilities (the probability that each player holds each of the 52 possible private card); an "action" is described by 156 probabilities (one per discrete action per private card). In general terms, a PBS is described by a joint probability distribution over the agents' possible infostates (Nayyar, Mahajan, and Teneketzis 2013; Oliehoek 2013; Dibangoye et al. 2016).[4] Formally, let $S_i(s_{\text{pub}})$ be the set of infostates that player i may be in given a public state s_{pub} and let $\Delta S_1(s_{\text{pub}})$ denote a probability distribution over the elements of $S_1(s_{\text{pub}})$. Then PBS $\beta = (\Delta S_1(s_{\text{pub}}), ..., \Delta S_N(s_{\text{pub}}))$. In perfect-information games, the discrete representation and belief representation are identical.

Since a PBS is a history of the perfect-information belief-representation game, a subgame can be rooted at a PBS.[5] The discrete-representation interpretation of such a subgame is that at the start of the subgame a history is sampled from the joint probability distribution of the PBS, and then the game proceeds as it would in the original game. The value for agent i of PBS β when all players play policy profile π is $V_i^\pi(\beta) = \sum_{h \in \mathcal{H}(s_{\text{pub}}(\beta))} (p(h|\beta) v_i^\pi(h))$. Just as world states have unique values in 2p0s perfect-information games, in 2p0s games (both perfect-information and imperfect-information) every PBS β has a unique value $V_i(\beta)$ for each agent i, where $V_1(\beta) = -V_2(\beta)$, defined by both players playing a Nash equilibrium in the subgame rooted at the PBS.

Since any imperfect-information game can be viewed as a perfect-information game consisting of PBSs (i.e., the belief representation), in theory we could approximate a solution of any 2p0s imperfect-information game by running a perfect-information RL+Search algorithm on a discretization of the belief representation. However, as shown in the example above, belief representations can be very high-dimensional continuous spaces, so conducting search (i.e., approximating the optimal policy in a depth-limited subgame) as is done in perfect-information games would be intractable. Fortunately, in 2p0s games, *these high-dimensional belief representations are convex optimization problems.* ReBeL leverages this fact by conducting search via an iterative gradient-ascent-like algorithm.

ReBeL's search algorithm operates on supergradients (subgradients but for concave functions) of the PBS value function at leaf nodes, rather than on PBS values directly. Specifically, the search algorithms require the values of *infostates* for PBSs (Burch, Johanson, and Bowling 2014; Moravčík et al. 2017). In a 2p0sum game, the value of infostate s_i in β assuming all other players play Nash equilibrium π^* is the maximum value that player i could obtain for s_i through any policy in the subgame rooted at β. Formally,

$$v_i^{\pi^*}(s_i|\beta) = \max_{\pi_i} \sum_{h \in \mathcal{H}(s_i)} p(h|s_i, \beta_{-i}) v_i^{\langle \pi_i, \pi^*_{-i} \rangle}(h) \quad (1)$$

where $p(h|s_i, \beta_{-i})$ is the probability of being in history h assuming s_i is reached and the joint probability distribution over infostates for players other than i is β_{-i}. Theorem 1 proves that infostate values can be interpreted as a supergradient of the PBS value function in 2p0s games.

Theorem 1. *For any PBS $\beta = (\beta_1, \beta_2)$ (for the beliefs over player 1 and 2 infostates respectively) and any policy π^* that is a Nash equilibrium of the subgame rooted at β,*

$$v_1^{\pi^*}(s_1|\beta) = V_1(\beta) + \bar{g} \cdot \hat{s}_1 \quad (2)$$

where $\bar{g}$ is a supergradient of an extension of $V_1(\beta)$ to unnormalized belief distributions and $\hat{s}_1$ is the unit vector in direction s_1.

