

# Evaluating Heuristics

## Summary

Three heuristic functions have been compared by using the supplied `tournament.py` script. I chose to base my heuristic functions on the number of blank spaces around the player. The overall results are as follows: \* `ID_Improved` scored 60.54% percent \* `Blank` scored 62.50%% \* `Blank IMP` scored 65.89% \* `Blank MOV` scored 67.57%

With a 7 point gap, I have created a player with a different approach that performs better than `ID_Improved` with the provided `tournament.py` script.

### A note on `tournament.py`

The implementation of the tournament script is random-based, meaning the results for each simulation is not reproducible.

## Evaluating `Blank MOV` against `ID_Improved`

```
Playing Matches:
-----
Match 1: Student Free Space vs ID_Improved    Result: 212 to 188

Results:
-----
Student Free Space      53.00%
```

When pitching the two agents against each other, it seems my agent barely beats `ID_Improved`.

## Detailed results for `ID_Improved`

Formula: `Moves - Opponent moves`

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

Playing Matches:
-----
Match 1: ID_Improved vs    Random    Result: 63 to 17
Match 2: ID_Improved vs    MM_Null   Result: 48 to 32
Match 3: ID_Improved vs    MM_Open   Result: 47 to 33
Match 4: ID_Improved vs MM_Improved Result: 36 to 44
Match 5: ID_Improved vs    AB_Null   Result: 50 to 30
Match 6: ID_Improved vs    AB_Open   Result: 43 to 37
Match 7: ID_Improved vs AB_Improved Result: 52 to 28
```

```
Results:
```

```
-----
```

```
ID_Improved      60.54%
```

Note that `ID_Improved` seems to be inferior to `MM_Improved`. This indicates that, at least on my hardware, the Iterative Deepening approach of `ID_Improved` does not give the expected speed up compared to a naive 3 level minimax approach. But looking at `ID_Improved` vs `AB_Improved` we do see an improvement in using Iterative Deepening compared to a level 5 alphabeta search. This discrepancy could also indicate that, given the `Improved` heuristic function, we do not see any particular gains for going deeper into the game state. This discrepancy warrants further investigation.

Given that `ID_Improved` beats all the `AB_*` agents, I see an indication that the iterative deepening approach of alphabeta search is an improvement over regular alphabeta search.

## Detailed results for `Blank`

Formula: `Blank spaces`

Given the restrivtive move for how the players move in this Isolation variant, I suspected that the number of blank spaces surrounding a player could be an indicator of how good a move it is, as the neighbouring cells for a given move would be available in moves ahead. I sought inspiration from this [https://en.wikipedia.org/wiki/Knight\\_\(chess\)](https://en.wikipedia.org/wiki/Knight_(chess)), but assigning the value 1 to each field.

```
*****
```

```
Evaluating: Blank
```

```
*****
```

```
Playing Matches:
```

```
-----
```

Match 1:	Blank	vs	Random	Result: 67 to 13
Match 2:	Blank	vs	MM_Null	Result: 57 to 23
Match 3:	Blank	vs	MM_Open	Result: 41 to 39
Match 4:	Blank	vs	MM_Improved	Result: 36 to 44
Match 5:	Blank	vs	AB_Null	Result: 53 to 27
Match 6:	Blank	vs	AB_Open	Result: 53 to 27
Match 7:	Blank	vs	AB_Improved	Result: 43 to 37

```
Results:
```

```
-----
```

```
Blank      62.50%
```

Again, we see that the `MM_Improved` proves quite the challenge. But worthy to note, is that the `Blank` heuristic is strong against the `Open` heuristic, indicating that I might be right in my hunch about using number of blank space as opposed to number of moves.

## Detailed results for **Blank IMP**

Formula: `Blank spaces - Opponent blank spaces`

The naive approach performed pretty well, but lets try the same tactic as in the `Improved` heuristic and subtract the opponent's blank spaces. This means, that in a situation where the number of blank spaces is equal between moves, it would break the tie by limiting the number of blank spaces available to the opponent.

```
*****
Evaluating: Blank IMP
*****

Playing Matches:
-----
Match 1:  Blank IMP  vs   Random      Result: 66 to 14
Match 2:  Blank IMP  vs   MM_Null     Result: 59 to 21
Match 3:  Blank IMP  vs   MM_Open     Result: 49 to 31
Match 4:  Blank IMP  vs  MM_Improved  Result: 42 to 38
Match 5:  Blank IMP  vs   AB_Null     Result: 56 to 24
Match 6:  Blank IMP  vs   AB_Open     Result: 46 to 34
Match 7:  Blank IMP  vs  AB_Improved  Result: 51 to 29

Results:
-----
Blank IMP          65.89%
```

Overall, this agent did better than the `Blank` agent. Not by much, but enough to indicate that breaking the ties by limiting the opponent's blank spaces is a good idea.

## Detailed results for **Blank MOV**

Formula: `Blank spaces / Opponent moves`

This heuristic builds upon the previous heuristic. Now we only break the ties if we actively restrict the opponent's available moves. But not only that, the division ensures that if we with a move can block the one of the opponent's legal moves, we will prioritise that move, even if it means fewer blank spaces.

```
*****
Evaluating: Blank MOV
*****

Playing Matches:
-----
Match 1:  Blank MOV  vs   Random      Result: 70 to 10
Match 2:  Blank MOV  vs   MM_Null     Result: 61 to 19
Match 3:  Blank MOV  vs   MM_Open     Result: 47 to 33
Match 4:  Blank MOV  vs  MM_Improved  Result: 41 to 39
```

```
Match 5:    Blank MOV vs    AB_Null    Result: 52 to 28
Match 6:    Blank MOV vs    AB_Open    Result: 53 to 27
Match 7:    Blank MOV vs AB_Improved  Result: 51 to 29
```

```
Results:
```

```
-----
```

```
Blank MOV          66.96%
```

Overall, this agent did better than the `Blank` and `Blank IMP` agents. Not by much, so I'm hesitant to draw any conclusions due to the randomness of the `tournament.py` script.

## Future work

- Investigate the values of each field surrounding the agent when evaluating the blank spaces
- Better tournament evaluation, for reproducible comparisons.