An Evolutionary Approach to Tetris

Niko Böhm* Gabriella Kókai* Stefan Mandl*

*Departments of Computer Science 8 and 2, University of Erlangen-Nuremberg
Niko.Boehm@stud.uni-erlangen.de, {kokai,mandl}@informatik.uni-erlangen.de

1 Introduction

This paper describes the application of evolutionary algorithms to the popular game of Tetris. Tetris is a falling block game in which the player's objective is to arrange a series of different shaped bricks seamlessly in order to survive and thus get as many points as possible (there is no way to *win* tetris). The computer chooses the best move by rating possible subsequent game boards based on a rating function. This function is primarily a weighted sum of several subratings. The evolutionary algorithm is used to find optimal weights for these. Our results compete pretty well compared to other documented tetris playing programs.

2 Basics and State of the Art

Tetris is a computer game invented by Alexey Pajitnov in 1985. The game is played on a game-board, which is a two-dimensional rectangular grid, usually with a width of 10 and height of 20. From the top enter so-called *tetraminos*. There are seven distinct tetraminos, each of which occupies four grid cells of the game-board. The tetraminos move downwards with a certain speed while the player can rotate them and move them horizontally. One tetramino stops moving as soon as it hits the ground or previously placed tetraminos. At this point each fully occupied horizontal line in the game-board is removed and all blocks above slip down by one line. Then a new tetramino enters the board. The game ends as soon as a new tetramino cannot enter the board, because it is instantly blocked by occupied cells. In order to collect points, the player has to clear as many lines as possible.

Tetris has been analysed theoretically by many authors. Most notably is the work of Demaine *et al.* [DHLN03], who showed that the solution of Tetris is NP-complete even if the whole sequence of tetraminos is known in advance. Heidi Burgiel [Bur97] showed that there are tetramino sequences that terminate the game, no matter how "good" they are placed. Tetris has also been used as test-case for machine learning algorithms, e.g. by Driessens [Dri01], or Tsitsiklis and Roy [TR96]. As far as Tetris-related results are published they cannot compete with specialised algorithms. On the world wide web [Fah03], Colin P. Fahey documents "world records " of tetris playing programs: He presents a one-piece (i.e. does not consider the preview tetramino) algorithm by Pierre Dellacherie, which clears 650,000 lines

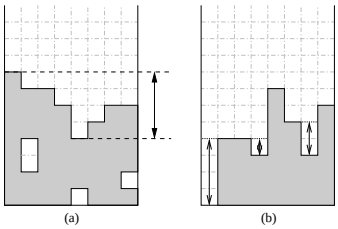

Figure 1: Altitude difference and wells

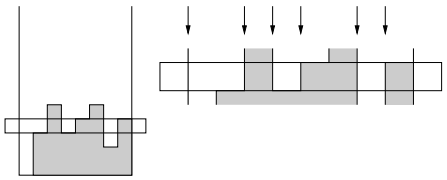

Figure 2: Rowtransitions

in average. Fahey himself reports a game clearing 7,216,290 lines using his own two-piece algorithm. Another one-piece algorithm by Roger Espel Llima clears an average of 42,000 lines over 150 games.

3 Evolutionary optimizing Tetris

To determine the best tetris move, we can chose a simple approach using two-level search for every possible game-board b that can be reached using the two known tetraminos, then rate all these boards and take the path to the best-rated one. The problem is thus reduced to finding a good rating function.

A hypothetically perfect rating function $R^*(b)$ depends on more than these two tetraminos, possibly on the infinite sequence of future tetraminos. As the computation of the perfect move is NP-complete even in the case where this sequence is finite and known in advance [DHLN03], we chose a heuristic approach to compute a rating function $R(b)$ in order to get results within a reasonable amout of time and under realistic (real games with random tetramino sequences) conditions. $R(b)$ is composed of n simpler rating functions $r_i(b); i \in \{1, ...n\}$ each of which rates the game-board according to one of the following criteria:

