All my three heuristics are computed based on two variables: 1. Number of my own available moves (own_moves). 2. Number of my opponent's available moves (opp_moves).

My first heuristic (AB_Custom) is **own_moves – 2 * opp_moves**. Two basic strategies of designing heuristics are Increasing my own moves and reducing the opponent's moves. The former one is a more "safe" strategy and the latter one is a more "aggressive" strategy. I choose to combine these two variables to balance the two strategies. I tested a bunch of weights and found that own_moves – 2 * opp_moves has the best performance with **70.1%** winning rate compared with AB_Improved's **68.4%.**

My second heuristic (AB_Custom_2) is **(own_moves + 1) / opp_moves**. Here I'm using the same strategy with heuristic 1, which is balancing safe and aggressive. However, I used divisive instead of subtractive, because when both players have relatively large number of available moves, the absolute difference might not be as important as when they have small number of moves. The "+1" term comes from testing experience. The winning rate of this heuristic is **69.3%.**

My third heuristic (AB_Custom_3) is to apply safe and aggressive strategy at different stage of game. When open space on the board is more than 25% of the total space, return **own_moves – 0.1 * opp_moves** as score, when open space is less than 25% of the total space, return **own_moves – opp_moves** as score. Since the maximum available moves for this game is 8, thus the factor 0.1 is only used to break tie. The logic behind this heuristic is, when there are a lot of space at the beginning, chasing behind your opponent is less likely to harmful. In contrast, in the late stage of the game, aggressive strategy might be more effective. The winning rate of this heuristic is **71.7%.**

In summary, all these three heuristics seems to perform better than the test agent AB_Improved. (see table and figure 1 below)

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

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	95	5	92	8	94	6	92	8
2	MM_Open	67	33	75	25	74	26	76	24
3	MM_Center	86	14	92	8	89	11	91	9
4	MM_Improved	71	29	80	20	70	30	70	30
5	AB_Open	47	53	48	52	53	47	56	44
6	AB_Center	61	39	56	44	54	46	60	40
7	AB_Improved	52	48	48	52	51	49	57	43
	Win Rate:	68.4%		70.1%		69.3%		71.7%	

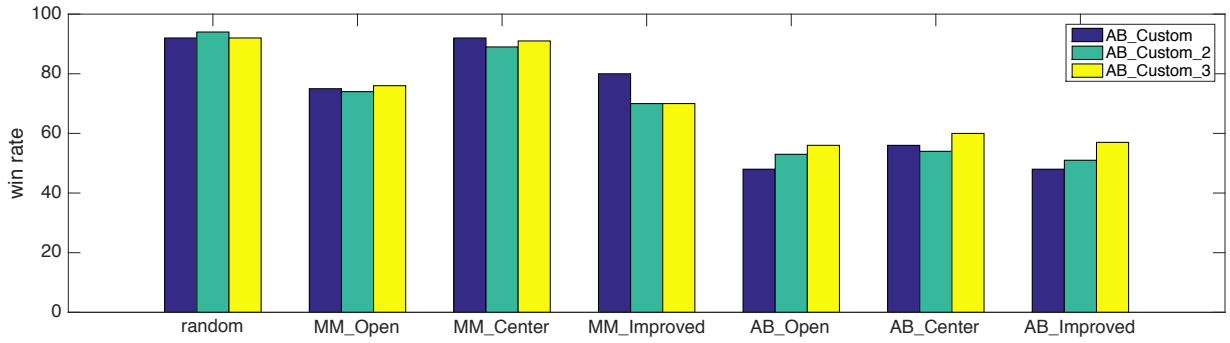

Figure 1. Win rate of 3 heuristics against 7 different test agents.

I would recommend using AB_Custom_3. **First**, it has the highest overall win rate (71.7%) among the three heuristics. **Second**, I compared the first move searching depth of these 3 heuristics playing against random agent (50 repeats). Although AB_Custom_3 is the most complex heuristic, its average searching depth is very close to the other two (figure 2. 6.96 vs 7.12 and 7.02), indicating that the searching depth is not significantly compromised. **Third**, I compared the rank of these heuristics playing against 7 different test agents. The average rank of AB_Custom_3 is significantly higher than the other two (1.42 vs 2.00 and 2.28), which indicates that AB_Custom_3 performs more stable than the other two. **Fourth**, among 7 test agents, AB_Custom_3 achieved highest win rate against all 3 AB agents (figure 1). Since AB performs better than MM and random agents, this indicates that AB_Custom_3 performs particularly better against relatively strong algorithms.

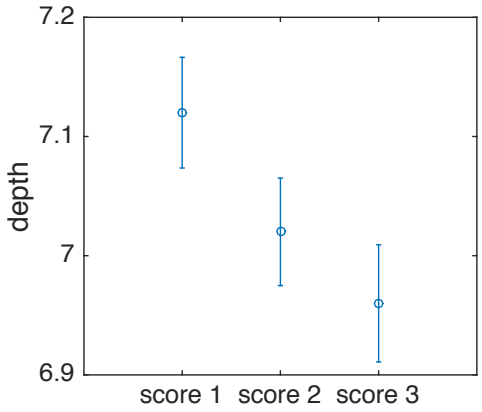

Figure 2. First move searching depth of 3 heuristics playing against random agent.

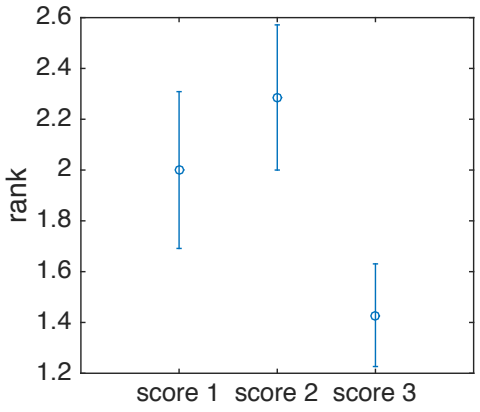

Figure 3. Rank of 3 heuristics playing against 7 different test agents.