Heuristic analysis

Jared Bowden

The following report provides an overview of 3 different evaluation heuristics proposed for use in the game, Isolation (with a brief overview of a 4th). The relative strength of each approach was tested by comparing the performance of our game agent against a host of AI opponents of increasing strength. In each case, higher performance was defined as a higher percentage win rate of our game agent over opponents.

Baseline testing methods

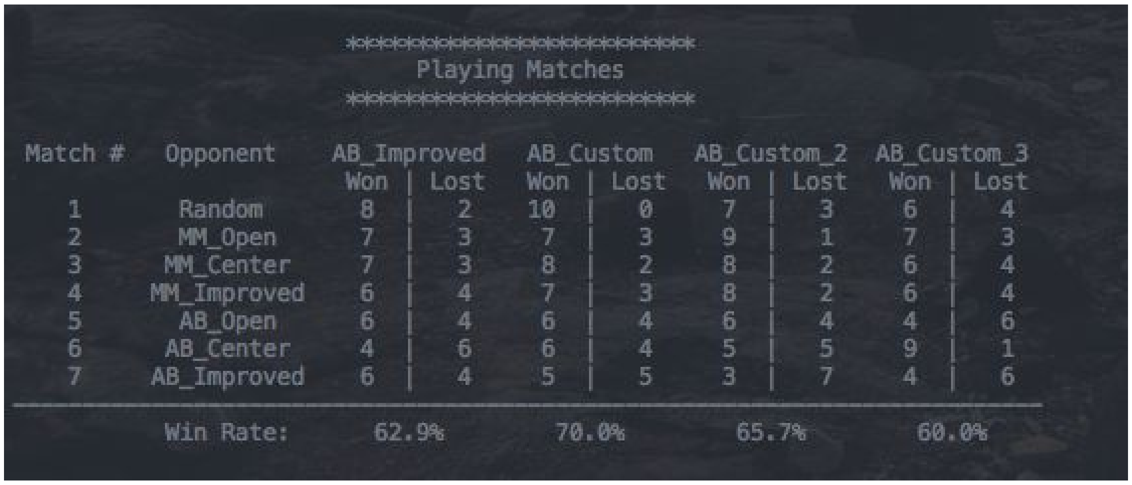
The first objective of the testing procedure was to establish the expected variability across multiple testing events. If the magnitude of variability across tests with identical conditions is large, is may be difficult to distinguish incremental increases in performance associated our evaluation heuristics from the general “noise” of testing.

In the following test, a simple weighted heuristic (Fig 1) was applied in AB_Custom, AB_Custom2, and AB_Custom_3:

```
weight = 2.0
my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
return my_moves - (opponent_moves * weight)
```

Figure 1. Simple weighted heuristic.

As we can see in Figure 2, identical heuristics can return large differences in performance across multiple runs.



```
                         *************************
                              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    10  |   0     7  |   3     6  |   4
    2       MM_Open      7  |   3     7  |   3     9  |   1     7  |   3
    3      MM_Center     7  |   3     8  |   2     8  |   2     6  |   4
    4     MM_Improved    6  |   4     7  |   3     8  |   2     6  |   4
    5       AB_Open      6  |   4     6  |   4     6  |   4     4  |   6
    6      AB_Center     4  |   6     6  |   4     5  |   5     9  |   1
    7     AB_Improved    6  |   4     5  |   5     3  |   7     4  |   6
--------------------------------------------------------------------------
           Win Rate:      62.9%        70.0%        65.7%        60.0%
```

Figure 2. Establishing an understanding of baseline variability across repeated runs. Identical heuristics can return different results across repeated runs. The same heuristic was applied in AB_Custom, AB_Custom2, and AB_Custom_3

The statistical approach to mitigating this variability would involve repeating the testing procedure for each individual heuristic several times, with the hope that signal would emerge as a measure of central tendency from the noise. Unfortunately, the compute time required to complete each run (~10 minutes) makes this approach unattractive.

