Heuristics Analysis

Lucas Cabral Rezende

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	38	2	37	3	37	3	35	5
2	MM_Open	33	7	29	11	25	15	28	12
3	MM_Center	33	7	36	4	33	7	36	4
4	MM_Improved	27	13	24	16	29	11	28	12
5	AB_Open	22	18	22	18	22	18	24	16
6	AB_Center	22	18	26	14	24	16	26	14
7	AB_Improved	19	21	17	23	20	20	24	16
	Win Rate:	69.3%		68.2%		67.9%		71.8%	

Tournument.py results

Agents comparison

From the table above we can easily see that the alphabeta pruning agent easily beats the minimax agent for every evaluation function. The alphabeta agent must outperform the minimax agent due to its pruning techniques and the iterative deepening method implemented in this agent.

custom_score()

This heuristics subtracts the square of the opponent's legal moves from the agent's legal moves, so it gives more emphasis on reducing the ability of the opponent to move and could be seen as a more aggressive heuristics.

custom_score_2()

This heuristics subtracts the square of the opponent's legal moves from the square of the agent's legal moves, as both scores can grow at the same rate a search method using this heuristics will give preference to moves that enhance the ability of our agent to move in the future.

custom_score_3()

This heuristics subtracts the cubic of the opponent's legal moves from the square of the agent's legal moves, like custom_score() it gives more emphasis on reducing the ability of the opponents to move, but it also tries to find moves where our agent has a lot of options to move, being a sort of hybrid of custom_score() and custom_score_2().

Conclusion

From the heuristics I've created, the third one is the best to go with, it yield the best results, about 72% winning rate on average in the tournament.py, and it is relatively simple, since it is basically a combination from the first two heuristics. In my opinion, this balance of aggressiveness and defensiveness is what makes it the best choice.