Isolation Game Heuristic Analysis

For this project, I have tested three different evaluation functions to calculate custom scores.

Among my three heuristics, I have chosen custom_score (Heuristic 3) as my best evaluation function based on the results and some factors.

The reason why custom_score(Heuristic 3) is relatively better because it uses positional advantage to create certain L-shape moves in the game. This heuristic takes advantage of player's position who is closer to the center of the board. Since it is more probable that this player can do better than a player whose remaining positions are near the edge of the board.

Here are details of functionalities for each heuristic and this report shows how each performs against the "ID_improved" heuristic.

Heuristic 1 (custom_score2):
Implementation:

```
def custom_score_2(game, player):
    if game.is_winner(player):
        return float("inf")

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

    player_mv= game.get_legal_moves(player)
    opponent_mv = game.get_legal_moves(game.get_opponent(player))
    # check if we have moves more than our opponent
    player_moves_remaining = len(player_mv)
    opponent_moves_remaining = len(opponent_mv)
    return float(player_moves_remaining - opponent_moves_remaining)
```

Analysis:
This heuristic is easy to interpret and fast to compute, it is clear to the notion of positional advantage but it only allows knight moves. So, this method is not great.

Heuristic 2: (custom_score3):
Implementation:

```
def custom_score_3(game, player):
    if game.is_winner(player):
        return float("inf")

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

    opponent = game.get_opponent(player)
    player_mv = len(game.get_legal_moves(player))
    opponent_mv = len(game.get_legal_moves(opponent))
    # check if we have moves more than our opponent

     return float(player_mv - (2.0 * opponent_mv))
```

Analysis:
This heuristic will cause our computer player to chase after the opponent, the winning move now has the highest evaluation function result. It keeps opponent closer, but it is not guaranteed to be the best evaluation without seeing different variants of other evaluation functions.

Heuristic 3: (custom_score):
Implementation:

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

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

    opponent = game.get_opponent(player)
    player_mv = game.get_legal_moves(player)
    opponent_mv = game.get_legal_moves(opponent)
    player_mv_left = len(player_mv)
    opponent_mv_left = len(opponent_mv)

    if player_mv_left != opponent_mv_left:
        return float(player_mv_left - opponent_mv_left)
    else:
        # look for a positional advantage if player and its opponent have the same number of
moves left
```

```
# Access positional advantage by using Manhattan distance to the center of the board
center_y, center_x = int(game.height / 2), int(game.width / 2)
player_y, player_x = game.get_player_location(player)
opponent_y, opponent_x = game.get_player_location(opponent)
player_distance = abs(player_y - center_y) + abs(player_x - center_x)
opponent_distance = abs(opponent_y - center_y) + abs(opponent_x - center_x)
return float(opponent_distance - player_distance) / 10
```

Analysis:
This heuristic increases our ability to survive because the player's remaining moves are near the center of the board due to the positional advantage. However, it still needs some improvements like speeding up the runtime execution. We can use the Manhattan distance instead of the Euclidean distance.

General results after 5 runs:

Run 1:

```
(aind) Changtongs-MacBook-Air:AIND-Isolation Nicole_Zhou$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    **************************
                         Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      20  |   0    20  |   0    18  |   2    18  |   2
    2       MM_Open     18  |   2    17  |   3    17  |   3    11  |   9
    3      MM_Center    17  |   3    17  |   3    18  |   2    18  |   2
    4     MM_Improved   15  |   5    15  |   5    13  |   7    14  |   6
    5       AB_Open     10  |  10    12  |   8    12  |   8    13  |   7
    6      AB_Center    11  |   9    12  |   8    14  |   6    14  |   6
    7     AB_Improved   11  |   9    14  |   6    10  |  10    12  |   8
--------------------------------------------------------------------------
          Win Rate:      72.9%        76.4%        72.9%        71.4%
```

Run 2:

```
(aind) Changtongs-MacBook-Air:AIND-Isolation Nicole_Zhou$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                      **************************
                           Playing Matches
                      **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      19  |   1    19  |   1    16  |   4    19  |   1
    2       MM_Open     13  |   7    12  |   8    12  |   8    14  |   6
    3      MM_Center    18  |   2    17  |   3    18  |   2    15  |   5
    4     MM_Improved   13  |   7    15  |   5    17  |   3    15  |   5
    5       AB_Open      8  |  12    10  |  10    11  |   9    11  |   9
    6      AB_Center    13  |   7    13  |   7    11  |   9    11  |   9
    7     AB_Improved   11  |   9    11  |   9    10  |  10     9  |  11
----------------------------------------------------------------------------
           Win Rate:      67.9%        69.3%        67.9%        67.1%
```

Run 3:

```
(aind) Changtongs-MacBook-Air:AIND-Isolation Nicole_Zhou$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                      **************************
                           Playing Matches
                      **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      20  |   0    17  |   3    19  |   1    18  |   2
    2       MM_Open     15  |   5    15  |   5    16  |   4    12  |   8
    3      MM_Center    18  |   2    17  |   3    20  |   0    16  |   4
    4     MM_Improved   16  |   4    15  |   5    12  |   8    13  |   7
    5       AB_Open     14  |   6    13  |   7    12  |   8     9  |  11
    6      AB_Center    11  |   9    14  |   6    13  |   7    13  |   7
    7     AB_Improved    9  |  11    11  |   9    13  |   7     9  |  11
----------------------------------------------------------------------------
           Win Rate:      73.6%        72.9%        75.0%        64.3%
```

Run 4:

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    **************************
                         Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       18  |   2    18  |   2    19  |   1    20  |   0
    2       MM_Open      14  |   6    15  |   5    13  |   7    16  |   4
    3      MM_Center     20  |   0    19  |   1    14  |   6    16  |   4
    4     MM_Improved    15  |   5    13  |   7    15  |   5    17  |   3
    5       AB_Open      10  |  10     9  |  11    10  |  10     9  |  11
    6      AB_Center     10  |  10    11  |   9    13  |   7    10  |  10
    7     AB_Improved     7  |  13    13  |   7    11  |   9    10  |  10
--------------------------------------------------------------------------
          Win Rate:      67.1%        70.0%        67.9%        70.0%
```

Run 5:

```
(aind) Changtongs-MacBook-Air:AIND-Isolation Nicole_Zhou$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    **************************
                         Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       20  |   0    19  |   1    20  |   0    18  |   2
    2       MM_Open      16  |   4    16  |   4    11  |   9    16  |   4
    3      MM_Center     19  |   1    17  |   3    15  |   5    16  |   4
    4     MM_Improved    16  |   4    16  |   4    13  |   7    12  |   8
    5       AB_Open      10  |  10    12  |   8     8  |  12    11  |   9
    6      AB_Center     16  |   4    13  |   7    12  |   8    14  |   6
    7     AB_Improved    10  |  10     7  |  13    12  |   8     9  |  11
--------------------------------------------------------------------------
          Win Rate:      76.4%        71.4%        65.0%        68.6%
```