# Isolation game heuristic analysis

For this project, I have implemented 4 different heuristics

I recommend the gameStragety3, and the following are the reasons:

1. It translates the notion of positional advantage to the specific L-shape knight-like moves allowed in the game
2. By counting L-shape jumps, it matches the use of a proven distance measure (Manhattan vs Euclidean)
3. It can leverage sophisticated game mechanics (max number of moves over min number of squares) to increase survival rate toward the end of the game
4. It can leverage function inlining and loop unrolling to explore more branches before timeouts

Now, I provide the justification for each game strategies implemented in the game_agent.py

## 1. gameStrategy1

In this heuristic, the more available moves player has available from the evaluated position, the better the player's position in the game. The 'gameStrategy1() function simply returns the difference in number of legal moves left between the players. If the player and its opponent have the same number of moves, then the returned value is zero. If the returned value is positive (negative), then player is doing better than its opponent.

## Results:

| Match # | Opponent | AB_Improved | | AB_Custom | | AB_Custom_2 | | AB_Custom_3 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Won | Lost | Won | Lost | Won | Lost | Won | Lost |
| 1 | Random | 10 | 0 | 9 | 1 | 10 | 0 | 10 | 0 |
| 2 | MM_Open | 9 | 1 | 8 | 2 | 6 | 4 | 10 | 0 |
| 3 | MM_Center | 7 | 3 | 9 | 1 | 10 | 0 | 10 | 0 |
| 4 | MM_Improved | 9 | 1 | 9 | 1 | 10 | 0 | 6 | 4 |
| 5 | AB_Open | 5 | 5 | 7 | 3 | 6 | 4 | 3 | 7 |
| 6 | AB_Center | 5 | 5 | 7 | 3 | 7 | 3 | 3 | 7 |
| 7 | AB_Improved | 5 | 5 | 5 | 5 | 6 | 4 | 6 | 4 |
| | Win Rate: | 71.4% | | 77.1% | | 78.6% | | 68.6% | |

## Analysis

This is not a great heuristic. Its benefits are that it's easily interpretable and fast to compute. On the downside, it is not really "game aware". It is oblivious to the notion of positional advantage and isn't influenced at all by the specific mechanics of the game (only knight moves are allowed).

## 2. gameStrategy2

In this heuristic, as in **gameStrategy2()**, the more moves the player has available from the evaluated position, the better, but not all starting positions are equal. If a player's position is closer to the centre of the board, it is more probable that this player can do better than a player whose remaining moves are near the edge of the board (where they will have less options to move down the line).

To speed up the runtime execution of this heuristic, we use the **Manhattan distance instead of the Euclidean distance.**

## Results:

| Match # | Opponent | AB_Improved | AB_Custom | AB_Custom_2 | AB_Custom_3 |
|---|---|---|---|---|---|
| | | Won \| Lost | Won \| Lost | Won \| Lost | Won \| Lost |
| 1 | Random | 10 \| 0 | 8 \| 2 | 8 \| 2 | 9 \| 1 |
| 2 | MM_Open | 9 \| 1 | 8 \| 2 | 9 \| 1 | 8 \| 2 |
| 3 | MM_Center | 10 \| 0 | 9 \| 1 | 10 \| 0 | 10 \| 0 |
| 4 | MM_Improved | 7 \| 3 | 7 \| 3 | 10 \| 0 | 9 \| 1 |
| 5 | AB_Open | 6 \| 4 | 5 \| 5 | 5 \| 5 | 6 \| 4 |
| 6 | AB_Center | 6 \| 4 | 6 \| 4 | 4 \| 6 | 7 \| 3 |
| 7 | AB_Improved | 5 \| 5 | 6 \| 4 | 6 \| 4 | 4 \| 6 |
| | | --- | --- | --- | --- |
| | Win Rate: | 68.6% | 70.0% | 74.3% | 75.7% |

## Analysis:

This heuristic performs better than the first one but not effective much . Yes, it does benefit from positional advantage, but still isn't really "game aware". What good is a position near the centre of the board if you can't really move?

