Heuristic Analysis

Project name: Advanced game playing

Task: Implement the requested functions in game_agent.py

The submitted file *game_agent.py* has the implementation of the requested functions, namely the *custom_score* heuristic function, the *get_move* iterative deepening adversarial search function, the *minimax* adversarial search algorithm and finally the *alphabeta* pruning minimax search.

The comments in the code provide more explanation. The main difficulties I faced, were in the get_move function as it took me some time and trial and error to pass all the tests especially for the iterative deep search.

The minimax and alphabeta functions pass all seven test cases.

Task: Implement and analyse at least three evaluation functions

The following four evaluation functions have been implemented.

$Score\ A = moves_{player} - moves_{opponent}$

$Score\ B = moves_{player} - moves_{opponent} - distance_{center}$

$Score\ C = moves_{player} - moves_{opponent} - distance^2_{center}$

$Score\ D = moves_{player} - 2\ moves_{opponent} - distance_{center}$

Score A is a straightforward one and is based on the reasonable assumption that the more options available to an agent the higher the chance to win. It is a quite simplistic approach though, as in theory the position also should have some impact.

For that reason, it is intuitive to think that closer to the centre provides a more favourable position as again more options should be available as implemented in Score B.

Score C adds stronger importance to the distance from the centre by squaring it.

After evaluating the functions performance an extra one, Score D was added according a particular point from the lectures, to get a more aggressive agent, by doubling the contribution of the opponent's moves available.

In summary, the *tournament.py* script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on the hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

The explanation of the opponents is:

- **Random:** An agent that randomly chooses a move each turn.

- **MM_Null:** CustomPlayer agent using fixed-depth minimax search and the null_score heuristic
- **MM_Open:** CustomPlayer agent using fixed-depth minimax search and the open_move_score heuristic
- **MM_Improved:** CustomPlayer agent using fixed-depth minimax search and the improved_score heuristic
- **AB_Null:** CustomPlayer agent using fixed-depth alpha-beta search and the null_score heuristic
- **AB_Open:** CustomPlayer agent using fixed-depth alpha-beta search and the open_move_score heuristic
- **AB_Improved:** CustomPlayer agent using fixed-depth alpha-beta search and the improved_score heuristic

Figure 1 and Table 1 show the performance of the *ID improved* case. The simplest function Score A achieves slightly better result compared to the B case. However, I would trust more the B case, as its performance it is not far from the best case and it is relatively constant while all the other cases are having much larger variation.

