Libratus — Superhuman AI for Heads-Up No-Limit Poker

1. Introduction to HUNL Poker

Heads-Up No-Limit Texas Hold’em (HUNL) is one of the most strategically complex games in existence. It is a two-player variant of Texas Hold’em poker with the following characteristics:

- Each player receives two private “hole” cards.
- Five community cards are revealed in three stages:
  - **Flop**: 3 cards
  - **Turn**: 1 card
  - **River**: 1 card
- Players build the best 5-card poker hand using any combination of their private and community cards.
- Players can bet any amount at any time — hence the term “No-Limit.”
- A player wins either by holding the best hand at showdown or by inducing the opponent to fold.

Each hand consists of four betting rounds:

1. **Pre-Flop**
2. **Flop**
3. **Turn**
4. **River**

In each round, players can fold, check, call, bet, or raise. The combination of imperfect information and strategic freedom leads to immense decision complexity — over 10^160 decision points, more than that of chess or Go.

2. Motivation and Challenge

AI has already surpassed human performance in perfect-information games like chess and Go. However, in games like poker, hidden information and strategic deception create unique challenges:

- AI must reason not just about outcomes, but also about the opponent’s beliefs.
- Bluffing, balancing, and probabilistic reasoning are essential.
- An optimal move depends not only on the current state but also on hypothetical states that haven’t yet occurred.

Poker, particularly HUNL, became the benchmark problem for imperfect-information AI. No AI had previously defeated elite human professionals in this game — until Libratus :)

3. Overview of Libratus

Libratus is the first AI system to defeat top human professionals in HUNL. It even does not use human gameplay data, poker-specific heuristics, or handcrafted rules.

The system contains three main modules:

1. **Blueprint Strategy**

2. **Safe Subgame Solving**
3. **Self-Improver**

4. Blueprint Strategy

The blueprint strategy is basically based on an abstraction of the game, which is smaller and easier to solve, and then computes game-theoretic strategies for the abstraction.

4.1 Game Abstraction

To reduce the enormous state space of HUNL, Libratus constructs two types of abstractions:

- **Action Abstraction**: Only a limited set of bet sizes (e.g., $100, $200, pot-sized bets) are considered, rather than every dollar amount. (Use roundings)
- **Card Abstraction**: Similar private hands are grouped together into “buckets” to reduce complexity.

For example, 55 million possible hands in the turn round are grouped into 2.5 million buckets.

4.2 Solving with MCCFR

Once the abstraction was constructed, we computed the blueprint strategy for Libratus by using an algorithm called Monte Carlo Counterfactual Regret Minimization (MCCFR):

- It simulates millions of games between copies of itself.
- Regret values are updated for each decision to indicate how much better other choices would have been. When a decision point is encountered during self play, the AI chooses actions with higher regret with higher probability
- Over time, it converges to a low-exploitability strategy. (with high probability a player’s average regret for any action approaches zero)

An improved version of MCCFR skips clearly suboptimal actions and focuses computation on relevant branches, achieving a 3x speedup in practice.

4.3 When the Blueprint is Used

- The blueprint strategy is used primarily in the early rounds (Pre-Flop and Flop).
- In later rounds (Turn and River), Libratus switches to real-time subgame solving.

5. Safe Subgame Solving

When the game reaches a later round, or when the opponent takes an off-tree action (i.e., a move not included in the abstraction), Libratus constructs a new subgame in real-time.

5.1 The Challenge

In imperfect-information games, subgames are interdependent (solving a subgame in isolation can lead to exploitation).

5.2 Augmented Subgame Construction

To safely solve subgames, Libratus used an “augmented subgame” technique:

- The opponent is offered a choice between:
  1. Entering the subgame, or
  2. Taking an alternative payoff equal to the expected value in the blueprint strategy.
- Libratus finds a strategy such that, for all opponent hands, the opponent cannot do better by choosing either option.

This ensures the subgame solution is no worse than the blueprint, which is basically the essence of “safe” subgame solving :)
(But! All of this relied on the assumption that we have accurate estimates of player’s values against optimal strategy)

6. Estimated-Maxmargin Technique

Libratus then further improves safety using a novel method called **Estimated-Maxmargin**.
For each possible opponent hand h, define:
\text{Margin}(h) = \text{Blueprint EV}(h) - \text{Subgame EV}(h)
The algorithm finds a strategy that maximizes the minimum of all margins — ensuring no hand benefits significantly from the new strategy.

Theoretical Guarantee:

If the estimated values are within±Δ \pm \Delta of the true values, then the overall increase in exploitability is at most
2Δ (2\Delta).
This technique achieves practical strength while maintaining provable safety.
Question: Why?
Although we describe safe subgame solving as using estimates of Player1 values, past techniques used upper bounds. Using upper bounds guarantees that the subgame solution has exploitability no higher than the blueprint strategy. However, it tends to lead to overly conservative strategies in practice. Using estimates can, in theory, result in strategies with higher exploitability than the blueprint strategy, and this 2*delta bounds how much higher this exploitability can be.

7. Nested Subgame Solving

When an opponent takes an off-tree action (e.g., bets $101 instead of $100), Libratus does not round the action. Instead:

- It creates and solves a new subgame including that exact action.
- This is done recursively — each subsequent off-tree action results in a new nested subgame.
- ***Question: Why?***
- e.g. $150, then it is likely significantly different from the response to a bet of $100 or a bet of $200. In principle one could simply increase the number of actions in the abstraction, perhaps by considering bets in increments of $10 rather than $100, so that the error from rounding is smaller. However, the size of the abstraction, and the time needed to solve it, increases prohibitively as more actions are added.

-> This real-time response ensures that Libratus remains robust to any betting pattern.

8. Self-Improver

Libratus includes a background module that refines the blueprint strategy over time.

- After each day of play, it analyzes which off-tree actions the opponents frequently used.
- It then runs computations overnight to solve those specific branches and adds them to the blueprint.
- These improvements are general-purpose, (not opponent-specific), so the strategy remains safe and broadly applicable, avoiding the overfitting problem.

This mechanism leverages human creativity to find weaknesses, which Libratus systematically patches.

9. Experimental Results vs Top Human Professionals

- Libratus played 120,000 hands over 20 days against four elite professionals.
- Final result: **+147 milli-big blinds per hand (mbb/hand)** in favor of Libratus.
- Statistical significance: **p = 0.0002**

vs Prior Top AI (Baby Tartanian8)

- Libratus defeated Baby Tartanian8 by **+63 mbb/hand**, showing clear superiority.
- Baby Tartanian8 had previously won the Annual Computer Poker Competition.

These results confirm that Libratus achieved superhuman performance in the most complex form of poker ever tackled by AI.

11. Conclusion

Libratus solved one of the most long-standing AI challenges: beating top humans in a massive imperfect-information game.
Its design combines abstraction, theoretical safety, real-time learning, and self-improvement (all without human data).
It demonstrates that AI can now reason strategically, probabilistically, and adaptively in domains with hidden information.