EVALUATION FUNCTION ANALYSIS

EVALUATION FUNCTIONS ANALYSIS

Introduction

I've came up with 3 evaluation functions as it is required in the task. At the end of analysis of each evaluation function you can find a comparison table between AB Improved and Custom Score evaluation functions, as well as a graph comparing two functions.

Analysis

Heuristics

In the following evaluation functions I've used a number of different heuristics.

Here's a list of them: 1. Heuristic 1: Difference between player's and opponent's moves left 2. Heuristic 2: Difference between player's and opponent's distances from the center of the game board - Manhattan - Euclidian 3. Heuristic 3: Partition Reward

According to my observations, the second heuristic performs better than other 2.

Heuristic 1

The first heuristic is just a difference between player's and opponent's legal moves. It's easy and fast to compute.

Heuristic 2

The second one is a bit more complex. It evaluates the difference between the player's and opponent's distances from the centre of game board. I tested 2 variants of this heuristic. One was computed using Manhattan distance, and another one — using Euclidian distance. The first way takes less time to compute, however, the second allows us to get better win rate. Therefore, I decided to stick with Euclidian method of calculating the distance.

Heuristic 3

With the third heuristic I tried to detect a partition of the game board. It's easy to compute. If such partition was present, I rewarded the agent with extra points. However, in reality, this haven't affected the performance of the agent much, and to say more, it had worsen the performance of the agent, when it was used with the first 2 heuristics.

custom_score_3

I'll start with the third one — custom_score_3. In this evaluation function I tried to add a couple of features. First one is detection of partition. As it was said in the lectures, if there's a partition, then we're likely to win the game. What I did is, I rewarded the agent if there was a presence of partition. I played around with coefficients that the partition award is multiplied by. The optimal value of the coefficient is 0.5. It's also easy to conclude that if the coefficient decreases, the win rate decreases as well. The same can be said about increasing the coefficient from the optimal value.

Apart from changing the partition_reward coefficient, I changed the way by which the distance of opponent and player is far from the center of the board. As you could see, in my case, the highest win rate for both Euclidian distance and Manhattan distance is the same. Overall, this evaluation function performed better than AB_Improved agent in 2 times out of 4.

Below are graph and table with comparison of AB_Improved and Custom_3 evaluation functions.

AB_Improved	Custom_3	Description
40 %	31.4 %	partition_reward * 0.1,

		Manhattan distance
35.7 %	38.6 %	partition_reward * 0.5, Manhattan distance
38.6 %	25.7 %	partition_reward * 1, Manhattan distance
38.6 %	38.6 %	partition_reward * 0.5, Euclidian distance

custom_score_2

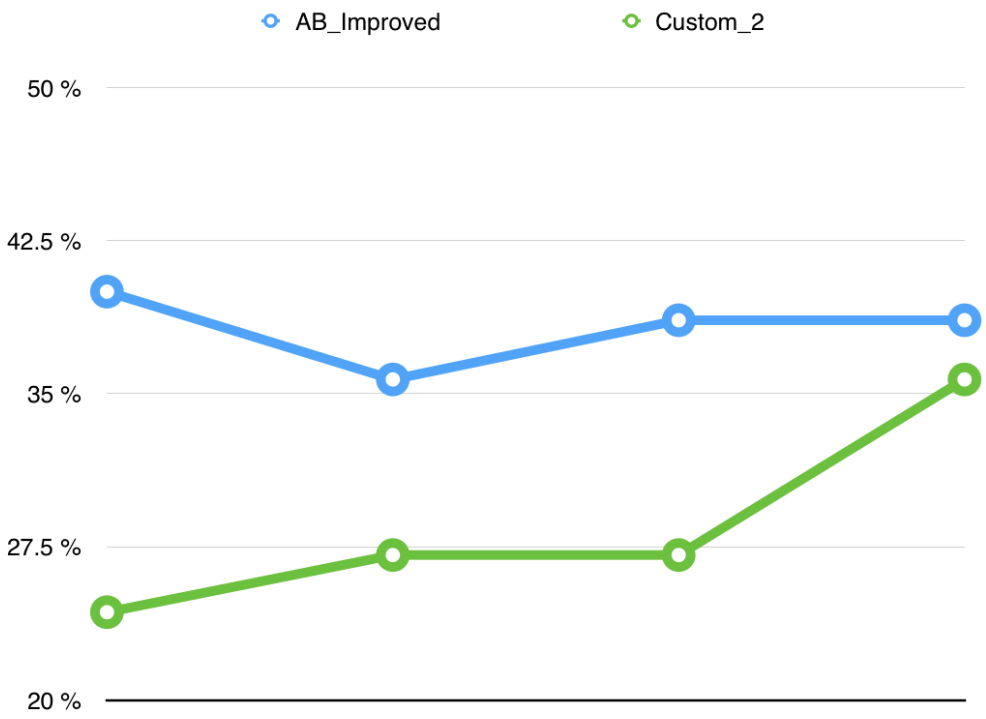
Let's move on to the next evaluation function — custom_score_2. From the 3 functions that I tried, this one performs the worst.

