Heuristic Analysis

1. Analysis:

In a nutshell, after close to100 trials applied the range of performance achieved was between 50% to 70% consistently. Given that the same heuristics gave multiple results, captured here (in supporting details) are some of the trials that are the best ones.

The heuristics used in the evaluation functions are:

- den_self: number of empty spaces in the board
- own_moves: number of legal moves available for the player
- opp_moves: number of legal moves available for the opponent
- int_len: number of legal moves that are overlapped with opponent's legal moves
- self_w, self_h: player's x-axis & y-axis board locations
- opp_w, opp_h: opponent's x-axis & y-axis board locations

A combination of own_moves, opp_moves and int_len proved to score high, with or without combination of varying formulae for the moves nearing the end game (based on 'den_self').

Use of 'int_len' could be used as a power of the player or loss of power for the player. Both 'own_moves + int_len' and 'own_moves – int_len' gave higher scores. These are the final implementations for AB_Custom_2 and AB_Custom_3 respectively.

The final implementations are:

Final implementation	***Evaluation Functions***
AB_Custom	up to 32 empty spaces - **return** own_moves up to 14 empty spaces – **return** (own_moves – opp_moves) < 14 empty spaces – **return** (own_moves – int_moves)
AB_Custom_2	own_moves + int_len
AB_Custom_3	own_moves - int_len

And the outcome of the tournament was well in the zones of 55% to 70% for all the custom scores and with no timeout messages.

```
Start Time: 2017-07-09 18:42:06
                      ************************
                           Playing Matches
                      ************************

 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     7  |   3     7  |   3
    2       MM_Open      2  |   8     1  |   9     4  |   6     2  |   8
    3      MM_Center    10  |   0    10  |   0     9  |   1     5  |   5
    4     MM_Improved    1  |   9     2  |   8     3  |   7     4  |   6
    5       AB_Open      5  |   5     7  |   3     4  |   6     4  |   6
    6      AB_Center    10  |   0    10  |   0    10  |   0     9  |   1
    7     AB_Improved    6  |   4     5  |   5     7  |   3     8  |   2
--------------------------------------------------------------------------
          Win Rate:      62.9%        61.4%        62.9%        55.7%
End Time: 2017-07-09 18:53:54

Process finished with exit code 0
```

This is one of the sample runs using the final implementations, having all the scores above 55% and with at least six 10 | 0 results was one of the best.

The easier (comparatively!!) opponents that could be won at least 50% (5 | 5) are:

- Random (was tough occasionally),
- MM_center and
- AB_center (always been very easy opponent)

The tough opponents that could not be won more than 50% (5 | 5) at times and very rarely at (9 | 1) are:

- MM_improved and
- AB_improved

Improvements in scores were seen when

- A calculation intensive formula was applied while reaching end of game using a gradual score using the empty_spaces in the board
- When the original AlphaBeta class was cleaned for unnecessary checks (to reduce the time consumption)

2. Brief Report of scores:

Performance matrix for Evaluation function used in AB_Custom

AB_Custom: Functions	*AB_Custom*	*Trial #*
own_moves	62.9%	1
*own_moves * (abs(self_h - opp_h) + abs(self_w - opp_w)*	51.4%	2
*len(own_moves) * (abs(self_h - opp_h) ** 2 + abs(self_w - opp_w) ** 2)*	44.3%	3
own_moves - int_moves	54.3%	4
*up to 42 empty spaces - **return** len(game.get_legal_moves(player))* *up to 28 empty spaces – **return** (own_moves – opp_moves)* *< 28 empty spaces – **return** (own_moves – int_moves)*	58.6%	5
*up to 32 empty spaces - **return** len(game.get_legal_moves(player))* *up to 14 empty spaces – **return** (own_moves – opp_moves)* *< 14 empty spaces – **return** (own_moves – int_moves)*	**67.1%**	6
(own_moves – opp_moves) ^ 2	64.3%	7
*len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25*	52.9%	8
*len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25*	48.6%	9
*len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25*	42.9%	10
(own_moves) / den_self	60%	11

