**Heuristic Analysis for Isolation Board Game**

## **Executive Summary:**

The main objective of this study is to perform a heuristic analysis to find a good evaluation function for the Isolation game-playing agent. Various combinations of probable affecting factors were considered to make the evaluation/heuristic function.

Finally, ‘Heuristic\_4’, an improved version of the given ‘improved\_score’ heuristic was found to perform the best, consistently beating ‘AB\_Improved’ and other opponent agents. The sample size of fair games played w.r.t to each opponent was increased to 30. ‘Heuristic\_4’ performs almost 10% better than ‘AB\_Improved’ agent overall. Also, ‘Heuristic\_4’ beats ‘AB\_Improved’ agent as opponent almost 57% of the times (multiple runs resulted close to this figure of wins).

## **Introduction:**

This report analyses various possible heuristic functions for the board game ‘Isolation’ which is a deterministic, two-player game of perfect information.

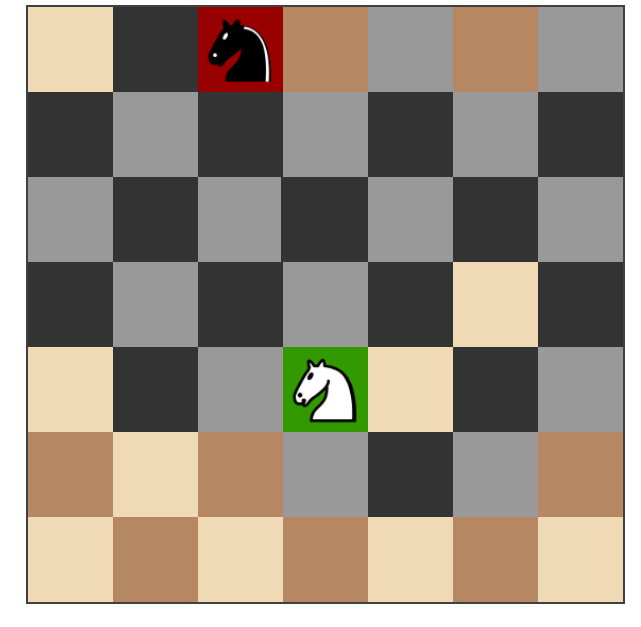
The evaluation function given is:

The objective of this study to find an improved evaluation function which consistently performs better than ‘improved\_score’ defined above.

There were many potential variables I considered for defining various evaluation functions. Some of them are as given below:

1. my\_moves – legal moves available to a player at a given state of the game
2. opp\_moves – legal moves available to the opponent player at a given state of the game
3. percent\_game\_completed – a factor to account for the progression of the game
4. dist\_from\_opp – calculates the euclidean distance of player from the opponent
5. dist\_from\_center – calculates the distance of player from center of game board.
6. my\_max\_moves – the maximum no: of moves given a set of legal moves available for the player
7. opp\_max\_moves – the maximum no: of moves given a set of legal moves available for the opponent
8. my\_sum\_future\_moves – sum of current legal moves + count of legal moves for each of these current legal moves for player
9. opp\_sum\_future\_moves – sum of current legal moves + count of legal moves for each of these current legal moves for opponent
10. my\_no\_future\_moves – count of any losing moves in the list of current legal moves for the player
11. opp\_no\_future\_moves – count of any losing moves in the list of current legal moves for the opponent.

Most of the game loses analysed using the ‘sample\_players.py’ file and the visualization API shows partitioning or walling effect of the board. I found predicting partitioning with simple evaluation functions very difficult. Hence, instead of trying to predict partitioning effect, my attempt has been to change strategy for end game to find the maximum moves to last longest possible.



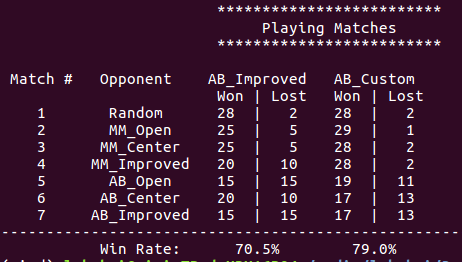
**Figure**: Example partitioning/walling effect (black knight loses and hence white knight wins)

The Heuristics below detail some of the efforts towards finding a better evaluation function which beats the ‘improved\_score’ function consistently.

