Evolving a Tetris Agent

Sam Davies
Department of Computer Science
University of Massachusetts Lowell
1 University Avenue, Lowell MA 01854
sam_davies@student.uml.edu

Sean Nishi
Department of Computer Science
University of Massachusetts Lowell
1 University Avenue, Lowell MA 01854
sean_nishi@student.uml.edu

Abstract

We created a genetic algorithm to improve the playstyle of an agent to be successful in a game of Tetris. The first test used an evaluation function based off human intuition on how a game of Tetris is supposed to be played successfully. The second test used the evaluation function of the first test with additional heuristics added. By testing different weight functions we were able to show improvement in the population as a whole and in the top scoring individuals when using an expanded heuristic function. Different reproductive strategies, weight representations as well as variations of algorithm parameters were further explored during testing and will be discussed in this paper.

Introduction

Tetris is a video game created by an AI researcher named Alexey Pajitnov in June 1984. The game consists of a board made up of 220 cells (10 columns, 22 rows), and a sequence of pieces that enter the board from the top and slowly descend to the bottom of the board. There are seven of these pieces, aptly named tetrominos - which each occupy 4 cells of board space. Once a piece can no longer fall any lower due to contact with the bottom of the board or a previously fallen piece, the next tetromino will arrive at the top of the board and begin its descent. By placing the tetrominos such that an entire row is occupied by tetromino cells, that row is then removed from the board, causing all board cells above it to shift down one row. If a tetromino is placed on the top of the board, the game ends. The goal of the game is to clear as many rows as possible. As this paper will discuss, there has been a multitude of AI techniques applied to creating an agent that can clear lines efficiently.

As this is a genetic algorithm, the simulations were set up so that an agent receives the current board object and an organism from the population. The agent would then utilize the weightings of the heuristics encoded by each allele of the organism that it was passed. The combination of the heuristic functions and the organism's weights for the values returned by those functions allows the agent to determine, according to organism's weights, which next board state has the highest utility. Specifically, the agent has the job of placing a single tetromino at a time. It carries out this task by checking each viable

placement of that tetromino and temporarily storing the score for each placement option. The score for each placement is determined by the heuristic functions, which each evaluate a Tetris board state in a unique manner, combined with the current organisms weightings for each of those heuristics. After the agent has considered each viable placement of the current tetromino, it chooses to carry out the set of actions necessary to achieve the placement which gave the highest score according to the weighted heuristic values. The agent continues to play the game with an organism until the game ends. Once the game has ended the number of lines cleared in the game instance is stored as the fitness value of the organism which the agent was imbued with. This process continues until the agent has completed a game with each organism in the population.

In this study, we will test two different sets of heuristic functions on the same genetic algorithm with the same initial population. The first set only contains four heuristics that stem from our human intuition about the mechanics of the game. The second trial will take the base function used in the first trial and will add additional heuristics that the agent will use. By adding more things for the agent to keep track of, we hoped to improve the performance of our agent.

The third trial incorporated a host of changes to the algorithm used in trial two, which includes changes to core processes, such as the crossover and mutation functions, which will be covered in detail in our results section.

Literature review

Previous studies have shown that when comparing genetic algorithms against other methods for creating an adept Tetris playing agent, genetic algorithms rein supreme. In Figure 1, the genetic algorithms presented were compared to other algorithms such as harmony search and ant colony optimization. The comparison showed that genetic algorithms were the most successful in creating an agent that could play Tetris well. Included in [6] were 32 criteria which were used in the utility functions of each algorithm. The specific criteria utilized for each algorithm was a mix of the 32 total criteria. From this study, we decided to use heuristics that were used in the top two performers, one of which was a genetic algorithm. The weights we included are the number of holes on the board and the number of removed rows.

TABLE I: Summary of Works Found in Literature

Method	Criteria utilized	Lines Cleared
Feature Based Dynamic Programming [7]	1, 6	32
Neuro Dynamic Programming [8]	1, 6, 9, 27	80
Kernels Based Regression + RRL [10]	1, 6, 9, 27, 3, 15, 2, 4, 5, 8,	≈ 50
Least Square Policy Iteration [9]	1, 3, 6, 18, 26, 10, Δ1, Δ3, Δ6, Δ18, Δ26,	< 3000
Ant Colony Optimization [14]	N/A[2]	17586
FFNN + PSO [15]	10, 1, 6 7, 11, 13, 15, 17, 19, 20, 21, 22, 28	1286705
Genetic Algorithms [19]	32	N/A[3]
Genetic Algorithms [12]	1, 6, 7, 10, 11, 13, 14, 17, 20, 21, 22	5498703
CMA-ES [17], [18]	6, 10, 20, 15, 19, 17, 28, 21, 22, 16, 30, 29, 31	≈ 3600000
Harmony Search [1]	1, 6, 7, 10, 11, 13, 12, 17, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30	416928

[2]The author did not specify the feature set utilized.
[3]The result measured was not number of lines cleared, but rather how well the Agent was able to fit 50 pieces in the board.