# 3. gameStrategy3()

This heuristic build on the **gameStrategy2()** and with an additional knowledge about the game awareness is induced. As with, **gameStrategy2(),** the more moves player has available from the evaluated position, the better, but not all starting positions are equal.

If a player's position is closer to the centre of the board, it is more probable that this player can do better than a player whose remaining moves are near the edge of the board. If there is no clear positional advantage (i.e. both players are at the same distance from the centre, then we measure the longest run of moves we can safely perform inside a 3x3 square defined by the starting position and any of the legal moves we have left. The longest run one can hope to reach is 7.

Note that we don't try going beyond finding more than one run of seven moves with any one available move.

## Results:

| Match # | Opponent | AB_Improved | AB_Custom | AB_Custom_2 | AB_Custom_3 |
|---|---|---|---|---|---|
| | | Won \| Lost | Won \| Lost | Won \| Lost | Won \| Lost |
| 1 | Random | 9 \| 1 | 10 \| 0 | 9 \| 1 | 10 \| 0 |
| 2 | MM_Open | 9 \| 1 | 7 \| 3 | 9 \| 1 | 9 \| 1 |
| 3 | MM_Center | 10 \| 0 | 10 \| 0 | 9 \| 1 | 9 \| 1 |
| 4 | MM_Improved | 8 \| 2 | 8 \| 2 | 8 \| 2 | 10 \| 0 |
| 5 | AB_Open | 5 \| 5 | 5 \| 5 | 5 \| 5 | 3 \| 7 |
| 6 | AB_Center | 6 \| 4 | 5 \| 5 | 3 \| 7 | 7 \| 3 |
| 7 | AB_Improved | 4 \| 6 | 5 \| 5 | 7 \| 3 | 3 \| 7 |
| | Win Rate: | 72.9% | 71.4% | 71.4% | 72.9% |

# Analysis

In this heuristic, we catch with all players. We use positional advantage and are "game aware". Still, being able to find one square where one can make seven moves, doesn't say much, especially at the beginning of the game where that constraint is easy to satisfy for both players.

For the implementation of `get_longest_jumping_run () `, please see game_agent.py.

# 4. gameStrategy4()

In this heuristic, we keep exploring game tactics.  Specifically, we assess our ability to survive the longest. We look at all the 3x3 squares in which the player's current position appears and sum the runs of moves that can be performed over all these squares (jumping back and forth up to seven times in a 3x3 square). This allows us to evaluate how long we can survive if we're cornered in a tight zone.

## Results:

| Match # | Opponent | AB_Improved | AB_Custom | AB_Custom_2 | AB_Custom_3 |
|---|---|---|---|---|---|
| | | Won \| Lost | Won \| Lost | Won \| Lost | Won \| Lost |
| 1 | Random | 10 \| 0 | 10 \| 0 | 9 \| 1 | 9 \| 1 |
| 2 | MM_Open | 8 \| 2 | 8 \| 2 | 7 \| 3 | 9 \| 1 |
| 3 | MM_Center | 10 \| 0 | 10 \| 0 | 10 \| 0 | 10 \| 0 |
| 4 | MM_Improved | 6 \| 4 | 10 \| 0 | 9 \| 1 | 8 \| 2 |
| 5 | AB_Open | 6 \| 4 | 4 \| 6 | 6 \| 4 | 6 \| 4 |
| 6 | AB_Center | 5 \| 5 | 3 \| 7 | 7 \| 3 | 7 \| 3 |
| 7 | AB_Improved | 5 \| 5 | 3 \| 7 | 6 \| 4 | 5 \| 5 |
| | | --- | --- | --- | --- |
| | Win Rate: | 71.4% | 68.6% | 77.1% | 77.1% |

## Analysis:

With this heuristic, we systematically beat the `ID_Improved` player. Using more sophisticated game mechanics, we make sure that our player can keep moving for as long as possible, even if cornered.