Udacity

Artificial Intelligence Nanodegree

Report for Project 4: Build an Adversarial Game Playing Agent

Prepared by: Sascha Metzger

1. June 2020

EXPERIMENTS

My experiments involved implementing a baseline agent and an additional heuristic and having them play each other for 50 fair games. A fair game is a set of two games where players swap initial positions to mitigate any advantages of a particular starting position.

Baseline Player

The baseline player is the minimax player from the sample_players.py file and improved by a alpha beta pruning algorithm. It uses the my_moves - opponent_moves heuristic.

Custom Player

There are two customer player impelmentations:

1. The first one maximises its own number of moves and follows a very defensive strategy.

2. The second one is a more aggressive player and tries to minimise the moves the opponent has left.

RESULTS

Baseline Agent

Opponent	% Wins baseline agent
GREEDY Agent	10,0 %
MINIMAX Agent	29,0 %
DEFENSE Agent	45,0 %
OFFENSE Agent	57,0 %
Mean	**35,3 %**

Defensive Agent

Opponent	% Wins baseline agent
GREEDY Agent	7,0 %
MINIMAX Agent	44,0 %
OFFENSE Agent	66,0 %
BASELINE Agent	69,0 %
Mean	**46,5 %**

Offense Agent

Opponent	% Wins baseline agent
GREEDY Agent	11,0 %
MINIMAX Agent	21,0 %
DEFENSE Agent	35,0 %
BASELINE Agent	43,0 %
Mean	**27,5 %**

DISCUSSION

What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?

One of my custom heuristic is purely based on defence as it tries to maximise the remaining number of moves. It matters because the more moves the player has available, the higher its chance of winning is as the game progresses.
The other heuristic tires to minimise the number of moves the opponent has left. If the number of moves your opponent has left is smaller than the number of moves you have left, your chance of winning is higher.

It is interesting to see that the more offensive player is the worst player, even the baseline agent has more wins.

Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?

Because my players only focuses on maximising its own number of moves or decreasing the number of moves the opponent has left, I think speed is not that critical for the performance of my player.