Performance matrix for Evaluation function used in AB_Custom_2

AB_Custom_2: Functions	*AB_Custom_2*	*Trial #*
own_moves/ den_self	55.7%	1
*(own_moves/den_self) * (abs(self_w - opp_w) + abs(self_h - opp_h)*	58.6%	2
*(own_moves - opp_moves) * (abs(self_w - opp_w) + abs(self_h - opp_h))*	44.3%	3
den_self - opp_moves	25.7%	4
*up to 42 empty spaces - **return** len(game.get_legal_moves(player))* *< 42 empty spaces – **return** (own_moves / den_self)*	64.3%	5
*up to 32 empty spaces - **return** len(game.get_legal_moves(player))* *< 32 empty spaces – **return** (own_moves / den_self)*	62.9%	6
*up to 54 empty spaces - **return** len(game.get_legal_moves(player))* *up to 34 empty spaces – **return** (own_moves ^2 – int_moves)* *< 34 empty spaces – **return** ((own_moves - int_moves)/ den_self)*	60%	7
*(own_moves) / den_self) - (int_len / den_self) ** 2*	57.1%	8
10 - len(game.get_legal_moves(player)) + int_len	10%	9
own_moves + int_len	**65.7%**	**10**
own_moves + int_len	62.9%	11

Performance matrix for Evaluation function used in AB_Custom_3

AB_Custom_3: Functions	*AB_Custom_3*	*Trial #*
own_moves – int_len	57.1%	1
own_moves – int_len	**67.1%**	**2**
*den_self * (own_moves - int_len)*	60%	3
*den_self * (own_moves - int_len)*	54.3%	4
*up to 42 empty spaces - **return** len(game.get_legal_moves(player))* *up to 28 empty spaces – **return** (own_moves / den_self)* *< 28 empty spaces – **return** float(den_self * (len(own_moves) - int_len)*	50%	5
*up to 32 empty spaces - **return** len(game.get_legal_moves(player))* *up to 14 empty spaces – **return** (own_moves / den_self)* *< 14 empty spaces – **return** float(den_self * (len(own_moves) - int_len)*	55.7%	6
own_moves – int_len	65.7%	7
own_moves – int_len	44.3%	8
10 – own_moves	12.9%	9
own_moves + den_self + int_len	60%	10
own_moves + den_self + int_len	51.4%	11

3. Recommendation:

Evaluation Functions	***Reasons***
up to 32 empty spaces - ***return*** *own_moves* *up to 14 empty spaces* – ***return*** *(own_moves – opp_moves)* *< 14 empty spaces* – ***return*** *(own_moves – int_moves)*	• Consistently gave better win rate • Not too complex to calculate; adds intensive calculations at the end •
own_moves + int_len	• Represents power of player over the opponent • Consistently scored better and there are no intersection of moves represents the original opportunity • Highest of other scores for AB_Custom_2
own_moves - int_len	• Represents loss of power or remaining opportunity in the game • Consistently scored better • Highest of the other scores for AB_Custom_3

4. Supporting Details

Performance of game board: Using the default settings of the tournament, {NUM_MATCHES = 5 # number of matches against each opponent and TIME_LIMIT = 150 # number of milliseconds before timeout}, the average time consumption stands at 12 minute for the whole tournament.

Further Observations: Behavior of one of the custom scores seem to influence the performance of other custom scores (Is it possible? Exclusion – sample trial 9). The sample Trial 9 was tried with an improving score from a constant value, but the scores performed badly when a benchmark score of 10 was used as the highest score when the game reaches depth. It is reasonable to state that imputing an artificial heuristic based on a constant value did not lead to meaningful decisions, just like using random numbers for score (there were few posts in Slack about this.)

Using the location of the current move compared to the center of the board, did not influence the results greatly. Hence, after few trials the usage of self_w, self_h, opp_w & opp_h were dropped from the evaluations.

Not having the # of player's moves (referred as 'own_moves' below) actually brought the results down. (Sample Trial 4 – AB_Custom_2)

Sample Trial 6 has the combination of the best heuristic identified so far (using 'own_moves', 'int_moves', 'den_self' in the score and a score based on number of empty spaces).

MM_improved & AB_improved seemed to be behaving in a similar way. Using the intersection of moves (same moves available for player & inactive_player, referred as 'int_moves' below) helped improving the score (AB_custom_3 @ sample Trial 2 is 67.1%).

5. Trials

The following are the few of the number of trials (out of about 100 different trials, captured here are with some reasonable win score) and the heuristics applied to calculate the scores.

