Heuristic analysis

The heuristic functions I chose combine the two ideas presented in the video tutorials. These evaluation functions kept a count of the difference between the number of player's and opponent's moves, as well as awarding each move a bonus term defined in terms of Euclidian distance from the centre of the board.

AB_Custom awards a slight bonus to moves closer to the centre of the board, whereas the other two heuristics award bonuses to moves closer to the edges of the board. The difference between AB_Custom_2 and AB_Custom_3 is the significance of the bonus term; AB_Custom_2 slightly favours moves closer to the edges of the board, while AB_Custom_3 strongly favours such moves.

The three heuristics achieved similar results based on a sample of 1,000 games against each opponent. It seems that irrespective of the evaluation functions I implemented, the win-rates of all agents plateau at around 70%. I suspect that this plateau has less to do with the heuristic function, and more with the random initialization of the first move. This, I believe can be seen through the relatively low average win-rate of the four agents against the random opponent of 93.2% (Please see Fig. 1). The fact that in about 7% of instances an agent loses against a random opponent suggests to me that the random position initialisation has a significant impact in the outcome of the game.

Despite the 0.2% decrease in win-rate of AB_Custom_3 as opposed to AB_Custom, I still prefer AB_Custom_3. This evaluation function reliably predicts the outcome of the game, and is relatively stable against all opponents. The heuristic function is reasonably easy to compute, which allows for a deeper exploration of the game tree. I think this heuristic function strikes a balance between evaluating the position and doing so without great additional computational strains. Moreover, I think it touches an idea that is closer to the way I think about winning strategies in the middle-game. As opposed to looking for moves which maximise "my moves – opponent moves", I often look for the longest path which cannot be interfered with by my opponent, in a similar fashion to the instance of queens separation in the other variant of Isolation described in the video tutorials. From my experience, by looking to play closer to the edges of the board, it usually is the case that one naturally maximises the length of the path that cannot be interfered by the opponent, while not necessarily maximising "my moves – opponent moves". This is an a posteriori observation of analysing games I played and games the agents described above played. It appears that it is often the case that close to the edges of the board one often only has one or two moves available, but the path that follows by playing these moves cannot be interfered with by the opponent and is often long enough to win the game.

An additional feature I would like to add to this heuristic is, however, that it uses AB_Custom in the early stages of the game and therefore prefers playing closer to the centre, and transitions to AB_Custom_3 in the later stages of the game where path lengths that cannot be interfered with by the opponent start to show up.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	929	71	946	54	928	72	924	76
2	MM_Open	739	261	772	228	738	262	768	232
3	MM_Center	884	116	885	115	887	113	884	116
4	MM_Improved	721	279	744	256	706	294	723	277
5	AB_Open	520	480	528	472	509	491	522	478
6	AB_Center	566	434	559	441	551	449	558	442
7	AB_Improved	497	503	490	510	465	535	528	472
	Win Rate:	69.4%		70.3%		68.3%		70.1%	

Fig. 1. Results from tournament.py over 1,000 games