Heuristic Analysis

Match #	Opponent	AB_Improved Won	AB_Improved Lost	AB_Custom Won	AB_Custom Lost	AB_Custom_2 Won	AB_Custom_2 Lost	AB_Custom_3 Won	AB_Custom_3 Lost
1	Random	6	4	8	2	8	2	9	1
2	MM_Open	8	2	8	2	6	4	6	4
3	MM_Center	7	3	8	2	9	1	7	3
4	MM_Improved	7	3	7	3	4	6	8	2
5	AB_Open	8	2	5	5	4	6	3	7
6	AB_Center	6	4	5	5	7	3	4	6
7	AB_Improved	3	7	5	5	7	3	5	5
	Win Rate:	64.3%		65.7%		64.3%		60.0%	

There were 2.0 timeouts during the tournament -- make sure your agent handles search timeout correctly, and consider increasing the timeout margin for your agent.

Your agents forfeited 247.0 games while there were still legal moves available to play.

Heuristic #3 returns a score based on the moves remaining for each player. This heuristic is fairly quick to calculate and has a 60.0% win rate. This heuristic rewards the player with the more moves remaining. Heuristic #2 takes a step further from Heuristic #3 in the cases in which the moves remaining are equal for each player. Heuristic #2 calculates the Manhattan distance from the center for each player. This heuristic takes more time to calculate than Heuristic #3, but is slightly more successful with a 64.3% win rate. This heuristic rewards the player with more distance from the center. Finally, Heuristic #1 takes a step further from Heuristic #2 and squares the Manhattan distance for each player. This heuristic takes more time to calculate than Heuristic #2 only slightly, but is more successful with a 65.7% win rate. Like Heuristic #2, this heuristic rewards the player with more distance from the center. Here, the heuristics reflect that the more complex heuristics are better suites to represent the win rate of the game. There were only 2 timeouts during the tournament, which reflects that the heuristics are working efficiently.

I would recommend using Heuristic #1 as the evaluation function to use. It is fairly successful as it has a 65.7% win rate (which just barely beat out AB_Improved!). According to the table, when using Heuristic #1, the player was able to stay above 0.500 with all opponents reflecting its stability. The heuristic takes into account both players as it calculates the Manhattan distance and squares it for each player. I believe that this heuristic is a good evaluation function for my player.

Heuristic #1

```
  if game.is_winner(player):
      return float("inf")

  if game.is_loser(player):
      return float("-inf")

  player_moves_remaining = len(game.get_legal_moves(player))
  player2_moves_remaining = len(game.get_legal_moves(game.get_opponent(player)))

  if player_moves_remaining != player2_moves_remaining:
      return float(player_moves_remaining - player2_moves_remaining)
  else:
      middle_x = int(game.width/2)
      middle_y = int(game.height/2)

      player_y, player_x = game.get_player_location(player)
      player2_y, player2_x = game.get_player_location(game.get_opponent(player))

      player_distance_from_center = abs(player_y - middle_y) + abs(player_x - middle_x)
      player2_distance_from_center = abs(player2_y - middle_y) + abs(player2_x - middle_x)

      return float( (player_distance_from_center^2) - (player2_distance_from_center^2) )
```