In order to get a better understanding of which heuristic function strategies would work best, I played my own game of isolation. Doing this helped me see how the strategies discussed in the lectures as well as new approaches I may approach would make sense. When looking at different information I could experiment with I broke it down into:

- Current player position, current player moves
- Opponent player position, opponent player moves
- Board state (squares taken, squares left)
- Distance from centre, distance from opponent

```
**************************
    Playing Matches
**************************
```

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	10	0	8	2	8	2
2	MM_Open	5	5	6	4	7	3	7	3
3	MM_Center	7	3	8	2	5	5	7	3
4	MM_Improved	8	2	4	6	3	7	8	2
5	AB_Open	6	4	7	3	5	5	7	3
6	AB_Center	6	4	6	4	6	4	7	3
7	AB_Improved	6	4	5	5	6	4	5	5
	Win Rate:	65.7%		65.7%		57.1%		70.0%	

The custom heuristics that resulted in the highest win rate were the following:

1. AB Custom - Maximize the distance from opponent

- Find the location of the opponent from the location of the current player and add together the squared difference between the two
- Logic: getting the algorithm to focus on getting away from the opponent could play a defensive strategy
- Result: This strategy seems to be as effective as AB_Improved, with a win rate of 65.7%

2. AB Custom 2 - Get as close to opponent player as possible

- Similar to the first custom function, however it is the opposite - playing a more aggressive strategy by scoring moves that are closer to the position of the opponent higher
- Logic: the idea is if we are moving toward the opponent we can perhaps get it in a defensive strategy and corner it more quickly
- Result: the results of this algorithm were not very effective, maxing at a win rate of 57.1%, the least effective of all the strategies

3. AB Custom 3 - Minimize the number of opponents moves

- Look at the number of moves both the current player and opponent have, and subtract the legal moves of the opponent from the player, forcing the algorithm to find the moves
- Result: Seems to be the most effective algorithm, at 70%

Given these results, I would recommend the third custom algorithm strategy is followed, i.e. minimize the number of moves the opponent has, for three reasons

1. It always performs at the same rate or better than AB_Improved when I've tested it
2. Its performance gives us the highest level of win-rates
3. The strategy seems logically sound - minimizing the moves the opponent has is in line with the game's strategy of minimizing the opponents moves to eventually 0