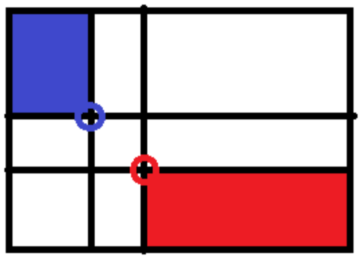
[2.1] Isolation Heuristic Analysis

Thursday, September 21, 2017 9:48 PM

- Comparison of board dominant functions:
  - Purpose
    - To evaluate various board dominance functions using the area of player controlled spaces or territories
  - Functions
    - AB_Custom - _board_dominance(game, player)
      - Function that evaluates how much area a player controls behind him relative to the opponent
    - AB_Custom2 - _improved_board_dominance(game, player)
      - Evaluates the difference between how much area a player controls compared to the opponent
    - AB_Custom3 - _openboard_dominance(game, player)
      - Evaluates the difference between how much open spaces a player controls behind him relative to the opponent
  - Visualization of generic dominance function
    - Depicts the area controlled in blue by the circled blue player compared to the red area controlled by the circled red player. The red player "controls" more of the board in this instance.



  - Results (Two Tournaments)

```
 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    20  |   0    20  |   0
    2        MM_Open     15  |   5    14  |   6    19  |   1    19  |   1
    3       MM_Center    18  |   2    17  |   3    20  |   0    19  |   1
    4      MM_Improved   17  |   3    14  |   6    15  |   5    14  |   6
    5        AB_Open     10  |  10    10  |  10    11  |   9    10  |  10
    6       AB_Center    11  |   9    10  |  10     9  |  11    12  |   8
    7      AB_Improved   10  |  10     4  |  16    10  |  10     7  |  13
--------------------------------------------------------------------------
            Win Rate:      72.1%        63.6%        74.3%        72.1%

 Match #    Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random      18  |   2    20  |   0    20  |   0    20  |   0
    2        MM_Open     16  |   4    17  |   3    17  |   3    14  |   6
    3       MM_Center    20  |   0    18  |   2    19  |   1    20  |   0
    4      MM_Improved   15  |   5    15  |   5    14  |   6    15  |   5
    5        AB_Open     11  |   9     8  |  12    13  |   7    10  |  10
    6       AB_Center    15  |   5    13  |   7    10  |  10    11  |   9
    7      AB_Improved    7  |  13     9  |  11    12  |   8     7  |  13
--------------------------------------------------------------------------
            Win Rate:      72.9%        71.4%        75.0%        69.3%
```

  - Discussion
    - All evaluation functions perform poorer or slightly better than AB_Improved. The use of taking the difference in evaluation scores between players improves the board dominance function just like improved for the open move score function. The high performance against MM_* functions that use only the minmax algorithm is due to the potentially lower search depth, inefficiency of the search algorithm (no pruning), and lack of finding "best" move before the time runs out with iterative deepening. These dominant heuristics does not optimally perform since the interpretation of board dominance may not accurately apply to knights that can jump over spaces and quickly invade the opponents previously controlled area for more

spaces; and especially when dominance itself is not directly related to the isolation objective.

- Comparison of recursive improved score functions:
  - Purpose
    - To build upon AB_Improved and evaluate improved score recursion by using additional move iterations along with counting the number of open spaces available along the paths. Instead of counting the number of open spaces available in one move, the open spaces, or move scores, available n-moves deep, are also accumulated.
  - Experiment 1
    - Functions
      - AB_Custom - _improved_multi_open_move(game, player, max_depth=2, memorized=False)
        - Like the improved_score function, but evaluates by the number of possible open moves 2 moves deep and compares against the number of possible open moves of the component
      - AB_Custom1 - _improved_multi_open_move(game, player, max_depth=3, memorized=False)
        - Evaluates by the number of moves 3 layers deep compared to the opponents'
      - AB_Custom2 - _improved_multi_open_move(game, player, max_depth=3, memorized=True)
        - Evaluates by the number of moves 3 layers deep while remembering the past moves for each path to prevent moving back
    - Results

```
 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    20  |   0    19  |   1
    2       MM_Open     16  |   4    18  |   2    16  |   4    19  |   1
    3      MM_Center    19  |   1    19  |   1    20  |   0    19  |   1
    4     MM_Improved   15  |   5    16  |   4    14  |   6    15  |   5
    5       AB_Open     13  |   7    14  |   6    13  |   7    14  |   6
    6      AB_Center    12  |   8    14  |   6    15  |   5    13  |   7
    7     AB_Improved    8  |  12    15  |   5     8  |  12    10  |  10
--------------------------------------------------------------------------
           Win Rate:      73.6%        82.9%        75.7%        77.9%
```