Sample Trial 1:

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

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      8   |  2     9   |  1     8   |  2     8   |  2
    2       MM_Open     5   |  5     2   |  8     2   |  8     4   |  6
    3      MM_Center    7   |  3     8   |  2     8   |  2     7   |  3
    4     MM_Improved   3   |  7     1   |  9     3   |  7     0   |  10
    5       AB_Open     6   |  4     7   |  3     4   |  6     6   |  4
    6      AB_Center    9   |  1     9   |  1     9   |  1     8   |  2
    7     AB_Improved   6   |  4     8   |  2     5   |  5     7   |  3
--------------------------------------------------------------------------
          Win Rate:      62.9%        62.9%        55.7%        57.1%
```

AB_Custom : len(own_moves)

AB_Custom_2 : float(own_moves/ den_self) ; den_self → total blank spaces

AB_Custom_3 : own_moves – int_len ; int_len → intersecting moves possible in the open spaces.

Sample Trial 2:

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

 Match #    Opponent    AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won | Lost    Won | Lost    Won | Lost   Won | Lost
    1       Random        7  |   3      9  |   1      6  |   4     10  |   0
    2       MM_Open       3  |   7      2  |   8      4  |   6      5  |   5
    3       MM_Center     8  |   2      7  |   3      6  |   4     10  |   0
    4       MM_Improved   5  |   5      2  |   8      1  |   9      4  |   6
    5       AB_Open       6  |   4      5  |   5      7  |   3      5  |   5
    6       AB_Center    10  |   0      8  |   2      9  |   1      9  |   1
    7       AB_Improved   4  |   6      3  |   7      1  |   9      4  |   6
-------------------------------------------------------------------------------
          Win Rate:      61.4%        51.4%        48.6%        67.1%

There were 3.0 timeouts during the tournament -- make sure your agent handles
```

AB_Csutom : **return** float(own_moves * (abs(self_h - opp_h) + abs(self_w - opp_w)))

AB_Custom_2: **return** float((own_moves/den_self) * (abs(self_w - opp_w) + abs(self_h - opp_h)))

AB_Custom_3: own_moves – int_len ; int_len → intersecting moves possible in the open spaces.

Sample Trial 3:

```
                    **************************
                        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      7  |   3      6  |   4      9  |   1
    2       MM_Open       4  |   6      3  |   7      2  |   8      1  |   9
    3       MM_Center     7  |   3      3  |   7      7  |   3      7  |   3
    4       MM_Improved   2  |   8      3  |   7      1  |   9      5  |   5
    5       AB_Open       3  |   7      3  |   7      2  |   8      6  |   4
    6       AB_Center     9  |   1      9  |   1      9  |   1      9  |   1
    7       AB_Improved   6  |   4      3  |   7      4  |   6      5  |   5
-------------------------------------------------------------------------------
          Win Rate:      57.1%        44.3%        44.3%        60.0%

There were 5.0 timeouts during the tournament -- make sure your agent handles
```

AB_Csutom : **return** float(len(own_moves) * (abs(self_h - opp_h) ** 2 + abs(self_w - opp_w) ** 2))

AB_Custom_2: **return** float((own_moves - opp_moves) * (abs(self_w - opp_w) + abs(self_h - opp_h)))

AB_Custom_3: **return** float(den_self * (len(own_moves) - int_len))

Sample Trial 4:

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      7  |   3    10  |   0     5  |   5     7  |   3
    2       MM_Open     2  |   8     3  |   7     0  |  10     1  |   9
    3      MM_Center    6  |   4     6  |   4     2  |   8     6  |   4
    4     MM_Improved   4  |   6     3  |   7     0  |  10     4  |   6
    5       AB_Open     5  |   5     2  |   8     0  |  10     7  |   3
    6      AB_Center    9  |   1     9  |   1     9  |   1     8  |   2
    7     AB_Improved   5  |   5     5  |   5     2  |   8     5  |   5
-------------------------------------------------------------------------
           Win Rate:      54.3%        54.3%        25.7%        54.3%

There were 5.0 timeouts during the tournament -- make sure your agent handles
```

AB_Csutom : **return** float(len(own_moves) - int_moves)

AB_Custom_2: **return** float(den_self - opp_moves)

AB_Custom_3: **return** float(den_self * (len(own_moves) - int_len))

Sample Trial 5:

Observation (following image) :

Having hardware costly executions below 28 empty spaces, lead to board execution for 2 hours and 23 minutes approximately. (not sure, if my mac paused the execution as it went in to sleep during this time – **Note**: This was caused by mac's behavior while going to sleep!!.)

But, considering reducing the scope for costlier scores in next run.

