# Heuristics Analysis

Raw tournament data has been written to the following files:
tournament_results_(1-6).txt
The tournament was played a total of six times with the heuristics below

## Custom_score():

Firstly I implemented a heuristic based simply on no. of moves available to the agent and the opponent but also the number of empty spaces available on the board.
The basis of this evaluation was to score the no. of moves available to both players against the number of moves currently available to the board. I felt this was a more sturdy approach to evaluating the each different board state because not only does it maximise and minimise the agent's score but it also its does so in a way that takes into account the remaining spaces left. This allows the agent to effectively keep track of the score of the given board's state as the number of spaces decline after each move. In doing so the agent will look for moves that maximise its area.
The function custom_score() implements this concept. Testing with this function returned win rates of 60%+, the highest being 71.4% and lowest being 60.7% , averaging 64.8% - this could be due to hardware though. This heuristic could be improved by taking into account the distance of the player in relation to the center. I feel doing this could potentially strengthen the agent because it's not only taking into account the amount of spaces remaining on the board in relation to the distance from the center to the player.

Notably this heuristic performed poorly against the agent AB_Improved as shown by the data represented in the bar graph Figure 1. With a Win/Loss ratio of 0.6, this heuristic is certainly not be as effective at evaluating the different board states as the heuristic used for AB_Improved. This could be due to hardware in that the agent has to search deeper in order to evaluate good board states or simply that AB_Improved is a better heuristic.