Since ReBeL's search algorithm uses infostate values, so rather than learn a PBS value function ReBeL instead learns an infostate-value function $\hat{v} : \mathcal{B} \rightarrow \mathbb{R}^{|S_1|+|S_2|}$ that directly approximates for each s_i the average of the sampled $v_i^{\pi^*}(s_i|\beta)$ values produced by ReBeL at β.[6]

RL+Search for Public Belief States

In this section we describe ReBeL and prove that it approximates a Nash equilibrium in 2p0s games. At the start of the game, a depth-limited subgame rooted at the initial PBS β_r is generated. This subgame is solved (i.e., a Nash equilibrium is approximated) by running T iterations of an iterative equilibrium-finding algorithm in the discrete representation of the game, but using the learned value network $\hat{v}$ to approximate leaf values on every iteration. During training, the infostate values at β_r computed during search are added as training examples for $\hat{v}$ and (optionally) the subgame policies are added as training examples for the policy network. Next, a leaf node z is sampled and the process repeats with the PBS at z being the new subgame root.

Search in a depth-limited subgame

In this section we describe the search algorithm ReBeL uses to solve depth-limited subgames. We assume for simplicity

[4] One could alternatively define a PBS as a probability distribution over histories in $\mathcal{H}(s_{\text{pub}})$ for public state s_{pub}. However, it is proven that any PBS that can arise in play can always be described by a joint probability distribution over the agents' possible infostates (Oliehoek 2013; Seitz et al. 2019), so we use this latter definition for simplicity.

[5] Past work defines a subgame to be rooted at a public state (Burch, Johanson, and Bowling 2014; Brown and Sandholm 2015; Moravcik et al. 2016; Moravčík et al. 2017; Brown and Sandholm 2017a; Kovařík and Lisý 2019; Šustr, Kovařík, and Lisý 2019; Seitz et al. 2019). However, imperfect-information subgames rooted at a public state do not have well-defined values.

[6] Unlike the PBS value $V_i(\beta)$, the infostate values may not be unique and may depend on which Nash equilibrium is played in the subgame. Nevertheless, any linear combination of supergradients is itself a supergradient since the set of all supergradients is a convex set (Rockafellar 1970).

Algorithm 1 ReBeL

function SELFPLAY($\beta_r, \theta^v, \theta^\pi, D^v, D^\pi$)
  while !ISTERMINAL(β_r) **do**
    $G \leftarrow$ CONSTRUCTSUBGAME(β_r)
    $\bar{\pi}, \pi^{t_{\text{warm}}} \leftarrow$ INITIALIZEPOLICY(G, θ^π)
    $G \leftarrow$ SETLEAFVALUES($G, \bar{\pi}, \pi^{t_{\text{warm}}}, \theta^v$)
    $v(\beta_r) \leftarrow$ COMPUTEEV($G, \pi^{t_{\text{warm}}}$)
    $t_{sample} \sim \text{unif}\{t_{\text{warm}} + 1, T\}$
    for $t = (t_{\text{warm}} + 1)..T$ **do**
      if $t = t_{sample}$ **then**
        $\beta'_r \leftarrow$ SAMPLELEAF(G, π^{t-1})
      $\pi^t \leftarrow$ UPDATEPOLICY(G, π^{t-1})
      $\bar{\pi} \leftarrow \frac{t}{t+1}\bar{\pi} + \frac{1}{t+1}\pi^t$
      $G \leftarrow$ SETLEAFVALUES($G, \bar{\pi}, \pi^t, \theta^v$)
      $v(\beta_r) \leftarrow \frac{t}{t+1}v(\beta_r) + \frac{1}{t+1}$ GETEV(G, π^t)
    Add $\{\beta_r, v(\beta_r)\}$ to D^v
    for $\beta \in G$ **do**
      Add $\{\beta, \bar{\pi}(\beta)\}$ to D^π
    $\beta_r \leftarrow \beta'_r$

that the depth of the subgame is pre-determined and fixed. The subgame is solved in the discrete representation and the solution is then converted to the belief representation. There exist a number of iterative algorithms for solving imperfect-information games (Brown 1951; Zinkevich et al. 2008; Hoda et al. 2010; Kroer et al. 2018; Kroer, Farina, and Sandholm 2018). We describe ReBeL assuming the **counterfactual regret minimization - decomposition (CFR-D)** algorithm is used (Zinkevich et al. 2008; Burch, Johanson, and Bowling 2014; Moravčík et al. 2017). CFR is the most popular equilibrium-finding algorithm for imperfect-information games, and CFR-D is an algorithm that solves depth-limited subgames via CFR. However, ReBeL is flexible with respect to the choice of search algorithm and we also show experimental results for **fictitious play (FP)** (Brown 1951).