In this function I tried to play with such heuristics as distance between player and the center of the board and the difference between number of opponent's moves and player's moves. What I did is multiplied the first one by alpha coefficient, and the latter — by beta coefficient.

As you could see from the results table below, the win rate increases slightly, when we decrease the alpha value. What's interesting is that if we do the reverse thing, i.e. set alpha to 1.0, the win rate is the same as in the previous case.

Overall, I came to conclusion that this evaluation function doesn't really work, except for the case, when alpha is equal to 0, and there's only the difference between number of moves left. But still, even in this case, the agent loses to AB_Improved.

You can see how the custom_score_2 function performed below.



AB_Improved	Custom_2	Description
40 %	24.3 %	alpha = 0.2, beta = 0.8
35.7 %	27.1 %	alpha = 0.05, beta = 0.95
38.6 %	27.1 %	alpha = 1.0, beta = 0.0
38.6 %	35.7 %	alpha = 0.0, beta = 1.0

custom_score

The last evaluation function (custom_score) is the most interesting one. With this function I've been able to achieve 40% win rate, and perform better than the AB_Improved agent.

I've used a number of heuristics to evaluate the score. The first one is the difference between opponent's and player's distance from the board center, which was divided by a coefficient, which in the end, was assigned to width of the board, because the agent performs the best in such way. If the coefficient is lower or greater than board width, then the agent performs worse.

I've also tried to add a reward for partition detection as in the case with custom_score_3 function, but it lowered the win rate by 1.4%.

Another thing I did is similar to what was done with custom_score_3 evaluation function: I looked at how agent performs when using either Manhattan or Euclidian distance. I ended up sticking with Euclidian distance.

AB_Improved	Custom	Description
40 %	35.7 %	Manhattan distance, / game.width
35.7 %	28.6 %	Manhattan distance, / 10
38.6 %	27.1 %	Manhattan distance, / (game.width / 2)
38.6 %	40 %	Euclidian distance, / game.width
30 %	38.6 %	Euclidian distance, / game.width, partition reward * 0.5

Common between Custom and Custom_3 functions

There is one thing that is common between custom_score and custom_score_3 evaluation functions, which are described below. That is, they use different evaluation methods depending on the situation. To be more precise, I used the simple heuristic, which is defined as the difference between values of moves left for player and opponent, when they're not equal, and the heuristic which determines how closer the player to the center in comparison to opponent agent.

Conclusion

By looking at the performance of each evaluation function, we can conclude that the custom_score function is the best function to use when playing Isolation game, because it performed the best among other functions.

Below is the table comparing all 3 evaluation functions, as well as a graph representation of their performance.

AB_Improved	Custom	Custom_2	Custom_3
40 %	35.7 %	24.3 %	31.4 %
35.7 %	35.7 %	27.1 %	38.6 %
38.6 %	27.1 %	27.1 %	25.7 %
38.6 %	40 %	35.7 %	38.6 %
30 %	38.6 %	-	-