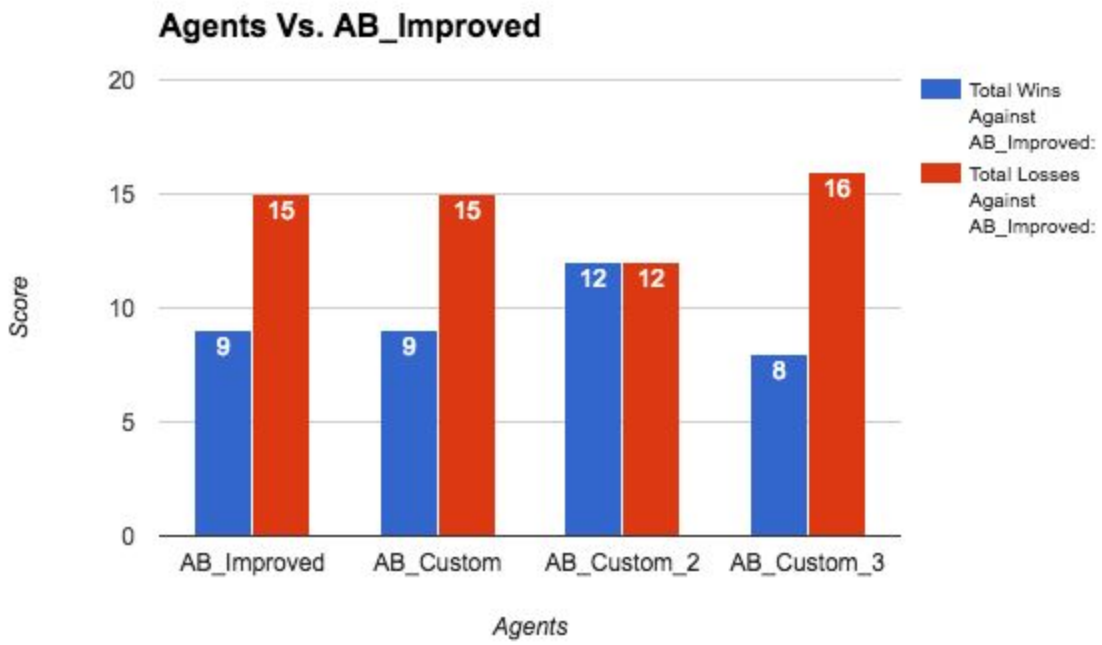


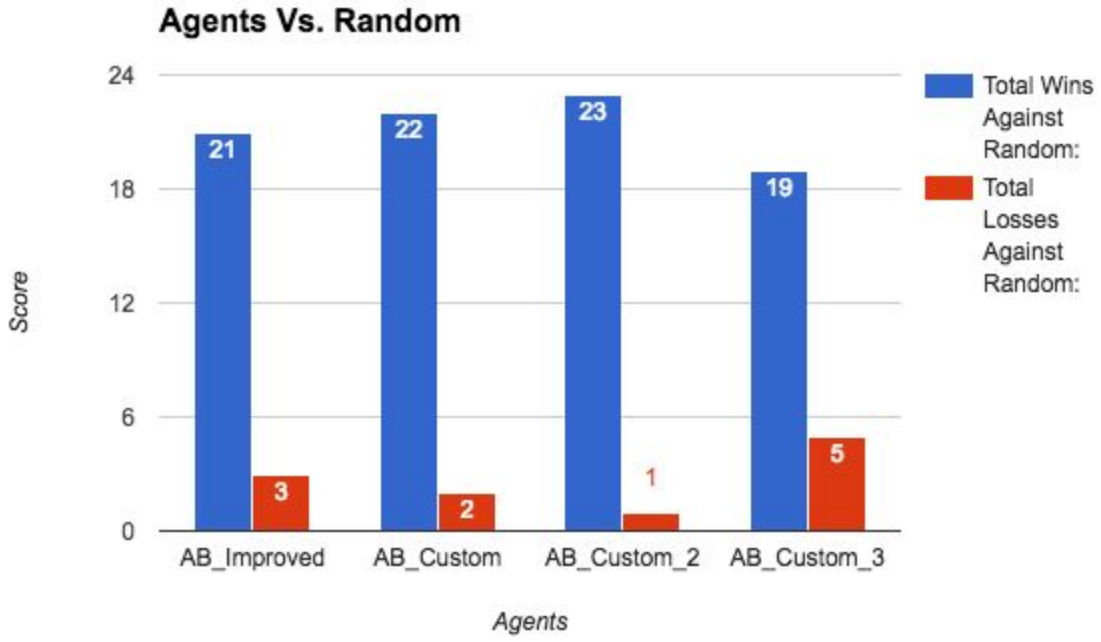

Figure 1.
Showing the total outcome of the matches played against the AB_Improved agent

## Custom_score_2():

I decided to take this concept of calculating the longest path from the center in relation to the player's location. Instead of using the agent's longest path the opponent's was used. This was used to minimize the opponent's path from the center by also taking into account the number of moves available to the opponent and scoring them as a whole. Doubling added increased aggression to the opponent in regards to the agent. For the the agent to pick moves correctly it has to be able to take into account it's own scoring based on its moves but also the the evaluation of the opponent's move score(as mentioned above). This is decent enough for evaluation of open board states but not mid-game board states. To solve this issue, the agent has to know the available spaces left on the board in each different state in order to more effectively judge the "goodness" of the board. Using this heuristic led to win rates of 60-70+%, highest being 85.7%, lowest being 64.3% and averaging a win rate of 73.2% - again could be down to hardware.

With data from the results, this heuristic yielded no below zero win/loss ratios against all the other competing agents in the tournament. It won & lost an equal amount of matches against the AB_Improved agent, resulting in a ratio of 1. It was the only heuristic to stand

its own against the AB_Improved agent given the same external variables, ie hardware limitations, time etc. The results justify that evaluating an isolation board with this heuristic not only increases the win rate drastically but also allows for effective evaluation of the board within a given amount of time. Being able to evaluate the opponents distance to the center it seems like a key factor in winning isolation games given that the player with the longest path wins given that this heuristic when paired with AB won 23 random matches losing only 1.




Though this heuristic had high win rates it could definitely be improved. For example it could factor in the agent's distance to the center evaluated as a score. Hypothetically speaking, with this implemented on the basis that it two has been doubled, would add aggression to agent in that it would seek moves that factored in maximising its score based on its position relative to the center.

## Custom_score_3():

With this heuristic, I based its implementation on combining the two concepts used in the two previous heuristics: evaluation of path to center relative to opponent's position and evaluation of number of moves available on the board.

With this heuristic the board's different states are evaluated according to the difference in moves available solely to the opposition in relation to the agent's position from the center. Doing this allows the agent to pick moves that are best suited to itself whilst

minimizing the opponents score at any given point in the game. It's not entirely effective at choosing winning moves against other agents, in straight sets sometimes. With this in mind the concepts used in two functions above do tend to outperform this function in rate of wins, this could be down to incorrect implementation or missing a key component in properly evaluating the board's different states.
On average this function provided the agent with a win rate of 53.6%, highest being 67.9% and lowest being 42.9%. This could be down to a factor of things, possible horizon effect on scoring the agent number of moves due to the added aggression, hardware or more likely the evaluation function isn't effective enough in evaluating the different states as scores.
Notably highlighted through results, this function could handle and win matches against agents whose evaluation of the board depended solely on the center. This could be due to the agent maximising his own distance from the center and plays moves that really damage to an opponent of such description. These assumptions can somewhat be backed up through data on match outcomes against MM_Center and AB_Center:

