A.I. Heads-up No Limit Texas Hold'em Poker Tournament Simulation

Ian Blanchard
Department of Computer Science
Kennesaw State University
Marietta, United States
iblanch1@students.kennesaw.edu

***Abstract*—This document is a proposition for building a simulation comparing Artificial Intelligences based on Loki and Cepheus, two AIs developed by two different teams at the University of Alberta. The AIs shall also compete against an AI that operates on Random chance, and another AI that always goes all in.**

I. Project Overview

Poker is a game that is both based on luck and skill. Unlike Chess, where all strategies and possibilities can be seen on the board, Poker is nondeterministic. This simulation lies within the A.I domain. CFR+, despite taking up less space than CFR, still takes up too much storage for the average person to compute/use. Can an bot based on Cepheus that hasn't completed the necessary compute to be comparable to nearly perfect play still compete against lesser A.I bots?

This project aims to simulate Heads-up No Limit Texas Hold'em Poker games between multiple A.Is: an incomplete A.I based on Cepheus, two A.Is based on Loki, one simple A.I that operates on random chance, and another simple A.I that always goes all in. Note: Cepheus and Loki were both developed by the University of Alberta's Poker research group. They don't use Reinforcement-learning or machine learning like most A.Is in the modern day and are greatly effective. In addition, this simulation shall also generate an output recording the hand initially dealt, the winner, and the winner's winnings. A G.U.I could be in scope for picking and choosing the A.Is in the game, but it depends on how smoothly development goes. If it is discovered that even a scaled-down version of Cepheus can't run on my hardware, then querying the original Cepheus will be suggested and investigated. Nothing else is in scope.

I will be calling my agent based on CFR+: Cepheus Lite because since it will lack a substantial amount of training data. I shall call my agent based on Loki: Loki-Lite to follow the same naming scheme.

II. System Description

A. System Components

Each type of Agent's code will be stored in their own file. In addition, a database of some kind will have do be constructed to store Cepheus Lite's regret and other data. Any additional libraries will also be stored in their own file or called.

B. System Dynamics

The main driver program will be responsible for setting up the deck and players. Functions calling other agents will send their current deck and community cards.

C. Core Models and Algorithms:

All programs will have to use a data structure of non-fixed size to hold each type of card. Currently, the plan is to store each type of card as a byte to minimize space. When a call is made, Henry R. Lee's poker evaluator will be used to see which player has the best hand.

D. Language

I have decided upon C++ due to the fact that HenryRLee's poker evaluator is written in that language and the fact that performance may be a crucial enough factor that only a fully compiled language (unlike Java and C# which uses JIT compilation) can fulfill the task efficiently. In addition, if we have other libraries or frameworks which use different languages, I can use GCC to combine the object files at the linking stage of compilation. In addition, C++ has a library allowing me to use something to fill the purpose of a byte literal without having to cast every time.

E. Libraries and Tools

HenryRLee's Poker Evaluator is a fast way to evaluate which player has a better hand. I will use the default SQLite C++ library for data collection. For programming I shall use Visual Studio using GCC for compilation.

F. Simulation Type

This is an agent-based, discrete-event simulation that goes turn-by turn in one or more Heads-up No Limit Texas Hold'em tournaments.

G. Data Collection Plan

The deck's card order before shuffling, the victor(s), and the winning hand(s). Data will be sent to a local SQLite database following each round.

III. Literature Review

Theory of Games and Economic Behavior (by John von Neumann and Oskar Morgenstern) is a foundational piece to the entire study of game theory. It introduced the Minimax theorem, which recommends that a player choose a strategy that minimizes the maximum possible loss against an adversary, which regret-matching and CFR are based on.

CFR+ is the core component of Cepheus. It was built off of Counterfactual Regret Minimization (CFR), which was built off of Regret-matching. Regret-matching is an algorithm that measures Regret, a measurement comparing the strategy chosen to some alternative strategy in hindsight. It stores regrets for each action. What we have below, $R^T(a)$ represents the external regret for all timestamps given action a.

$$\lim_{T\to\infty} \frac{R^T}{T} = 0$$

Regret-matching is effective for finding a solution that minimizes regret because the line converges and approaches the Nash Equilibrium. However, the exponential size in representation in regret for games with a large amount of possibilities like poker make it impractical.

CFR is an approach to solve regret-matching's fatal flaw by decomposing overall regret at each information set. The University of Alberta's Computer Poker Research Group took this algorithm a step further in order to fit the regret values and strategy (and not have to spend centuries training it), by creating CFR+.

CFR+ uses a weighted average strategy and no sampling techniques, unlike its predecessor, CFR. CFR+ also does alternating regret updates for both players, also unlike it's predecessor. CFR+ uses regret-matching+ in place of regret-matching. Regret-matching+ does not store regrets, but instead tracks another value that acts like regret. CFR+ uses a massive recursive procedure that uses the current known information to run subgame probabilities on other threads. The model is "trained" by running it against itself attempting to minimize exploitability over many iterations.

Loki consists of a hand evaluator and a betting strategy. Loki calculates its current hand strength, potential for hand to improve, effective hand strength, and pot odds. In addition, it may compute weights or modify those weights to evaluate an opponent's "strength." This second part of Loki can vary depending on the "personality" that the designers want, and the paper on it modifies this aspect of it to varying results.

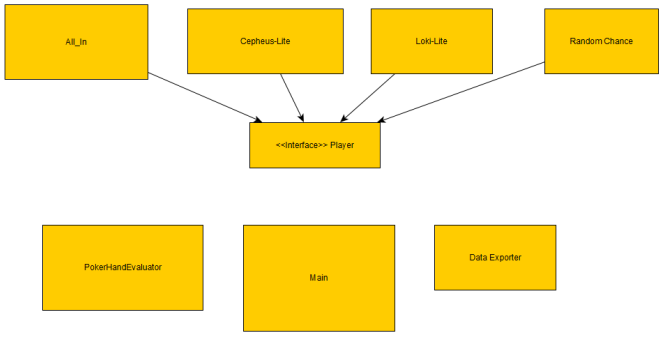

Fig. 1. Class Structure

Fig. 2. Basic Program flow

References

[1] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Opponent Modeling in Poker." Available: https://cdn.aaai.org/AAAI/1998/AAAI98-070.pdf

[2] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information." Accessed: Feb. 07, 2026. [Online]. Available: https://johanson.ca/publications/poker/2007-nips-cfr/2007-nips-cfr.pdf

[3] "Theory Of Games And Economic Behavior : Neumann, john Von. : Free Download, Borrow, and Streaming : Internet Archive," Internet Archive, 2017. https://archive.org/details/in.ernet.dli.2015.215284/page/n5/mode/2up

[4] O. Tammelin, N. Burch, M. Johanson, and M. Bowling, "Solving Heads-up Limit Texas Hold'em." Accessed: Feb. 07, 2026. [Online]. Available: https://poker.cs.ualberta.ca/publications/2015-ijcai-cfrplus.pdf

Additional Note: I saw a Python program on GitHub that claims to be able to access Cepheus via. console, which could open the door for something similar to API calls. Further investigation is needed.