Evaluation Heuristics

The purpose of the game is being the last player to play a legal move, so a good heuristic tries to maximize the number of valid moves available to play. It is also very important to try to minimize the opponent's number of available moves, so that the opponent has always less moves than our AI. A good heuristic that combines the previous ones has been shown previously in the course:

$$\#\ of\ player's\ available\ moves\ -\ \#\ of\ opponent's\ available\ moves$$

This heuristic works alright, nevertheless it assumes that the number of available moves for both the player and opponent are equally important. It could be the case that prioritizing the opponent's number of available moves could give better results, so I decided to test the `custom_score` function with different variations of the following formula:

$$\#\ of\ player's\ available\ moves\ -\ \alpha\ \times \#\ of\ opponent's\ available\ moves$$

Where α is a parameter that was varied to find the best results. It is important to mention that if the score being evaluated is a leaf node, it returns infinite if the player wins and minus infinite if the opponent wins.

The following table shows the win rates after trying the new score function for different values:

α	Student %
1	56.43
1.5	67.14
2	67.86
2.5	61.43
3	58.57

According to the results, it is recommended to use the following heuristic to improve the win rate:

$$\#\ of\ player's\ available\ moves\ -\ 2\ \times \#\ of\ opponent's\ available\ moves$$

This, given that the heuristic when $\alpha = 2$ has a higher win rate than the rest. Additionally it is equally as hard to compute (it is still a multiplication, no timeouts), and the chance that there's a better value for α that improves over this solution is low, given that several values were tested.

Supporting files

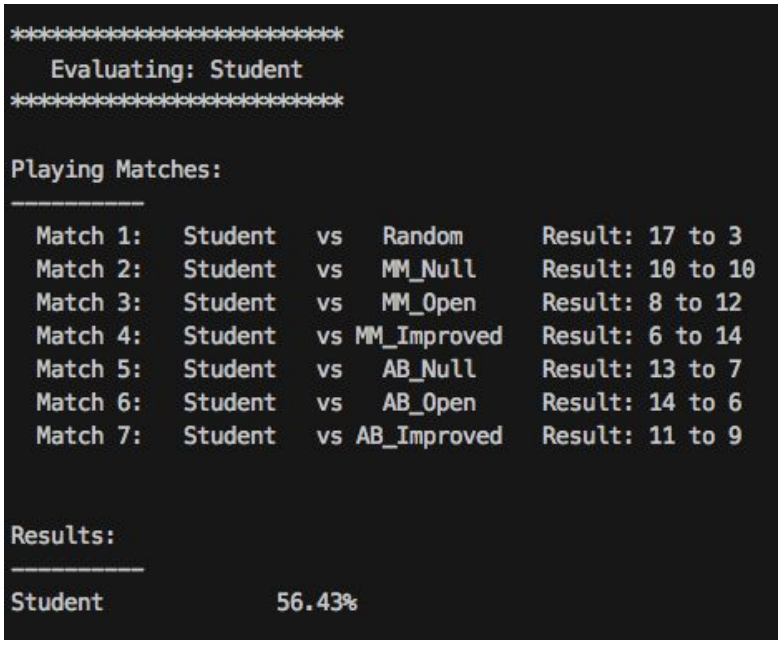

Image 1. Results Alpha equals 1.

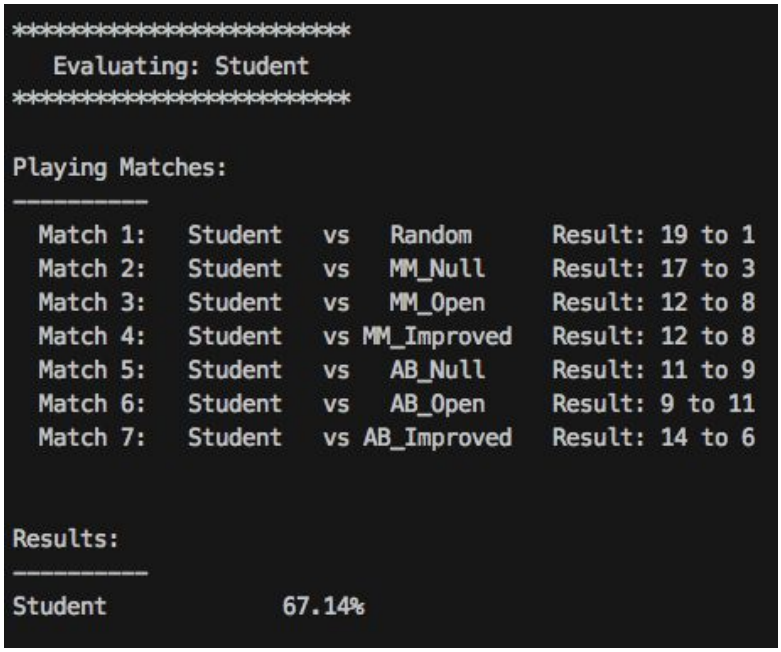

Image 2. Results Alpha equals 1,5.