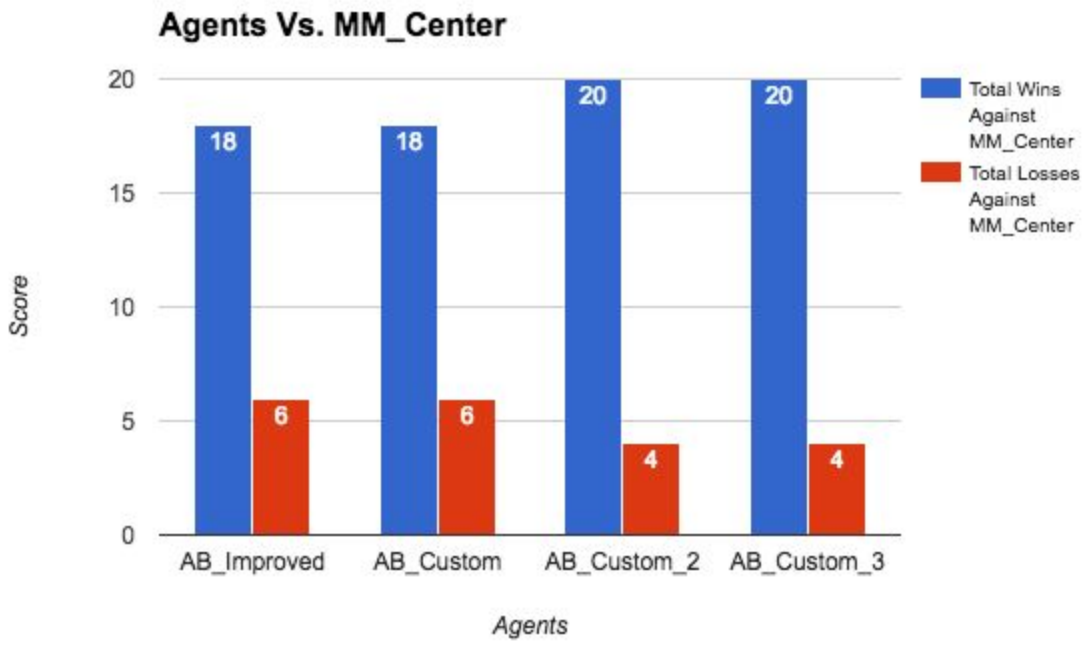



Figure 3.
Highlighting this heuristics evaluation of the board is capable at winning matches against a center focused agent.

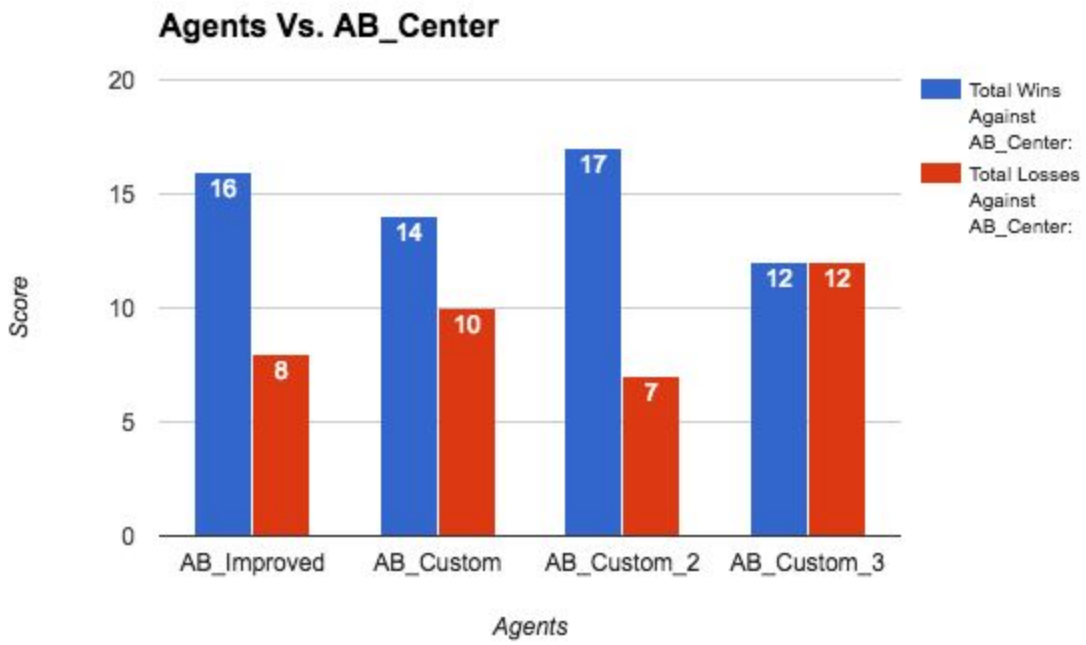
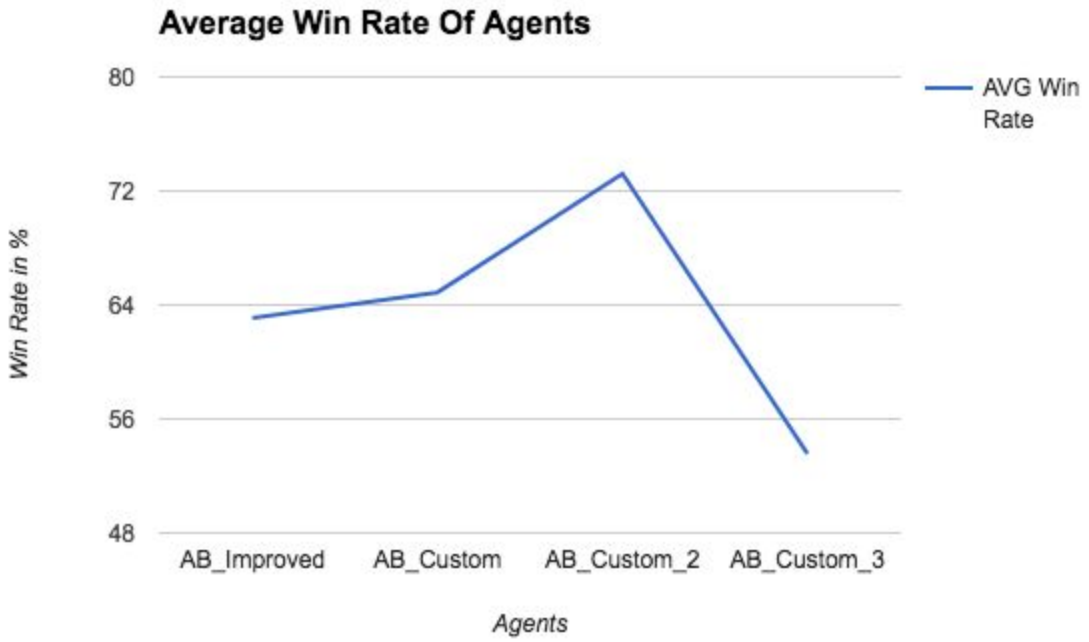