On each iteration t, CFR-D determines a policy profile π^t in the subgame. Next, the value of every discrete representation leaf node z is set to $\hat{v}(s_i(z)|\beta_z^{\pi^t})$, where $\beta_z^{\pi^t}$ denotes the PBS at z when agents play according to π^t. This means that the value of a leaf node during search is conditional on π^t. Thus, the leaf node values change every iteration. Given π^t and the leaf node values, each infostate in β_r has a well-defined value. This vector of values, denoted $v^{\pi^t}(\beta_r)$, is stored. Next, CFR-D chooses a new policy profile π^{t+1}, and the process repeats for T iterations.

When using CFR-D, the *average* policy profile $\bar{\pi}^T$ converges to a Nash equilibrium as $T \to \infty$, rather than the policy on the final iteration. Therefore, after running CFR-D for T iterations in the subgame rooted at PBS β_r, the value vector $(\sum_{t=1}^{T} v^{\pi^t}(\beta_r))/T$ is added to the training data for $\hat{v}(\beta_r)$.

Self-play reinforcement learning

We now explain how ReBeL trains a PBS value network through self play. After solving a subgame rooted at PBS β_r via search, the value vector for the root infostates is added to the training dataset for $\hat{v}$. Next, a leaf PBS β'_r is sampled and a new subgame rooted at β'_r is solved. This process repeats until the game ends.

Since the subgames are solved using an iterative algorithm, we want $\hat{v}$ to be accurate for leaf PBSs on every iteration. Therefore, a leaf node z is sampled according to π^t on a *random* iteration $t \sim \text{unif}\{0, T-1\}$, where T is the number of iterations of the search algorithm.[7] To ensure sufficient exploration, one agent samples random actions with probabilility $\epsilon > 0$.[8] In CFR-D $\beta'_r = \beta_z^{\pi^t}$, while in CFR-AVG and FP $\beta'_r = \beta_z^{\bar{\pi}^t}$.

Theorem 2 states that, with perfect function approximation, running Algorithm 1 will produce a value network whose error is bounded by $\mathcal{O}(\frac{1}{\sqrt{T}})$ for any PBS that could be encountered during play, where T is the number of CFR iterations being run in subgames.

Theorem 2. *Consider an idealized value approximator that returns the most recent sample of the value for sampled PBSs, and 0 otherwise. Running Algorithm 1 with T iterations of CFR in each subgame will produce a value approximator that has error of at most $\frac{C}{\sqrt{T}}$ for any PBS that could be encountered during play, where C is a game-dependent constant.*

ReBeL as described so far trains the value network through bootstrapping. One could alternatively train the value network using rewards actually received over the course of the game when the agents do not go off-policy. There is a trade-off between bias and variance between these two approaches (Schulman et al. 2016).

Adding a policy network

Algorithm 1 will result in $\hat{v}$ converging correctly even if a policy network is not used. However, initializing the subgame policy via a policy network may reduce the number of iterations needed to closely approximate a Nash equilibrium. Additionally, it may improve the accuracy of the value network by allowing the value network to focus on predicting PBS values over a more narrow domain.

Algorithm 1 can train a policy network $\hat{\Pi} : \beta \to (\Delta\mathcal{A})^{|S_1|+|S_2|}$ by adding $\bar{\pi}^T(\beta)$ for each PBS β in the subgame to a training dataset each time a subgame is solved (i.e., T iterations of CFR have been run in the subgame). Using techniques based on (Brown and Sandholm 2016b), it is possible to warm start equilibrium finding given the initial policy from the policy network.

Playing an Equilibrium at Test Time

This section proves that running Algorithm 1 at test time with an accurately trained PBS value network will result in playing a Nash equilibrium policy in expectation even if we do not know the opponent's policy. During self play training

[7] For FP, we pick a random agent i and sample according to $(\pi_i^t, \bar{\pi}_{-i}^t)$ to reflect the search operation.

