# Heuristic Analysis

## Adversarial Game Playing Agent for Isolation

Udacity Artificial Intelligence Nanodegree

by: Mayank Prakash

# Introduction:

Isolation is a deterministic two player board game in which a player or agent alternately moves single piece from one cell to another on the board. An occupied cell stays blocked for the remainder of the game. The first player who doesn't have a valid position to move on the board loses. In this specific project, the movements of each of the pieces are restricted to an L-shape.
In order to implement an agent that plays the isolation game, following algorithms are implemented:

- Minimax Search
- Alpha-Beta Pruning
- Iterative Deepening with Alpha-Beta

In order to score, evaluation function is implemented as heuristics in order to evaluate the performance of the agent. The heuristics defined are:

- custom_score():
- custom_score_2():
- custom_score_3():

# Evaluation:

**tournament.py** is run in order to evaluate the performance of the heuristic defined as 'Student'. custom_score() contains the best performing heuristic function. The opponent against which the 'Student' agent plays are as follows:

1. Random
2. MM_Open
3. MM_Center
4. MM_Improved
5. AB_Open
6. AB_Center
7. AB_Improved

# Opponents:

1. Random: agent that randomly chooses from list of available options.
2. MM_Open: Minimax player with open_move_score heuristics with search depth 3.
3. MM_Center: Minimax player with center_move_score heuristics with search depth 3.
4. MM_Improved: Minimax player with improved_score heuristics with search depth 3.
5. AB_Open: AlphaBeta player using Iterative Deepening Alpha-Beta Search with open_move_score heuristics.
6. AB_Center: AlphaBeta player using Iterative Deepening Alpha-Beta Search with center_move_score heuristics.
7. AB_Improved: AlphaBeta player using Iterative Deepening Alpha-Beta Search with improved_score heuristics.

## Performance:

- Student_1: this is more of an aggressive approach where the agent tries chasing its opponent to try and minimise the opponent's legal moves. The evaluation is done as follows:

$$score = agentsMove - 1.5 * opponentsMove$$

The Student_1 here in this case performed poorly when compared to ID_Improved, and also it performed poorly in case of test agent MM_Improved.

| Opponent | ID_Improved: Win \| Loss | AB_Custom: Win \| Loss |
|---|---|---|
| Random | 10 \| 0 | 10 \| 0 |
| MM_Open | 8 \| 2 | 8 \| 2 |
| MM_Center | 7 \| 3 | 10 \| 0 |
| MM_Improved | 8 \| 2 | 3 \| 7 |
| AB_Open | 6 \| 4 | 6 \| 4 |
| AB_Center | 6 \| 4 | 5 \| 5 |
| AB_Improved | 6 \| 4 | 5 \| 5 |
| Win Rate: | 72.9% | 67.1% |

- Student_2: weighted heuristics is used to evaluate the performance of the game playing agent in this case, where agent is penalised for any choice it makes that results in reduced chances of its win. The evaluation is done as follows:

$$score = agentsMove^2 - 1.5 * opponentsMove^2$$

The Student_2 performs better when compared to ID_Improved with marginal difference in winning of the game played against test agents.

| Opponent | ID_Improved: Win \| Loss | AB_Custom: Win \| Loss |
|---|---|---|
| Random | 8 \| 2 | 9 \| 1 |
| MM_Open | 6 \| 4 | 9 \| 1 |
| MM_Center | 8 \| 2 | 10 \| 0 |
| MM_Improved | 5 \| 5 | 7 \| 3 |
| AB_Open | 5 \| 5 | 7 \| 3 |
| AB_Center | 5 \| 5 | 4 \| 6 |
| AB_Improved | 8 \| 2 | 6 \| 4 |
| Win Rate: | 64.3% | 74.3% |

- Student_3: defensive heuristics is used as to finally compare the performance of the agent against ID_Improved. In this case the game agent tries defensive approach by trying to maximise its own number of legal moves rather than chasing the opponent. The evaluation of this is done as follows:

$$score = 1.5 * agentsMove - opponentsMove$$

The Student_3 here performs slightly better than ID_Improved agent so this can also be a viable option.

| Opponent | ID_Improved: Win \| Loss | AB_Custom: Win \| Loss |
|---|---|---|
| **Random** | 9 \| 1 | 9 \| 1 |
| **MM_Open** | 7 \| 3 | 8 \| 2 |
| **MM_Center** | 10 \| 0 | 9 \| 1 |
| **MM_Improved** | 5 \| 5 | 7 \| 3 |
| **AB_Open** | 4 \| 6 | 3 \| 7 |
| **AB_Center** | 5 \| 5 | 7 \| 3 |
| **AB_Improved** | 6 \| 4 | 5 \| 5 |
| **Win Rate** | 65.7% | 68.6% |

## **Conclusion:**

Out of the above three heuristics, which were tested against ID_Improved, **Student_2** is selected for custom_score().

A. Student_2 has marginal difference in its performance when compared with ID_Improved, which provides a viable reason that it will outperform ID_Improved most of the times.

B. Student_3 could have also been chosen, but there isn't a very significant difference in their performance so appropriate number of trails are required to be sure whether it is reliable or not, which isn't the case with Student_2.

C. Despite the fact that Student_2 is computationally expensive when compared to Student_1 and Student_3, which might affect the number of iterations in Iterative Deepening but still it performs better than both the defined heuristics, so computation doesn't create an issue.