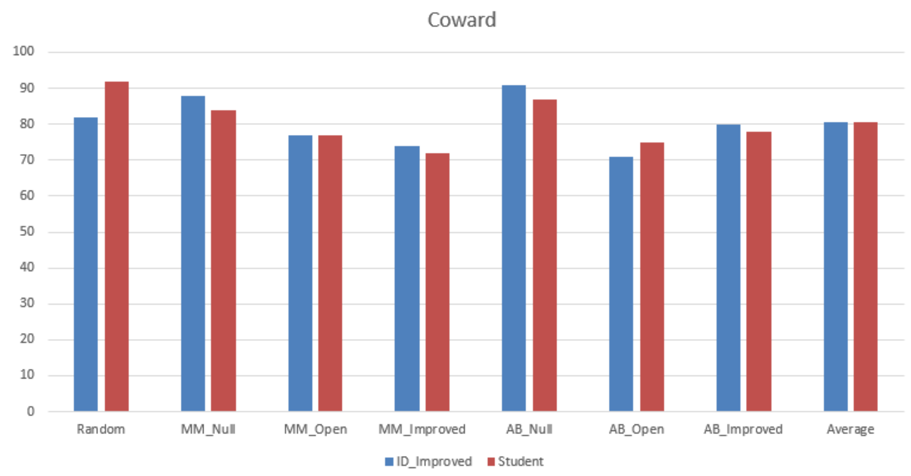
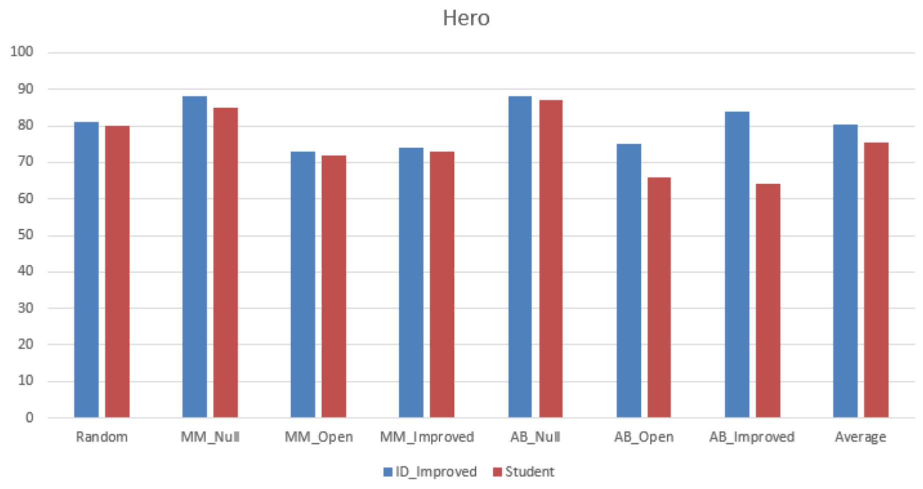
I first wanted to test some simple heuristics unrelated to the number of possible moves in order to see which one could compete the best with the heuristic that takes the difference in the number of our and our opponent's moves. The idea was that I could then combine my best heuristic and the ID_Improved one to get a heuristic that outperforms both. One of the reasons why I wanted to always use the ID_Improved heuristic is that I believe that the number of possible moves (both ours and our opponent's) should always play a part in our evaluation of the game state. All the following results are gained from running the tournament.py script with the number of matches set to 25.

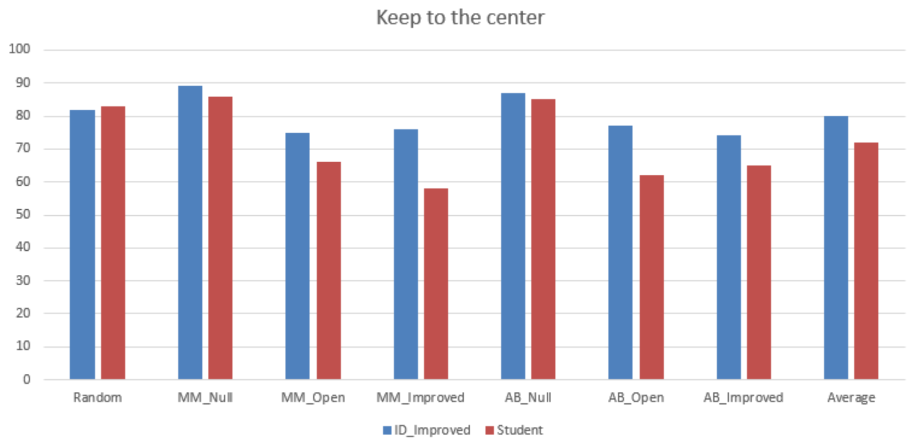
The first heuristic I tested was what I named the "Coward" heuristic, as it tries to maximise the distance between us and the opponent. It performed at a level comparable to the ID_Improved heuristic, which was a lot better than my initial expectation.



