Heuristic Analysis for Game Playing Agent

Vatsal Srivastava

The project aims at developing an adversarial search agent to play the game **Isolation**.

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.

Custom Heuristics:

1. **aggressive_heuristic (AB_Custom_2) – Minimizing opponent's moves**

This heuristic is based on the notion that at any time, the number of moves with the opponent must be minimized. The mathematical expression for this can be represented as:

$$len(self\_moves) - \alpha * len(opponent\_moves)$$

where α has been chosen as 5.0 after some experimentation.

2. **aggressive_heuristic_2 (AB_Custom_3) – Minimizing opponents moves and maximizing our moves**

This heuristic is based on the notion that at any time, the number of moves with the opponent must be minimized and some effort must be made to maximize our moves. The mathematical expression for this can be represented as:

$$len(self\_moves)^2 - \alpha * len(opponent\_moves)^2$$

where α has been chosen as 5.0 after some experimentation.

3. **defensive_heuristic (AB_Custom_4) – Maximizing our moves**

This heuristic is based on the notion that at any time, the number of our moves must be maximized. The mathematical expression for this can be represented as:

$$\alpha * len(self\_moves) - len(opponent\_moves)$$

where α has been chosen as 5.0 after some experimentation.

4. **ratio_distance_heuristic (AB_Custom_5) – Maintain Moves Ratio in our favor and keep our player near the center and away from the edges**

This heuristic was based on two notions. First, at any point, the number of moves for our player must be more than the number of moves for our opponent. Second, since most Isolation games end with the player getting restricted to the edges, we should try to keep our player close to the center. The mathematical expression can be represented as:

$$\alpha * distance\_from\_center + \beta * \frac{len(self\ moves)}{len(opponent\ moves)}$$

where α has been chosen as -0.5 and β has been chosen as 1.5 after some experimentation.

5. **custom_distance_heuristic (AB_Custom_6) – Maximize our moves and keep away from opponent**

This heuristic was based in the notion that is we try to maximize the Euclidean Distance between our player and the opponent, then we are less likely to be blocked. The mathematical expression can be represented as:

$$len(self\_moves) - len(opponent\_moves) + \alpha * dist\_from\_opponent$$

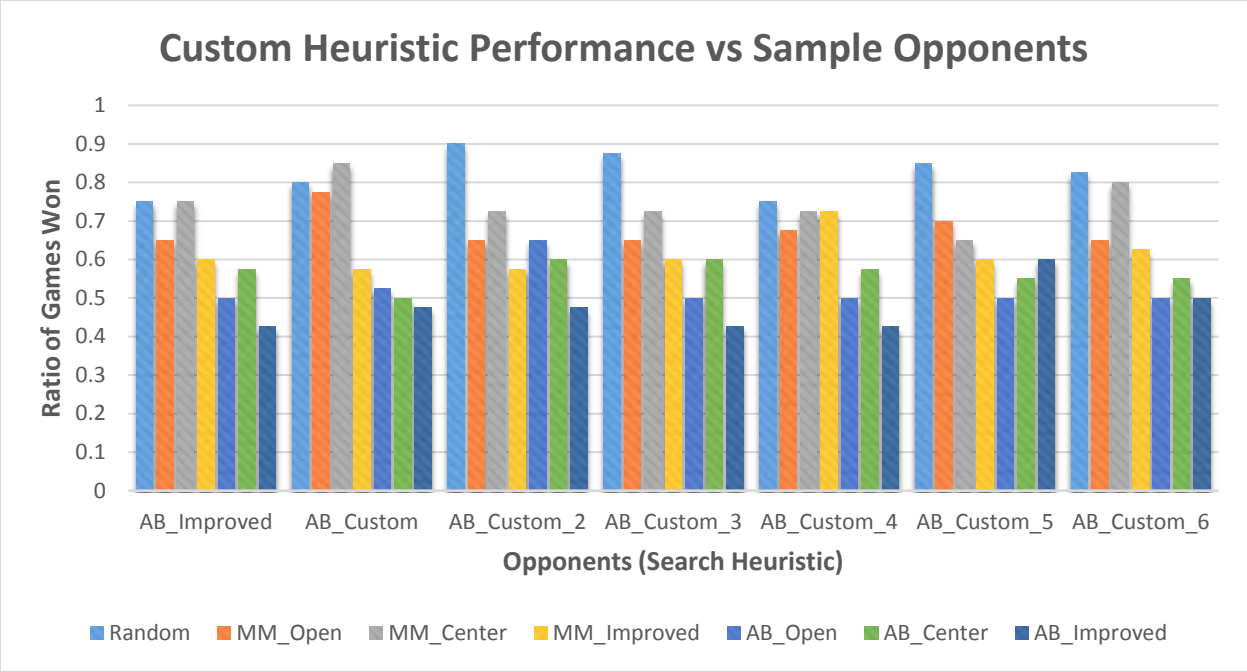
where α has been chosen as 0.5 after some experimentation.

Performance Evaluation:

The table below shows the program output.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3		AB_Custom_4		AB_Custom_5		AB_Custom_6	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	30	10	32	8	36	4	35	5	30	10	34	6	33	7
2	MM_Open	26	14	31	9	26	14	26	14	27	13	28	12	26	14
3	MM_Center	30	10	34	6	29	11	29	11	29	11	26	14	32	8
4	MM_Improved	24	16	23	17	23	17	24	16	27	13	24	16	25	15
5	AB_Open	20	20	21	19	26	14	20	20	20	20	20	20	20	20
6	AB_Center	23	17	20	20	24	16	24	16	23	17	22	18	22	18
7	AB_Improved	17	23	19	21	19	21	17	23	17	23	24	16	20	20
	Win Rate:	60.7%		64.3%		65.4%		62.5%		61.8%		63.6%		63.6%	

Your ID search forfeited 1730.0 games while there were still legal moves available to play.



As can be seen from the table and the graph above, the performance against the Random opponent is always the best. On average, more than 80% of the games played against the random opponent as won.

The overall win percentages are almost the same for various heuristics. The top three performers seem to be heuristic that

1. Minimizes opponent's moves
2. Minimizes the opponent's moves and maximizes the distance between the players
3. Maintains a favorable moves ratio in our favor and minimizes the distance from the center.

To minimize the variance due to small number of plays, I increased the number of matches against each opponent to batches of 20 and increased the timeout to 200 milliseconds. The best performer had a score of **65.4%**, while the other two gave a comparable performance of **64.3%** and **63.6%**.

I would therefore recommend the ratio_distance_heuristic which tries to maximize the moves and keep our player near the center and away from edges. This is because,

1. It is computationally simple which allows more time to be allocated to searching more levels.
2. The heuristic also takes into account the moves of both the players and the distances from the center as most of the games seem to end with our player getting stuck to the edges.
3. The heuristic's runtime was also comparative to others and its implementation was simple as well as it involved only a few simple computations.
4. It depends on only the current state of the problem and no additional memory is required.