**Libratus: The Superhuman AI for No-Limit Poker**  
(Demonstration)

Noam Brown  
Computer Science Department  
Carnegie Mellon University  
noamb@cs.cmu.edu  

Tuomas Sandholm  
Computer Science Department  
Carnegie Mellon University  
Strategic Machine, Inc.  
sandholm@cs.cmu.edu

**Abstract**

No-limit Texas Hold' em is the most popular variant of poker in the world. Heads-up no-limit Texas Hold’em is the main benchmark challenge for AI in imperfect-information games. We present Libratus, the first—and so far only—AI to defeat top human professionals in that game. Libratus’s architecture features three main modules, each of which has new algorithms: pre-computing a solution to an abstraction of the game which provides a high-level blueprint for the strategy of the AI, a new nested subgame-solving algorithm which repeatedly calculates a more detailed strategy as play progresses, and a self-improving module which augments the pre-computed blueprint over time.

**1 Introduction**

Recreational games have long been used in AI as benchmarks to evaluate the progress of the field. AIs have beaten top humans in chess [Campbell et al., 2002] and Go [Silver et al., 2016]. Checkers was even completely solved [Schaef-fer et al., 2007]. However, these are perfect-information games: both players know the exact state of the game at every point. In contrast, poker is an imperfect-information game: part of the state is hidden from a player because the other player has private information. Many real-world applications can be modeled as imperfect-information games, such as negotiations, business strategy, security interactions, and auctions. Indeed, imperfect information is common in the real world, while perfect information is rare, making imperfect-information games particularly suitable for modeling real-world strategic interactions. Dealing with hidden information requires drastically different AI techniques. Heads-up no-limit Texas Hold’em has long been the primary benchmark challenge for imperfect-information games.

In January 2017 Libratus beat a team of four top-10 heads-up no-limit specialist professionals in a 120,000-hand Brains vs. AI challenge match over 20 days. This was the first time an AI had beaten top human players in this game. Libratus beat the humans by a large margin (147 mbb/hand), with 99.98% statistical significance. It also beat each of the humans individually.

**2 Architecture of Libratus**

Libratus’s strategy was not programmed in, but rather generated algorithmically. The algorithms are domain-independent and have applicability to a variety of imperfect-information games. Libratus features three main modules, and is powered by new algorithms in each of the three:

1. Computing approximate Nash equilibrium strategies before the event.
2. Subgame solving during play.
3. Improving Libratus’s own strategy to play even closer to equilibrium based on what holes the opponents have been able to identify and exploit.

The next three subsections discuss these, respectively.

**2.1 Abstraction and Equilibrium Finding**

It is infeasible to pre-compute a strategy for each of the $10^{161}$ different decision points in heads-up no-limit Texas hold’em.\(^1\)\(^2\) However, many situations are strategically similar and can be treated identically at only a small cost. For example, there is little difference between a bet of $500 and a bet of $501. Rather than come up with a unique strategy for both of those situations, it is standard to group them together and treat them identically, so that only one strategy is generated for them. There are two kinds of such abstraction: *action abstraction* and *card abstraction*.

In action abstraction, only a few of the nearly 20,000 possible actions available at any point in the game are included in the abstraction for both the agent and the opponent. If, during actual play, the opponent chooses an action that is not in the abstraction, then it is standard to map that action to a nearby action that is in the abstraction. The actions that we included in the abstraction were determined by analyzing the most common actions taken by prior top AIs in the *Annual Computer Poker Competition (ACPC)*. For the first few actions of the game, the actions to include in the abstraction (i.e., bet sizes) were determined by a parameter optimization algorithm which converged to a locally optimal set of bet sizes [Brown and Sandholm, 2014].

\(^1\)The standard version of the game has $10^{161}$ because both players have $20,000, and are limited to dollar-increment bets.

\(^2\)Heads-up limit Texas Hold’em, a significantly smaller game with $10^{13}$ decision points, was essentially solved in 2015 [Bowling et al., 2015; Tammelin et al., 2015].
In card abstraction, similar poker hands are bucketed together and treated identically. Libratus does not use any card abstraction on the first (preflop) and second (flop) betting rounds. The last two betting rounds, which are exponentially larger, are more coarsely abstracted. The 55 million different hand possibilities on the third round are grouped into 2.5 million buckets, and the 2.4 billion different possibilities on the fourth round are grouped into 1.25 million buckets. The idea is that solving this abstraction gives a detailed strategy for the first two betting rounds and a blueprint for the remaining two betting rounds; the subgame solver, discussed in the next subsection, will then refine the blueprint into a detailed strategy. The card abstraction algorithm was similar to that used in Baby Tartanian8 [Brown and Sandholm, 2016a] (the winner of the 2016 ACPC) and Tartanian7 [Brown et al., 2015] (the winner of the 2014 ACPC). The abstraction algorithm took the game size from $10^{161}$ decision points down to $10^{12}$.