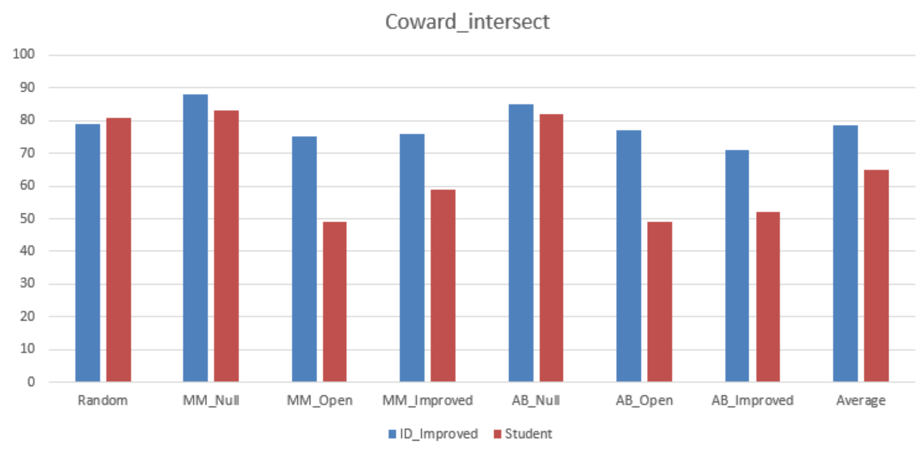
The logical next heuristic to test was the "Hero". It tries to minimize the distance between the agents. It performed worse than the coward heuristic.



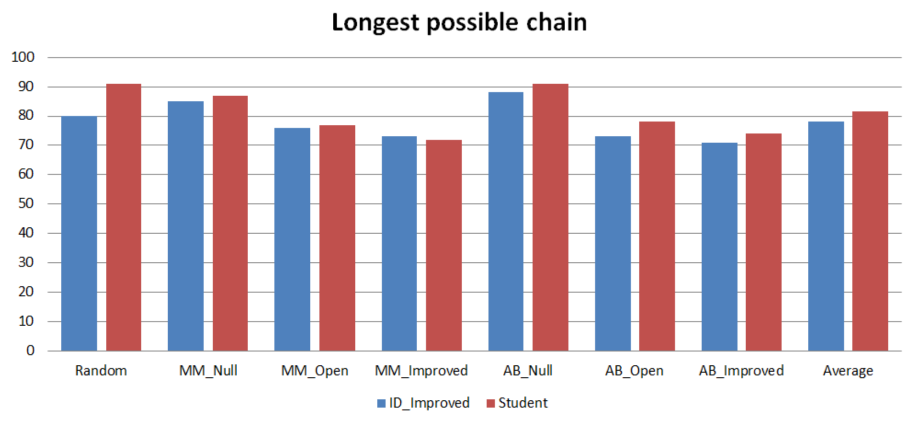
The next heuristic was ”Keep to the center” where we try to maximise the distance to the edges of the game board. This heuristic failed to perform on the level of ID_Improved which is not so suprising as this heuristic is not likely to lead to optimal moves at the later stages of the game.



The worst performing heuristic I tried was the ”No intersect” one. The idea was to minimize the number of intersecting possible moves between our agent and the opponent. This could be thought of as the modification of the ”Coward” heuristic as we are also trying to avoid the opponent.



In the end I implemented the "Longest possible chain" heuristic that uses the ID_Improved heuristic until the number of blank spaces on the board falls under 25% of all fields. The heuristic then combines the ID_Improved score with the difference in the length of the longest possible chain of moves on the current state of the game board (not taking into account future moves of the opponent) between our agent and the opponent. This heuristic produced the best results and is the one that my agent is using in the final version as it on average performed slightly better than ID_Improved on the test computer.



I believe there are several reasons why the "Longest possible chain" heuristic performs well. First of all, it is based on the ID_Improved heuristic which already gives a very good evaluation of the state of the game board. But as the game is reaching the end and the number of blank spaces gets lower, we are able to get additional valuable information about the game board. In my opinion, the length of the longest chain of possible moves provides an excellent prediction of the final outcome of the game as it tells us how far we are from defeat (in ideal case scenario where the opponent doesn't break our longest chain of moves). As we are looking at the difference in the lengths of the longest chains for our agent and our opponent, we actually see which agent is further away from defeat which enables us to make better decisions towards the end of the game. Since the number of blank fields is small (less then a quarter of all fields), the added computational complexity is not affecting our search depth in a way that would deteriorate the performance of our agent. The difference in the number of available moves is still important as it prevents us from favoring game board states where we have a long possible chain of moves, but only one available move which could allow our opponent to easily win the game by occupying that field.