Build a Game-Playing Agent
Heuristic Analysis

Alberto Gaona

September 29, 2017

1 Introduction

A central challenge in build a game-playing agent is that of constructing heuristic evaluation functions from game description. An heuristic evaluation function estimates how good the current board configuration is for a player, just a function that return a single number to indicate how well or bad your player is doing the game, a high value indicates your player is in a good situation while a low value indicates otherwise.

In this analysis we evaluate three heuristic evaluation functions for a game called knight-isolation a variation of Isolation, where each agent is restricted to L-shaped movements (like a knight in a chess) on a seven by seven board.

2 Basic heuristic evaluation function

Typically, evaluate how good is the board for the player and how good it is for the opponent, then subtract the opponent's score from the player's is the simplest heuristic used in game-playing agents.

$$my\_moves - opponent\_moves$$

This heuristic would penalize our computer player with more potential moves. which is counter productive. It continues to take into account boards where the current player can make a larger number of moves and also penalizes boards where the opponent can make a lager number of moves.

We can even weight components of the formula to try to encourage more aggressive or less aggressive game play, that means, if we change coefficients around that subtraction we can improve the heuristic. With this in mind we try weight the opponent's moves with a value of two, three and four. Where:

AB_custom = my_moves - 4*opponent_moves
AB_custom_2 = my_moves - 3*opponent_moves
AB_custom_3 = my_moves - 2*opponent_moves

We evaluate which of the three variations obtains a better percentage of win rate, competing our agent against seven agents more and comparing it with the original basic heuristic evaluation function called AB_Improved:

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	7	3	7	3	8	2	9	1
2	MM_Open	8	2	5	5	5	5	8	2
3	MM_Center	4	6	8	2	7	3	8	2
4	MM_Improved	7	3	7	3	5	5	8	2
5	AB_Open	4	6	5	5	5	5	6	4
6	AB_Center	6	4	6	4	8	2	7	3
7	AB_Improved	4	6	6	4	3	7	6	4
	Win Rate:	57.1%		62.9%		58.6%		74.3%	

Figure 1: Test one

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	7	3	9	1	8	2	8	2
2	MM_Open	6	4	6	4	8	2	9	1
3	MM_Center	7	3	7	3	9	1	9	1
4	MM_Improved	8	2	7	3	5	5	8	2
5	AB_Open	7	3	6	4	5	5	5	5
6	AB_Center	6	4	3	7	8	2	6	4
7	AB_Improved	5	5	5	5	7	3	6	4
	Win Rate:	65.7%		61.4%		71.4%		72.9%	

Figure 2: Test two

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	6	4	10	0	7	3
2	MM_Open	6	4	8	2	6	4	9	1
3	MM_Center	8	2	7	3	8	2	8	2
4	MM_Improved	6	4	5	5	6	4	5	5
5	AB_Open	5	5	5	5	5	5	3	7
6	AB_Center	5	5	8	2	5	5	6	4
7	AB_Improved	4	6	6	4	6	4	8	2
	Win Rate:	61.4%		64.3%		65.7%		65.7%	

Figure 3: Test three

And conclude that function $my\_moves - 2 * opponent\_moves$ was the best, also we discover if we add a sum on the number of moves we will get a better result compared to having the exact number of moves, as we can see in the figure 4.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2	
		Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	9	1	7	3
2	MM_Open	8	2	8	2	6	4
3	MM_Center	8	2	10	0	6	4
4	MM_Improved	7	3	7	3	9	1
5	AB_Open	4	6	6	4	6	4
6	AB_Center	5	5	5	5	5	5
7	AB_Improved	8	2	7	3	4	6
	Win Rate:	70.0%		74.3%		61.4%	

Figure 4: Difference between function one (with sum) and function two (without sum) compared with the basic heuristic evaluation function AB_improved

As we can see we get an improvement of around 10% , so, our basic heuristic evaluation function for knight-isolation is as follows:

$$(my\_moves + 1) - 2 * (opponent\_moves + 1)$$

3 Lucky heuristic evaluation function

The lucky heuristic takes into account the number of locations that are still available on the board, however this number of available spaces don't reflect the goodness of the board so we use it to experimenting how the win rate change and confirm the next statement: "Both the number of open spaces and the number of moves made are the same for every position for the same depth, thus any heuristic that relies only on these two is exactly as effective (minus very slight performance penalty) as a zero score function"

A null heuristic evaluation function or zero score function is one that always returns the value of 0.

We confirm the statement by modifying again and again an operation that only takes into account the number of open spaces and the number of moves, so finally our lucky heuristic evaluation function is as follows:

$$((free\_moves + 1/my\_moves + 1) + (free\_moves + 1/opponent\_moves + 1))^2$$

notice that we need the sum of one to avoid a zero division error.

4 Coward heuristic evaluation function

The last heuristic is called coward because the goal is get as far away from the opponent as possible, here is the otherwise of our basic heuristic evaluation function.

To explain this, let's analyze some board configurations.

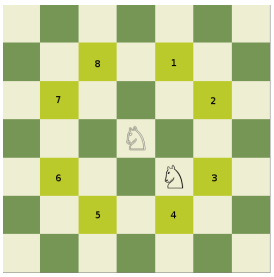

Figure 5: Second move, our player turn, where the light green are our available moves

In the figure 5 we can notice that our player has eight options to move and each of these options has its own number of available moves assuming that move are selected as shown in the following table:

Move	my moves	opponent moves	opponent moves - my moves
1	5	8	3
2	4	7	3
3	5	8	3
4	5	8	3
5	4	7	3
6	5	8	3
7	5	8	3
8	5	8	3

And our goal is to get away so we need move to the available move number 7 or 8, but since it is the second movement of our player we cannot yet declare a way to help our player to select the most distant movements.