We solved the abstract game via a distributed version of an improvement over Monte Carlo Counterfactual Regret Minimization (MCCFR) [Zinkevich et al., 2007; Lanctot et al., 2009; Brown et al., 2015]. MCCFR is an iterative algorithm which independently minimizes regret at every decision point. If both players play according to MCCFR in a two-player zero-sum game, then their average strategies provably converge to a Nash equilibrium. Libratus improves over vanilla MCCFR through a sampled form of Regret-Based Pruning (RBP) [Brown and Sandholm, 2015] (which we also used in our Baby Tartanian8 agent [Brown and Sandholm, 2016a]). At a high level, our improvement is that paths in the tree that have very negative regret (for the player that is being updated on the current iteration) are visited less often. This leads to a significant speed improvement, thereby enabling a large fine-grained abstraction to be solved. It also mitigates the downsides of imperfect-recall abstraction (which is the state of the art) because the effective in-degree of abstract states decreases as some paths to them get de-emphasized.

2.2 Nested Safe Subgame Solving

The second module solves a finer-grained abstraction of the remaining game, taking into account the blueprint of the strategy for the entire game, when the third round is reached. Unlike perfect-information games, an imperfect-information subgame cannot be solved in isolation. The Nash equilibrium strategy in other subgames affects the optimal strategy in the subgame that is reached during play. Nevertheless, we can approximate a good strategy in the subgame that is reached if we have a good estimate of the value of reaching the subgame in an equilibrium. The first module estimated this value for every subgame. Using these subgame values as input, subgame solving creates and solves a finer-grained abstraction in the subgame that is reached.

This finer-grained abstraction does not use any card abstraction and uses a dense action abstraction.

Also, rather than apply action translation as was done on the first two rounds, and as has been done in prior poker AIs, Libratus instead constructs and solves a new subgame every time an opponent chooses an action that is not in the finer-grained abstraction (in practice, it constructs a new subgame every time the opponent bets). This allows it to avoid the rounding error due to action translation and leads to much lower exploitability [Brown and Sandholm, 2017b].

Another novel aspect of the subgame solver is that it guarantees that the solution is no worse than the precomputed equilibrium approximation, taking into account the magnitude of opponent’s mistakes in the hand so far to enlarge the strategy polytope that can be safely optimized over. This leads to better strategies [Brown and Sandholm, 2017b] than prior subgame-solving techniques [Ganzfried and Sandholm, 2015; Burch et al., 2014; Jackson, 2014; Moravcik et al., 2016].

A further novel aspect is that Libratus changes its action abstraction in each subgame. Thus the opponents must adapt to new bet sizes in each subgame.

2.3 Self-Improvement

As described in Section 2.1, Libratus uses a dense action abstraction on the first two betting rounds. If an opponent does not bet an amount that is in the abstraction, the bet is rounded to a nearby size that is in the abstraction. This causes the AI’s strategy and estimates of the values of reaching subgames, to be slightly off. To improve upon this, in the background, the AI determined a small number of actions to add to the abstraction that would reduce this rounding error as much as possible. The choice of actions was based on a combination of which actions the opponents were choosing most frequently and how far those actions were from their nearest actions in the abstraction. Once an action was selected, a strategy was calculated for it in a similar manner to subgame solving, described in Section 2.2. From that point on, if that action (or a nearby one) were chosen by an opponent, then the newly solved subgame strategy would be used.

3 Agent Construction

In total, Libratus used about 25 million core hours. Of those, about 13 million core hours were used for exploratory experiments and evaluation. About 6 million core hours were spent on the initial abstraction and equilibrium finding component, another 3 million were used for nested subgame solving, and about 3 million were used on the self-improvement algorithm.

The equilibrium finding and self-improvement algorithms used 196 nodes on the Bridges supercomputer at the Pittsburgh Supercomputing Center. Each node has 128 GB of memory and 28 cores, but only 14 cores are used by the agent. An asymmetric abstraction was used that had more actions for the opponent, to better reduce the error resulting from action translation [Bard et al., 2014].

The subgame solver used 50 nodes per game. Here we used CFR+ [Tammelin et al., 2015] combined with specialized optimizations [Johanson et al., 2011] for equilibrium finding. Recent work suggests that subgame solving could be even faster by leveraging warm starting [Brown and Sandholm, 2016b], new pruning techniques [Brown and Sandholm, 2017a], or first-order methods [Nesterov, 2005; Kroer et al., 2017].
Acknowledgments
The research was supported by NSF grant 1617590, ARO award W911NF-17-1-0082, and XSEDE computing resources by the Pittsburgh Supercomputing Center. The Brain s vs. AI match was sponsored by CMU, Rivers Casino, GreatPoint Ventures, Avenue4Analytics, TNG Technology Consulting, Artificial Intelligence, Intel, and Optimized Markets. Ben Clayman computed statistics of the play of our AIs.

References