```
Start Time: 2017-07-05 22:59:48
                    **************************
                        Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       7  |  3     8  |  2     9  |  1     5  |  5
    2       MM_Open      1  |  9     4  |  6     3  |  7     3  |  7
    3      MM_Center     7  |  3     6  |  4     6  |  4     8  |  2
    4     MM_Improved    1  |  9     4  |  6     4  |  6     2  |  8
    5       AB_Open      4  |  6     5  |  5     8  |  2     3  |  7
    6      AB_Center    10  |  0     9  |  1    10  |  0     9  |  1
    7     AB_Improved    5  |  5     5  |  5     5  |  5     5  |  5
--------------------------------------------------------------------------
          Win Rate:      50.0%        58.6%        64.3%        50.0%

There were 4.0 timeouts during the tournament -- make sure your agent handles

End Time: 2017-07-06 02:23:16
```

AB_Csutom :

up to 42 empty spaces - **return** len(game.get_legal_moves(player))

up to 28 empty spaces – **return** (own_moves – opp_moves)

< 28 empty spaces – **return** (own_moves – int_moves)

AB_Custom_2:

up to 42 empty spaces - **return** len(game.get_legal_moves(player))

< 42 empty spaces – **return** (own_moves / den_self)

AB_Custom_3:

up to 42 empty spaces - **return** len(game.get_legal_moves(player))

up to 28 empty spaces – **return** (own_moves / den_self)

< 28 empty spaces – **return** float(den_self * (len(own_moves) - int_len)

Sample Trial 6:

```
Start Time: 2017-07-06 19:57:19
                    ************************
                        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    10  |   0     8  |   2     9  |   1
    2       MM_Open     2  |   8     3  |   7     5  |   5     5  |   5
    3      MM_Center    9  |   1     8  |   2     8  |   2     8  |   2
    4     MM_Improved   2  |   8     3  |   7     5  |   5     4  |   6
    5       AB_Open     5  |   5     7  |   3     5  |   5     3  |   7
    6      AB_Center    9  |   1     9  |   1     9  |   1     6  |   4
    7     AB_Improved   6  |   4     7  |   3     4  |   6     4  |   6
--------------------------------------------------------------------------
          Win Rate:      60.0%        67.1%        62.9%        55.7%

There were 1.0 timeouts during the tournament -- make sure your agent handles

End Time: 2017-07-06 20:08:59
```

AB_Custom :

up to 32 empty spaces - **return** len(game.get_legal_moves(player))

up to 14 empty spaces – **return** (own_moves – opp_moves)

< 14 empty spaces – **return** (own_moves – int_moves)

AB_Custom_2:

up to 32 empty spaces - **return** len(game.get_legal_moves(player))

< 32 empty spaces – **return** (own_moves / den_self)

AB_Custom_3:

up to 32 empty spaces - **return** len(game.get_legal_moves(player))

up to 14 empty spaces – **return** (own_moves / den_self)

< 14 empty spaces – **return** float(den_self * (len(own_moves) - int_len)

Sample Trial 7:

```
Start Time: 2017-07-08 07:40:38
                        **************************
                             Playing Matches
                        **************************

 Match #    Opponent    AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won  | Lost    Won  | Lost    Won  | Lost   Won  | Lost
    1        Random       3   |   7      9   |   1      7   |   3     9   |   1
    2        MM_Open      1   |   9      1   |   9      3   |   7     2   |   8
    3       MM_Center     2   |   8      7   |   3      7   |   3     7   |   3
    4     MM_Improved     1   |   9      6   |   4      7   |   3     5   |   5
    5        AB_Open      3   |   7      5   |   5      2   |   8     4   |   6
    6       AB_Center     6   |   4      8   |   2      8   |   2    10   |   0
    7     AB_Improved     4   |   6      9   |   1      8   |   2     9   |   1
--------------------------------------------------------------------------------
           Win Rate:      28.6%        64.3%        60.0%        65.7%

There were 5.0 timeouts during the tournament -- make sure your agent handles

End Time: 2017-07-08 07:51:36
```

AB improved : return (own_moves – opp_moves) ^ 2

AB_Csutom :

up to 54 empty spaces - **return** own_moves ^ 2 – opp_moves

up to 34 empty spaces – **return** (own_moves ^2 – int_moves)

< 34 empty spaces – **return** (own_moves ^2 - abs(h - y) + abs(w - x) - int_moves)

AB_Custom_2:

up to 54 empty spaces - **return** len(game.get_legal_moves(player))

up to 34 empty spaces – **return** (own_moves ^2 – int_moves)

< 34 empty spaces – **return** ((own_moves - int_moves)/ den_self)

AB_Custom_3: **return** (own_moves) - int_len)

Sample Trial 8:

```
Start Time: 2017-07-08 12:42:09
                    **************************
                        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     8  |  2     9  |  1     6  |  4
    2       MM_Open     3  |  7     3  |  7     2  |  8     3  |  7
    3      MM_Center    8  |  2     6  |  4     7  |  3     7  |  3
    4     MM_Improved   3  |  7     2  |  8     4  |  6     3  |  7
    5       AB_Open     4  |  6     5  |  5     4  |  6     5  |  5
    6       AB_Center   9  |  1    10  |  0    10  |  0     5  |  5
    7     AB_Improved   6  |  4     3  |  7     4  |  6     2  |  8
----------------------------------------------------------------------
          Win Rate:      60.0%        52.9%        57.1%        44.3%
End Time: 2017-07-08 12:54:13
```

AB_Csutom: **return** float(len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25)

AB_Csutom_2: **return** float((len(own_moves) / den_self) - (int_len / den_self) ** 2)

AB_Csutom_3: **return** float((len(own_moves) - int_len))

Sample Trial 9:

```
Start Time: 2017-07-08 13:04:25
                    **********
                    Playing Matches
                    **********

 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      4  |   6      4  |   6
    2       MM_Open       5  |   5      3  |   7      0  |  10      0  |  10
    3      MM_Center      7  |   3      6  |   4      1  |   9      0  |  10
    4     MM_Improved     6  |   4      0  |  10      0  |  10      0  |  10
    5       AB_Open       3  |   7      4  |   6      0  |  10      1  |   9
    6       AB_Center     9  |   1      9  |   1      2  |   8      3  |   7
    7      AB_Improved    4  |   6      4  |   6      0  |  10      1  |   9
--------------------------------------------------------------------------------
          Win Rate:      62.9%        48.6%        10.0%        12.9%
End Time: 2017-07-08 13:16:31
```

AB_Csutom: **return** float(len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25)

AB_Csutom_2: **return** float(10 - len(game.get_legal_moves(player)) + int_len)

AB_Csutom_3: **return** float(10-len(game.get_legal_moves(player)))

Sample Trial 10:

```
Start Time: 2017-07-08 13:20:26
                        **************************
                             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     8  |   2     9  |   1     8  |   2
    2       MM_Open      1  |   9     3  |   7     5  |   5     4  |   6
    3      MM_Center     8  |   2     6  |   4     8  |   2     9  |   1
    4     MM_Improved    4  |   6     1  |   9     5  |   5     5  |   5
    5       AB_Open      5  |   5     3  |   7     5  |   5     2  |   8
    6       AB_Center    8  |   2     7  |   3    10  |   0     9  |   1
    7     AB_Improved    7  |   3     2  |   8     4  |   6     5  |   5
----------------------------------------------------------------------------
          Win Rate:      60.0%        42.9%        65.7%        60.0%

There were 1.0 timeouts during the tournament -- make sure your agent handles

End Time: 2017-07-08 13:32:08
```

AB_Csutom: **return** float(len(own_moves) / den_self - (int_len + len(opp_moves)) ** 0.25)

AB_Csutom_2: **return** float(len(game.get_legal_moves(player)) + int_len)

AB_Csutom_3: **return** float(len(game.get_legal_moves(player)) + den_self + int_len)

Sample Trial 11:

```
Start Time: 2017-07-08 13:38:30
                        **************************
                             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     10  |   0     9  |   1     9  |   1
    2       MM_Open      1  |   9      4  |   6     4  |   6     3  |   7
    3      MM_Center     6  |   4      7  |   3     6  |   4     5  |   5
    4     MM_Improved    3  |   7      3  |   7     3  |   7     3  |   7
    5       AB_Open      4  |   6      4  |   6     6  |   4     6  |   4
    6      AB_Center    10  |   0     10  |   0     8  |   2     7  |   3
    7     AB_Improved    7  |   3      4  |   6     8  |   2     3  |   7
------------------------------------------------------------------------------
          Win Rate:      57.1%        60.0%        62.9%        51.4%

There were 1.0 timeouts during the tournament -- make sure your agent handles

End Time: 2017-07-08 13:49:59
```

AB_Csutom: **return** float(len(own_moves) / den_self)

AB_Csutom_2: **return** float(len(game.get_legal_moves(player)) + int_len)

AB_Csutom_3: **return** float(len(game.get_legal_moves(player)) + den_self + int_len)