Figure 3.2
Evaluation function handle its own against AB_Center, winning and losing a total amount of matches.

With all this being said this evaluation function was the least effective in evaluating good board states for the agent.

Conclusion




To conclude, heuristic two which evaluated the opponents distance from the center in relation to the difference of number of moves available to the board and the number of remaining spaces left deemed a more sturdier and effective means of evaluating the board at its different states.
With an average win rate of 73.2% over a total of 6 tournaments, this would be the evaluation function of choice at this current point in time. I say this because it outperforms the AB_Improved agent in terms of average win rate, even though it doesn't necessarily beat it head-on(draws - see custom_score_2 section above) it is more consistent and effective in its evaluation of different board states. The same could be said for the heuristic used in custom_score_1, as it too yields a greater win rate than the AB_Improved agent. This could be down to the difference in varied but related constraints of which are being evaluated in the two functions. By using these constraints the agent is provided with a more effective means of checking each game state for a best move whilst minimizing their opponents chances of winning. With this in mind, these two functions, custom_score() and custom_score_2() were the only ones which provided somewhat consistent win rates, more so custom_score_2:

| Agent | Tournament 1 Win Rate | Tournament 2 Win Rate | Tournament 3 Win Rate | Tournament 4 Win Rate | Tournament 5 Win Rate | Tournament 6 Win Rate | AVG Win Rate |
|---|---|---|---|---|---|---|---|
| AB_Improved | 75 | 71.4 | 64.3 | 67.9 | 53.6 | 46.4 | 63.1 |
| AB_Custom | 64.3 | 60.7 | 60.7 | 67.9 | 71.4 | 64.3 | 64.88333333 |
| AB_Custom_2 | 64.3 | 64.3 | 75 | 75 | 75 | 85.7 | 73.21666667 |
| AB_Custom_3 | 64.3 | 57.1 | 57.1 | 50 | 50 | 42.9 | 53.56666667 |

Figure 4. Another measure of proof that this heuristic is the most efficient and effective out of current implemented evaluation functions.

Though not with a 100% win rate that I'd like, I feel that the heuristic used and implemented in custom_score_2 is both efficient and effective at helping the agent evaluate winning moves at different turns within the game, Though not perfect, suffering heavy defeats at times(Win: 1, Loss: 3) to some agents it outperforms the other heuristics and implements even more so justified through the data provided through played tournaments.