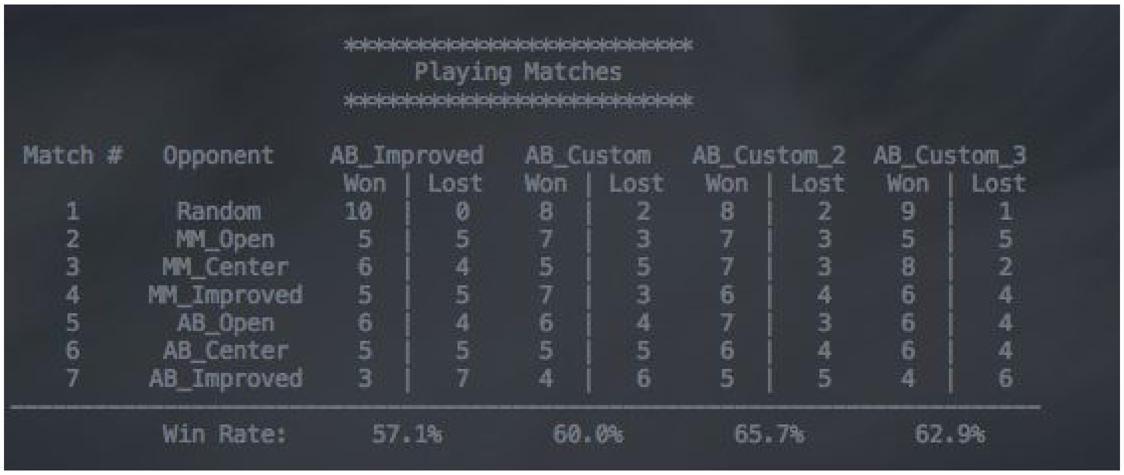
Long test times also limits the use of optimization routines that iterate across a range of different parameters to settle on values that return maximum performance.

With these caveats in mind, the following evaluation heuristics were explored:

1. Simple agent vs opponent weighted heuristic
As shown in Figure 1, adjusting the weight factor as follows:

Column name	**Weight value**
AB_Custom	3
AB_Custom2	4
AB_Custom_3	5

As shown in Figure 3, changes in the weight factor of this heuristic analysis did appear to correlate with significant improvements in performance. The highest performance seen here was with a weight of 4 (64%).



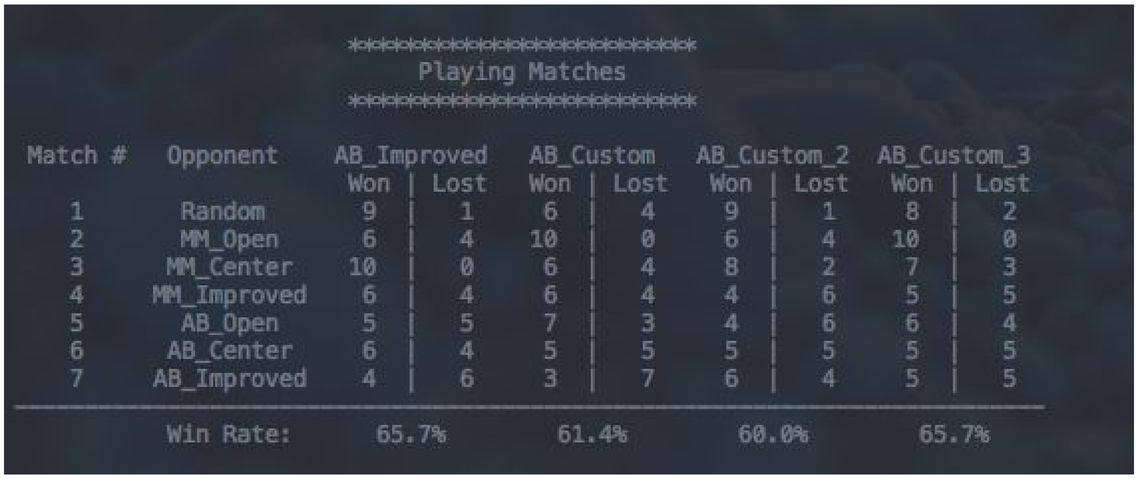
```
                         *************************
                              Playing Matches
                         *************************

 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     8  |   2     9  |   1
    2       MM_Open      5  |   5     7  |   3     7  |   3     5  |   5
    3      MM_Center     6  |   4     5  |   5     7  |   3     8  |   2
    4     MM_Improved    5  |   5     7  |   3     6  |   4     6  |   4
    5       AB_Open      6  |   4     6  |   4     7  |   3     6  |   4
    6      AB_Center     5  |   5     5  |   5     6  |   4     6  |   4
    7     AB_Improved    3  |   7     4  |   6     5  |   5     4  |   6
--------------------------------------------------------------------------
           Win Rate:      57.1%        60.0%        65.7%        62.9%
```