  - Experiment 2 - increasing max depth
    - Functions
      - AB_Custom - _improved_multi_open_move(game, player, max_depth=3, memorized=False)
      - AB_Custom1 - _improved_multi_open_move(game, player, max_depth=4, memorized=False)
      - AB_Custom2 - _improved_multi_open_move(game, player, max_depth=4, memorized=True)
    - Results

```
 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    20  |   0    19  |   1
    2       MM_Open     17  |   3    18  |   2    17  |   3    19  |   1
    3      MM_Center    19  |   1    20  |   0    18  |   2    18  |   2
    4     MM_Improved   15  |   5    14  |   6    16  |   4    14  |   6
    5       AB_Open      8  |  12    11  |   9     8  |  12    12  |   8
    6      AB_Center    13  |   7    14  |   6    13  |   7    11  |   9
    7     AB_Improved   13  |   7    10  |  10    10  |  10    10  |  10
--------------------------------------------------------------------------
           Win Rate:      75.0%        76.4%        72.9%        73.6%
```

  - Discussion
    - Improving upon the improved_score function appears to be possible by accounting for deeper move recursion and calculating the number of possible moves n-moves deep instead of just counting the number of open moves 1 move deep. Optimal max depth appears to be 2-3, with both max depths beating AB_Improved in both experiments as highest contesters. Accounting for past moves for each move path improve upon the evaluation function, but considerable amount of tests would be needed to separate from statistical noise. The improvement in AB_Improved makes intuitive since as the number of moves available after the next move, and so forth, increases the chance the agent will have additional moves compared to the opponent.

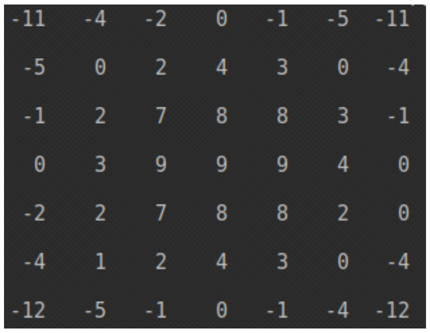
- Comparison of improved score functions using modified score values:
  - Purpose:
    - To evaluate the effect of counting the sum of positions values instead of just the open positions. A scorer function is adde d to the improved score recursion function that decides the score for any given position.

- Functions
  - AB_Custom - _improved_multi_open_move(game, player, max_depth=1, False)
    - Intended to simulate AB_Improved using a version of the improved score function that evaluates the number of moves one-move-level deep
  - AB_Custom1- _improved_multi_open_move(game, player, max_depth=1, False, scorer=_historical_position_value)
    - Like the recursive improved score functions tested above, but places different values on positions based on the scorer function. This particular call simulates the AB_Improved opponent with the special scoring function as it goes only one move deep.
  - AB_Custom2 - _improved_multi_open_move(game, player, max_depth=2, False, scorer=_historical_position_value)
    - Evaluates with the historical position value function going two-moves deep
  - _historical_position_value
    - A function that values a position's score using a historical list of probabilities of that position being involved in a wining game compared to a losing game. An overnight run log of about 8,000 games was compiled to see the portions of winning games a given position was involved compared to losing games using the equation below. The net percentage of winning games that position was involved in was then used to value the position.