1. *Pile Height*: The row of the highest occupied cell in the board.
2. *Holes*: The number of all unoccupied cells that have at least one occupied above them.
3. *Connectd Holes*: Same as *Holes* above, however vertically connected unoccupied cells only count as one hole.
4. *Removed Lines*: The number of lines that were cleared in the last step to get to the current board.
5. *Altitude Difference*: The difference between the highest occupied and lowest free cell that are directly reachable from the top. See figure 1 (a).
6. *Maximum Well Depth*: The depth of the deepest well (with a width of one) on the board. In figure 1 (b) this is the leftmost well and has a depth of 4.
7. *Sum of all Wells* (CF): Sum of all wells on the board. This is 7 in figure 1 (b).
8. *Landing Height* (PD): The height at which the last tetramino has been placed.
9. *Blocks* (CF): Number of occupied cells on the board.
10. *Weighted Blocks* (CF): Same as *Blocks* above, but blocks in row n count n-times as much as blocks in row 1 (counting from bottom to top).

11. *Row Transitions* (PD): Sum of all horizontal occupied/unoccupied-transitions on the board. The outside to the left and right counts as occupied.
12. *Column Transitions* (PD): As *Row Transitions* above, but counts vertical transitions. The outside below the game-board is considered occupied.

Criteria 1–6 were originally discovered by ourselves, but we found some of them being used by others, too. However, the results only slowly approached the 1,000,000 mark, so we added some other criteria (7–12) found in Colin Faheys "Standard Tetris Application" originating from Colin Fahey (CF) and Pierre Dellacherie (PD) [Fah03]. These criteria were combined to the following overall rating-functions:

1. *linear rating-function* (dot product): $R_l(b) = \sum_{i=1}^{n} \omega_i r_i(b)$
2. *exponential rating-function*: $R_e(b) = \sum_{i=1}^{n} \omega_i r_i(b)^{e_i}$
3. *exponential rating-function with displacement*: $R_d(b) = \sum_{i=1}^{n} \omega_i |r_i(b) - d_i|^{e_i}$

The first one is straightforward but may not be really good to approximate tetris, because from the view of a (human) tetris player, there's no big difference if the pileheight is 2 or 4, but there's a huge difference between height 15 and 17. Both are considered equally with R_l. To cover this, we use the exponential rating-function R_e, which gives a better approximation. The third rating-function R_d is based on the idea, that the optimum value for a certain criteria might not be zero. If the objective would be clearing four lines at a time as often as possible, the desired value for *Maximum Well Depth* would be at least four. The weights, exponents, and displacements, respectively, are to be discovered by the evolutionary algorithm.

So the genotype consists of up to three chromosomes depending on the very form of the combined rating function under consideration: The *weights*-chromosome $(\omega_1, ...\omega_n)$, the *exponents*-chromosome $(e_1, ...e_n)$ and the *displacements*-chromosome $(d_1, ...d_n)$. Each of the ω_i is an integer number, whereas each of the e_i, and d_i is a real number.

3.1 Determining the fitness

The fitness of a genotype is measured on the phenotype. In our case, the chromosomes of the genotype are used to create an instance of the board rating-function guiding the search for the next tetramino location. The fitness function has to be designed so that higher fitness values correspond to better tetris performance. The performance of a single game of tetris can be determined in a couple of ways. These include counting placed tetraminos, counting cleared lines or counting "points" for different kinds of actions, e.g. clearing more than on line at a time, dropping tetraminos from a certain height, etc. We chose the number of placed tetraminos as the performance measure, which is (in large scales) proportional to the number of cleared lines, the measure mostly used. However, this provides no advantage for clearing more than one line at a time, so that greedy clearing of lines is enforced. The fitness of an individual is computed as the arithmetic mean of the performance of multiple (in our case five to twelve) runs of tetris. To smoothen the result, the best and worst game performance is ignored when at least seven games are played. Due to the low number of games this average is not perfect. We observed individuals that reached a certain number of cleared lines as mean of ten games being noticeably better or worse on ten other games. Playing whole games for fitness determination hast a downside, though. The better the individuals get during evolution, the

longer every single game of tetris lasts. A really good game, clearing millions of lines can last several days. To compesate for this, we reduced the size of the gameboard to 6×12, which makes Tetris much harder and games seriously shorter.

