Heuristic Analysis

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

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       6  |  4
   2       MM_Open     8  |  2       6  |  4      9  |  1       3  |  7
   3       MM_Center   8  |  2      10  |  0      7  |  3       6  |  4
   4      MM_Improved  6  |  4       8  |  2      8  |  2       6  |  4
   5       AB_Open     7  |  3       6  |  4      4  |  6       3  |  7
   6       AB_Center   4  |  6       8  |  2      5  |  5       3  |  7
   7      AB_Improved  6  |  4       3  |  7      5  |  5       4  |  6
-----------------------------------------------------------------------
          Win Rate:     68.6%        71.4%        64.3%         44.3%
```

Heuristic 1 (custom_score)

```python
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
if own_moves > opp_moves:
    return float(100 * (own_moves/(own_moves + opp_moves)))
elif own_moves < opp_moves:
    return -float(100 * (opp_moves/(own_moves + opp_moves)))
else:
    return 0
```

This heuristic considers about the remaining step of the opponent as well as own. If the steps of own are more than the opponent's. We mark it as a positive weight. Otherwise, we mark it as a negative weight. According to the heuristic, we can know about this is the linear computation to find the possible moves within linear time. Even it is losing in AB_Improved, it is better than others. The win rate is nearly 70%.

Heuristic 2 (custom_score_2)

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

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

own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
if own_moves + opp_moves != 0:
    return 100 * (own_moves - opp_moves)/(own_moves + opp_moves)
else:
    return 0
```

This heuristic tries to compare the player's move and the opponent's move. The win rate is nearly 60%. In constant time you could control the number of possible steps for yourself and know your opponent's remaining time and steps.

Heuristic 3 (custom_score_3)

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

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

w, h = game.width /2., game.height / 2.
y, x = game.get_player_location(player)
return float(abs(h - y) + abs(w - x))
```

This heuristic tries to use Manhattan Distance to calculate the distance form the center block of the board. The win rate is nearly 40%. Though it will lose in some situation, it still have a little chance to win the opponent.

In Conclusion

Over all, the best performance of the heuristic is Heuristic 1. It increases the chances of winning to almost 70% in all kinds of competitions. It has some advantage in below:

1. It can hand with most kinds of competitions.
2. It was not depending on the performance of hardware.
3. It compares the player with the opponent and the remaining moves in the board, which is the core of the isolation competition.
4. Its win rate is better than heuristics 2 and 3.