Report Isolation game

1. Implementation

Depth limited Minmax

The implementation of the minmax follows the algorithm suggested in: [link]. It makes use of the extra function min-value et programmed recursively.

Alpha beta search with pruning

The implementation of the alpha beta follows the algorithm suggested in: [link]. It makes use of the extra function min-value and max-value et programmed recursively.

2. Overall strategy of the customs evaluations functions

Only 3 functions were programmed but many more combination were tried.

- Custom_1

This is a very simple improvement to the improved evaluation function presented in class (number of my moves – number of moves of my opponent) where the function gives a premium in the case we are cornering the opening meaning that it gives a very large score for case when, for instance, the 2 coordinates x,y of the opponent are both smaller than our coordinated. This should be a decent heuristic because when the opponent is cornered it restricted its move for future rounds and we may have more moves on the rest of the board.

- Custom_2

This one aims to achieve 2 things: be close to your enemy in order to restrict it moves and stay on the center side.

- Custom_3

This one also aims to achieve 2 things: be close to your enemy in order to restrict it moves and maximize your number of moves comparatively to your opponent's number of moves.

- Custom_4

For comparison, we want to know if simply staying close to you enemy is not better than Custom and Custom_3.

3. Results

The interesting part in the result is that it is not only the overall wining rate that matter but also against which agent is a particular agent wining. Overall winning rate of the customs methods against each other seems to remain within the margin of error and against the pre-implemented evaluation functions, they constantly win but with variation in the winning rates. I investigated the dependency of the results on the time each search is allowed, although I had to reduce the number of simulations. From the number below, it is hard to conclude which one is the best custom algorithm because the numbers remain within the margin of error. But the AB_custom seems to perform the best.

NUM_MATCHES = 80 # number of matches against each opponent
TIME_LIMIT = 40 # number of milliseconds before timeout

	AB_Improved		AB_Custom		AB_custom_2		AB_custom_3		
	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Loss rate
Random	144	16	150	10	144	16	147	13	91%
MM_Open	79	81	95	65	109	51	105	55	61%
MM_Center	127	33	141	19	130	30	140	20	84%
AB_Custom	66	94	76	84	90	70	88	72	50%
MM_improved	93	67	91	69	90	70	92	68	57%
AB_Open	93	67	91	69	88	72	104	56	59%
AB_Center	85	75	94	66	86	74	89	71	55%
AB_Improved	87	73	90	70	94	66	90	70	56%
AB_custom_2	72	88	78	82	73	87	91	69	49%
AB_custom_3	66	94	79	81	80	80	85	75	48%
AB_custom_4	71	89	83	77	74	86	83	77	49%
	56%	44%	61%	39%	60%	40%	63%	37%	

NUM_MATCHES = 30 # number of matches against each opponent
TIME_LIMIT = 125

	AB_Improved		AB_Custom		AB_custom_2		AB_custom_3		
	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Loss rate
Random	48	12	48	12	43	17	48	12	78%
MM_Open	38	22	36	24	41	19	35	25	63%
MM_Center	45	15	51	9	49	11	46	14	80%
AB_Custom	15	25	26	34	32	28	29	31	46%
MM_improved	28	32	34	26	35	25	37	23	56%
AB_Open	33	27	38	22	38	22	32	28	59%
AB_Center	26	34	33	27	32	28	34	26	52%
AB_Improved	38	22	34	26	38	22	33	27	60%
AB_custom_2	30	30	33	27	27	33	29	31	50%
AB_custom_3	28	32	34	26	28	32	26	34	48%
AB_custom_4	27	33	35	25	28	32	29	31	50%
	54%	43%	61%	39%	59%	41%	57%	43%	