Figure 3. Results of simple weighted heuristic, with different weight values

Earlier baseline testing was conducted with a weight factor of 2, and achieved a maximum performance of 70% (Fig 2), leading to the hypothesis that lower weight factors could possibly result in better performance. In the following test the weights that resulted in the highest performance (2 and 4) were tested against a weight factor of 1, as summarized in the following table:

Column name	**Weight value**
AB_Custom	4
AB_Custom2	2
AB_Custom_3	1

As shown in Figure 4, at least within the range tested, this weighted heuristic does not exhibit improvements in performance worth considering further. The highest score observed here was actually a weight of 1 (equivalent to no weight). Furthermore, the performance of this test appears to have been skewed by a particularly large win rate against an earlier weak opponent (Fig. 4, AB_Custom_3, row 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       9  |  1      6  |  4      9  |  1      8  |  2
    2       MM_Open      6  |  4     10  |  0      6  |  4     10  |  0
    3      MM_Center    10  |  0      6  |  4      8  |  2      7  |  3
    4     MM_Improved    6  |  4      6  |  4      4  |  6      5  |  5
    5       AB_Open      5  |  5      7  |  3      4  |  6      6  |  4
    6      AB_Center     6  |  4      5  |  5      5  |  5      5  |  5
    7     AB_Improved    4  |  6      3  |  7      6  |  4      5  |  5
--------------------------------------------------------------------------
           Win Rate:      65.7%        61.4%        60.0%        65.7%
```

Figure 4. Simple weighted heuristic, with weight factors of 4, 2, and 1, set to AB_Custom, AB_Custom2, and AB_Custom_3, respectively.

2. Distance to the center heuristic

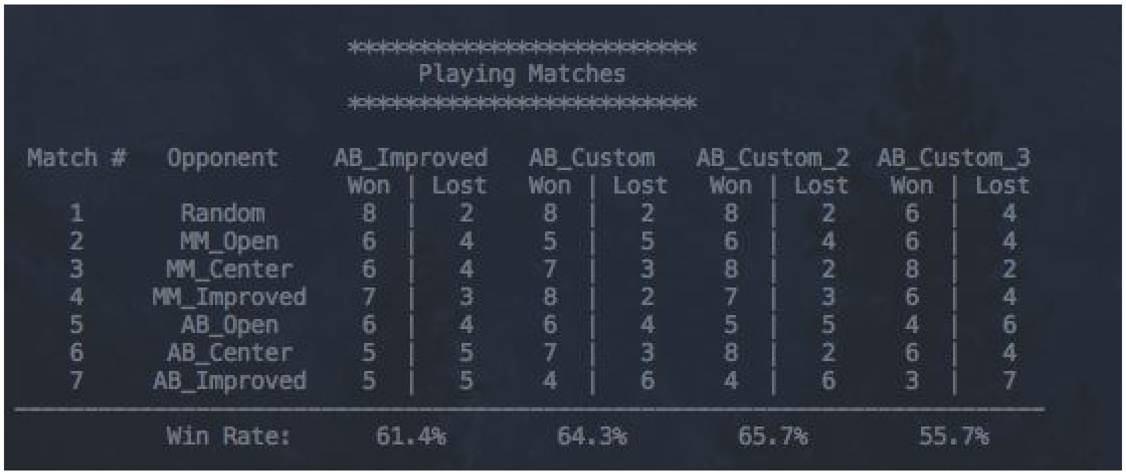
This heuristic was designed to test the hypothesis that branches containing moves closer to the middle of the board offer more down-stream opportunity than moves closer to the edges of the board (Fig 5.). Here, we're computing the total number of moves available to a player, subtracting a "distance from the center" metric. Distance from center is computed as the absolute difference from the center square, averaged across a players available legal moves.

```
def distance_from_center(legal_moves):
       """
            Compute the mean absolute distance of all available legal moves
            from the center square.