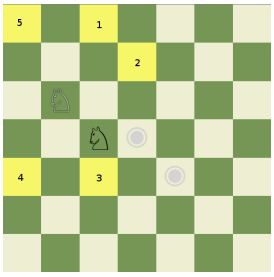

Figure 6: Third move, our player turn

Now, our player has five movements available and our table is as follows:

Move	my moves	opponent moves	opponent moves - my moves
1	3	7	4
2	4	6	2
3	7	7	0
4	2	6	4
5	1	7	6

This time our player should choose the movement 5 and we begin notice that the movement we should choose is the one that has the biggest difference between opponent's moves and our player moves. This makes a lot of sense as we are doing the reverse of our basic heuristic evaluation function.

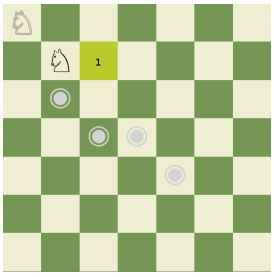

Figure 7: Fourth move, our player turn

Move	my moves	opponent moves	opponent moves - my moves
1	4	3	-1

Now we only have one option to move, to this version of isolation our movements are restricted to L-shaped so we can continue to run away even get to flip the board around as shown in the following image.

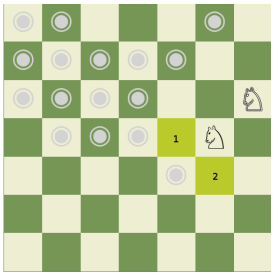

Figure 8: advanced board, our player turn

Move	my moves	opponent moves	opponent moves - my moves
1	2	5	3
2	6	6	0
3	4	5	1
4	2	6	4

Finally we can confirm that the move that has the greatest difference between the opponent's moves and our player moves is the option we have to choose. To make this more reliable we can add the difference between the current position of the opponent and the position of the movement, therefore, our coward heuristic evaluation function is as follows:

$$(y2 - y1) + (x2 - x1) + (opponent\_moves - my\_moves)$$

5 Conclusion

As conclusion I recommend our basic heuristic for the following reasons:

- It obtains a better win rate than AB_Improved which is the **general** basic heuristic in board games.
- It is a very simple function no computationally complex therefore it does not consume much computational time.
- Evaluate very well how good the current board configuration is for our player.

Also comparing the Lucky heuristic there is not much difference respect to the null function since it leaves in a certain way to the luck the decision of whether it is a good game or not, although this sometimes surpasses the win score of AB_Improved.

Meanwhile the Coward heuristic is not recommended, just like in real life: running away is never the solution. It has a low win rate compared with our other two heuristic and with AB_Improved although it could be used for a combat of beginners level.

Note: For the figure 10 and figure 11, AB_Custom is our Basic heuristic, AB_Custom_2 is our Lucky heuristic and AB_Custom_3 is our Coward heuristic.

Match #	Opponent	AB_Improved		Basic		Lucky		Coward	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	86	14	89	11	83	17	84	16
2	MM_Open	64	36	64	36	66	34	45	55
3	MM_Center	82	18	89	11	87	13	84	16
4	MM_Improved	59	41	66	34	55	45	49	51
5	AB_Open	48	52	48	52	52	48	39	61
6	AB_Center	59	41	53	47	56	44	51	49
7	AB_Improved	51	49	52	48	52	48	41	59
	Win Rate:	64.1%		65.9%		64.4%		56.1%	

Figure 9: Comparison of the win rate of our three heuristics versus AB_Improved, as we can see the basic and the lucky heuristic achieved a better win rate while the coward heuristic has a low performance.

****
Playing Matches
****

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	77	23	85	15	80	20	81	19
2	MM_Open	70	30	68	32	67	33	52	48
3	MM_Center	81	19	80	20	81	19	66	34
4	MM_Improved	65	35	57	43	60	40	56	44
5	AB_Open	43	57	51	49	53	47	45	55
6	AB_Center	68	32	59	41	63	37	48	52
7	AB_Improved	50	50	61	39	49	51	40	60
	Win Rate:	64.9%		65.9%		64.7%		55.4%	

Figure 10: Comparison two of the win rate of our three heuristics versus AB_Improved, here we can notice that our basic heuristic has a better win rate than AB_Improved, whereas there is not much difference between AB_Improved and our Lucky heuristic, finally the Coward heuristic has a lower win rate than the others.

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	33	7	33	7	33	7	34	6
2	MM_Open	26	14	28	12	26	14	28	12
3	MM_Center	32	8	32	8	33	7	37	3
4	MM_Improved	21	19	26	14	24	16	26	14
5	AB_Open	23	17	25	15	23	17	16	24
6	AB_Center	22	18	21	19	25	15	23	17
7	AB_Improved	22	18	21	19	14	26	23	17
	Win Rate:	63.9%		66.4%		63.6%		66.8%	

Figure 11: In this test surprisingly our Coward heuristic had a better win rate than AB_Improved like our Basic heuristic, meanwhile the Lucky heuristic continues vary slightly to the win score of AB_Improved