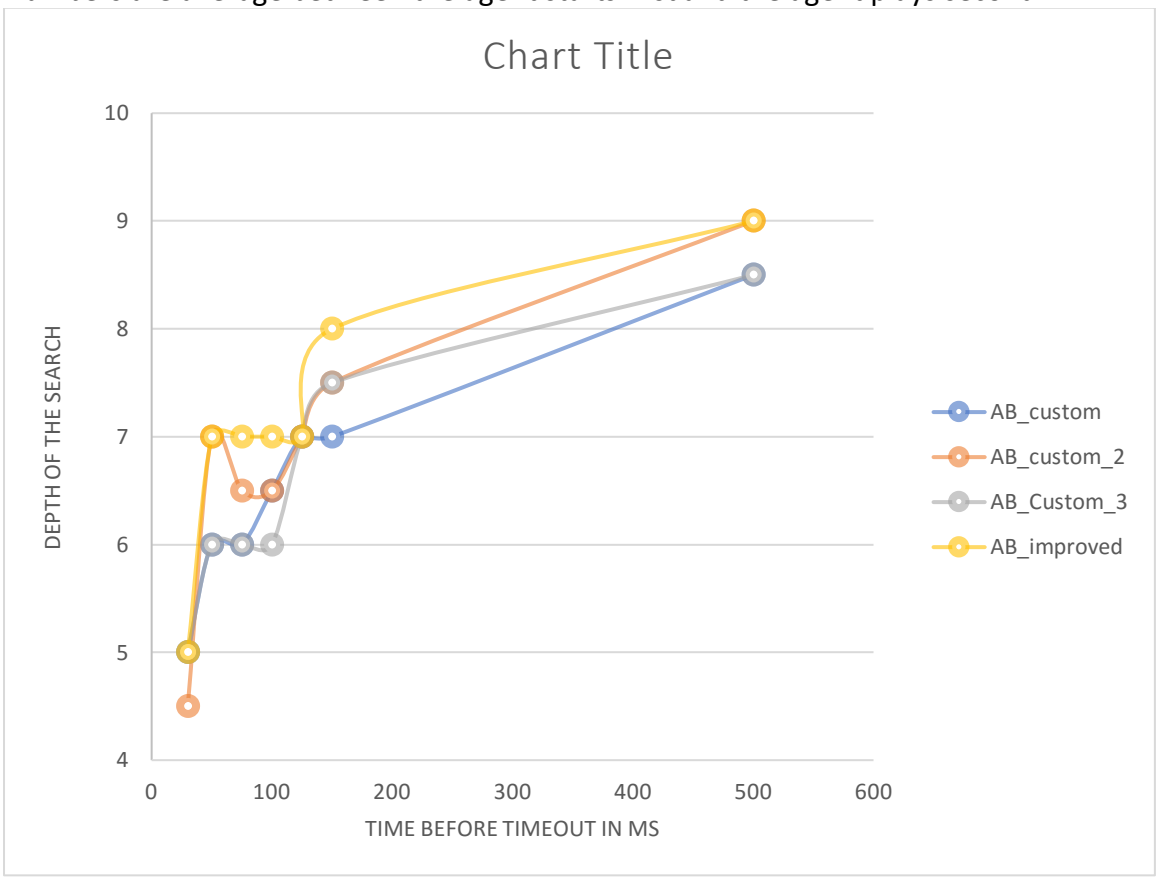
NUM_MATCHES = 30 # number of matches against each opponent
TIME_LIMIT = 160

	AB_Improved	AB_Custom	AB_custom_2	AB_custom_3

	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Loss rate
Random	27	3	29	1	24	6	25	5	88%
MM_Open	17	13	19	11	19	11	15	15	58%
MM_Center	24	6	27	3	22	8	25	5	82%
AB_Custom	12	18	12	18	13	17	15	15	43%
MM_improved	17	13	21	9	16	14	18	12	60%
AB_Open	17	13	17	13	17	13	16	14	56%
AB_Center	16	14	14	16	11	19	18	12	49%
AB_Improved	17	13	13	17	17	13	17	13	53%
AB_custom_2	15	15	15	15	16	14	15	15	51%
AB_custom_3	15	15	18	12	14	16	15	15	52%
AB_custom_4	11	19	14	16	16	14	15	15	47%
	57%	43%	60%	40%	56%	44%	59%	41%	

4. Depth of the Search

Through this short analysis, I wanted to see how deep was the search going and if the complexity of the evaluation function had an impact on the depth of the search. Below are the number for the first search being performed. Very typically, in virtue of the law of large numbers, we should get a convergence of the outcome of the game towards the “true” performance of the evaluation functions. The impact seems rather limited although the AB_improved clearly goes deeper by 1 quite often due to the simple evaluation function. Numbers are average between the agent starts first and the agent plays second.



5. Error estimation in the Results

In the end, I noted quite some variations in the results so I did a very quick check on the error estimation. It is rather large. A 95% confidence interval tells us that for 30 games, the error may be as large as +-6 game in the number of winning games (and losing games by symmetry).

	AB_Improved		AB_Custom		AB_custom_2		AB_custom_3	
	Won	Lost	Won	Lost	Won	Lost	Won	Lost
AB_Improved	15	15	15	15	13	17	17	13
AB_Improved	16	14	20	10	16	14	16	14
AB_Improved	16	14	20	10	14	16	20	10
AB_Improved	14	16	12	18	17	13	20	10
AB_Improved	12	18	14	16	15	15	15	15
AB_Improved	16	14	16	14	14	16	15	15
mean	15	15	16	14	15	15	17	13
stdev	2	2	3	3	1	1	2	2
95 % interval for 30 sim +-	3	3	6	6	3	3	5	5