

---

# HEURISTIC ANALYSIS

---

For an Adversarial Game Playing Agent for Isolation

Submitted by  
**Sumit Binnani**  
as part of Artificial Intelligence Nanodegree, Udacity

# TABLE OF CONTENTS

---

Synopsis .....	2
Custom Heuristics .....	3
1.    Heuristic 1: Minimizing Opponent's Moves (see code) .....	3
2.    Heuristic 2: Maximizing Player's Moves (see code) .....	3
3.    Heuristic 3: Maximizing Ratio of Player to Opponent Move (see code) .....	3
4.    Heuristic 4: Minimizing Ratio of Opponent to Player Move (see code) .....	3
5.    Heuristic 5: Combining Heuristic 3 and 4 (see code) .....	3
6.    Heuristic 6: Weighted Combination of Heuristic 3 and 4 (see code) .....	4
7.    Heuristic 7: Weighted Combination of Heuristic 3 and 4 (see code) .....	4
Evaluating Heuristics .....	5
Results .....	6
Appendices .....	7
A. Appendix: Evaluation Result .....	7

# SYNOPSIS

---

The project aims at developing an adversarial search agent to play the game "Isolation". This project report focusses on the heuristics to be used in A\* Search for minimax and alphabeta pruning.

Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent is declared the winner.

This project uses a version of Isolation where each agent is restricted to L-shaped movements (like a knight in chess) on a rectangular grid (like a chess or checkerboard). The agents can move to any open cell on the board that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. Movements are blocked at the edges of the board (the board does not wrap around), however, the player can "jump" blocked or occupied spaces (just like a knight in chess).

Additionally, agents will have a fixed time limit each turn to search for the best move and respond. If the time limit expires during a player's turn, that player forfeits the match, and the opponent wins. These rules are implemented in the `isolation.Board` class provided in the repository.

# CUSTOM HEURISTICS

---

## 1. HEURISTIC 1: MINIMIZING OPPONENT'S MOVES ([SEE CODE](#))

The heuristic is based on the logic that opponent's moves should be minimized. It can be mathematically expressed as:

$$\text{len}(\text{my available moves}) - \alpha \text{len}(\text{available opponent moves}), \quad \text{where } \alpha \in (1, \infty)$$

The value of  $\alpha$  was empirically chosen as 1.5.

## 2. HEURISTIC 2: MAXIMIZING PLAYER'S MOVES ([SEE CODE](#))

The heuristic is based on the logic that player's moves should be maximized. It can be mathematically expressed as:

$$\alpha \text{len}(\text{my available moves}) - \text{len}(\text{available opponent moves}), \quad \text{where } \alpha \in (1, \infty)$$

The value of  $\alpha$  was empirically chosen as 1.5.

## 3. HEURISTIC 3: MAXIMIZING RATIO OF PLAYER TO OPPONENT MOVE ([SEE CODE](#))

The heuristic is based on the logic that player should have more moves in comparison to opponent. It can be mathematically expressed as:

$$\frac{\text{len}(\text{my available moves})}{\text{len}(\text{available opponent moves})}$$

## 4. HEURISTIC 4: MINIMIZING RATIO OF OPPONENT TO PLAYER MOVE ([SEE CODE](#))

The heuristic is based on the logic that opponent should have less moves in comparison to player. It can be mathematically expressed as:

$$- \frac{\text{len}(\text{available opponent moves})}{\text{len}(\text{my available moves})}$$

## 5. HEURISTIC 5: COMBINING HEURISTIC 3 AND 4 ([SEE CODE](#))

Can be mathematically expressed as:

$$\frac{\text{len}(\text{my available moves})}{\text{len}(\text{available opponent moves})} - \frac{\text{len}(\text{available opponent moves})}{\text{len}(\text{my available moves})}$$

Maximizing above equation is equivalent to maximizing:

$$[\text{len}(\text{my available moves})]^2 - [\text{len}(\text{available opponent moves})]^2$$

The latter form has been implemented in the code.

## 6. HEURISTIC 6: WEIGHTED COMBINATION OF HEURISTIC 3 AND 4 ([SEE CODE](#))

Can be mathematically expressed as:

$$\frac{\text{len}(\text{my available moves})}{\text{len}(\text{available opponent moves})} - \alpha \frac{\text{len}(\text{available opponent moves})}{\text{len}(\text{my available moves})}, \quad \text{where } \alpha \in (1, \infty)$$

Maximizing above equation is equivalent to maximizing:

$$[\text{len}(\text{my available moves})]^2 - \beta [\text{len}(\text{available opponent moves})]^2, \quad \text{where } \beta \in (1, \infty)$$

The latter form has been implemented in the code with  $\beta$  chosen as 1.5 empirically.

