ISOLATION HEURISTIC EVALUATION

Methods

In the lectures the following heuristics were introduced:

1. Own moves vs. opponent moves: Takes the number of own moves and subtracts the number of the opponents moves. This is the AB_Improved heuristic that all other heuristics should evaluated against.
2. Higher weight of opponent moves: Takes the number of own moves and subtracts twice the number of opponent moves.

Based on those evaluations the following ideas have been evaluated in the tournament setting:

1. Own_moves – weight * opp_moves: The factor of 2 for weighing the opponent moves seems rather arbitrary. So, I tried several other values.
2. Another way of combining the moves of both players is to compute the ratio of both values and possible weighing them with the formula own_move / (weight * (opp_moves + 1). The +1 avoids division by zero. Again, the weights have been varied.
3. It could make sense to vary the influence of each factor during the game. To accomplish this, I use the number of moves that have already been done and compute a linear equation which is defined by the two values at the start of the game (move_count == 0) and the end of the game (move_count == 25).
   weight = a * move_count + b
   This one factor is used to weigh both values in the opposite direction.
   result = weight * own_moves – (1 – weight) * opp_moves

As a further extension (but not implemented), a neural network could be trained to evaluate the positions as in the AlphaGo paper.

Results

Generally, the different heuristics are not easily compared as the random nature of the start states in the tournament setting generates a lot of variance. The AB_Improved showed results ranging from 62.9% to 75.7%. The evaluation against the selected opponents in tournament.py using random, minimax or simple alphabeta strategies favors heuristics, that perform very well against those opponents. The influence of more advanced agents is not so significant.

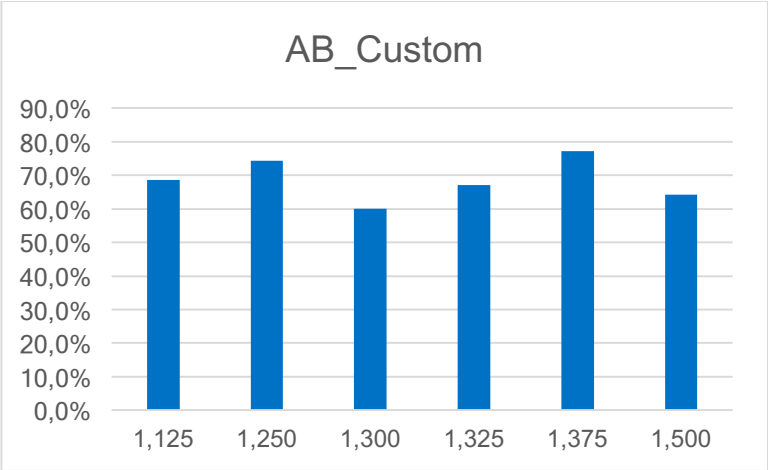

Figure 1

AB_Custom

The results show that this heuristic can model the game better than the other heuristics given in such a degree that it can win a significant amount of games. It takes both players into account and the experiments show that there is some difference when adapting the weights of the opponent moves. I did not weigh the moves of my own agent as the important part is to relate the own moves vs. the opponent moves. The scaling does not matter. This led to the idea of AB_Custom_2 using a ratio. This heuristic is very easy to implement an also very easy to compute. It doesn't traverse the game tree. Generally, the alphabeta search should do the traversing of the game tree and not the heuristic. There is however, some tradeoff. Putting more computation into the heuristic and therefore generating better evaluation scores will lead to less evaluated nodes, which can have a much bigger influence on the overall computation time. The evaluation shows that the best overall result with **77,1%** can be achieved with a value of 1.375 for weighing the opponents moves. However, the results are

not really consistent for small variations and setting the weight to 1.3 shows a significant performance drop to only 60%.

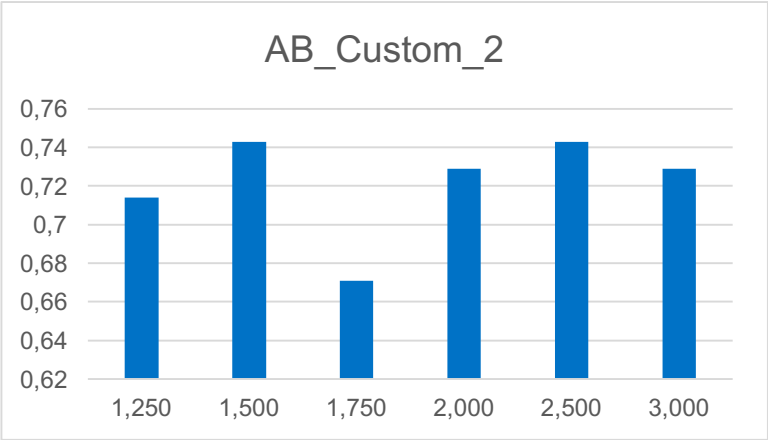

Figure 2

AB_Custom_2

In contrast to the first heuristic, this one optimizes the ratio between my own moves and the weighted opponent moves. The general idea behind this is similar to the AB_Custom heuristic, but it uses a 1/x instead of a linear function. A nice comparison can be seen in figure 4. It is easy to implement and easy to compute as it does not traverse the game tree. It seems that the 1/x function can model the true dependency slightly better than a linear function. Using a ratio between the agents and the opponents moves showed consistently high results for large values of the weight, i.e. 2 and above and outperformed AB_Improved significantly for the runs with values 2.5 and 3.

