Heuristic analysis

Overview

The “improved_score” heuristic function performs well in the isolation game. So, my custom score functions are also based on the formula,

#Own Move - #Opposite Move

And following scenarios and evaluation factors are used to find the custom scores.

- Overlap move
- Corner move
- Diagonal chain
- Players distance

Scenarios and Evaluation Factors

Overlap Move

If two players are close to each others, their future moves may be overlapped. And the active player may occupy the future move of the enemy.
In order to reflect this phenomenon, when the next move is my turn and overlap occurs, the number of opposite move is reduced by one. On the other hand, when the next move is opposite turn and overlap exists, the number of my move is reduced by one.

#Opposite Move = #Opposite Move -1 (if overlap and next move = my turn)
#Own Move = #Own Move – 1 (if overlap and next move = opposite turn)

One extreme case is that a player has only one move. When that move is occupied by the future move of the enemy, the player will lose.
So, if my player has only one move and overlap exists, the score returns negative infinity immedicatly. But if the opposite player has only one move and overlap exists, the score returns positive infinity immedicatly.

Corner Move

Corner move is dangerous. After moved to the corner, the player has at most 1 move available in most of the situations. (Except the first move, which has two future moves.)
Therefore, weighted future move is introduced. The value of each future corner move is 0.2. And the value of each future non-corner move is 1.

Corner Move = 0.2
Non-corner Move = 1

Diagonal Chain

“Diagonal Chain” is a kind of aggressive move. It aims at reduced the possible move of players. Figure1 shows the possible moves of a knight. It has 8 available moves with diagonal pattern. Filling the spaces diagonally is efficient to reduce number of possible moves of players. (Figure2, Figure3)

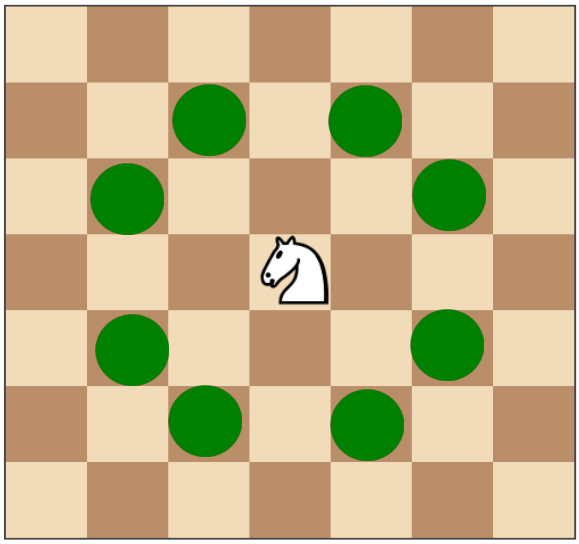

Figure1. Possible move of knight (green circles)

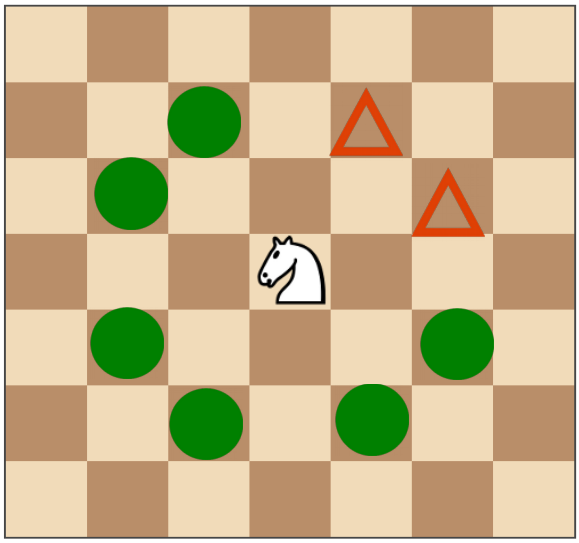

Figure2. Possible move of knight (green circles) and diagonal filled spaces (red triangles)