### **Heuristic\_1:**

**Definition:**

The maximum legal moves when we start the game is 8. As the game progresses, the maximum legal moves reduces to 7 or less. This heuristic is an addition over the given heuristic ‘improved\_score’. I felt this as a good starting point as ‘imporved\_score’ = len(my\_moves) – len(opp\_moves) was not really aware of the game progression. At the beginning of the game when there are multiple legal moves available, it really doesn’t make a difference when the player chooses to chase after the opponent or not. There are enough moves for both players. Towards the end of the game though, it is better for the player to maximize his own moves and minimize the opponents move, wherever possible.

**Performance:**

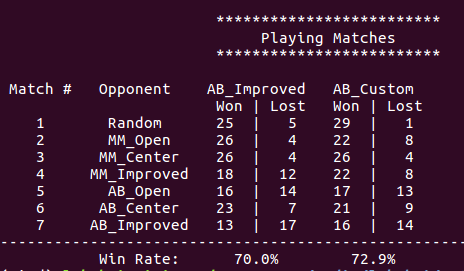
This heuristic seems to be working very well as shown above. This function beats ‘minimax\_only’ agents easily. It also does perform adequately well even when the opponent is AB\_Improved. Overall, a good improvement.

**Heuristic\_2:**

**Definition:**

Trying to improve over heuristic\_1, the next option considered was to develop a heuristic based on count of current legal moves and maximum possible future legal moves out of the current legal move. Similarly, the same counts for the opponent is also calculated. My idea was to include a ‘move quality’ factor into the heuristic so that the player bases its moves not just on the count of legal moves but also on the quality of each of these legal moves. If any current legal move results in a no future move (game loss), that count is subtracted from the above total score. The same is not done for the opponent as we can assume the opponent will most likely not choose that move anyways and hence considering this does not give us much of an advantage (For clarity, I did try both options but this performed better).

**Performance:**



Although this heuristic performs better than ‘AB\_improved’, it does not beat ‘heuristic\_1’. I believe ‘heuristic\_2’ includes a lot more computations than ideal for a good evaluation function.

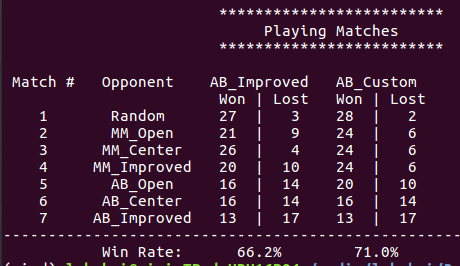
**Heuristic\_3:**

**Definition:**

From this heuristic onwards, I decided to split the game into two. One strategy for the first 90% of the game and another for the last 10% of the game. The idea was to include an end game tactics. Towards the end of the game when the player would majorly be surrounded by blocked cells, it is essential for the player to maximize its own survival and not consider chasing the opponent. The factor of 90% (%game\_completion) was arrived at after various experiments.

For the last 10% of the game, I found that len(my\_moves) + my\_max\_moves works the best, out of the various combinations I tried, for maximizing player benefit and increasing survival. This is probably because it includes the quality of a move using ‘my\_max\_moves’ factor together with len(my\_moves). For the rest 90% of the game, I changed heuristic 2 to include total current moves and sum of possible future legal moves for each of the current move.

**Performance:**



This seems to be performing better than heuristic\_2 but still not as good a heuristic\_1 which is much simpler in complexity and better in performance.

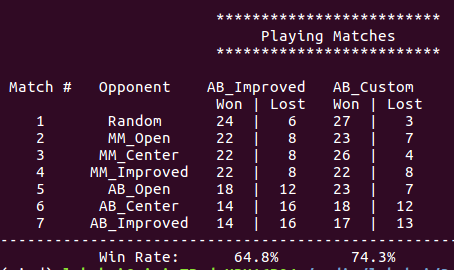
**Heuristic\_4:**

**Definition:**

The next attempt at improving the heuristic function was to preserve the simplicity of the heuristic\_1 and add the end game tactics from heuristic\_3.

(Various combinations of variables from heuristic\_3 were tried but none performed better than heuristic\_1. Hence I decided to go back to improving heuristic\_1).

**Performance:**



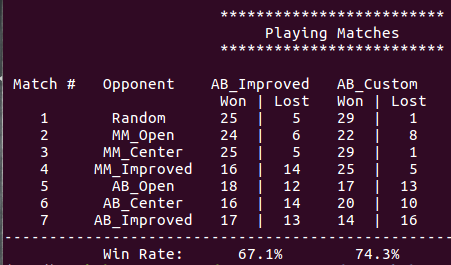
As seen from figure above, the addition of end game tactic did improve heuristic\_1 slightly. Overall, this heuristic performs best so far against the given agents including AB\_Improved.

