Heuristics analysis
by Luka Anicin

In this project our goal was to create AI agent who is going to play and, hopefully, win Isolation game.

The main part was to find a good evaluation function, or heuristic, which is going to be used to determine what move is next to be performed by our agent.

In the process of testing and implementing, I've made many different heuristics but most of them weren't good enough for the project. Heuristic number 3, that didn't even perform very well, it made the cut so you can have the view of the process of thinking that I've had while working on this project.

What heuristic should be used?

From those which I implemented, 'Reward from center' is performing the best, so my recommendation is to use that one (in the code: custom_score).

Nevertheless, this heuristic is specially made for the rules provided in Knight - Isolation (Isolation games where players are able to move like horse figure from chess). Let's consider center of our 7x7 playing table. In that position our player is able to move up to 8 positions which is the maximum number of legal moves from any position on the table. As our agent moves to the position which provides more possible legal moves for it, agent will receive a bigger reward. The behavior of our agent trying to find the spot which provides more possible moves for it coincides with our winning progress. By pushing our opponent to spots where it won't have many or at all spots to move we are one step closer to winning the game.

Let's consider time complexity for this heuristic. The biggest time consuming parts are two 'for' loops (one for our moves and one for opponent's moves). The maximum size of our legal moves list is 8, which is also the variable that we iterate around in this heuristic. This is the part where we need to decide what we want. It is better for us when the agent has more opportunities (bigger score) to play but also run time of this heuristic will be longer. As the game progresses and number of free spaces are drastically decreased our heuristic function will run faster.

Generally the complexity is linear.

More details about it in the analysis part.

Heuristic number 1:

'Reward from center' – the idea behind this heuristic is to give our agent a reward for every legal move position. If the position of the legal move is close to the center it will receive a bigger reward, as we go further from the center it will receive less and less.

On the other side, we are doing the same process for our opponent player, but of course, we want our opponent to be close to the edge, where it has less moves to play.

The function has beaten ID_improved:

Match #	Opponent	AB_Improved		AB_Custom	
		Won	Lost	Won	Lost
1	Random	15	5	18	2
2	MM_Open	14	6	12	8
3	MM_Center	15	5	16	4
4	MM_Improved	12	8	17	3
5	AB_Open	12	8	15	5
6	AB_Center	14	6	11	9
7	AB_Improved	6	14	10	10
	Win Rate:	62.9%		70.7%	

Analysis:

By maximizing the number of our legal moves close to the center, our agent will have more options to choose from for its next move. As we approach closer to the center we have more possible moves to play. The same rule is for our opponent, so we would like to minimize moves of our opponent which are close to the center. By doing so the difference is bigger and our agent has more moves to play.

Heuristic number 2:

In code (custom_score_2). The idea for this heuristic is to check how many moves our opponent will have after we play some of our legal moves from the current position. We would like to choose the move which provides our opponent with least chances to play its next move.

Match #	Opponent	AB_Improved		AB_Custom_2	
		Won	Lost	Won	Lost
1	Random	14	6	16	4
2	MM_Open	12	8	12	8
3	MM_Center	15	5	13	7
4	MM_Improved	12	8	11	9
5	AB_Open	9	11	9	11
6	AB_Center	10	10	10	10
7	AB_Improved	10	10	8	12
	Win Rate:	58.6%		56.4%	

As you can see, this function performs poorly in the tournament and didn't receive 60% + score.

Analysis:

The goal of this heuristic is to choose that move for our player which will output the smallest number of possible (legal) moves for our opponent after we try and play that move.

Heuristic number 3:

In code (custom_score_3). This was the first version of my heuristic which is performing the best (custom_score). The idea is to maximize number of moves for our player which are close or in the center of the game table.

Match #	Opponent	AB_Improved		AB_Custom_3	
		Won	Lost	Won	Lost
1	Random	19	1	17	3
2	MM_Open	12	8	13	7
3	MM_Center	19	1	15	5
4	MM_Improved	12	8	14	6
5	AB_Open	9	11	9	11
6	AB_Center	13	7	10	10
7	AB_Improved	14	6	9	11
	Win Rate:	70.0%		62.1%	

Analysis:

By maximizing the number of our legal position close to the center, our agent, in the next move will have more possible legal moves.

Heuristic number 4:

In code (custom_score_4). This heuristic was mentioned in the class. I was impressed by its performance. Follow the enemy. (Explanation in the analysis part)

Match #	Opponent	AB_Improved		AB_Custom_4	
		Won	Lost	Won	Lost
1	Random	18	2	15	5
2	MM_Open	13	7	12	8
3	MM_Center	16	4	15	5
4	MM_Improved	13	7	13	7
5	AB_Open	11	9	13	7
6	AB_Center	11	9	13	7
7	AB_Improved	11	9	10	10
	Win Rate:	66.4%		65.0%	

Analysis:

Firstly, we multiple the number of opponent's legal moves by 2 and then from the number of our legal moves, subtract that number, because our goal is to minimize the difference. Our agent will start to perform moves that are close to the opponent. By doing so, our agent will start to follow it.