Figure 1, Table 1 [6]

Some previous research on applying genetic algorithms to Tetris has been done at our university. From our peer's work we see that an algorithm with many heuristics takes

a lot of time to go through a generation [5]. Their model consisted of 14 heuristics which led to a good learning rate but quickly required more time for a single evolution than the time allotted for their entire project [5].

The works of Heidi Burgiel has shown that there exists at least one sequence of tetrominoes that will force the game to end, even if the agent is playing with complete optimality [2]. From her work, we were able to confirm that the piece sequences given to our algorithm would not force the agent into a game ending state before it could be fully tested.

In our research we discovered we also had the option of improving upon the base genetic algorithm by passing values through a directed search method, such as the Nelder-Mead method [4]. However, we were interested in testing a pure genetic algorithm. Additionally, we experimented with various crossover and mutation techniques found in our referenced literature. Small technical variations were also investigated such as the benefit of using floating point weights, the value range of weights, etc. [1][3].

Methodology

After conducting our research we decided to use a pure genetic algorithm that would test two sets of heuristic functions. Unlike a regular game of Tetris, our Tetris agent did not have knowledge of the next piece. A look-ahead piece was not included in the evaluation of a piece placement because we were concerned about how long it would take to test organisms. Without an additional piece to take into account, the state space will stay relatively small and will not take as much time to evaluate the fitness of a placement.

We tested two different sets heuristic functions in order to compare how well the agent would play with additional constraints. The first evaluation function contained two heuristics from previous works [6] but also included two others that were based on our human understanding of how to be successful when playing the game. The second evaluation function took the same base function as the first test but with additional variables to weight.

For both simulations, the initial population and the piece sequences the organisms experienced from generation to generation were kept the same. We decided to keep the initial population and the piece sequences the same for both trials so we could determine how well the agents played with respect to each other.

In our first trial, the utility of a piece placement was determined using a linear sum of weighted heuristics. The function accounted for aggregate height, number of completed lines, number of holes on the board, and the height difference of adjacent columns, referred to as ‘bumpiness’, of the board. As explained in the literature review, the number of holes and the number of removed rows were included based on previous successful findings. The other two variables that were weighed were based off of our own intuition of the game. As a player wants to reduce the number of holes on the board in order to increase the number of rows removed per piece placed we knew

that previous studies were using some human intuition in their heuristics. However, we decided to account for aggregate height and the bumpiness of the columns. From our understanding of the game mechanics, keeping a low aggregate height means we can drop more pieces into the grid before reaching a terminal state. The bumpiness of the grid is also important as keeping the grid as monotone as possible allows more pieces to be placed.

In the second trial, we kept the initial evaluation function and added some heuristics that may not be as intuitive. The second evaluation function included: connected holes, blockades, the maximum altitude difference, weighted blocks as well as vertical and horizontal roughness. Connected holes was included to check the number of vertically connected holes. By reducing this number, we predict it will work similarly to the number of holes created. Weighted blocks is the sum of occupied cells on the board with a block on the n-th row counted n times. We expected this heuristic to further help with keeping the total height of the board low. By testing adding additional heuristics from a base algorithm, we were hoping to increase the top scores of the best organisms in the final population. The vertical and horizontal roughness heuristics counted the number of tetronmino cells that were between two empty board cells either vertically or horizontally respectively.

For each trial there were a few parameters that had to be chosen for our algorithm, and it was during our third trial that we began to experiment with variations of them. These parameters were the crossover rate, the mutation rate, the population size, and our elitism value. We had to make many adjustments to our genetic algorithm. Apart from adjusting the crossover and mutation rates, the functions driving these processes were also changed. Adjustments to these processes were first made after we referenced previous research into strategies that helped stave off or avoid early convergence in the population. Our hypothesis stated that a genetic algorithm was an apt solution to creating a proficient Tetris playing agent. In this case, a more fit set of weights results in a higher number of lines cleared by the agent.

Results

```
Generation: 185 , Average Lines Cleared: 24
Alva Rudge, Age: 13 Weights: [-45, -25, -112, 83] - Lines Cleared:93
```

Trial 1 Fittest Organism

```
Generation: 43 , Average Lines Cleared: 19
Marion Fells, Age: 23 Weights: [-32, -22, -99, 76] - Lines Cleared:94
```

Trial 2 Fittest Organism

Figure 2

For the first trial, in which each organism played the game with a random sequence of tetrominoes, we let the algorithm run for 318 generations. The highest scoring organism cleared 93 lines in generation 185. Although that organism was eventually replaced in a later generation, it was one of the most fit organisms for multiple generations.

The second trial, which also utilized a random sequence, was run for 267 generations which received maximum of 94 lines cleared in generation 43. The additional dependencies added to the heuristic function appear to have cut the

number of generations needed to produce a similarly scoring organism in half.

The average number of lines cleared for the final 10 generations for the first trial was 19.2 lines cleared. For the second trial, the average number of lines cleared for the last 10 generations was 20.7. This is significant as the first trial had more generations to produce an organism with a higher utility than the second trial but was unable to do so. This means the improved utility function was vital to improving the performance of the agent.

