Heuristic Analysis

Guruprasad Sridharan

May 2017

1 Heuristic Functions

1.0.1 Modified Center Score

It is not hard to notice that opening moves around the center square have a higher branching factor. So it is better to open somewhere near the center. One good heuristic based on this assumption is to play moves such that the distance from the center square is minimized. I experimented with different measures for distance such as Euclidean distance, Manhattan distance and radial distance. Manhattan distance based center heuristic outperformed all the others.
MD(c1, c2) = $|c1.x - c2.x| + |c1.y - c2.y|$

1.0.2 Longest Move Chain Length

As I was thinking about heuristic functions whose value is directly correlated with the probability of survival for the given player. One metric directly related to this is the longest chain of moves possible for the given player from the current position of his knight on the board. I limit the depth of the chains we explore so that we don't cause timeouts. Memoization is used to store max chain length beginning at already visited squares. The results of max chain length calculation are memoized into a dictionary. The actual max chain length estimated may be an under approximation of the actual max chain length for two reasons. One, the algorithm doesn't explore all the way since we limit the depth to 7. Second, the algorithm memoize results a little aggressively without considering directions so as to make the function evaluate faster and this inherently doesn't explore all possible chains. The function actually calculates the max chain for both players and returns the difference between the max chain length for current player and the opponent player as the heuristic value. There were some problems with this. First the difference between max chain length is a very inaccurate estimation of the chances of winning for the current player. For example, let the longest chain length for current player be 3 and for the opponent player let it be 2. This heuristic function would return a value of 1. Imagine in this case, the number of possible chains for the opponent was only one. So, the heuristic value for the current player should be higher. This heuristic fails to consider the degrees of freedom (number of possible chains) for the players. Even though a player

might have a long chain of moves, if that chain is somehow blocked by the other player, he is at risk to lose the game. Capturing these kind of situations becomes harder with the max chain length heuristic. It is no surprise that this was the worst performing heuristic in the tournament for my ID Search player.

1.0.3 Move Closure Cardinality

When designing a heuristic function, there are multiple tenets to be considered. First, the heuristic function should accurately capture the final outcome of the game. Second, it should be possible to evaluate the heuristic function quickly. A move closure is defined as the set of all legal moves possible for a particular player from the current location of his knight on the board. Here, we assume that the opponent's position doesn't change between multiple plies. It's a crude approximation that helps us do a quick lookahead. Since in the initial stages of the game, the number of levels we may lookahead in this heuristic can be huge, it is hard to set a strict upper bound on the time required to evaluate the function. That's why we limit the number of levels we lookahead to 3. This guarantees that the function always completes within a stipulated time. Why specifically three levels only? Well, because it doesn't make sense to go further deeper. Since it is hard to predict so much into the future. As I evaluated this heuristic, I found that it made more sense to combine it with the modified center score heuristic. So for the first nine moves, we use the modified center score since it will sort of nudge the agent to choose opening moves closer to the center. For all moves thereafter, we use the difference in move closure length between the current player and the opponent.

1.0.4 Tournament Evaluation

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	6	4	5	5
2	MM_Open	8	2	10	0	9	1	7	3
3	MM_Center	9	1	9	1	10	0	6	4
4	MM_Improved	9	1	8	2	6	4	2	8
5	AB_Open	4	6	5	5	4	6	2	8
6	AB_Center	6	4	5	5	6	4	2	8
7	AB_Improved	3	7	8	2	5	5	0	10
	Win Rate:	68.6%		78.6%		65.7%		34.3%	

Please take a look at the figure above. It shows the win rate of different variants of the ID search using different heuristics. AB_Custom player uses Move Closure Cardinality heuristic. AB_Custom_2 uses Modified Center Score heuristic. AB_Custom_3 uses Longest Move Chain Length heuristic. I have ran the tournament against the sample players multiple times and found that the ranking remains the same more or less.

1.0.5 Summary

It is evident from the tournament evaluation against other players that the move closure cardinality heuristic is the best performing one. Using this heuristic, I was consistently able to win around 75%-80% of all the games. Moreover, the evaluation of the heuristic function takes very less time which makes it an ideal choice for time limited plies and use in Alpha-Beta Iterative Deepening Search. It also accurately captures the winning chance of the current player. There are avenues for improvement though. There is some room for incorporating complex end game strategies. The agent can be forced to play offensive or defensive depending on the current board state. For example, if the opponent is cramped for space, then instead of moving into more open positions, the agent should try to play offensive and capture available position for the opponent to claim victory. Though we can go for such complex heuristic functions, it is better to stay with simpler and effective heuristic functions since more often that not such complex analysis comes at the cost of more expensive computation.