            Assumes a 7x7 board
            """
            distance_from_center = [abs(x[0] - 3) + abs(x[1] - 3) for x in legal_moves]

            try:
                return sum(distance_from_center) / float(len(distance_from_center))
            except:
                return 0

my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))

my_abs_center_dist = distance_from_center(game.get_legal_moves(player))
opponent_abs_center_dist = distance_from_center(game.get_legal_moves(game.get_opponent(player)))

# Return within top-level function
# NOTE: in alternative versions of this heuristic, the center_dist values are
# weighted according to specified parameters.
return (my_moves - my_abs_center_dist) - (opponent_moves - opponent_abs_center_dist)
```

Figure 5. Distance to the center heuristic

AB_Custom and AB_Custom2 represent the replication of the same base settings (Fig. 5). In AB_Custom_3, the opponent player's distance from the center metric is weighted by a multiple of 2. See comments in Figure 5.

Column name	Distance from the center weight value
AB_Custom	Null
AB_Custom2	Null
AB_Custom_3	2



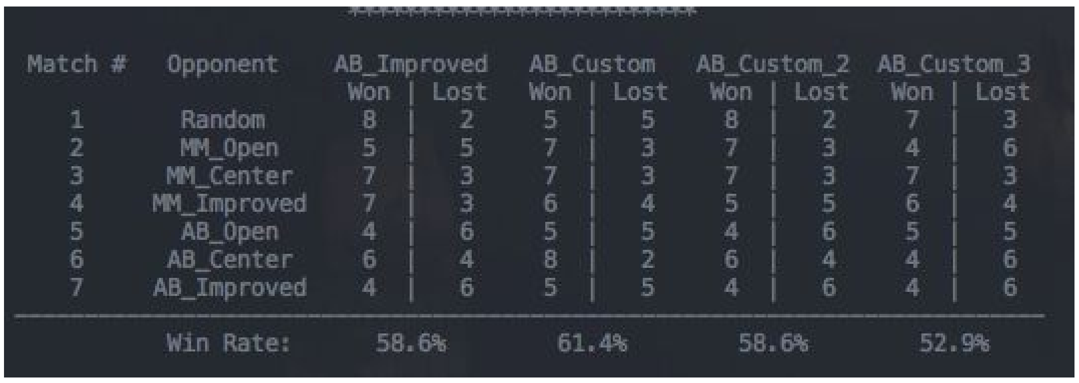
```
                          *************************
                               Playing Matches
                          *************************

 Match #   Opponent    AB_Improved   AB_Custom    AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |  2      8  |  2      8  |  2      6  |  4
    2       MM_Open      6  |  4      5  |  5      6  |  4      6  |  4
    3      MM_Center     6  |  4      7  |  3      8  |  2      8  |  2
    4     MM_Improved    7  |  3      8  |  2      7  |  3      6  |  4
    5       AB_Open      6  |  4      6  |  4      5  |  5      4  |  6
    6      AB_Center     5  |  5      7  |  3      8  |  2      6  |  4
    7     AB_Improved    5  |  5      4  |  6      4  |  6      3  |  7
--------------------------------------------------------------------------
           Win Rate:      61.4%        64.3%        65.7%        55.7%
```

Figure 6. Distance to the center heuristic. AB_Custom and AB_Custom2 represent the replication of the same base settings. In AB_Custom_3, the opponent player's distance from the center metric is weighted by a multiple of 2.

The percentage win rate resulting from this approach did not appear promising. However, the magnitude of mean absolute distance metric is typically quite a bit lower than the total number of available moves (particularly at early moves of the game). This observation lends itself to the hypothesis that more effect could be realized from this approach by increasing the weight factor applied to the absolute distance metric. See comments in Figure 5 for more detail.

Here, a larger weight factor was applied to the opponent's relative distance than player's:

Column name	Distance from the center weight value (player, opponent)
AB_Custom	(5, 7)
AB_Custom2	(6, 8)
AB_Custom_3	(7, 9)