## 7. HEURISTIC 7: WEIGHTED COMBINATION OF HEURISTIC 3 AND 4 ([SEE CODE](#))

Can be mathematically expressed as:

$$\alpha \frac{\text{len}(\text{my available moves})}{\text{len}(\text{available opponent moves})} - \frac{\text{len}(\text{available opponent moves})}{\text{len}(\text{my available moves})}, \quad \text{where } \alpha \in (1, \infty)$$

Maximizing above equation is equivalent to maximizing:

$$\beta [\text{len}(\text{my available moves})]^2 - [\text{len}(\text{available opponent moves})]^2, \quad \text{where } \beta \in (1, \infty)$$

The latter form has been implemented in the code with  $\beta$  chosen as 1.5 empirically.

# EVALUATING HEURISTICS

---

The tournament.py script is used to evaluate the effectiveness of heuristic. The script measures relative performance of player in a round-robin tournament against several other pre-defined agents.

The performance of time-limited iterative deepening search is hardware dependent (faster hardware is expected to search deeper than slower hardware in the same amount of time). The script controls for these effects by also measuring the baseline performance of an agent called "ID\_Improved" that uses Iterative Deepening and the improved\_score heuristic from sample\_players.py.

The tournament opponents are listed below:

- Random: An agent that randomly chooses a move each turn.
- MM\_Null: CustomPlayer agent using fixed-depth minimax search and the null\_score heuristic
- MM\_Open: CustomPlayer agent using fixed-depth minimax search and the open\_move\_score heuristic
- MM\_Improved: CustomPlayer agent using fixed-depth minimax search and the improved\_score heuristic
- AB\_Null: CustomPlayer agent using fixed-depth alpha-beta search and the null\_score heuristic
- AB\_Open: CustomPlayer agent using fixed-depth alpha-beta search and the open\_move\_score heuristic
- AB\_Improved: CustomPlayer agent using fixed-depth alpha-beta search and the improved\_score heuristic
- ID\_Improved: CustomPlayer agent using iterative alpha-beta search and the improved\_score heuristic
- Student1: CustomPlayer agent using iterative alpha-beta search and the [heuristic 1](#)
- Student2: CustomPlayer agent using iterative alpha-beta search and the [heuristic 2](#)
- Student3: CustomPlayer agent using iterative alpha-beta search and the [heuristic 3](#)
- Student4: CustomPlayer agent using iterative alpha-beta search and the [heuristic 4](#)
- Student5: CustomPlayer agent using iterative alpha-beta search and the [heuristic 5](#)
- Student6: CustomPlayer agent using iterative alpha-beta search and the [heuristic 6](#)
- Student7: CustomPlayer agent using iterative alpha-beta search and the [heuristic 7](#)

Since, running only a few matches gave different results, the number of matches were increased from 20 to 500 to get more sample points.

# RESULTS

---

The performance of various agents is as follow:

Agent	Performance	Rank
ID_Improved	61.94%	8
Student1	65.89%	2
Student2	65.41%	4
Student3	64.21%	6
Student4	63.51%	7
Student5	64.48%	5
Student6	66.46%	1
Student7	65.45%	3

All the custom heuristics perform better than ID\_Improved by a reasonable margin as can be seen in the above table.

The *custom\_score* function have been implemented to execute [heuristic 6](#) as:

1. It outperforms all other heuristics with win-rate of 66.46% which is higher than all the other heuristics being considered.
2. It depends on current state of the problem only and no additional memory (history) is required.
3. It is comparable with time taken to calculate other heuristics in consideration.
4. It is easier to implement and involves merely a few operations.
5. It aligns with the intuition that the player having higher chances of winning would have higher available moves as compared to the opponent.

The raw evaluation result can be found in A. Appendix: Evaluation Result.

# APPENDICES

---

## A. APPENDIX: EVALUATION RESULT

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic

functions. The `ID\_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

```
*****
Evaluating: ID_Improved
*****
```

Playing Matches:

```
-----
Match 1: ID_Improved vs Random Result: 1724 to 276
Match 2: ID_Improved vs MM_Null Result: 1395 to 605
Match 3: ID_Improved vs MM_Open Result: 1012 to 988
Match 4: ID_Improved vs MM_Improved Result: 952 to 1048
Match 5: ID_Improved vs AB_Null Result: 1297 to 703
Match 6: ID_Improved vs AB_Open Result: 1150 to 850
Match 7: ID_Improved vs AB_Improved Result: 1142 to 858
```

