Heuristic analysis for AIND isolation

General outline

The goal is to find and describe different scoring heuristics for a virtual game of isolation in which two players move in patterns similar to a knight in chess. The game is defined by Udacity's AIND Isolation project at https://github.com/Hopfensaft/AIND-isolation.
The heuristics can consist of any method to determine the value of a current game state, that is then used as to make a decision which move is best for the active player at any given time.

All ideas outlined below have in common that they use the number of moves the player has available as well as infinitely larger numbers for a lost/won game into account. As winning is the ultimate goal, these moves should always be prioritised.

The results

The three different heuristics tried in the example are all scored against an "AB_Improved" score as a benchmark. After running several benchmarks, the custom scores appear to outperform AB_Improved generally. On some occasions however, it seems the heuristics give inferior results. Win rates over multiple games for all heuristics can vary up to +/- 7%.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	10	0	8	2
2	MM_Open	7	3	8	2	7	3	7	3
3	MM_Center	8	2	9	1	8	2	7	3
4	MM_Improved	6	4	10	0	6	4	8	2
5	AB_Open	4	6	7	3	4	6	5	5
6	AB_Center	5	5	6	4	5	5	4	6
7	AB_Improved	4	6	5	5	4	6	4	6
	Win Rate:	61.4%		78.6%		62.9%		61.4%	

AB_custom

This heuristic is a twist on the AB_Improved scoring by valuing the number of available player moves more highly.

len(player_moves)*2 - len(opponent_moves)

the idea behind this test is that regardless of how many options the opponent has, the most

important part of the game is that the active player has possible moves at his disposal.
The win rate leads to believe a higher focus on player movement options leads to better results.

AB_custom_2

With this heuristic, the idea is to test a more aggressive approach to game playing. Game states that allow the player to "steal" a move from the opponent. This is determined by looking at potential movesets of both players, should they contain the same position, the player can take this possible move away from the opponent. A possibility to steal multiple moves is not awarded more highly as only one move can be taken away from the opponent at any given time.

A move steal is (arbitrarily) set to be worth an equivalent of 2 available moves.
len(player_moves) - len(opponent_moves) + 2 * move_steal

This strategy seems to lead to consistently slightly better results than AB_Improved. Stealing an opponent move appears to be a viable option.

AB_custom_3

Lastly, I wanted to try a completely different approach. While still taking into account available moves of the player, this heuristic also rewards player positioning on the board.

Early tests focussed on a central position, with the theory in mind that a centred stance allows for movement in various directions, depending on where openings are found. Inconsistent results of this approach prompted me to add another component. The max squared distance to all unused positions on the game board in the second half of the game.
My reasoning is that focussing on the centre position leads to these fields getting used early on in the game to get tactical advantage. It might be equally important in the later game stages to be "near" clusters of empty spaces as good as possible.

Good positioning (centre or near large, clear clusters) is worth a maximum of three moves.
len(player_moves) + 3 / positioning_bonus

Win rates are decent, but not great. Generally being at least on par with AB_Improved and sometimes beating it. Depending on the game board, this heuristic either outperforms or underperforms others on MM_Center and AB_Center players. More tweaking of this heuristic should be considered.