Tetris AI

Al Jamil Arazas *
Ateneo de Naga University
Ateneo Avenue,
Naga City, Philippines

Louie Lester Pascual *
Ateneo de Naga University
Ateneo Avenue,
Naga City, Philippines

John Dinnex Valeros *
Ateneo de Naga University
Ateneo Avenue,
Naga City, Philippines

ABSTRACT

Tetris is a one-player video game created by Alexey Pazhitnov in 1984. In this project, we created an agent that plays the game using four heuristics: Aggregate Height, Complete Line, Holes, and Bumpiness. The weight of each heuristic was manually tuned to make the agent more intelligent. The resulting agent was able to score 459 so far, but since the state of the art agents in this game runs indefinitely, the agent created in this project is not in par with the state of the art. To improve the AI, the weight of each heuristic must be optimized using various optimization algorithms.

Keywords

Tetris; Artificial Intelligence;

1. INTRODUCTION

Tetris [4] is a popular video game created by Russian game designer Alexey Pazhitnov released in 1984.

The game is played on a two-dimensional grid, usually with a width of 10 and a height of 20. To play the game, the player must place a set of tiles called Tetrominoes, as shown in Figure 1, to the board. The goal of the game is to prevent the Tetrominoes from stacking up to the top of the board. Each time a row of tiles is filled with Tetrominoes, it will be cleared away and the Tetrominoes above it will fall down to take place of the cleared row.

In this project, we have created a Java-based Tetris AI that will play the game using four heuristics stated in [6].

2. RELATED WORKS

Studies [2][3][5] regarding Tetris mostly focuses on producing the best heuristics by using various algorithms to create its optimal parameters.

Yiyuan [6] produced a nearly-perfect bot that uses four heuristics: (1) Aggregate Height, or the sum of height in each column, (2) Complete Lines, the number of rows that are full of tiles, (3) Holes, the number of empty spaces with at least one tile above it and (4) Bumpiness, or the difference between the height of adjacent columns. The author then used a Genetic Algorithm to produce the optimal weights for each heuristic. The results of these heuristics multiplied with their corresponding weights were then summed up to compute the total cost of a move. The move with the least cost was the one chosen by the bot.

Figure 1: The tetrominoes O, J, L, Z, S, I, and T.

3. METHODOLOGY

The method that we used is basically a 1-level breadth-first search with heuristics and weights attached.

The first thing we did was to create a tetris game, which would be the environment of the AI. In this case, we took the Tetris code from Zetcode [1]. The code from the site is a basic tetris code which does not let you see the next block.

The second step was to represent the tetris in a 1 dimensional array, with the cells which contain a block having a value of 1 and those empty being 0. This updates whenever the board is updated. Since the code already has a representation of the board, made our 2nd representation by copying its format.

The third step is to create the successor function. The successor function creates all the possible moves that a player can do. The function would fill the blanks containing the temporary block with 7 to mark them as temporaries. The successor function also creates a list of the number of movements it needs to to get to the right position.

After creating the successor function, the next step would be to compute the heuristics. We took these heuristics from Yiyaun. The first heuristic is the number of lines cleared. This is computed by checking if all the elemnts in a row have a value greater than 0. The second heuristic is called the number of holes. This is counted by looking at all cells that have a value of 0 and checking if there is a cell with a value greater than 0 above it. Another heuristic is total height. This is calculated by adding all of the heights of each column. The last heuristic is called bumpiness. This is the sum of the change in height for every adjacent column.

After computing for these heuristics, we attach weights to them. We attach these weights as follows: 2.25 for number

*Email: {aarazas,lpascual,jvaleros}@gbox.adnu.edu.ph

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

of lines cleared -1.5 for total height -1.0 for number of holes -0.6 for bumpiness

As you may notice, most weights are negative. This is because you wouldn't really want the AI to have a high value for these heuristics. After computing the "score" for each successor, we take the one with the highest and execute the moves needed to attain the state of the successor.

4. RESULTS AND DISCUSSION

In our test runs, the highest number of lines that the AI cleared was 459 as seen on Figure 2. In this test run, the AI could still had the total height at a low value. However, the AI suddenly "stupefies," as if it cannot see a column and ignores it. We have 2 guesses in why this happens. The first one is because the AI was made inside the Board of the tetris game. It was not an AI that runs in parallel with the tetris, but runs with it. Whenever a new piece is created, the function that created the new piece also calls the function that creates the successor, then computes the compute heuristics function, and lastly call the function that does the move. This could have affected the AI in some manner that we may have overlooked. The second is that there might be a programmatic error inside, and that we might overlooked it too. Or maybe even the combination of both.

5. CONCLUSION

We created an Artificial Intelligence that plays the game of Tetris. We then created a function that makes it decide using the heuristics that we dictated. The AI showed good results since it has been able to gain score up to a little more than 450. However, the state of the art AI for this game could run indefinitely. With regards to this, the AI produced in this project may not be possibly state of the art, but it has been able to tackle the challenges of the game which is to create a bot that decides based on a given heuristic.

Additionally if this project will then be continued, we need to apply optimization algorithms to speed up and the make the AI a little smarter.

6. REFERENCES

[1] J. Bodnar. Tetris, 2008.

[2] N. Böhm, G. Kókai, and S. Mandl. An evolutionary approach to tetris. In *The Sixth Metaheuristics International Conference (MIC2005)*, page 5, 2005.

[3] T. Bousch and J. Mairesse. Asymptotic height optimization for topical ifs, tetris heaps, and the finiteness conjecture. *Journal of the American Mathematical Society*, 15(1):77–111, 2002.

[4] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *Computing and combinatorics*, pages 351–363. Springer, 2003.

[5] I. Szita and A. Lörincz. Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941, 2006.

[6] L. Yiyuan. Tetris ai - the (near) perfect bot, 2013.

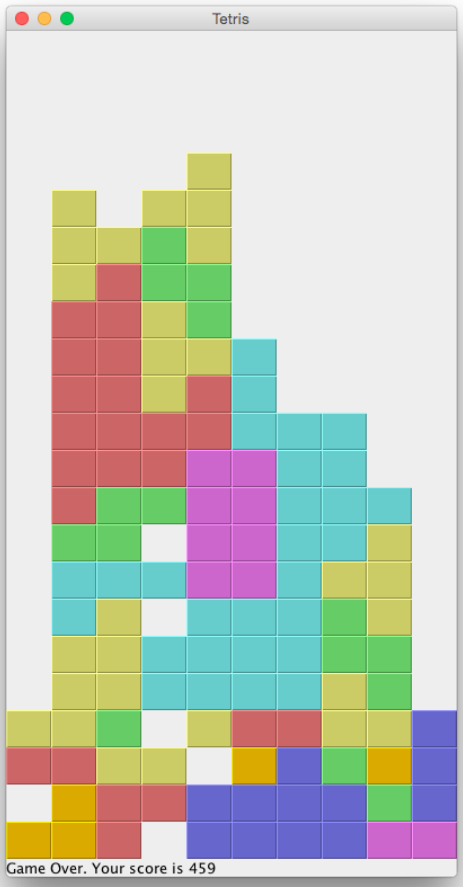

Figure 2: Best Run of the AI.