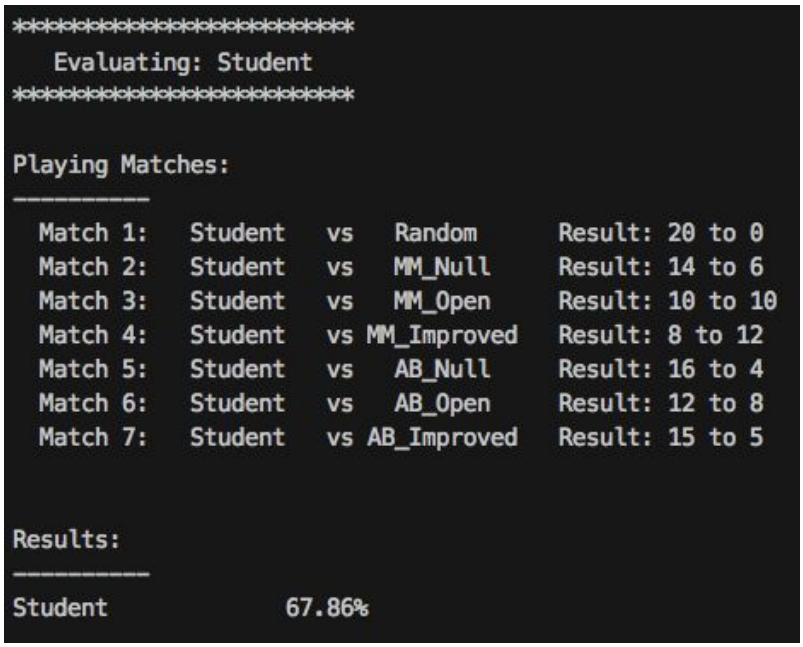

Image 3. Results Alpha equals 2.

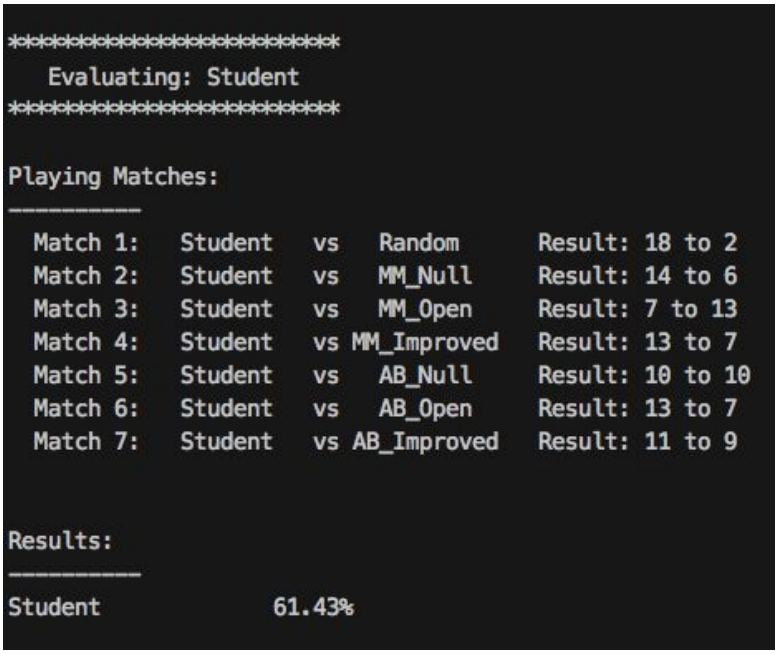

Image 4. Results Alpha equals 2,5.

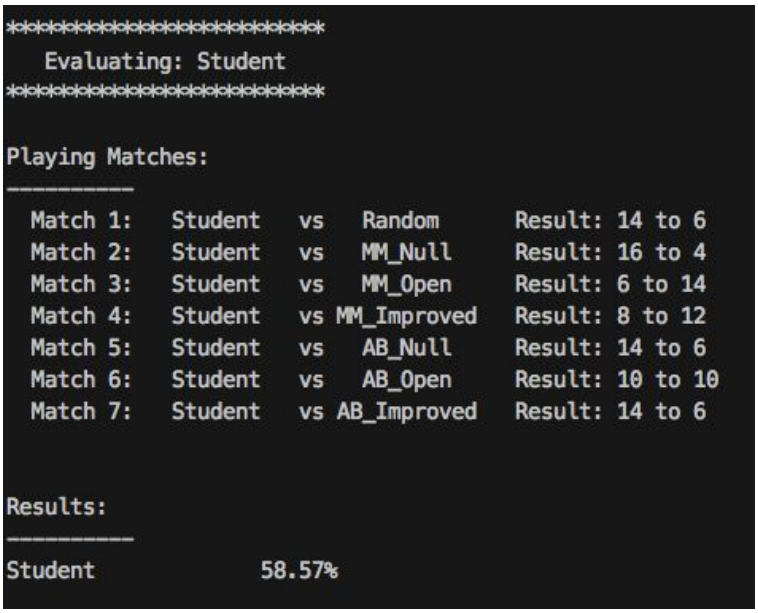

Image 5. Results alpha equals 3