# Heuristics Overview

While exploring the new to me Isolation game, I tried various heuristics as evaluation functions. Since I am quite familiar with Chess and Computer Chess, although evaluation functions are important in Chess they are not so important as standalone functions. Their most important role is guiding search, initiating extensions or pruning

For evaluation the influence of the various evaluations functions I tested, I used the provided tournament.py

At first I played two round robin tournaments covering all considered heuristics

# Tournament Tables for the heuristics used

**table 1:**

| Match | Opponent | AB_Improved | | Combined_1 | | Combined_2 | | Combined_3 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Won | Lost | Won | Lost | Won | Lost | Won | Lost |
| 1 | Random | 37 | 3 | 36 | 4 | 36 | 4 | 35 | 5 |
| 2 | MM_Open | 29 | 11 | 34 | 6 | 35 | 5 | 31 | 9 |
| 3 | MM_Center | 36 | 4 | 31 | 9 | 34 | 6 | 36 | 4 |
| 4 | MM_Improved | 31 | 9 | 29 | 11 | 30 | 10 | 26 | 14 |
| 5 | AB_Open | 19 | 21 | 19 | 21 | 16 | 24 | 22 | 18 |
| 6 | AB_Center | 29 | 11 | 23 | 17 | 23 | 17 | 24 | 16 |
| 7 | AB_Improved | 20 | 20 | 22 | 18 | 17 | 23 | 15 | 25 |
| | Win Rate | 71.8% | | 69.3% | | 68.2% | | 67.5% | |

**table 2:**

| Match | Opponent | AB_Improved | | Distance | | Combined_4 | | Common Moves | |
|---|---|---|---|---|---|---|---|---|---|
| | | Won | Lost | Won | Lost | Won | Lost | Won | Lost |
| 1 | Random | 37 | 3 | 36 | 4 | 39 | 1 | 33 | 7 |
| 2 | MM_Open | 29 | 11 | 28 | 12 | 32 | 8 | 27 | 13 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | MM_Center | 36 | 4 | 34 | 6 | 36 | 4 | 32 | 8 |
| 4 | MM_Improved | 27 | 13 | 29 | 11 | 26 | 14 | 22 | 18 |
| 5 | AB_Open | 23 | 17 | 22 | 18 | 17 | 23 | 15 | 25 |
| 6 | AB_Center | 20 | 20 | 20 | 20 | 19 | 21 | 20 | 20 |
| 7 | AB_Improved | 16 | 24 | 21 | 19 | 20 | 20 | 18 | 18 |
| | Win Rate | 67.1% | | 67.9% | | 67.5% | | 59.8% | |

### Statistical Significance of Results.

To make any valid assumptions about the effectiveness of the heuristics, one should play **a lot** of games, because as one can see, even self playing can result in results far away from the expected 50%

This is a very common situation in evaluating chess engines and one can read more here

To do that evaluation I implemented the function mentioned in the link to evaluate if there is any significance in the results (you can find the function in los.py in the repository)

Of course for any meaningful results much longer matches should be played. For this reason I created 2 tournament matches 1000 games long each , to get some meaningful results for the heuristics.

**match 1:**

| **Match** | **Opponent** | **Combined_1** | | **Combined_2** | | **Combined_3** | | **Combined_4** | |
|---|---|---|---|---|---|---|---|---|---|
| **.....** | **.....** | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** |
| 1 | AB_Improved | 503 | 497 | 504 | 496 | 516 | 484 | 0 | 0 |

**match 2:**

| **Match** | **Opponent** | **Inv_Distance** | | **Distance** | | **Open_move** | | **Common Moves** | |
|---|---|---|---|---|---|---|---|---|---|
| **.....** | **.....** | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** |
| 1 | AB_Improved | 461 | 539 | 0 | 0 | 476 | 524 | 0 | 0 |

## Heuristics Reference and performance analysis

1. **Open move score**: Number of legal moves available for the player.
    - **Performance**: fast and simple
    - **ELO Difference**:-16.69
    - **Likelihood of Superiority**: 6.45%
2. **Mobility**: The difference between number of own legal moves - opponent legal moves. This is also used in the tournament.py by the AB_Improved agent

- **Performance**: fast and simple. This is the ID_improved heuristic
- **ELO Difference**: 0
- **Likelihood of Superiority**: 0

3. **Common Moves**: This simple heuristic gives a higher score to positions where both players have squares they can both go. It counts the common squares and give that as a score for the position

    - **Performance**: slightly slower
    - **ELO Difference**: -61.79
    - **Likelihood of Superiority**: 1.30%

4. **Distance**: This returns the distance between the two players in the board. Trying to stay as far away from the opponent as possible

    - **Performance**: fast and simple
    - **ELO Difference**: -13.20
    - **Likelihood of Superiority**: 11.47%

5. **Inverse Distance**: This returns the 1/distance between the two players in the board. Trying to stay as close to the opponent as possible

    - **Performance**: fast and simple
    - **ELO Difference**: -27.155
    - **Likelihood of Superiority**: 0.68%

6. **Combined_1**: A linear combination of (1), (2) and (3).

    - **Performance**: slightly slower
    - **ELO Difference**: 2.08
    - **Likelihood of Superiority**: 57.52%

7. **Combined_2**: This heuristic gives (number of legal moves)^2 - number of opponent legal moves, when the board is more than half full (that means Its more important to have open moves when the board is full) and (number of legal moves) - (number of opponent legal move)^2 when the board is not , which means I care more about restricting the opponent in the early stages. This gives a slight improvement over AB_Improved and it can be more fine tuned

    - **Performance**: much slower than ID_improved
    - **ELO Difference**: 2.08
    - **Likelihood of Superiority**: 57.52%

8. **Combined_3**: Expanding on the previous idea this returns:

```
    if moves_to_board >= 0.8:  # closer to endgame (most squares are occupied
        return float(len(own_moves) * 4 - len(opponent_moves))
    elif 0.4 <= moves_to_board < 0.8:
        return float(len(own_moves) * 2 - len(opponent_moves))
    else:
        return float(len(own_moves) - len(opponent_moves) * 2)
```

```
    which does not change much compare to the previous heuristic


- **Performance**: much slower than ID_improved
- **ELO Difference**: 11.12
- **Likelihood of Superiority**: 84.42%
```

9. **Combined_4**: This was just a crazy idea as a shot in the dark. In short it calculates the following:

   ```
    score = float(own_moves ** 2 / (1 + opp_moves)) + float(own_moves / (1 +
   opp_moves ** 2))
   ```

   - **Performance**: fast and simple
   - **ELO Difference**:
   - **Likelihood of Superiority**:

## Summary

The results are summarized in the following graph

Based on the above analysis, the heuristic of choice (based on those shallow depth matches) is because its the only one showing statistically significant evidence of superiority against the reference heuristic