[8] The algorithm is correct if all agents sample random actions with probability ϵ, but that is inefficient because the value of a leaf node that is not reached by either agent's policy is irrelevant.

we assumed that both players' policies are common knowledge. This allows us to exactly compute the PBS we are in. However, at test time we do not know our opponent's entire policy, and therefore we do not know the PBS. This is a problem for conducting search, because search is always rooted at a PBS. For example, consider again the game of modified Rock-Paper-Scissors illustrated in Figure 1. For simplicity, assume that $\hat{v}$ is perfect. Suppose that we are player 2 and player 1 has just acted. In order to now conduct search as player 2, our algorithm requires a root PBS. What should this PBS be?

An intuitive choice, referred to as **unsafe** search (Gilpin and Sandholm 2006; Ganzfried and Sandholm 2015), is to first run CFR for T iterations for player 1's first move (for some large T), which results in a player 1 policy such as $(R = 0.4001, P = 0.3999, S = 0.2)$. Unsafe search passes down the beliefs resulting from that policy, and then computes our optimal policy as player 2. This would result in a policy of $(R = 0, P = 1, S = 0)$ for player 2. Clearly, this is not a Nash equilibrium. Moreover, if our opponent knew we would end up playing this policy (which we assume they would know since we assume they know the algorithm we run to generate the policy), then they could exploit us by playing $(R = 0, P = 0, S = 1)$.

This problem demonstrates the need for **safe** search, which is a search algorithm that ensures we play a Nash equilibrium policy in expectation. Importantly, it is *not* necessary for the policy that the algorithm outputs to always be a Nash equilibrium. It is only necessary that the algorithm outputs a Nash equilibrium policy *in expectation*. For example, in modified Rock-Paper-Scissors it is fine for an algorithm to output a policy of 100% Rock, so long as the probability it outputs that policy is 40%.

All past safe search approaches introduce constraints to the search algorithm (Burch, Johanson, and Bowling 2014; Moravcik et al. 2016; Brown and Sandholm 2017a; Šustr, Kovařík, and Lisỳ 2019). Those constraints hurt performance in practice compared to unsafe search (Burch, Johanson, and Bowling 2014; Brown and Sandholm 2017a) and greatly complicate search, so they were never fully used in any competitive agent. Instead, all previous search-based imperfect-information game agents used unsafe search either partially or entirely (Moravčík et al. 2017; Brown and Sandholm 2017b; Brown, Sandholm, and Amos 2018; Brown and Sandholm 2019b; Serrino et al. 2019). Moreover, using prior safe search techniques at test time may result in the agent encountering PBSs that were not encountered during self-play training and therefore may result in poor approximations from the value and policy network.

We now prove that safe search can be achieved without any additional constraints by simply *running the same algorithm at test time that we described for training*. This result applies regardless of how the value network was trained and so can be applied to prior algorithms that use PBS value functions (Moravčík et al. 2017; Serrino et al. 2019). Specifically, when conducting search at test time we pick a *random* iteration and assume all players' policies match the policies on that iteration. Theorem 3 states that once a value network is trained according to Theorem 2, using Algorithm 1 at test time (without off-policy exploration) will approximate a Nash equilibrium.

Theorem 3. *If Algorithm 1 is run at test time with no off-policy exploration, a value network with error at most δ for any leaf PBS that was trained to convergence as described in Theorem 2, and with T iterations of CFR being used to solve subgames, then the algorithm plays a $(\delta C_1 + \frac{\delta C_2}{\sqrt{T}})$-Nash equilibrium, where C_1, C_2 are game-specific constants.*

Since a random iteration is selected, we may select an early iteration in which the policy is poor. We can mitigate this by using modern equilibrium-finding algorithms, such as Linear CFR (Brown and Sandholm 2019a), that assign little or no weight to early iterations.

Experimental Setup

We measure **exploitability** of a policy π^*, which is $\sum_{i \in \mathcal{N}} \max_\pi v_i(\pi, \pi^*_{-i}) / |\mathcal{N}|$. All CFR experiments use alternating-updates Linear CFR (Brown and Sandholm 2019a). All FP experiments use alternating-updates Linear Optimistic FP.