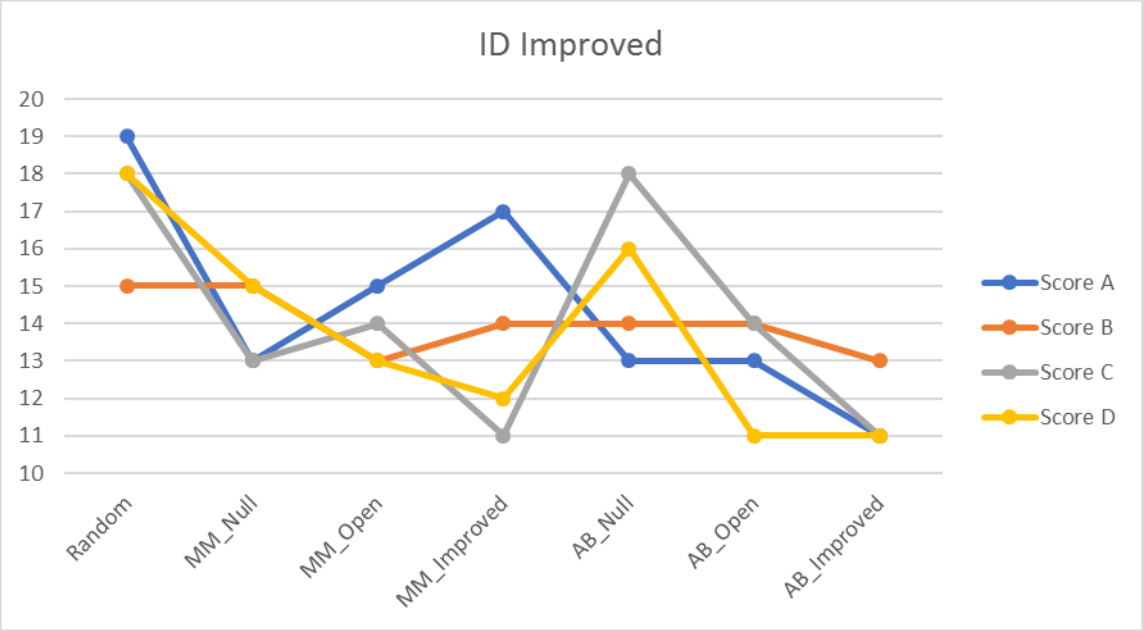

Figure 1: Score of wins out of 20 games of custom agent with Iterative Deepening against agents using other heuristic functions

Table 1: Detailed data for ID Improved case (wins out of 20 games played)

ID_improved Vs	Score A	Score B	Score C	Score D
Random	19	15	18	18
MM_Null	13	15	13	15
MM_Open	15	13	14	13
MM_Improved	17	14	11	12
AB_Null	13	14	18	16
AB_Open	13	14	14	11
AB_Improved	11	13	11	11
% Result	**72.14**	**70**	**70.71**	**68.57**

Figure 2 and Table 2***Table 1*** show the performance of the *Student* case, where the custom heuristic is used against same agents. In this case, Score B dominates the rest by more than 8%. It is an easy choice therefore to choose Score B as the best achieving heuristic function.

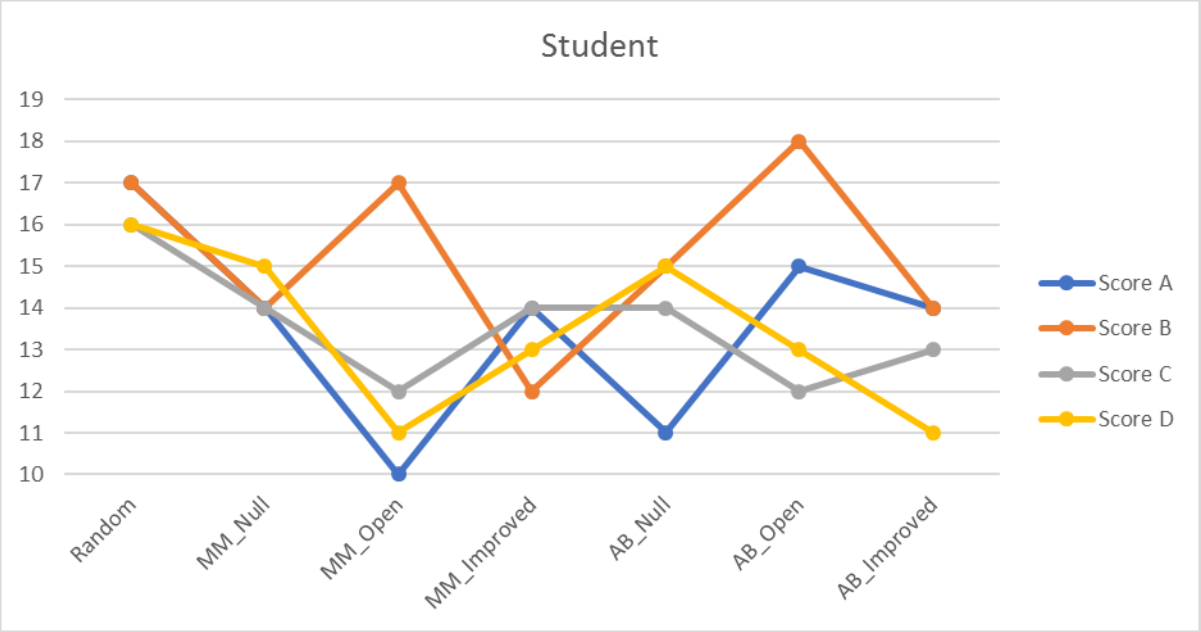

Figure 2: Score of wins out of 20 games of custom agent with Iterative Deepening against agents using same heuristic function