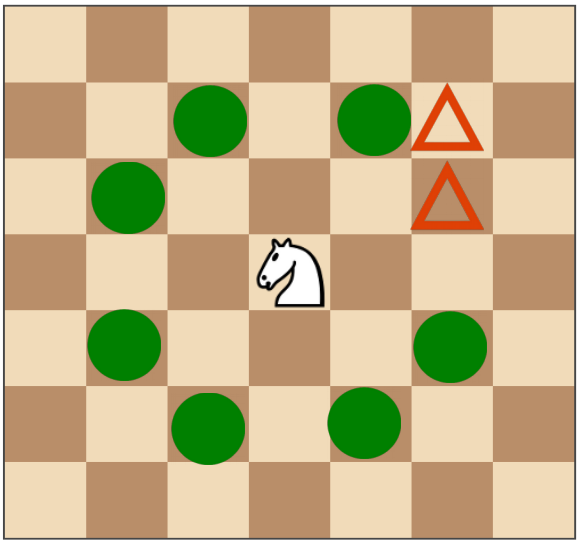

Figure3. Possible move of knight (green circles) and vertical filled spaces (red triangles)

So, “diagonal chain factor” is built to measure the aggressiveness. The range of this factor is between 0 and 1. And 1 means it is the most aggressive move.
For instance, if all of my player’s diagonal neighbours(top-left, top-right, bottom-left and bottom-right) are filled, the “diagonal chain factor” is 1.
The calcuation of the “diagonal chain factor” is the following,

is top-left neighbour filled? Yes: +0.25, No: +0
is top-right neighbour filled? Yes: +0.25, No: +0
is bottom-left neighbour filled? Yes: +0.25, No: +0
is bottom-right neighbour filled? Yes: +0.25, No: +0

Players Distance

Players distance is an important factor. When two players are close together, the uncertainty increases. And their future moves are easily overlapped. When two players are far away, the uncertainty decreases. And their future moves are “safer”.
Therefore, an evaluation factor is created to measure the distance of players. The range of this factor is between 0 and 1. The shorter the distance, the smaller the factor value, vice versa.

$$\textit{Longtest Distance} = (\textit{Board Width} - 1) + (\textit{Board Height} - 1)$$
$$\textit{Distance Factor} = (|X_{own} - X_{opposite}| + |Y_{own} - Y_{opposite}| - 1) / (\textit{Longtest Distance} - 1)$$

Heuristic Functions

custom_score_3

I try to implement a score function that can be reused in “custom_score_2” and “custom_score”. “custom_score_3” is an enhancement of “improve_score” function. And overlap move and corner move are adopted to make the score more representable to the game status. It can be simplified as following formula,

custom_score_3 = #Own Move(Adjusted) - #Opposite Move(Adjusted)

custom_score_2

I would like to build an aggressive player in this score function. It is an enhancement of “custom_score_3”. My player is willing to create diagonal chain. And after added distance factor, my player would move towards opposite player more often. Of course, Both diagonal chain and distance factor are affected by number of my future move. It would be less aggressive if number of my future move reduces.

custom_score_2 = #Own Move(Adjusted) - #Opposite Move(Adjusted)
*+ #Own Move(Adjusted) * diagonal chain factor * 0.25*
*+ #Own Move(Adjusted) * (1 – distance factor) * 0.25*

custom_score

Compared with “custom_score_2”, this score function attempt to find a balance between aggressive and conservative move. “Net move” is calculated in order to find the advantage side. If my player has an advantage, it would perform aggressive move. However, if opposite player has an advantage, my player would keep aways from the enemy and avoid diagonal chain.

Net Move = #Own Move(Adjusted) - #Opposite Move(Adjusted)
*custom_score = Net Move + Net Move * diagonal chain factor * 0.1*
*+ Net Move * (1 – distance factor) * 0.1*