```
 Match #    Opponent     AB_Improved   AB_Custom    AB_Custom_2  AB_Custom_3
                          Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random        8  |  2      5  |  5      8  |  2      7  |  3
    2        MM_Open       5  |  5      7  |  3      7  |  3      4  |  6
    3       MM_Center      7  |  3      7  |  3      7  |  3      7  |  3
    4      MM_Improved     7  |  3      6  |  4      5  |  5      6  |  4
    5        AB_Open       4  |  6      5  |  5      4  |  6      5  |  5
    6       AB_Center      6  |  4      8  |  2      6  |  4      4  |  6
    7      AB_Improved     4  |  6      5  |  5      4  |  6      4  |  6
--------------------------------------------------------------------------
           Win Rate:      58.6%        61.4%        58.6%        52.9%
```
:

Figure 7. Result from weighted distance to the center heuristic.

Overall, the distance to the center heuristic did not yield results worthy of pursuing. The concept of acknowledging the center square in evaluation was applied with greater success in the next iteration of testing.

3. Center square bonus

The distance to the center heuristic is based on the assumption that moves that have an overall closer proximity to the center square are more desirable. The center square bonus heuristic is a simplification of this hypothesis: here, a bonus value (tested across a range of values) is applied if the center square is available as a legal move.

```
def center_classification(legal_moves):
       """
       Helper function to apply bonus to values that include center
       Square in legal moves
       """
       # Set the strength of the bonus value
       Bonus_value = 5

       if (3,3) in legal_moves:
       return bonus_value
      else:
       return 0

my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))

my_center_bonus = center_classification(game.get_legal_moves(player))
opponent_center_bonus = center_classification(game.get_legal_moves(player))

# Return within top-level function
return (my_moves + my_center_bonus) - (opponent_moves - opponent_center_bonus)
```

Figure 8. Center square bonus heuristic

Here, 2 different bonus strengths were compared: 5 and 10. One final (unrelated) heuristic was also prototyped: a value that evaluates as a ratio between moves available to the player, and moves available to the opponent (Fig. 9)

```
my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
```

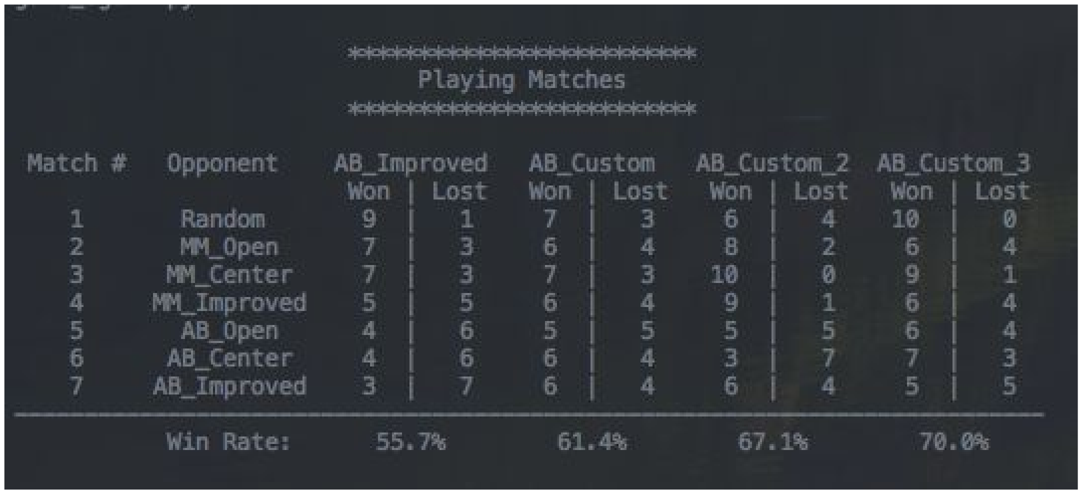
```
Try:
        return my_moves / opponent_moves
except:

