Heuristic Analysis

1. Test Results

There are ten tests carried out to compare performance between benchmark evaluation function AB_Improve and custom evaluation functions. The win rates for each test are presented in the table below:

	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3
Test 1	62.9%	61.4%	70.0%	62.9%
Test 2	61.4%	64.3%	64.3%	64.3%
Test 3	58.6%	72.9%	65.7%	62.9%
Test 4	72.9%	68.6%	62.9%	71.4%
Test 5	68.6%	77.1%	58.6%	64.3%
Test 6	64.3%	67.1%	64.9%	67.1%
Test 7	64.3%	62.9%	62.9%	64.3%
Test 8	65.7%	72.9%	58.6%	70.0%
Test 9	68.6%	74.3%	62.9%	62.9%
Test 10	58.6%	68.6%	62.9%	61.4%
Mean	64.9%	69.01%	63.37%	65.15%
Number of times reaching highest win rates among 10 tests	2	7	2	3

Table 1.1: Test results

The test results show that AB_Custom evaluation function presents the highest mean win rate and wins most of times during the tests.

2. Evaluation Function Analysis

2.1 AB_Custom

AB_Custom function tries to copy opponent's moves at the beginning of the game by placing it moves at the opposite side of the game board and occupying most available spaces.

It evaluates three factors: number of own available moves, number of opponent's available moves, and distance between own position and opponent's "mirror" position. It tries to maximize the number of own available moves, while minimizing the number of opponent's available moves and the distance.

The opponent's mirror position is the opposite position across the game board. For example, assuming that opponent’s position is row: 3 and column: 1 (index starts at 0), the mirror position will be row: game board height - 3 - 1, column: game board width - 1 – 1, as shown below:

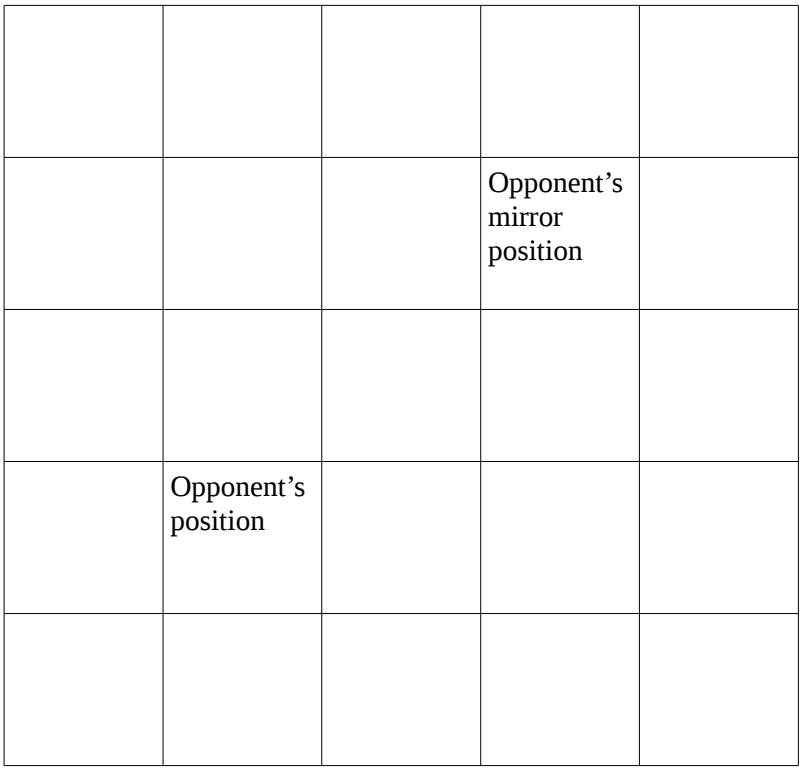

Figure 2.1: Illustration for how to calculate opponent’s mirror position

The preferred positions are the ones on the mirror position or close to the mirror position. Also, the distance is weighted by the ratio between number of available spaces and total spaces on the game board. As the available spaces become scarce, it leaves fewer choices and becomes less important to keep close to the opponent's mirror position.

2.2 AB_Custom_2

AB_Custom_2 function aims to confine the opponent by staying close to it and limit its available moves.

It evaluates three factors: number of own available moves, number of opponent's available moves, and distance between own position and opponent's position. It tries to maximize the number of own available moves, while minimizing the number of opponent's available moves and the distance.

The preferred own position are the ones that opponents next move can reach. As the players move in an L shape as the knights do in chess, the distance between preferred own position and opponent’s position is the square root of the sum of 2 to power of 2 and 1 to power of 2. This distance is used as an offset. Any distance between own position and opponent’s position that is shorter or longer than the offset will be penalized. Also, the distance is weighted by the ratio between number of available spaces and total spaces on the game board. As the available spaces become scarce, it leaves fewer choices and becomes less important to keep close to the opponent.

2.3 AB_Custom_3

AB_Custom_3 function is similar to the improved_score function presented in sample_players.py file. It evaluates four factors: number of own available moves, number of opponent's available moves, active player at current state, and whether there are overlaps between own available moves and opponent's available moves. It tries to maximize the number of own available moves and minimize the number of opponent's available moves. However, if opponent is the active player at current state and there are overlaps between own available moves and opponent's available moves, the number of own available moves will be penalized by lessing the value by 1. It assumes that opponent will try to occupy the overlapping position and tried to avoid this to happen.

2.4 Comparison of Evaluation Function

AB_Custom_3 function is a simple variant to the improved_score function. Its strategy is to avoid overlapping when the opponent is the active player at current state.

AB_Custom_2 function presents an aggressive strategy, especially at the beginning of the game, by trying to stay close to the opponent so as to limit the opponent’s moves.

AB_Custom function’s strategy is to stay away from the opponent and copy opponent’s moves at the beginning of the game. As the game proceeds, there are fewer and fewer space left, staying away from the opponent becomes less import and AB_Custom function becomes more aggressive in looking for moves that both maximize its number of available moves and minimize the opponent’s number of available moves. It presents a certain degree of adaptiveness during the game play. A similar adaptiveness is also presented by AB_Custom_2 function, as staying close to the opponent becomes less important in later stage of the game. However, AB_Custom_2 function adopts the aggressive strategy throughout the game without adjustment.

3 Conclusion

The heuristic of AB_Custom function is ultimately used in the submitted agent because its adaptiveness and adjustment during game play. The test results also confirm the choice by showing that the heuristic of AB_Custom function presents the best win rates comparing to other evaluation functions.