Heuristic Analysis

```
CENTER_SQUARES = [(x, y) for x in [2, 3, 4] for y in [2, 3, 4]]

def moves_with_centers_and_blanks(game, player):
    opponent               = game.get_opponent(player)
    opponent_moves         = game.get_legal_moves(opponent)
    player_moves           = game.get_legal_moves(player)
    num_opponent_moves     = float(len(opponent_moves))
    num_player_moves       = float(len(player_moves))
    blank_moves            = float(len(game.get_blank_spaces()))
    player_center_moves    = float(len([m for m in player_moves
if m in      CENTER_SQUARES]))
    opponent_center_moves = float(len([m for m in opponent_moves
if m in CENTER_SQUARES]))

    return ((num_player_moves + player_center_moves) -
(num_opponent_moves + opponent_center_moves) - blank_moves) \
           / (1 + blank_moves + num_player_moves -
num_opponent_moves)

def custom_score(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")

    return moves_with_centers_and_blanks(game, player)

def custom_score_2(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    opponent           = game.get_opponent(player)
    opponent_moves     = game.get_legal_moves(opponent)
    player_moves       = game.get_legal_moves(player)
    num_opponent_moves = float(len(opponent_moves))
    num_player_moves   = float(len(player_moves))
    return 4.0*num_player_moves - num_opponent_moves

def custom_score_3(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    opponent           = game.get_opponent(player)
    opponent_moves     = game.get_legal_moves(opponent)
    player_moves       = game.get_legal_moves(player)
    num_opponent_moves = float(len(opponent_moves))
    num_player_moves   = float(len(player_moves))
    return num_player_moves - 4.0*num_opponent_moves
```

custom_score is sums (both the difference between the number of player's moves and number of opponent's moves and the difference between the number of player's moves in the centre and number of opponent's moves in the centre minus the number of unvisited squares) divided by (the difference between the number of player's moves and number of opponent's moves plus the number of unvisited squares).

Scaling by the number of unvisited squares and subtracting the number of unvisited squares increases the importance of the heuristic towards the end game.

custom_score_2 which gives more weight to num_player_moves leads to more defensive play where the player would try to increase its own moves more than it tries to decrease the opponent's moves.

custom_score_3 which gives more weight to num_opponent_moves leads to more aggressive play where the player would try to decrease the opponent's moves more than it tries to increase its own moves.

Results

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

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      178 |  22    179 |  21    178 |  22    175 |  25
   2       MM_Open     170 |  30    175 |  25    170 |  30    167 |  33
   3      MM_Center    160 |  40    179 |  21    175 |  25    174 |  26
   4     MM_Improved   167 |  33    172 |  28    177 |  23    179 |  21
   5       AB_Open     98  | 102    94  | 106    99  | 101    102 |  98
   6      AB_Center    90  | 110    98  | 102    97  | 103    103 |  97
   7     AB_Improved   100 | 100    105 |  95    113 |  87    105 |  95
--------------------------------------------------------------------------
          Win Rate:      68.8%        71.6%        72.1%        71.8%
```

All the custom heuristics seem to outperform the Improved heuristic (num_player_moves-num_opponent_moves; neutral play) by a small margin of about 2-3%. When the alpha-beta algorithm is pitted against the minimax algorithm, the alpha-beta algorithm prevails 80-90% of the time. When the alpha-beta algorithm is pitted against itself, AB_Improved, AB_Custom, AB_Custom_2, and AB_Custom_3 does not seem to show any advantages against AB_Open, AB_Center, and AB_Improved.

AB_Custom_2 seem to have a slight edge over the other heuristic for this run. But since it is only better by less than 1%, I still believe that AB_Custom may be a better heuristic as it takes into account the position of the player and opponent as opposed to just the number of legal moves left.