Table 2: Detailed data for ID Improved case (wins out of 20 games played)

Student Vs	Score A	Score B	Score C	Score D
Random	17	17	16	16
MM_Null	14	14	14	15
MM_Open	10	17	12	11
MM_Improved	14	12	14	13
AB_Null	11	15	14	15
AB_Open	15	18	12	13
AB_Improved	14	14	13	11
% Result	**67.86**	**76.43**	**67.86**	**67.14**

Finally, the total performance across the two cases is shown in Figure 3, with the Score B function having an upward trend, while the others are performing worse in the second case and are actually converging to the same point.

As a conclusion, it is interesting to note the Score B is achieving a good balance between the move and distance from the centre components. Score A, is decent but is lacking the distance component. However, when the distance component is intensified the performance returns to worse numbers. Also, when the opponents move factor is favoured again suboptimal performance is extracted.

I would believe that there is an endless number of combinations to try and also some other types of contributor that has not been taken into account. In overall though, Score B is achieving a decent performance and it the submitted one.

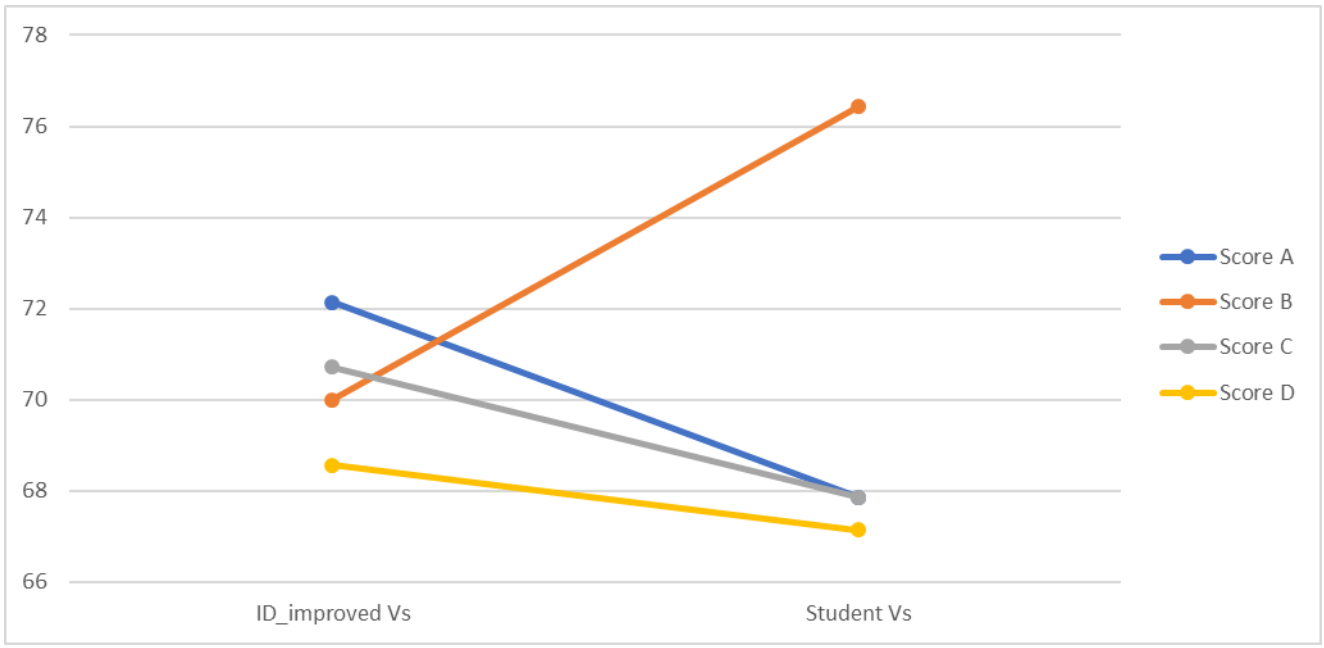

Figure 3: Performance between the two cases