Introduction
I've encountered problems to implement the Alpha-Beta pruning algorithm with iterative deepening.

First, I had a bug in the implementation that seemed that it was working OK, but it lost all matches against normal Minimax opponents:

Match #	Opponent	AB_Improved Won	Lost	AB_Custom Won	Lost	AB_Custom_2 Won	Lost	AB_Custom_3 Won	Lost
1	Random	0	10	0	10	0	10	0	10
2	MM_Open	0	10	0	10	0	10	0	10
3	MM_Center	0	10	0	10	0	10	0	10
4	MM_Improved	0	10	0	10	0	10	0	10
5	AB_Open	5	5	5	5	5	5	5	5
6	AB_Center	5	5	5	5	5	5	5	5
7	AB_Improved	5	5	5	5	5	5	5	5
	Win Rate:	21.4%		21.4%		21.4%		21.4%	

Even AB_Improved was losing.

Then I had problems in order to find which Timeout to use. After executing the tournament, I was getting 30+/40+/50+ timeouts and they were not deterministic, same code would give two different consecutive 20->40 timeouts in two different executions.
After some analysis I discovered also that tweaking the custom scores also affected the timeouts.

I also noticed that the Win Rate of games with same code could vary from 45% to 70% using the same code in different executions for the tournament.
After trying to achieve some stable baseline in which compare scores, I settled into increasing the number of matches from 5 to 50, which gives enough number of matches (>30) per tournament so that we can compare with enough confidence. Drawback: iterations on exploration of utility heuristic function slowed down.

Then I decided to help the MM opponents, just in case the Timeouts were affecting them, and returning (-1,-1) if minimax was not completing in time, so I changed the code for them that if timeout occurred, then instead of (-1,-1) they would move randomly, so at least they had a chance.

Heuristics
Finally, for the heuristic study, I observed that Improved score worked well for Random and MM opponents (R-MM from now on) but it was not that strong against AB in "open" and "center".
I remembered that in the lectures it was suggested to try to occupy the central positions at the beginning to the game, which would give advantage. So my approach has been to dynamically

choose one of these score tactics, trying to achieve different approaches for the game in different times. So:

- For custom_score (AB_Custom), if there are more than 5 movements available, then use the "center" scoring, trying to "push" the decision towards the center of the board. If the movements start to be scarce (<5), then use "Improved" scoring, which is quite strong.
- For custom_score_2 (AB_Custom_2), similar approach, but if movements are below 5, then use "open" scoring.
- For custom_score_3 (AB_Custom_3), try to increase the number of 'steps' in the tactic, so, if number of available moves are:
    - >5 → use Center
    - 3,4 → use Open
    - 1,2 → ise Improved

These are the results (using the 100 games approach already mentioned):

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	82	18	84	16	76	24	86	14
2	MM_Open	63	37	74	26	59	41	59	41
3	MM_Center	72	28	78	22	68	32	79	21
4	MM_Improved	62	38	61	39	60	40	56	44
5	AB_Open	57	43	55	45	46	54	53	47
6	AB_Center	62	38	61	39	56	44	58	42
7	AB_Improved	51	49	53	47	43	57	44	56
	Win Rate:	64.1%		66.6%		58.3%		62.1%	

Overall, there is little improvement in my scoring approach, being AB_Custom the best of the three, with 2.5% more wins than AB_Improved.

What is interesting to observe, is how different tactics work better with other tactics, and with others don't.
For example, custom_score_3 is beaten by AB_Improved (44-55), but then custom_score_3 does better than AB_Improved against Random (86-14) vs (82-18). Same situation for AB_Center (79-21) for custom_score_3 vs (72-28).

Conclusion
Using combinations of well-defined basic tactics as building blocks for more sophisticated scoring functions on the Isolation game, produces marginal gains (~2.5%) in number of matches played against all other strategies.