Heuristic Analysis for "Isolation" Game Agent

1. Methodology

The starting point was to lay out all the metrics that could be useful in identifying winning paths while searching the game tree. The tournament was run with each of these metrics in isolation and sample games were analysed to understand the strengths and weaknesses of each of them. The idea was then to combine the metrics that proved to be the most useful to build up a smarter heuristic that could outperform the sample heuristics given during the course. To simplify combining metrics that are different in value space, such as metrics that calculate distances vs metrics that calculate number of moves, all metrics were standardised to 0-1 range by dividing each value to the maximum they can take.

The metrics that gave promising results are summarised below. This is then followed by the final three heuristics that were chosen as the final strategies to be submitted as part of the "Build a Game-Playing Agent" project.

2. Raw Metrics

2.1. Open Moves: This is the heuristic used by one of the sample agents. The function simply returns the number of moves available to the active player.

$$open\_moves_{player}$$

2.2. Distance to Centre: This is another heuristic leveraged by the sample agents. The squared Manhattan distance to the centre block is used.

$$(row_{player} - row_{centre})^2 + (column_{player} - column_{centre})^2$$

2.3. Distance to Blanks: An alternative to the "Distance to Centre" metric that keeps track of where the blank blocks are concentrated at and moves the agent that way. This is done by finding the centre point of all blank blocks as opposed to the centre point of the board.

$$(row_{player} - \frac{\sum row_{blank}}{count_{blank}})^2 + (column_{player} - \frac{\sum column_{blank}}{count_{blank}})^2$$

2.4. Distance to Opponent: The square Manhattan distance to the opponent. This metric can be used either to run away from the opponent to avoid being trapped, or to build an aggressive strategy to follow the opponent closely.

$$(row_{player} - row_{opponent})^2 + (column_{player} - column_{opponent})^2$$

2.5. Improved Open Moves: This is a derivation of the *Open Moves* metric that combines the open moves for each player.

$$open\_moves_{player} - open\_moves_{opponent}$$

2.6. Distance to Target: This metric tries to place the agent in between the centre of the board and the opponent, hoping to create a barrier to isolate the opponent away from the centre of the board. Hence, "target" in this context refers to the middle-point between opponent's location and the centre of the board.

$$row_{target} = \frac{\left(\frac{\sum row_{blank}}{count_{blank}} + row_{opponent}\right)}{2} \qquad column_{target} = \frac{\left(\frac{\sum column_{blank}}{count_{blank}} + column_{opponent}\right)}{2}$$

$$(row_{player} - row_{target})^2 + (column_{player} - column_{target})^2$$

3. Final Heuristics

3.1. Custom Score: The most effective heuristic in tests was a combination of *Improved Open Moves* and *Distance to Blanks*. *Improved Open Moves* consistently proved to be a strong heuristic despite its simplicity, and most of the testing was performed on modifications to this metric to make it more robust. One place that this heuristic fell short was the beginning of the game, when being closer to the centre (of blanks) was also of high importance. Failing to pick more central moves over moves to the edges that had more moving opportunities caused the agent to be trapped in edges and corners early in the game.

On the other hand, being close to the centre proved to be less important through the end of the game when the blanks are scattered around the board and being at centre doesn't increase chances of finding an open path. This required using a dynamic weight for this metric and making sure the weight gradually decayed as the game progressed. The simplest way to achieve this was to use the percentage of blank squares as a weighting scheme.

$$open\_moves_{player} - open_{moves_{opponent}} - distance_{blanks} * \left(\frac{count_{blank}}{(board_{width} * board_{height})}\right) * 0.5$$

Note that the weight was halved by multiplying it with 0.5. This was to reduce the effect of the *Distance to Blanks* metric further, as tests proved that weighting it higher causes the agent to pick suboptimal moves more often.

This heuristic yielded around 70% win rate against the CPU agents created by the tournament class given in the project and consistently outperformed the AB_Improved agent. However, it was also observed that occasionally the win rate had significant drops. This was primarily caused by starting move sequences that tricked the heuristic to pick squares that had high number of open moves around it, but were within isolated parts of the board that limited the number of moves after it. This highlighted the importance of picking a time-effective heuristic function to be able to go deeper in the search tree within the allowed time frame to avoid picking moves that lead to dead-ends within a couple moves.

3.2. Custom Score 2: A more aggressive heuristic was picked as the second scoring function. The idea was to keep close to the opponent while maintaining high number of open moves to the player. This is to increase the chances to trapping the opponent in an isolated part of the board. Similar to the first heuristic, a further fix weighting was used to increase the importance of Open Moves heuristic relative to the Distance to Opponent heuristic.

$$distance_{opponent} + open\_moves_{player} * 5$$

The tests showed mixed success with this heuristic and although it gave promising performance in some games, it lacked consistency.

3.3. Custom Score 3: The final heuristic that was tested was a modification to the second heuristic in an attempt to maintain a more consistent success rate. The Distance to Opponent metric was replaced with Distance to Target metric to avoid following the opponent to hard to get out of corners. This yielded higher success rate as anticipated. However, the results weren't as good as the results of the first heuristic function.

4. Conclusion and Future Improvements

The tests showed that each metric performed well in different scenarios and the main difficulty was combining these metrics in an efficient way. This could be seen as picking different strategies for different cases, similar to how a human player would play. Using a linear function with fixed/decaying weights was a promising improvement on using raw metrics. This could be improved further by employing machine learning techniques to optimise the weights of each of the metric over thousands/millions of games as opposed to updating them manually by trial and error. A further improvement could be to directly use a machine learning algorithm as the heuristic function. One example could be to use a simple decision tree, that could decide which metric to use based on the state of the board identified through a pre-defined set of features.