Heuristic Analysis

I played 100 matches per opponent, as opposed to the default 5 matches, because I found that there was a lot of variance in the results with such a small number of matches. I developed four heuristics to evaluate game state and below are the performance of the game agent using each of the heuristics.

Block_score:

*(# my moves – 2 * # opp moves)*

This heuristic encourages blocking the opponent's avaiable moves, as there is a higher reward for blocking the opponent's moves than attaining a position with more legal moves. The result was a win rate of 67.86%, which beat the default ID_Improved agent, but not by a significant amount.

Slow_block_score

*(# my moves – (1 + 1 / # free spaces) * # opp moves)*

This heuristic is the same as the previous one except with an additional parameter: the # of free spaces. This allows us to track how far into the game we are. As the game goes longer, the agent puts more emphasis on blocking the opponent's moves. I thought this might increase the chances of going for a "killer move" later in the game, when there are less moves available. However, the win rate was only 67.71%, which is almost identical to the previous heuristic, and only slightly beating ID_Improved.

Lookahead_score:

(# own future moves – # opp future moves)

This heuristic gets the available legal moves, and then for each of those moves, it looks ahead for the next available moves (the future moves). It then sums up the count of available moves from the second level (the level at which we are looking ahead). The win rate using this heuristic was 70.29%, which is significantly higher than that of the previous two heuristics and the ID_Improved agent. Looking ahead to evaluate the game state seemed to have increased the performance of the game agent. Something I noticed though is that in my code, I summed up the count of available legal moves on the second level, which leads to recounting the same moves sometimes. The game agent may improve even more if I did not recount the same moves.

Slow_lookahead_score:

*(# own future moves – (1 + 1/ # free spaces) * # opp future moves)*

This heuristic is essentially a combination of the last two heuristics, as it looks ahead and sums up the number of available moves at the second level, which also encouraging blocking the opponent's moves more as the game progresses. With this heuristic, the agent yielded its highest win rate at 72.79%.

Performance Table

	ID_Improved	Block	Slow_block	Lookahead	Slow_lookahead
Random	160-40	169-31	163-37	159-41	172-28
MM_Null	146-54	151-49	149-51	159-41	157-43
MM_Open	121-79	133-67	132-68	130-70	139-61
MM_Improved	122-78	115-85	123-77	128-72	124-76
AB_Null	142-58	143-57	135-65	147-53	157-43
AB_Open	122-78	129-71	120-80	132-68	138-62
AB_Improved	116-84	110-90	126-74	129-71	132-68
Win Rate	66.36%	67.86%	67.71%	70.29%	72.79%

Chosen heuristic:

I chose the Slow_lookahead_score heuristic in the end as it yielded the highest win rate. I believe this heuristic does the best job in evaluating the game state, as shown by the win rate. This heuristic is not as simple as ID_improved (or Block_score and Slow_block_score), so it doesn't search as deep into the search tree, but this is outweighed by being a more complex, better heuristic. Looking ahead to evaluate the game state seems to increase the performance of the agent, as seen by the performance of Lookahead_score and Slow_lookahead_score. This heuristic seems to evaluate the game state even better than Lookahead_score, with the complexity between the two being nearly identical and the only difference being that it also blocks the opponent's moves more often as the game goes on.