# Return within top-level function
return my_moves
```

Figure 9. Ratio of number of player moves compared to opponent moves

The layout used was as follows:

Column name	Comment
AB_Custom	Ratio of number of player moves compared to opponent moves
AB_Custom2	Center square bonus heuristic, bonus value of 5
AB_Custom_3	Center square bonus heuristic, bonus value of 10

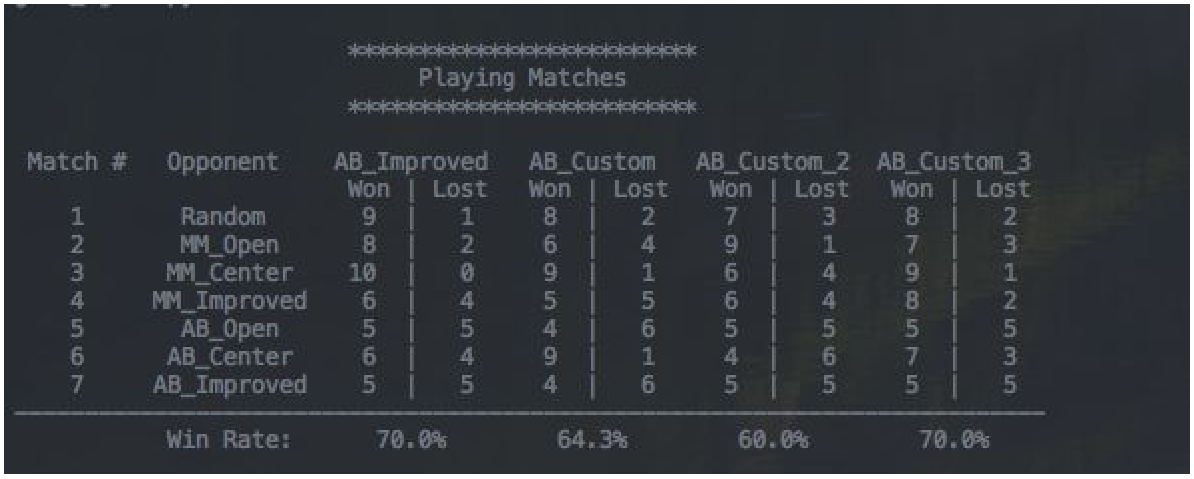
As shown in Figure 10, the center square bonus heuristic (bonus value of 10) yielded maximal results. What's more, this performance was repeated in a second round of repeated sampling (Fig 11).



```
                         *************************
                              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     10  |  0
    2         MM_Open       7  |  3      6  |  4     8  |  2      6  |  4
    3        MM_Center      7  |  3      7  |  3    10  |  0      9  |  1
    4       MM_Improved     5  |  5      6  |  4     9  |  1      6  |  4
    5         AB_Open       4  |  6      5  |  5     5  |  5      6  |  4
    6        AB_Center      4  |  6      6  |  4     3  |  7      7  |  3
    7       AB_Improved     3  |  7      6  |  4     6  |  4      5  |  5
--------------------------------------------------------------------------
           Win Rate:      55.7%        61.4%        67.1%        70.0%
```

Figure 10. Results of ratio of number of player moves compared to opponent moves heuristic, and center square bonus heuristic.

Let's run this again for the sake of consistency.



```
                    *************************
                         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     7  |   3     8  |   2
    2       MM_Open      8  |   2     6  |   4     9  |   1     7  |   3
    3      MM_Center    10  |   0     9  |   1     6  |   4     9  |   1
    4     MM_Improved    6  |   4     5  |   5     6  |   4     8  |   2
    5       AB_Open      5  |   5     4  |   6     5  |   5     5  |   5
    6      AB_Center     6  |   4     9  |   1     4  |   6     7  |   3
    7     AB_Improved    5  |   5     4  |   6     5  |   5     5  |   5
--------------------------------------------------------------------------
           Win Rate:      70.0%        64.3%        60.0%        70.0%
```

Figure 11. Second round of testing: results of ratio of number of player moves compared to opponent moves heuristic, and center square bonus heuristic.

Conclusion

Overall, this analysis demonstrates that considerable variability may exist between rounds of isolation testing. This variability made the process of evaluation of the different heuristic difficult, and required multiple rounds of sampling to establish the consistency of results.

Of the heuristics tested, the Center square bonus heuristic function returned the most consistently high win rate (~70%). This heuristic was selected for use in the final model.

Final heuristic settings implemented in code:

Column name	**Heuristic name**
AB_Custom	Center square bonus heuristic (bonus value of 10)
AB_Custom2	Distance to the center heuristic (no weight)
AB_Custom_3	Simple agent vs opponent weighted heuristic (weight 2)