**Heuristics Analysis**

For the Isolation Game Agent Project I used more than three different heuristics. I mark with an asterisk (\*) the one I decided to choose:

**H1:** X2 Weighted Score.

**H2:** X4 Weighted Score.

**H3:** Center Score.

**H4:** Proportion Score.

**H5\*:** H3 + H2.

All heuristics were tested using two different computers:

**PC1:** Linux Ubuntu 64-Bit. 16 GB RAM. Core i7 Processor.

**PC2:** Linux Ubuntu 64-Bit. 16 GB RAM. Core i5 Processor.

**Chosen Heuristic**

For my implementation I used H5, a combination of H3 for the first moves in the game and H2 for the rest. The results with this heuristic were better than the rest. Other heuristics worked and gave better results than the ID\_Improved agent, but they were not as good as H2.

**Heuristic Tournament Results**

**H1: X2 Weighted Score**

For this heuristic I used the following formula:

**own\_moves – (2 \* opponent\_moves)**

The overall performance was good, but I stated to notice some cases where my agent did not performed well and even loose against the ID\_Improved agent. This looses were very rare though.

**Results:**

|  |  |  |
| --- | --- | --- |
| **ATTEMPT** | **PC1** | **PC2** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

**H2: X4 Weighted Score**

For this heuristic I used the following formula:

**own\_moves – (4 \* opponent\_moves)**

After using the H1 heuristic I decided to start increasing the weight constant. I tweaked the values until I found a point where the weight was giving me sufficiently good results. Too high values performed badly. Anything below 2 was not good enough. The final desition was to set te weight with a value of 4.

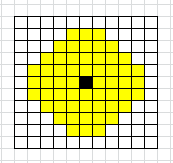
In comparison with H1, H2 offered much better performance. I also thought this could be a good way to approach the Horizon Effect problem in a more aggressive fashion.

**Results:**

|  |  |  |
| --- | --- | --- |
| **ATTEMPT** | **PC1** | **PC2** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

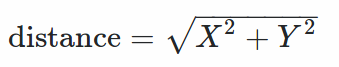
**H3: Center Score**

For this heuristic I had to first find a way to find the distance between two points in a grid assuming of course that the second point will always be the center of it. Figure 1 illustrates what I nedded.



**Figure 1:** All yellow squares are examples of what could represent a good move. The closer to the center square the better given that more moves are available from the center. However those moves could be only available in early stages of the game.

After doing a quick search I found a formula that could be used on H3 (Figure 2), however the version I used to compute the distance is slightly different.

****

**Figure 2:** Distance of one point to the center of the grid, if the two points are X grids apart horizontally and Y grids apart vertically. The version used on this heuristic subtracts the half of the board width (h) to X and Y, what give us the folloing formula: sqrt( (x-h)^2 + (y-h)^2).

Therefore score function used is:

**opponent\_distance - own\_distance**

This heuristic performed better than ID\_Improved only if it was applied at the first stages of the game, but even like that the difference between both agents was very narrow. As I said before, those moves were only available for a limited time. That's why I implemented H3 only at the very beginning of the game. But still, wasn't good enough.

**Results:**

|  |  |  |
| --- | --- | --- |
| **ATTEMPT** | **PC1** | **PC2** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

**H4: Proportion Score**

This heuristic is the proportion of moves available for each player with respect to all available moves

For this heuristic I used the following formula:

**my\_proportion \* 10 - opponent\_proportion \* 10**

This heuristic worked good on average, however I got some looses more frequently than H1. Also some extra validations needed to be added in order to avoid errors.

In order to improve performance of H4 I decided to add a weighted value to the opponents move value.

Results weren't better, the difference between the too agents was even smaller. So I stuck with the regular values.

**Results:**

|  |  |  |
| --- | --- | --- |
| **ATTEMPT** | **PC1** | **PC2** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

**H5\*: H3 + H2**

This heuristic uses H3 for the first 6 turns and then applies H2 for the rest of the turns. The formula can be simply described like this:

if move\_count > 6:

return H2()

else:

return H3()