We evaluate on the benchmark imperfect-information games of heads-up no-limit Texas hold'em poker (HUNL) and Liar's Dice. We also evaluate our techniques on turn endgame hold'em (TEH), a variant of no-limit Texas hold'em in which both players automatically check/call for the first two of the four betting rounds in the game.

In HUNL and TEH, we reduce the action space to at most nine actions using domain knowledge of typical bet sizes. However, our agent responds to any "off-tree" action at test time by adding the action to the subgame (Brown, Sandholm, and Amos 2018; Brown and Sandholm 2019b). The bet sizes and stack sizes are randomized during training. For TEH we train on the full game and measure exploitability on the case of both players having \$20,000, unperturbed bet sizes, and the first four board cards being 3♠7♡T♢K♠. For HUNL, our agent uses far less domain knowledge than any prior competitive AI agent.

We approximate the value and policy functions using artificial neural networks. Both networks are MLPs with GeLU (Hendrycks and Gimpel 2016) activation functions and LayerNorm (Ba, Kiros, and Hinton 2016). Both networks are trained with Adam (Kingma and Ba 2014). We use pointwise Huber loss as the criterion for the value function and mean squared error (MSE) over probabilities for the policy. In preliminary experiments we found MSE for the value net and cross entropy for the policy net did worse.

We use PyTorch (Paszke et al. 2019) to train the networks. We found data generation to be the bottleneck due to the sequential nature of the FP and CFR algorithms and the evaluation of all leaf nodes on each iteration. For this reason we use a single machine for training and up to 128 machines with 8 GPUs each for data generation.

Experimental Results

Figure 3 shows ReBeL reaches a level of exploitability in TEH equivalent to running about 125 iterations of full-game tabular CFR. For context, top poker agents typically use between 100 and 1,000 tabular CFR iterations (Bowling et al.

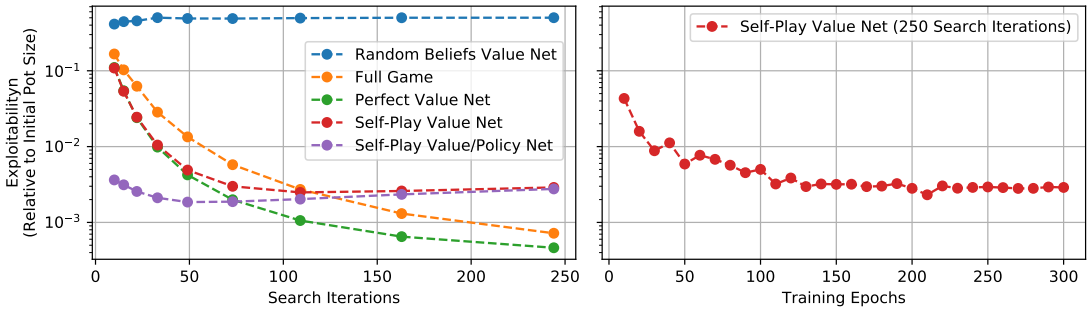

Figure 3: Convergence of different techniques in TEH. All subgames are solved using CFR-AVG. Perfect Value Net uses an oracle function to return the exact value of leaf nodes on each iteration. Self-Play Value Net uses a value function trained through self play. Self-Play Value/Policy Net additionally uses a policy network to warm start CFR. Random Beliefs trains the value net by sampling PBSs at random.

Bot Name	**Slumbot**	**BabyTartanian8**	**LBR**	**Top Humans**
DeepStack	-	-	383 ± 112	-
Libratus	-	63 ± 14	-	147 ± 39
Modicum	11 ± 5	6 ± 3	-	-
ReBeL *(Ours)*	45 ± 5	9 ± 4	881 ± 94	165 ± 69

Table 1: Head-to-head results of our agent against benchmark bots BabyTartanian8 and Slumbot, as well as top human expert Dong Kim, measured in thousandths of a big blind per game. We also show performance against LBR (Lisy and Bowling 2017) where the LBR agent must call for the first two betting rounds, and can either fold, call, bet $1\times$ pot, or bet all-in on the last two rounds. The $\pm$ shows one standard deviation. For Libratus, we list the score against all top humans in aggregate; Libratus beat Dong Kim by 29 with an estimated $\pm$ of 78.

