Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	9	1	7	3	8	2
2	MM_Open	7	3	6	4	7	3	8	2
3	MM_Center	10	0	5	5	7	3	7	3
4	MM_Improved	5	5	8	2	6	4	7	3
5	AB_Open	6	4	3	7	5	5	6	4
6	AB_Center	5	5	4	6	4	6	7	3
7	AB_Improved	5	5	4	6	4	6	4	6
	Win Rate:	67.1%		55.7%		57.1%		67.1%	

Comparing the different performances of my 3 custom heuristics, with:
AB_Custom evaluating positions based on the remaining moves left for the opponent player;
AB_Custom_2 evaluating positions based on the difference between remaining moves left for the computer player and opponent player, while giving the number of moves of the opponent an increased weightage;
AB_Custom_3 evaluating positions based on the percentage of total available moves belonging to the computer player,
it is clear that when faced with opponents both implementing Minimax and Alpha-Beta pruning, heuristic Custom_3 works the best with a win rate of 67.1%, but on average, the custom heuristics still score lower than a test heuristic AB_Improved.
Looking at the data, it is clear that heuristic 3 works the best. Heuristic 3 maximizes the number of legal moves available to our player while considering the total number of legal moves available to our player and the opponent, and seeks to maximise the ratio of legal moves available to our player to total legal moves available to give our player the most advantage. In an adversarial game such as isolation, the heuristic works well because we are not only considering the survivability of our own player, but also trying to minimize the amount of moves left for the opponent player in order to maximise our own chances of winning.
The reason that heuristic 1 does not perform as well could be that only considering the number of moves available to the opponent player ignores the number of moves available to our player to a large enough extent that it is causing bad moves for our own player.
On the other hand, the reason that heuristic 2 does not perform as well could be that only focusing on the difference between the number of moves available to our player and the opponent neglects our own number of moves and thus fails to maximise the number of moves available to our player in the first place. Although it is an aggressive version of the ID_Improved implementation as it gives double weightage to the number of opponent moves and seeks to minimize opponent moves as much as possible in relation to our own number of moves, it is possible that a balance between a conservative playing style and an aggressive stance is required and different strategies are required at different points of the game.
It is hard to come up with a heuristic that far outperforms ID_Improved given the time limit of the search. While a more complex heuristic such as heuristic 3 will give

a better estimate at a shallow depth, it can be more advantageous to implement a simpler and faster heuristic such as heuristic 1,
which will proceed deeper into the game tree in the same amount of time. Considering this trade-off, heuristic 2 is recommended as it balances the complexity and speed of the search well. Heuristic 2 has a decent winning rate on average, and is easy to implement. It also trades some complexity for speed, making it the best heuristic to use in general.