As shown in Figure 2 , our produced material includes logged results in the form of text files containing relevant generational information including population size, crossover and mutation rate, elitism percentage, and the weights and fitnesses for each organism in the population. These results confirmed our hypothesis that using our genetic algorithm, the evolutionary process produces not only a more fit population on average but more importantly, improves the fitness score of the most fit organism through generational cycles. In other words, the top performing organism was repeatedly outclassed by a newly produced organism in the subsequent generation(s).

Generation: 95, Cycle Time: 0:03:59.068000 Elite Average Lines Cleared in 1 Games: 136
Timothy Worthen, Age: 0 Weights: [-2.1613, -0.8431, -5.0451, 7.139, -5.1046, -1.0808, 1.2055, -1.1661, 1.2067, 4.0212, -3.3167, -2.2948] - Lines Cleared:151
Trial 3 Fittest Organism

Figure 3

This became increasingly evident as we tested beneficial adjustments to our crossover and mutation processes. Trial three incorporated our biggest set of changes to our algorithm. To begin, this trial kept the same sequence of tetrominoes for every generation, whereas our first two trials saw a new sequence for each generation.

Additionally, the third trial utilized new crossover and mutation methods. The crossover method in this trial randomly averaged the weights of two parents to produce offspring. Prior to this change we had implemented and tested a simple two-point crossover. We found that this crossover method was highly preferable and perhaps more akin to biological reproduction, as the newly produced weight contained information from both of the parents instead of just one.

Next, our mutation function was altered from bitflipping to gaussian mutation, which was able to produce more stable mutation. By this we mean that the mutation was now guaranteed to be within certain bounds, whereas randomly flipping a bit could cause a very large value change. These alterations to the algorithm proved worthwhile, creating our best organism yet in 95 evolutionary cycles. This organism was able to clear 151 lines, as shown in Figure 3.

Discussion

As shown in our logs, the first trial was able to produce an organism that was able to clear 93 lines. This means basing the heuristic function on human intuition of the game led to a successful agent. However, with so many generations we were expecting an organism that would clear hundreds of lines as the best genetic algorithm analyzed in [6] had millions of lines cleared as their best. As their heuristic function had more

than four heuristics to weight for an organism, it is likely a combination of the remaining heuristics they used that caused the algorithm to produce a fitter organism.

In the second trial, which ran for 267 generations and produced an agent that cleared 94 lines as its maximum, it appears the added heuristics helped produce a successful agent in less than half the generations required by the first trial. Unfortunately, it appears the additional heuristics did not significantly improve the number of lines cleared at the end of each run. Instead, the additional heuristics reduced the number of generations needed to produce an organism with a similar number of lines cleared.

Even though the results from the second trial did not produce an organism with a significantly higher score, it is clear by adding additional heuristics the average score per generations increases. Even with 16% less generation tested in the second trial, the average score already overtook the average score for the first trial. We are confident that after running the second trial until generation 318, like the first trial, the average number of lines cleared would increase. However, we cannot say if that algorithm would produce an organism that is better than its highest scoring one within the other generations that would be generated.

Trial three made it evident that reproductive processes are the most instrumental in creating a successful genetic algorithm. The changes introduced during this phase of testing created a significantly better organism in the same time frame as the second trial.

With minimal changes to our codebase we can set up our algorithm in a manner that would allow us to determine the most effective combination of the heuristics that we have incorporated and encountered in literature.

Conclusion

From our tests, we can conclude that a genetic algorithm is a good way to create an agent that can play Tetris well. The different heuristics that were tested confirmed that human intuition of how to play the game is a good starting point when playing Tetris.

When testing variations of an evaluation function, it is clear that adding additional heuristics to the evaluation function reduces the number of generations required to obtain an organism of similar success. By adding additional heuristics, we are able to improve upon the time the algorithm takes to produce a similar scoring organism by more than half.

Additionally, as shown in our third trial, the key to creating an efficient and successful genetic algorithm lies in the reproductive functions. Implementing these functions properly ensures that the best information from the surviving organisms propagates through the generational cycles, and that new information is introduced in a controlled manner in order to avoid early convergence in the population.

Update

After closer inspection, we identified an issue in which some configurations were being skipped in the placement evaluation phase. After a simple correction to our code, we were able to achieve thousands of lines cleared.

References

[1]Budin, Golub, Budin (1996) “Traditional Techniques of Genetic Algorithms Applied to Floating-Point Chromosome Representations”

[2]Burgiel, H. (1997). How to lose at Tetris. *The Mathematical Gazette, 81*(491), 194-200. doi:10.2307/3619195

[3]Michalewicz & Janikow (1991) “An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms”

[4]Rollinson, David and Glenn Wagner. (2010). “Tetris AI Generation Using Nelder-Mead and Genetic Algorithms.

[5]Shahar & West. (2010) “Evolutionary AI for Tetris.”

[6]Silva, Renan & Parpinelli, Rafael. (2017). Playing the Original Game Boy Tetris Using a Real Coded Genetic Algorithm. 282-287. 10.1109/BRACIS.2017.15.