3.2 Reproduction

The individuals of the next generation are derived from the individuals of the previous generation using recombination, mutation, and scaling. However, some of the fittest individuals are copied literally to keep a minimum number of good individuals (elite-selection). Selection of parents is done randomly by using a fitness proportionate selection function. Two distinct individuals are chosen and their chromosomes are recombined with the standard two-point crossover operator. Each gene (i.e. one weight, or exponent, or displacement value) is mutated with a certain probability. Mutation works by multiplying the current value of the gene with a gaussian distributed random number with $\mu = 1$ and adjustable standard derivation. By allowing a value of 0 and using multiplication for mutation, this mutation operator leaves the possibility that genes are virtually turned off in the case the gene gets zero. This behavior delivers hints at which criteria are essential for good tetris performance and which are not. Another operator is only applied to the weights-chromosome, scaling each gene with a (per chromosome) constant factor. This does not affect the fitness of an individual, but is used to create more variety in the gene pool.

4 Results and Analysis

Using the linear function R_l with criteria 1–6 on a 10×20 game-board resulted in the best games of 859,520 cleared lines with a top performance of the best individuals of about 170,000 cleared lines on an avergage of five played games. Population size was usually 100 individuals. Due to the lack of time, most runs were terminated between 20th and 30th generation after the strong improvements were over, so that we could try other parameter settings. After using criteria 1–6 to achieve the results above, we introduced criteria 7–12. All further tetris-games took place on a 6×12 game-board unless stated otherwise. We started over again with the linear rating-function R_l, so that we could draw comparisons to the results on the regular-sized board. Since the games were also much shorter we were finally able to let them run for a great number of generations. In nearly all runs the performance gains decreased between generation 20 and 30. The best observed results were 4,007 cleared lines as average of 12 runs of the best individual, and 24,311 cleared lines in the very best game. An example run is shown in figure 3(left). The highest line shows the very best game of tetris played in the generation. The middle line shows the results of the best individual, i.e. the average of the games of this individual. The last and lowest line shows the mean of all games of all individuals. All results are displayed as the number of cleared lines.

To improve these results, we considered the exponential rating-function R_e on the 10×20 board. There was only one run with this rating-function, as it took three months to complete 30 generations. The results as seen in figure 3(right) corroborate our theory that this function might yield better tetris-performance: a best game with 5,498,703 cleared lines and a best individual with an average of 1,359,184 lines in five games, which exceeds all results of the

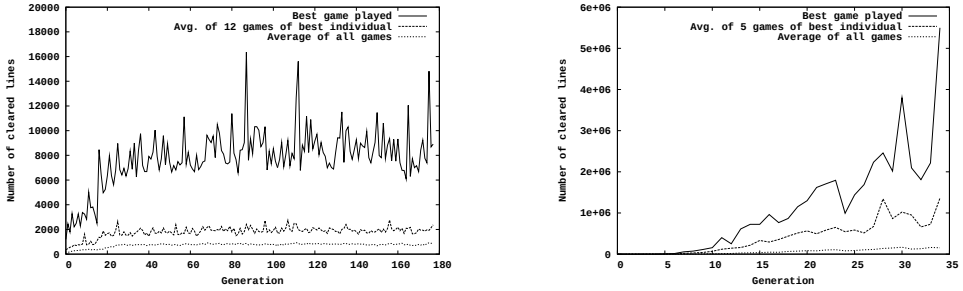

Figure 3: Two examplary evolutions

evolutionary runs with R_l as rating-function. As expected the exponential rating-function R_e achieved significantly higher results on the 6×12 game-board, too. The best individual has achieved an average of 15,146 cleared lines in nine games, and the best game observed had 68,421 cleared lines. Unlike the runs with R_l the results sometimes continue to improve later.