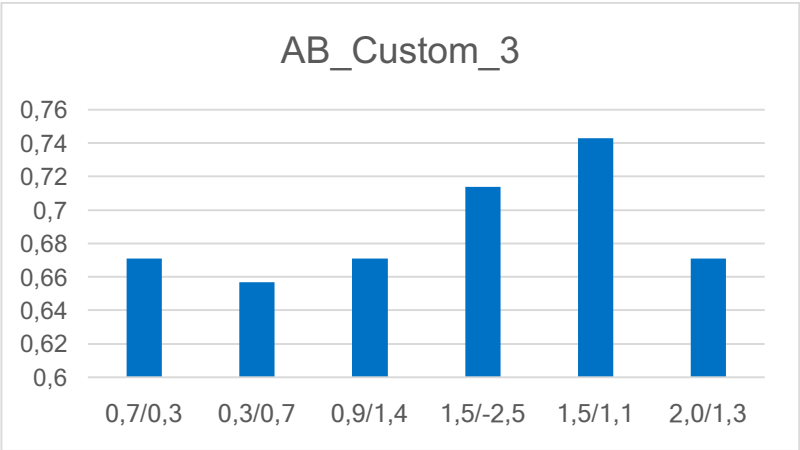

Figure 3

AB_Custom_3

Changing the weights during the game did not work well for the values that I initially designed it for, e.g. 0.7 and 0.3 so that the opponents moves become more important at the end of the game. However, with some trial and error also this heuristic could outperform AB_Improved for the cases where a high weight was used for the agents moves and the variations increased the importance of the opponents move towards the end of the game. This heuristic is slightly more complicated to compute, but still easy to implement and does not traverse the game tree. Figure 4 shows the evaluation of this heuristic for the start and end of the game. In contrast to the other heuristics, it increases the values if the number of opponent moves increases which makes it very aggressive to reduce the number of opponent moves.