Performance

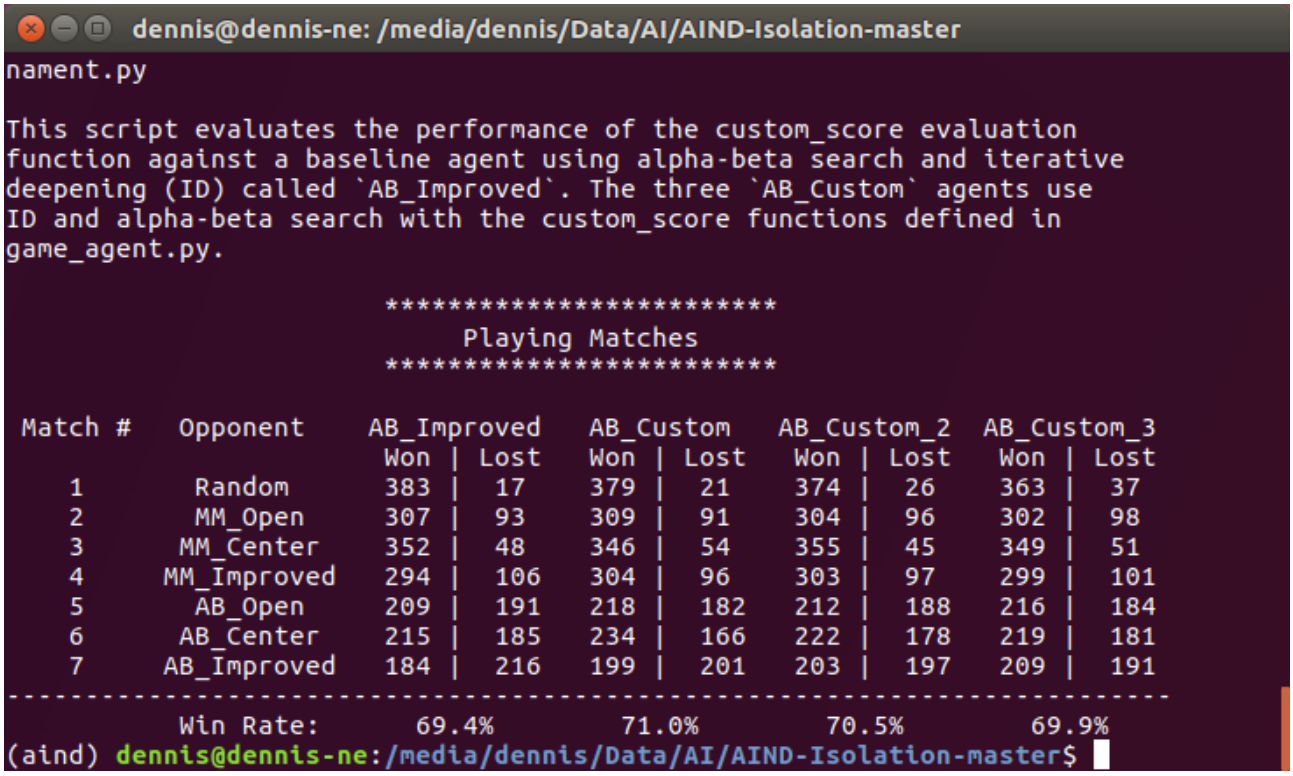

Figure4. Result of tournament

Figure4 shows the result of tournament. 400 matches are performed against each opposite player. Their winning rates are very simliar (i.e. around 70%). And the result is same as my expectation.

The agents perform well for random and minimax opposite players. And the average winning rate is higher than 80%. However, the average winning rate against alpha-beta pruning opposite players is not good. It is around 50%.

About the custom heuristic functions, we can notice that when the score function is evolved, the winning rate increases.

Name	Based On	Winning rate
AB_Improved	-	69.4%
AB_Custom_3	AB_Improved	69.9%
AB_Custom_2	AB_Custom_3	70.5%
AB_Custom	AB_Custom_2	71.0%

“AB_Custom” has the highest winning rate (71%). It is 1.6% better than “AB_Improved”. It performs much better especically against alpha-beta pruning players.

However, the winning rate of “AB_Improved” vs “AB_Improved” should be around 50%. But the actual result is 46%. Perhaps more matches are needed to produce a more reliable result.

Conclusion

“custom_score” is the enhancement of “improved_score”, which performs well in the isolation game. Also, it has both aggressive and conservative strategies in different game states. And it considers overlap move, corner move, diagonal chain and players distance.
And in practice, it works better than other evaluation functions. It obtains the highest winning rate (i.e. 71%). Thus, I would recommend to use “custom_score” as the evaluation function.