Heuristic Analysis for Isolation Game – Playing Agent

Synopsis

The aim of this project was to build a game-playing agent for a game called Isolation. The game is played by two players on a 7x7 chess game board. Each player has a knight which in turns is moved in L shaped moves. The goal of the game is to 'trap' the opponent – move the figure in certain positions that the opponent would run out of possible moves. Once the player occupy a cell, it can be no longer occupied by the same or other player. The first player who runs out of possible moves, loses the game.

The agent was built using two core AI techniques – Minimax and Alpha – Beta pruning. Minimax algorithm was used to search the game tree and calculate the best move for the player so that it would maximize player's wining chances and would minimize opponent's winning chances. Alpha – Beta pruning was used to optimize the search by decreasing the number of nodes which have to be searched.

After the agent was created, we tested the agent against the group of other agents withdifferent evaluation functions:

- Random: An agent that randomly chooses a move each turn.
- MM_Open: MinimaxPlayer agent using the open_move_score heuristic with search depth 3
- MM_Center: MinimaxPlayer agent using the center_score heuristic with search depth 3
- MM_Improved: MinimaxPlayer agent using the improved_score heuristic with search depth 3
- AB_Open: AlphaBetaPlayer using iterative deepening alpha-beta search and the open_move_score heuristic
- AB_Center: AlphaBetaPlayer using iterative deepening alpha-beta search and the center_score heuristic
- AB_Improved: AlphaBetaPlayer using iterative deepening alpha-beta search and the improved_score heuristic

To make our agent more efficient, we implemented three heuristic evaluation functions which were used to test our agent's performance during the tournament.

Heuristic evaluation functions

To test our agent we implemented three different heuristic evaluation functions:

1) The difference between our agent's moves and 2 times the opponents moves (See the code – custom_score_3)

$$\#\text{my\_moves} - 2(\#\text{opponent\_moves})$$

The logic behind selecting this heuristic function is that we are trying to minimize the number of moves left to our opponent. By multiplying opponent's moves by two we are implementing more aggressive gameplay – we are looking for the ways to block the player faster.

2) Ratio of our agent's moves and opponents moves (see the code – custom_score_2)

$$\frac{\#my\_moves}{\#opponent\_moves}$$

Just like the previous one, this heuristic considers the number of moves of both tournament players. The logic is similar - we are trying to minimize opponent's moves and maximize our agent's moves.

3) Ratio of the number of agent's moves and number of opponent's moves weighted by the value depending on where in the board the player is (central area or non-central area). (see the code – custom_score)

$$\frac{\#my\_moves}{\#opponent\_moves} \times weight, \qquad weight \in \begin{cases} 1{,}5 \; if \; player \; is \; in \; central \; area \\ 0.5 \; if \; player \; is \; outside \; the \; central \; area \end{cases}$$

This is the extension of the heuristics above. Since, there are more options for possible moves when a player is in the centre of the board than when the player is outside the central area, it makes sense to include this assumption to heuristic evaluation function.

Results

In order to get consistent results, we ran our test on 400 matches. Below is the output of all matches:

Playing Matches									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	731	69	737	63	730	70	728	72
2	MM_Open	352	448	401	399	384	416	345	455
3	MM_Center	616	184	677	123	644	156	630	170
4	MM_Improved	340	460	366	434	365	435	318	482
5	AB_Open	387	413	417	383	424	376	387	413
6	AB_Center	632	168	644	156	657	143	619	181
7	AB_Improved	365	435	409	391	420	380	384	416
	Win Rate:	**61.1%**		**65.2%**		**64.7%**		**60.9%**	

As we can see, our third evaluation function (weighted ratio of agent's and opponent's moves) scored the best compared to AB_Improved score. However, second heuristic (ratio of agent's and opponent's moves) score very close to third heuristic. Our first and the simplest heuristic (number of my moves – 2*opponents moves) score the worst and didn't outperform AB_Improved score. Based on these results, we would suggest using third heuristic evaluation because it scored the highest and

because it performs the most in depth-checks for player's position and possibility to minimize opponent's moves. However, this heuristic is more computationally intense than second heuristic which would work as good alternative when computational resources is a limiting factor.