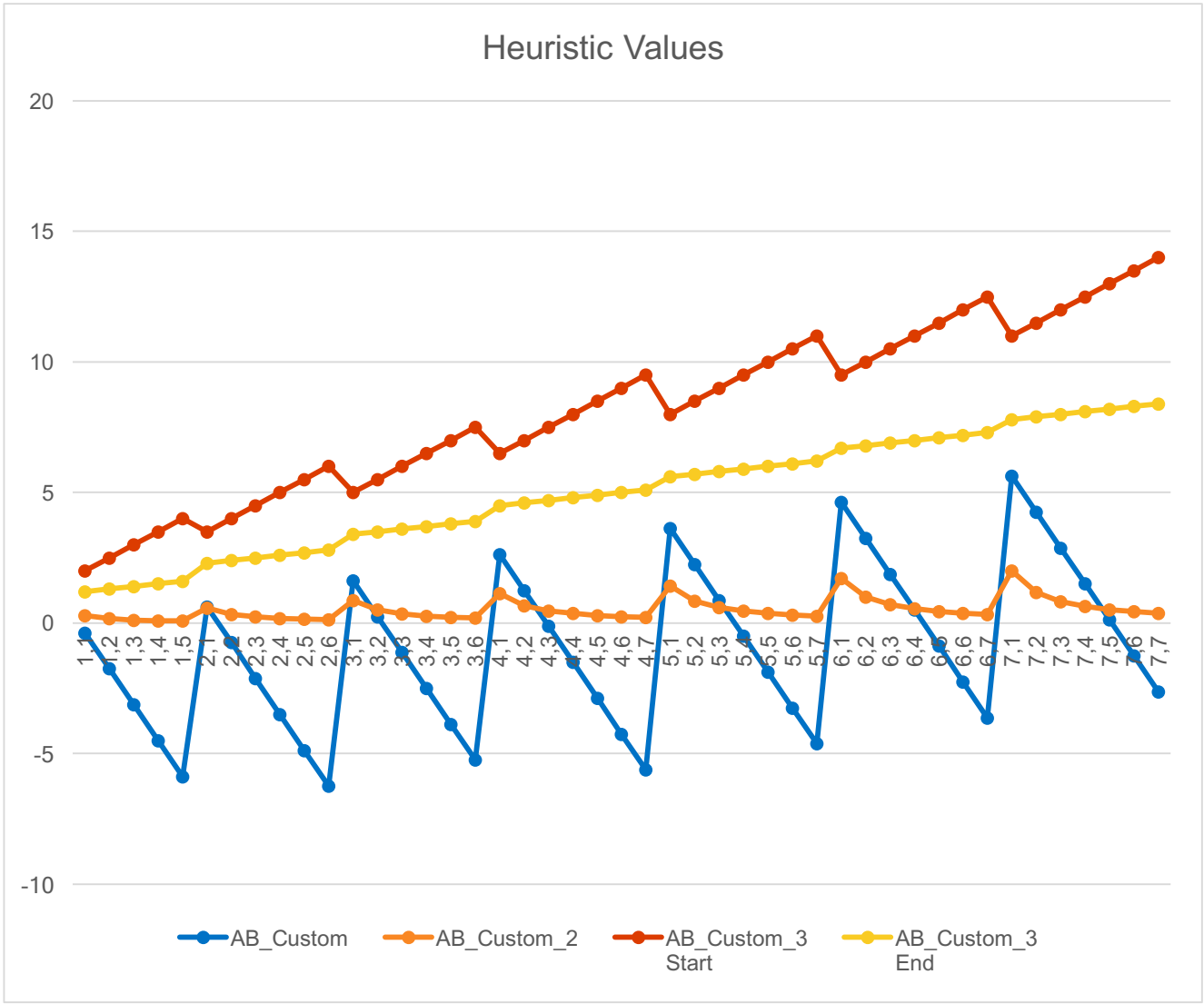

Figure 4: The x-axis plots (own_moves, opp_moves) and the y axis plots the value of each heuristic.

Recommendation

The general recommendation would be to use the ratio based heuristic with a weight of 2.5.

The main reasons for choosing this heuristic are:

1. Very good performance in the tournament
2. Consistently good performance in the tournament. Compared to AB_Custom the variations in performance showed less variance.
3. The performance against the best opponent (AB_Improved) was always at least a tie or more wins for AB_Custom_2.
4. The computation is easy to do and easy to understand.

Appendix

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.5		1.5		0.7 / 0.3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	9	1	9	1
2	MM_Open	7	3	8	2	8	2	8	2
3	MM_Center	10	0	7	3	10	0	9	1
4	MM_Improved	8	2	5	5	8	2	7	3
5	AB_Open	6	4	6	4	6	4	5	5
6	AB_Center	6	4	5	5	5	5	4	6
7	AB_Improved	6	4	4	6	6	4	5	5
	Win Rate:	75.7%		64.3%		74.3%		67.1%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.25		1.25		0.3 / 0.7	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	10	0	10	0
2	MM_Open	8	2	7	3	7	3	7	3
3	MM_Center	7	3	9	1	9	1	10	0
4	MM_Improved	7	3	6	4	6	4	7	3
5	AB_Open	8	2	7	3	7	3	4	6
6	AB_Center	7	3	6	4	5	5	3	7
7	AB_Improved	4	6	7	3	6	4	5	5
	Win Rate:	71.4%		74.3%		71.4%		65.7%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.125		2		0.9 / 1.4	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	9	1	10	0	10	0
2	MM_Open	7	3	8	2	8	2	8	2
3	MM_Center	10	0	10	0	9	1	7	3
4	MM_Improved	7	3	7	3	7	3	6	4
5	AB_Open	4	6	3	7	5	5	6	4
6	AB_Center	6	4	6	4	6	4	6	4
7	AB_Improved	6	4	5	5	6	4	4	6
	Win Rate:	71.4%		68.6%		72.9%		67.1%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.375		1.75		1.5 / -2.5 (error)	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	9	1	9	1
2	MM_Open	7	3	9	1	8	2	9	1
3	MM_Center	8	2	9	1	7	3	10	0
4	MM_Improved	7	3	9	1	7	3	5	5
5	AB_Open	6	4	4	6	6	4	8	2
6	AB_Center	4	6	5	5	4	6	5	5
7	AB_Improved	6	4	8	2	6	4	4	6
	Win Rate:	68.6%		77.1%		67.1%		71.4%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.325		2.5		1.5 / 0.9	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	9	1	10	0	10	0
2	MM_Open	7	3	8	2	8	2	7	3
3	MM_Center	8	2	10	0	7	3	9	1
4	MM_Improved	7	3	7	3	6	4	6	4
5	AB_Open	6	4	5	5	7	3	5	5
6	AB_Center	4	6	5	5	8	2	9	1
7	AB_Improved	4	6	3	7	6	4	6	4
	Win Rate:	62.9%		67.1%		74.3%		74.3%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
Factors:				1.3		3		2.0/1.3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	9	1	10	0	10	0
2	MM_Open	8	2	7	3	6	4	8	2
3	MM_Center	8	2	10	0	9	1	9	1
4	MM_Improved	6	4	6	4	7	3	8	2
5	AB_Open	4	6	4	6	9	1	6	4
6	AB_Center	6	4	2	8	5	5	3	7
7	AB_Improved	4	6	4	6	5	5	3	7
	Win Rate:	62.9%		60.0%		72.9%		67.1%	