**Heuristic\_5:**

**Definition:**

From heuristic\_4, I thought of replacing len(my\_moves) and len(opp\_moves) with total moves for both players (current count of legal moves and future possible count of legal moves). I also found that instead of 0.9 as the game completion factor, 0.8 worked better in this case. The end game tactic performed better with my\_total\_moves instead of my\_max\_moves + len(my\_moves).

**Performance**:

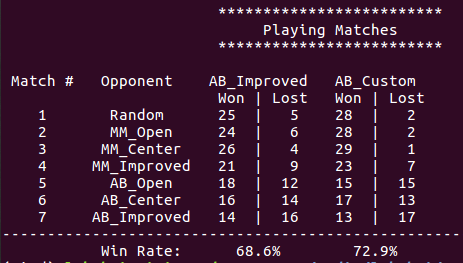


The heuristic is simple and performs well too. But it still doesn’t beat the simpler heuristic\_1 or heuristic\_4 (improved version of heuristic\_1). Hence, no further trial to improve this was done.

**Heuristic\_6:**

**Definition:**

This was another attempt to improve heuristic\_4. I added a factor to prefer moves further away from the centre at the beginning of the game so that the more moves are left near the centre towards the end of the game. This might avoid cornering or island partitioning towards the end of the game. For the last 10% of the game, the player just tries to maximize his moves based on the maximum future moves available like the previous heuristics.

**Performance:**

This heuristic also does perform better overall but doesn’t beat the simplicity of heuristic\_1 or improved heuristic\_4. I believe this is due to the fact that adding too many variables makes it computationally complex and hence loses out on performance.

## **Conclusion:**

Multiple combinations of probable variables listed at the beginning of this report were tried. Most of the non-performing ones were excluded from this report. Top six performing heuristics were selected to be presented for this study.

The heuristics are ordered as follows in the game\_agent.py file based on their descending order of performance.

1. custom\_score() - Heuristic\_4
2. custom\_score\_2() - Heuristic\_1
3. custom\_score\_3() - Heuristic\_5
4. custom\_score\_4() - Heuristic\_3
5. custom\_score\_5() - Heuristic\_6
6. custom\_score\_6() - Heuristic\_2

**Performance Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **AB\_Improved (% wins)** | **Custom Heuristic** | **Custom\_score (% wins)** | **Difference** |
| 64.8 % | Heuristic\_ 4 | 74.3 % | 9.5 % |
| 70.5 % | Heuristic\_ 1 | 79.0 % | 8.1 % |
| 67.1 % | Heuristic\_ 5 | 74.3 % | 7.2 % |
| 66.2 % | Heuristic\_ 3 | 71.0 % | 4.8 % |
| 68.6 % | Heuristic\_ 6 | 72.9 % | 4.3 % |
| 70.0 % | Heuristic\_ 2 | 72.9 % | 2.9 % |

Heuristic\_1 and Heuristic\_4 performs the best as shown below.

**Figure**: Heuristic Performance % difference w.r.t ‘improved\_score’

Although heuristic\_1 is the simplest, I would chose its improved version heuristic\_4 even though it adds only a bit more performance. This is due to the following:

1. Heuristic\_4 considers an end game strategy which I believe is very essential for success in board games. Instead of removing this, I would like to try to improve this strategy further as future work.
2. Heuristic\_4 although a bit more complex than heuristic\_1 is still simple enough to calculate compared to the various other combinations tried.
3. Heuristic\_4 performs the best, consistently beating not just minimax and other alpha-beta (AB) agents but also the ‘AB\_Improved’ opponent.
4. Heuristic\_4 maximizes player options by choosing max\_moves along with len(my\_moves) towards the end of the game. This adds a ‘quality’ factor to the move and helps the player to go after moves with maximum future moves, hence, maximizing its options for survival.
5. In code, heuristic\_4 also considers the option of avoiding one length moves which although might have the maximum future moves but is in the opponent’s move\_list. Otherwise, the opponent might move into this position essentially ending the game with a loss for the player.

Improving the heuristics further but at the same time keeping it computationally simple is a challenging task. Adopting different strategies for various game progression states by optimizing various combinations of the heuristics might be a good future work.