Hearthstone AI Competition

Project report

Mohammed Farhaan Shaikh

Matrikel-Nr. 226066

Hearthstone is a Card game and as most of the card games out there, this game doesn't only depend on a plane strategy but also on probability of the good cards you get during your play. Therefore, to get a good result from your Bot, you also have to consider the fact that sometimes you might not get the best set of cards for every game and each game might not be in your favour. Apart from the random distribution of cards, there are other factors to consider like the there are a lot of types of cards like minion cards, weapon cards, spell cards and each cards have their own properties. Each hero also has their own powers and every power is unique. Every move you make in each turn has a unique state and this state and this state should be best made in your own advantage. Therefore, to efficiently beat another player in this game, you should consider all these factors when programming your own bot.

The bot I developed uses a greedy approach and for each different hero class it has its own set of scores. This scoring function considers of a lot of factors like the hero and opponent hero hp, number of minions on the board, HP and attack of the minions on the board, hero attack, number of cards in the hand etc. This scoring functions returns a unique value for each of the states the moves take it into. Considering this scoring function, we can make moves based on the current state of the board, game. As I'm applying this method to solve the problem, we are considering only the current state. Since we can't look into the opponent hand and simulate a tree which proceeds to the end of a game and checks the end of the game for each current move, this is one of the best approaches to solve the game.

A greedy approach was used to design this bot. I had to decide a scoring function for the bot to beat each of the hero classes. To decide upon these values, I played the game multiple times and then came to a particular strategy for each of the heroes. I added certain values into the scoring function depending on my gameplay experience and let the bot play using these scoring functions. After letting the bots play for a certain number of games, I tweaked the scoring functions to make the bots play better. After multiple iterations I came to a scoring function which beats the opponent greedy bot and gains a win rate of at least 50 percent. For each hero class, there is a specific scoring function and this scoring function is tweaked in a way to beat the greedy bot as best as it can. I had to create a unique scoring function because each hearo had a specific type of deck and a unique hero power.

Other strategic games like chess can only have a certain set of moves for each turn and there is no probability involved like it is in the way a card is selected in the game. We can create a tree which leads to an end of the game easily and decide which move is best suited to take in that current state. The same cannot be said about Hearthstone because firstly we cannot see the hand of the opponent so we can't exactly know which move he might make. To make it worse, the cards are randomly given to the hand and theirs is no way to know which card anyone might get next and there a lot of cards to consider in each case. In order to go about this, I thought that the best way to beat the game is if I could maximise the profit for each move for the current turn. This would consider only the current state and not worry about anything else. Greedy approach was the best one which came to my mind to solve this problem. And it gave interesting results when I played the game against the opponent bot.

My agent uses the given information like the player hp, the opponent hp, number of minions on the board, health of the minions, attack of the minions, hero attack etc to decide each move to make. The score functions consider these factors and decides upon a certain score for each of the given move. The agent also considers the class of the hero and it uses a different scoring function for each of the hero class depending on it. These scores based on the aforementioned factors and calculated for the current opponent hero is returned and the move with the max score is chosen to decide each move in the current state of the game. This move doesn't consider the future states of the game but only the current state of the game and I try to maximise the profit.

There are multiple sets of score functions which this bot uses and each of the hero class has its own scoring function. To decide upon each of the scoring functions, firstly I had to play the game multiple times to understand how everything works and how each of the hero classes and decks affect the game. After that initial step, I decided upon a certain scoring function and let my bot play it multiple times against the opponent greedy bot. The results of this game gave me certain results and I used these results to tweak the scoring function even more to get better results each time until I was happy with my scoring function and its results. The final scoring functions have unique values for each of the factor and the are unique to the player classes.

The basic procedure how my bot makes decisions are as follows. First for each of the available options in the current state of the game, that move is simulated once, and the next state is obtained for each of the moves. After obtaining the states mentioned for each of the moves, we try to get a score for each of the states using a scoring function. This score tells us how good the move is to take for the current state. The bot then takes this move. I have coded the bot in such a way that it only chooses the end turn move only if there are no available moves left in the current turn.

The current agent can be improved by tweaking the scoring function better and applying better values for each of the factors. We can also use an evolutionary algorithm in helping us get a better scoring function and apply this scoring function for the greedy agent. If we know all the cards being played in the game, we can also use a neural network and train it with the factors provided in the game. This might also give interesting results which may be better than the ones I am currently getting with my greedy agent.

Test for 2000 games each as player 1 and player 2

Hero Class	As Player 1	As Player 2	Average
Mage	62%	54%	58%
Warrior	93%	76%	84.5%
Shaman	75%	56%	65.5%