4.1 Discussion of the resulting vectors

The criteria which get the highest weights in the evolutions on the 10×20 game-board are the criteria *Pile Height* (1), *of Holes* (2), *Connected Holes*(3) and *Maximum Well Depth* (6). These criteria are dominant for the rating of a game-board. As already observed by human players, the prevention of holes and keeping the pile low are considered good. Similar observations can be made with the evolutions applying R_e as rating-function. As stated above, the evolutionary runs on the 6×12 game-board were run for a long time even after the first phase of strong improvements. In all evolutions using R_l the weights of several criteria become or approach zero, and are therefore not affecting the rating. These are not always the same in all evolutions, though. Obviously, several local maxima in the search area are found, which are on a similar fitnes level, but in completely different places. An explanation for this would be that different criteria describe similar facets of the game-board.

The results with R_e are less clear. The reason for it may be the double number of dimensions of the search area. Especially the criteria *Connected Holes* and *Maximum Well Depth* got exponents larger than one, which means that become more important for the total rating. In contrary to our originally assumption, the *Pileheight*-criteria gets mostly exponents around one. Surprisingly the results developed with R_d as rating-function did not exceed those achieved with R_e, despite the fact that everything that is representable with R_e is also representable with R_d (using displacement-values of zero). The conclusion hereof is, that the search area contains too much local maxima which the evolutionary algorithm could not escape from.

4.2 The step back from small to large

The results developed on the small game-boards cannot all be verified on the large ones, because of the exhaustive time needed for such a test process. So only some random samples were run. Using the linear and exponential rating-function all games clear more than one million lines. There is a tendency that games using the linear rating-function play better on

the large game-board.

For example a game with R_e and the vectors

$$\begin{aligned} \omega &= (2382, -135028, -48491, -74343, 59739, -72528, -61591, 13344, -19737, -47653, -50015, -144688) \\ e &= (0.935, 1.839, 0.905, 1.078, 0.858, 2.207, 1.565, 0.023, 0.117, 0.894, 1.474, 1.346) \end{aligned}$$

clear 34,440,225 lines, while other tested games ended already between six and twenty million removed lines.

A game with R_l and the weight vector ω = (-62709, -30271, 0, -48621, 35395, -12, -43810, 0, 0, -4041, -44262, -5832) reached more than 480 million removed lines.

The results achieved with R_d are not nearly as good, which is no big surprise, considering that the displacement-values depend heavily on the codomain of the corresponding r_i. No attempt was taken to scale those values. Most tested vectors led to games that only removed really few ($<$ 100,000) lines.

5 Conclusion and Future Work

We report results on using evolutionary algorithms in order to evolve a heuristic function for the game of tetris. The performance of our evolved tetris game-board rating-function compares nicely with other reported results. By extending the simple, linear rating-function, we were achieving a significant increase in tetris performance. Using more complex rating-functions and criteria yield better performance but is more likely to be bound to the board-size on which the evolution takes place.

To improve the speed of the fitness determination on big boards, we plan to verify a hypothesis of Colin Fahey [Fah03], which claims that the number of cleared lines of a tetris game can be statistically estimated from the pileheight distribution in the beginning (about the 100,000 placed tetraminos) of a game. Another direction of future work would be the extension of the algorithm to multiplayer tetris.

References

[Bur97] Heidi Burgiel, *How to lose at Tetris*. Mathematical Gazette:p. 194 (**July 1997**).

[DHLN03] Erik D. Demaine, Susan Hohenberger and David Liben-Nowell, *Tetris is Hard, Even to Approximate*. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON 2003)* (**2003**).

[Dri01] Kurt Driessens, *Relational Reinforcement Learning*. Lecture Notes in Computer Science, 2086:pp. 271+ (**2001**).

[Fah03] Colin P. Fahey, *Tetris AI* (**2003**), URL `http://www.colinfahey.com/`

[TR96] John N. Tsitsiklis and Benjamin Van Roy, *Feature-Based Methods For Large Scale Dynamic Programming*. Machine Learning, 22:pp. 59–94 (**1996**).