2015; Moravčík et al. 2017; Brown and Sandholm 2017b; Brown, Sandholm, and Amos 2018; Brown and Sandholm 2019b). Our self-play algorithm is key to this success; Figure 3 shows a value network trained on random PBSs fails to learn anything valuable.

Table 1 shows results for ReBeL in HUNL. We compare ReBeL to BabyTartanian8 (Brown and Sandholm 2016a) and Slumbot, prior champions of the Computer Poker Competition, and to the local best response (LBR) (Lisy and Bowling 2017) algorithm. We also present results against Dong Kim, a top human HUNL expert that did best among the four top humans that played against Libratus. Kim played 7,500 hands. Variance was reduced by using AIVAT (Burch et al. 2018). ReBeL played faster than 2 seconds per hand and never needed more than 5 seconds for a decision. We compare this performance to DeepStack (Moravčík et al. 2017), Libratus (Brown and Sandholm 2017b), and Modicum (Brown, Sandholm, and Amos 2018).

Beyond just poker, Table 2 shows ReBeL also converges to an approximate Nash in several versions of Liar's Dice. Of course, tabular CFR does better than ReBeL when using the same number of CFR iterations, but tabular CFR quickly becomes intractable to run as the game grows in size.

Algorithm	**1x4f**	**1x5f**	**1x6f**	**2x3f**
Full-game FP	0.012	0.024	0.039	0.057
Full-game CFR	0.001	0.001	0.002	0.002
ReBeL FP	0.041	0.020	0.040	0.020
ReBeL CFR-D	0.017	0.015	0.024	0.017

Table 2: Exploitability on 4 variants of Liar's Dice: 1 die with 4, 5, or 6 faces and 2 dice with 3 faces. The top two rows represent baseline numbers when a tabular version of the algorithms is run on the entire game for 1,024 iterations. The bottom two rows show the performance of ReBeL operating on subgames of depth 2 with 1,024 search iterations. For exploitability computation of the bottom two rows, we averaged the policies of 1,024 playthroughs and thus the numbers are upper bounds on exploitability.

Conclusions

We present ReBeL, an algorithm that generalizes the paradigm of self-play reinforcement learning and search to imperfect-information games. We prove that ReBeL computes an approximate Nash equilibrium in two-player zero-sum games, demonstrate convergence in Liar's Dice, and demonstrate that it produces superhuman performance in the benchmark game of heads-up no-limit Texas hold'em.

ReBeL has limitations that present avenues for future research. Most prominently, the input to its value and policy functions currently grows linearly with the number of infostates in a public state. This is intractable in games such as Recon Chess (Newman et al. 2016) that have strategic depth but little common knowledge. ReBeL's theoretical guarantees are also limited only to two-player zero-sum games.

Nevertheless, ReBeL achieves low exploitability in benchmark games and superhuman performance in heads-up no-limit Texas hold'em while leveraging far less expert knowledge than any prior bot. We view this as a major step toward developing universal techniques for games.

References

Anthony, T.; Tian, Z.; and Barber, D. 2017. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, 5360–5370.

Aumann, R. J. 1976. Agreeing to disagree. *The annals of statistics* 1236–1239.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science* 347(6218): 145–149.

Brown, G. W. 1951. Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13(1): 374–376.

Brown, N.; and Sandholm, T. 2015. Simultaneous abstraction and equilibrium finding in games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Brown, N.; and Sandholm, T. 2016a. Baby Tartanian8: Winning Agent from the 2016 Annual Computer Poker Competition. In *IJCAI*, 4238–4239.

Brown, N.; and Sandholm, T. 2016b. Strategy-based warm starting for regret minimization in games. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Brown, N.; and Sandholm, T. 2017a. Safe and nested subgame solving for imperfect-information games. In *Advances in neural information processing systems*, 689–699.

Brown, N.; and Sandholm, T. 2017b. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

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

Brown, N.; and Sandholm, T. 2019b. Superhuman AI for multiplayer poker. *Science* eaay2400.

