Take5 Assignment Report

The Take5 assignment, asks to create a game AI for a card game, which has a higher complexity than the NoThanks assignment, given the increase of actions that may be taken. In light of this complexity, I stayed with the notion of creating an AI that plays the game in a similar way than I myself would play. To this end I created different implementations ("Close", "Increasing",…), in the end I chose the AI, playing as closely to my own take. Some interesting notions came up during testing which will be noted a bit later in the report.

The AI plays as follows:

- Choose the lowest card that is still playable while the rows have a size smaller than 5
- For the row, pick the one with the smallest penalty

The penalty implementation is the simplest to choose, given that no other strategy would come to my mind that would need you to explicitly choose a worse penalty.

The card choice resulted from testing different strategies. "close" would choose cards which have close cards on the board, thus if the 33 was on the board and I had a 35 that card would be chosen. This strategy performed miserably. I played around with some mixes of the beginner AI's yet found no stable well performing mix. "Increasing" is the name chosen for my final implementation, we increase from low to high to maintain higher cards for the end game and thus hopefully minimize the penalty.

An interesting notion that arises during testing was the fact of choosing a cut of for row size. When telling the AI to choose an increasing card when the row is smaller than 5, it would perform worse than telling it to choose an increasing card only if the row is smaller than 4. This happens given that my strategy does not outperform the HighToLow strategy already implemented. I played around with some variations of the beginner AI's but in the end HighToLow outperforms everyone, thus the diminishing row size increases the performance of my AI because the implementation becomes closer to the HIghToLow implementation. I will stick with my strategy given a high impact of luck in the HighToLow implementation. The latter will perform great in this scenario, yet given that the competition for my AI is set with other student implementations. My strategy might perform well against said implementations given the higher complexity of their strategies.

A main issue concerning this game is the random luck factor. It all depends on the hand that you were dealt and which card other players play. This makes stable strategies rather difficult, there would be more complex algorithms that one could apply to over engineer the problem yet one thing has to stay while creating these game AI's and that is the aspect of fun. Thus my final choice stays with this rather simple yet well thought out strategy.