Monte Carlo Methods for Poker

Course Project Report

Ilian Benaïssa-Lejay

April 15, 2025

1 Introduction

Poker involves **incomplete information**, unlike fully observable games like Chess. Here, we use *Monte Carlo* methods to estimate winning probabilities in a simplified two-player Texas Hold'em setup.

Simulations sample possible community cards (and hidden cards if needed) to estimate how often one player's hand beats the other's. The code could be adapted to smaller poker variants like Kuhn or Leduc for faster experimentation.

2 Theoretical Background

2.1 Monte Carlo Simulation

Monte Carlo simulations randomly generate possible outcomes. The fraction of trials where Player 1 wins gives an estimate of their win probability:

$$P(\text{win}) \approx \frac{\#\text{ Player 1 wins}}{\#\text{ simulations}}.$$

2.2 Policy Adaptation

Our implementation tracks which cards contribute to wins, then biases future draws toward such cards. This is a lightweight form of *Playout Policy Adaptation (PPAd)*.

3 Implementation Overview

The program:

- Encodes cards, hands, and ranking logic
- Samples remaining community cards
- Compares 5-card hands to determine winners
- Repeats simulations to estimate win rates
- Optionally biases card draws via adaptation
- Includes a basic GUI and produces `matplotlib` plots

Listing 1: Core Monte Carlo simulation

```
def monte_carlo_simulation(h1, h2,
    known_community, iterations=1000):
    results = [0, 0, 0]
    for _ in range(iterations):
        deck = generate_deck(h1 + h2 +
            known_community)
        drawn = random.sample(deck, 5 - len(
            known_community))
        full_community = known_community +
            drawn
        best1 = best_hand(h1, full_community)
        best2 = best_hand(h2, full_community)

        if best1 > best2: results[0] += 1
        elif best2 > best1: results[1] += 1
        else: results[2] += 1
    return results
```

4 Experimental Results

Typical settings:

- `iterations` = 5000+
- `step` = 100
- `alpha` in $[0.0, 1.0]$

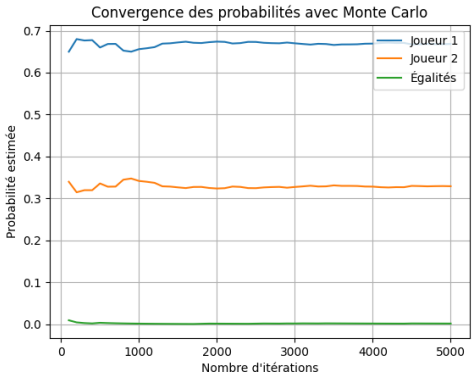

Figure 1: Convergence of win probabilities.

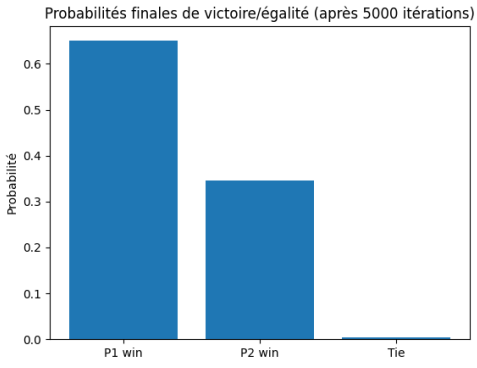

Figure 2: Final win rates.

5 Discussion

5.1 Observations

- Results stabilize after a few thousand simulations
- Adaptation increases win rates for the leading player
- More complex strategies would require CFR or IS-MCTS

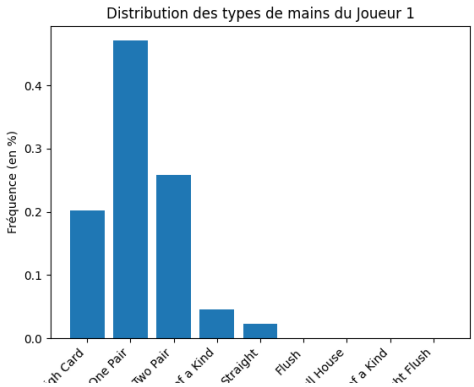

Figure 3: Player 1 hand type distribution.

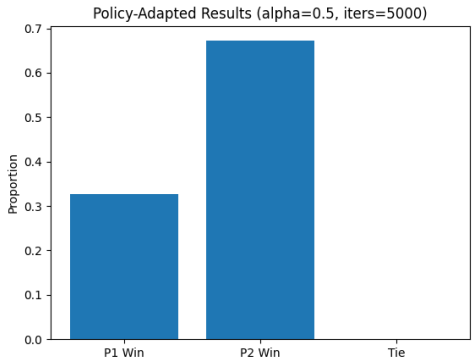

Figure 4: Win rates after adaptation ($\alpha = 0.5$).

5.2 Limitations

- Only end-of-round outcomes considered
- No betting, bluffing, or opponent modeling
- Simple biasing — not optimal play

6 Conclusion

We used Monte Carlo simulations to approximate poker win rates and explored a basic policy adaptation. Future work could include:

- Using Information Set MCTS
- Modeling betting rounds and multiple players
- Integrating learning-based evaluation