Brown, N.; Sandholm, T.; and Amos, B. 2018. Depth-limited solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, 7663–7674.

Burch, N.; Johanson, M.; and Bowling, M. 2014. Solving imperfect information games using decomposition. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Burch, N.; Schmid, M.; Moravcik, M.; Morill, D.; and Bowling, M. 2018. Aivat: A new variance reduction technique for agent evaluation in imperfect information games. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Dibangoye, J. S.; Amato, C.; Buffet, O.; and Charpillet, F. 2016. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research* 55: 443–497.

Foerster, J.; Song, F.; Hughes, E.; Burch, N.; Dunning, I.; Whiteson, S.; Botvinick, M.; and Bowling, M. 2019. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 1942–1951.

Ganzfried, S.; and Sandholm, T. 2015. Endgame solving in large imperfect-information games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 37–45. International Foundation for Autonomous Agents and Multiagent Systems.

Gelly, S.; and Silver, D. 2007. Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning*, 273–280.

Gilpin, A.; and Sandholm, T. 2006. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 1007. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, 709–715.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* .

Hoda, S.; Gilpin, A.; Pena, J.; and Sandholm, T. 2010. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research* 35(2): 494–512.

Horák, K.; and Bošanskỳ, B. 2019. Solving partially observable stochastic games with public observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2029–2036.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101(1-2): 99–134.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kovařík, V.; and Lisỳ, V. 2019. Problems with the EFG formalism: a solution attempt using observations. *arXiv preprint arXiv:1906.06291* .

Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisỳ, V. 2019. Rethinking Formal Models of Partially Observable Multiagent Decision Making. *arXiv preprint arXiv:1906.11110* .

Kroer, C.; Farina, G.; and Sandholm, T. 2018. Solving large sequential games with the excessive gap technique. In *Advances in Neural Information Processing Systems*, 864–874.

Kroer, C.; Waugh, K.; Kılınç-Karzan, F.; and Sandholm, T. 2018. Faster algorithms for extensive-form game solving via improved smoothing functions. *Mathematical Programming* 1–33.

Lerer, A.; Hu, H.; Foerster, J.; and Brown, N. 2020. Improving Policies via Search in Cooperative Partially Observable Games. In *AAAI Conference on Artificial Intelligence*.

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

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

Moravcik, M.; Schmid, M.; Ha, K.; Hladik, M.; and Gaukrodger, S. J. 2016. Refining subgames in large imperfect information games. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nash, J. 1951. Non-cooperative games. *Annals of mathematics* 286–295.

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

Newman, A. J.; Richardson, C. L.; Kain, S. M.; Stankiewicz, P. G.; Guseman, P. R.; Schreurs, B. A.; and Dunne, J. A. 2016. Reconnaissance blind multi-chess: an experimentation platform for ISR sensor fusion and resource management. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXV*, volume 9842, 984209. International Society for Optics and Photonics.

Oliehoek, F. A. 2013. Sufficient plan-time statistics for decentralized POMDPs. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 8026–8037.

Rockafellar, R. T. 1970. *Convex analysis*. 28. Princeton university press.

Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development* 3(3): 210–229.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2019. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265* .

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*. URL http://arxiv.org/abs/1506.02438.

Seitz, D.; Kovarík, V.; Lisỳ, V.; Rudolf, J.; Sun, S.; and Ha, K. 2019. Value Functions for Depth-Limited Solving in Imperfect-Information Games beyond Poker. *arXiv preprint arXiv:1906.06412* .

Serrino, J.; Kleiman-Weiner, M.; Parkes, D. C.; and Tenenbaum, J. 2019. Finding Friend and Foe in Multi-Agent Games. In *Advances in Neural Information Processing Systems*, 1249–1259.

Shannon, C. E. 1950. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41(314): 256–275.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362(6419): 1140–1144.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354.

Šustr, M.; Kovařík, V.; and Lisỳ, V. 2019. Monte carlo continual resolving for online strategy computation in imperfect information games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 224–232. International Foundation for Autonomous Agents and Multiagent Systems.

Tesauro, G. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* 6(2): 215–219.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.