Results:

```
-----
ID_Improved 61.94%
```

```
*****
Evaluating: Student1
*****
```

Playing Matches:

```
-----
Match 1: Student1 vs Random Result: 1736 to 264
Match 2: Student1 vs MM_Null Result: 1458 to 542
Match 3: Student1 vs MM_Open Result: 1136 to 864
Match 4: Student1 vs MM_Improved Result: 1036 to 964
Match 5: Student1 vs AB_Null Result: 1411 to 589
Match 6: Student1 vs AB_Open Result: 1230 to 770
Match 7: Student1 vs AB_Improved Result: 1218 to 782
```

Results:

```
-----
Student1 65.89%
```

```
*****
```



Evaluating: Student2  
\*\*\*\*\*

Playing Matches:

-----  
Match 1: Student2 vs Random Result: 1745 to 255  
Match 2: Student2 vs MM\_Null Result: 1433 to 567  
Match 3: Student2 vs MM\_Open Result: 1106 to 894  
Match 4: Student2 vs MM\_Improved Result: 1041 to 959  
Match 5: Student2 vs AB\_Null Result: 1388 to 612  
Match 6: Student2 vs AB\_Open Result: 1235 to 765  
Match 7: Student2 vs AB\_Improved Result: 1209 to 791

Results:

-----  
Student2 65.41%

\*\*\*\*\*

Evaluating: Student3  
\*\*\*\*\*

Playing Matches:

-----  
Match 1: Student3 vs Random Result: 1724 to 276  
Match 2: Student3 vs MM\_Null Result: 1412 to 588  
Match 3: Student3 vs MM\_Open Result: 1092 to 908  
Match 4: Student3 vs MM\_Improved Result: 1018 to 982  
Match 5: Student3 vs AB\_Null Result: 1336 to 664  
Match 6: Student3 vs AB\_Open Result: 1210 to 790  
Match 7: Student3 vs AB\_Improved Result: 1197 to 803

Results:

-----  
Student3 64.21%

\*\*\*\*\*

Evaluating: Student4  
\*\*\*\*\*

Playing Matches:

-----  
Match 1: Student4 vs Random Result: 1726 to 274  
Match 2: Student4 vs MM\_Null Result: 1392 to 608  
Match 3: Student4 vs MM\_Open Result: 1098 to 902  
Match 4: Student4 vs MM\_Improved Result: 986 to 1014  
Match 5: Student4 vs AB\_Null Result: 1317 to 683  
Match 6: Student4 vs AB\_Open Result: 1178 to 822  
Match 7: Student4 vs AB\_Improved Result: 1195 to 805

Results:

-----  
Student4 63.51%

\*\*\*\*\*

Evaluating: Student5

\*\*\*\*\*

Playing Matches:

-----

Match 1:	Student5	vs	Random	Result: 1719 to 281
Match 2:	Student5	vs	MM_Null	Result: 1439 to 561
Match 3:	Student5	vs	MM_Open	Result: 1088 to 912
Match 4:	Student5	vs	MM_Improved	Result: 998 to 1002
Match 5:	Student5	vs	AB_Null	Result: 1354 to 646
Match 6:	Student5	vs	AB_Open	Result: 1222 to 778
Match 7:	Student5	vs	AB_Improved	Result: 1207 to 793

Results:

-----

Student5                      64.48%

\*\*\*\*\*

Evaluating: Student6

\*\*\*\*\*

Playing Matches:

-----

Match 1:	Student6	vs	Random	Result: 1741 to 259
Match 2:	Student6	vs	MM_Null	Result: 1520 to 480
Match 3:	Student6	vs	MM_Open	Result: 1112 to 888
Match 4:	Student6	vs	MM_Improved	Result: 1059 to 941
Match 5:	Student6	vs	AB_Null	Result: 1418 to 582
Match 6:	Student6	vs	AB_Open	Result: 1245 to 755
Match 7:	Student6	vs	AB_Improved	Result: 1209 to 791

Results:

-----

Student6                      66.46%

\*\*\*\*\*

Evaluating: Student7

\*\*\*\*\*

Playing Matches:

-----

Match 1:	Student7	vs	Random	Result: 1714 to 286
Match 2:	Student7	vs	MM_Null	Result: 1435 to 565
Match 3:	Student7	vs	MM_Open	Result: 1138 to 862
Match 4:	Student7	vs	MM_Improved	Result: 1084 to 916
Match 5:	Student7	vs	AB_Null	Result: 1369 to 631
Match 6:	Student7	vs	AB_Open	Result: 1199 to 801
Match 7:	Student7	vs	AB_Improved	Result: 1224 to 776

Results:

-----

Student7                      65.45%