Project 2 Phase 2 Report

Tracy Karol

4/30/2018

Instructor: Mark Baldwin

**Description**

To improve upon the first iteration of my Blockhead Poker program, a more intelligent AI class was created. This Beta AI was added as another player type that can be played either by the user or in a Monte Carlo, where the Beta AI and Alpha AI will play against each other for a number of games (specified by the user). The Beta AI uses a random logic, which keeps the player from being able to predict how it will make its decisions, which is a key feature to having an intelligent AI.

**Implementation**

In order to test whether the Beta AI was smarter than the Alpha, a Monte Carlo simulation needed to be available to conduct experimental games between the two. A feature was added to the user interface that asks the user whether they would like to play a game or conduct a Monte Carlo. When the user selects a game, they can then choose whether they would like to play against the Alpha or Beta AI. Once that is chosen, the game is played as it was in the first implementation. When the user chooses Monte Carlo, they can input the number of games that they want to be ran. After the Monte Carlo ends, the resulting win averages of the alpha and beta are displayed. It is important to set the number of games to be over 100 to get accurate winning averages.

To create the Beta AI, I used the Alpha AI’s logic as a starting point. I copied and pasted the code already written for the last deliverable and made some modifications, which made the Beta perform significantly better. I decided to change some of the bet values that were chosen by the Alpha and I increased any bet of five to eight and increased some bets with a value of one to three. I also used a random number function that was implemented as a helper function to randomly choose a value for the “potFactor”. This impacted any decision made based off of this condition, so this made the Beta’s logic contain an element of randomness.

**Challenges/Interesting Solutions**

The solution that surprised me the most was how much of a difference adding the random element made to how well the AI performed. Before implementing the randomized pot factor, the AI was winning only about 55% of the time, but after this addition, the percent raised to consistently over 70%.

**Testing**

Test 1: When user selects to play a game, the correct path will be taken to start a game involving an AI and the user.

Expected: The user will be asked which type of AI they would like to play against and the game proceeds with the appropriate options

Actual: Got the expected result.

Test 2: When user selects to run a Monte Carlo simulation, the simulation begins.

Expected: The user chooses how many games they should play and the win averages will be displayed at the end.

Actual: Got expected result.

Test 3: Run the Monte Carlo simulation for 100 games.

Expected: Beta’s winning average will be higher than Alpha’s

Actual: Alpha- 17%

Beta- 83%

Test 4: Run the Monte Carlo simulation for 500 games.

Expected: Beta’s winning average will be higher than Alpha’s

Actual: Alpha- 26.2%

Beta- 73.8%