      $$Position\ Value = \frac{a - b}{a + b}$$

      - "a" is the number of games that position was used in to win a game
      - "b" is the number of games that position was involved in a losing game

- Visualization
  - Depicts the position values for each space on the game board using the Position Value equation given above after 8,000 games. Values make intuitive sense. Corners with the lowest score are the least likely to generate a winning game as they are easy to get trapped in with only one move available space to exit to once in a corner. The middle areas are the most likely to be involved in a winning game since they afford many available positions, and most of those available positions afford their own available opportunities once moved to.



```
-11   -4   -2    0   -1   -5  -11

 -5    0    2    4    3    0   -4

 -1    2    7    8    8    3   -1

  0    3    9    9    9    4    0

 -2    2    7    8    8    2    0

 -4    1    2    4    3    0   -4

-12   -5   -1    0   -1   -4  -12
```

- Experiment 1
  - Functions (see above)
  - Results

```
 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    20  |   0    19  |   1
    2       MM_Open     14  |   6    15  |   5    17  |   3    16  |   4
    3      MM_Center    19  |   1    17  |   3    20  |   0    19  |   1
    4     MM_Improved   13  |   7    18  |   2    17  |   3    17  |   3
    5       AB_Open     11  |   9    13  |   7    13  |   7    10  |  10
    6      AB_Center    16  |   4    10  |  10     9  |  11    14  |   6
    7     AB_Improved   11  |   9    12  |   8    10  |  10    10  |  10
--------------------------------------------------------------------------
           Win Rate:      73.6%        74.3%        75.7%        75.0%
```

- Experiment 2 - increasing max depth
  - Functions
    - AB_Custom - _improved_multi_open_move(game, player, max_depth=2, False)
    - AB_Custom1- _improved_multi_open_move(game, player, max_depth=2, False, scorer=_historical_position_value)
    - AB_Custom2- _improved_multi_open_move(game, player, max_depth=3, False, scorer=_historical_position_value)
  - Results

```
 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      19  |   1    20  |   0    20  |   0    20  |   0
    2       MM_Open     13  |   7    19  |   1    19  |   1    15  |   5
    3      MM_Center    20  |   0    19  |   1    19  |   1    19  |   1
    4     MM_Improved   18  |   2    19  |   1    17  |   3    15  |   5
    5       AB_Open     10  |  10    16  |   4    11  |   9    11  |   9
    6      AB_Center    12  |   8     8  |  12    13  |   7    13  |   7
    7     AB_Improved    9  |  11    12  |   8    10  |  10    11  |   9
--------------------------------------------------------------------------
           Win Rate:      72.1%        80.7%        77.9%        74.3%
```

- Discussion
  - The addition of a scoring function using the probability of a position being in a winning game relative to a losing game does not appear to have any positive or negative affect on the success of the agent. This may be do to similarity of scoring by open positions and scoring by the historical position values where the amount of moves available for a given position on the board roughly correlates to its historical position value.

- Conclusion and Recommendation for Best Agent
  - The evaluation function that builds upon the AB_Improved agent's improved_score function by counting available moves after moving additional spaces deep was the most successful, with 15/20 wins against AB_Improved in one test using a max move depth of 2. This function with a max depth of 2 also showed 80.7-82.9% success rate in two of the tests above, significantly more than the 72.0 - 73.6% success rate achieved by AB_Improved. Not only does this function and settings win head to head the majority of the time against AB_Improved, but it also wins more often when considering other tested heuristics. This improvement is due to looking further ahead in the improved score function, since best available moves now for a current position may not translate to best available moves in the next position. Improvements over AB_Improved are also observed with modifications of the recursive improved scoring function for other max depths, path memorization, and a scorer function for the value of each position. The use of board dominance, where territory control was deemed by the area of the board behind a given player and their opponent, provided the lowest success rate, most likely because it only indirectly evaluates the goal of the game in having more spaces available than your opponent.