**A Tetris Player Evolved from a Genetic Algorithm in Tandem with an Artificial Neural Network**

The computer game Tetris has been shown to be NP-Complete, meaning that an "exact" solution cannot be computed in any remotely reasonable amount of time. For this reason, the problem of playing Tetris is a popular topic in artificial intelligence research. This paper explores the creation and performance of a Tetris Player MKB (Mother Knows Best) that uses a genetic algorithm in tandem with an artificial neural network that has been trained on a human’s Tetris playing tendencies.

**Introduction:**

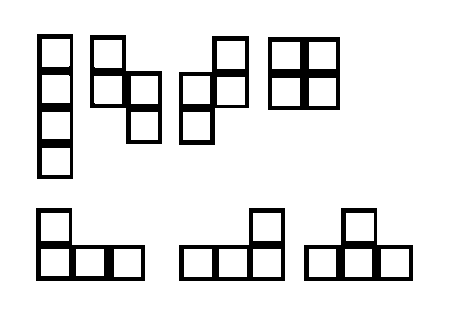
Tetris is a computer game developed in the 1980s by Alexey Pajitnov. There are many slight differences when implementing the game of Tetris that greatly influence performance [1]. The one described here is the implementation used throughout this paper. The game of Tetris is played on a 10x20 grid. One of seven tetrominos (see figure 1) appears at the top of the matrix and moves downward. The player may rotate the tetromino and move it laterally within the matrix so that it falls in the desired location. The game of Tetris is over when a tetromino’s placement results in any part lying outside the 10 x 20 grid. Game over is avoided by the player placing tetrominos so as to complete entire horizontal rows. When this occurs, the entire row disappears and all blocks (a tetromino is composed of four blocks) above the completed row move down by one unit. If the next piece is known to the player, this is known as “two-piece” Tetris (as opposed to one-piece). The chosen implementation of Tetris presented here concerns two-piece Tetris.

Figure : The Seven Possible Tetrominos

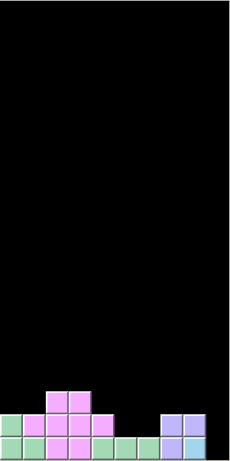
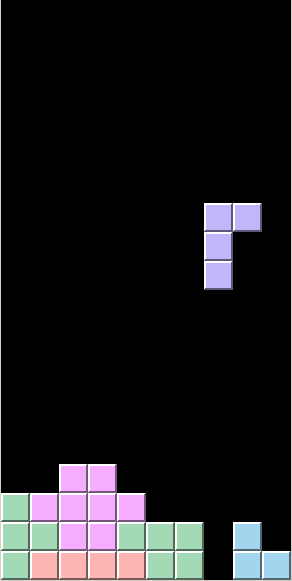


Figure :   
Left: Before placement Right: After placement

**Genetic Algorithm:**

Tetris has been shown to be NP-Complete *even if* all the pieces are known in advance [2]. Tetris is therefore a popular target of artificial intelligence techniques such as genetic algorithms. We can exhaustively list all possible tetromino placements of a currently falling tetromino in combination with all possible placements of the (known) next tetromino without being computationally burdensome. We refer to a particular pair of tetromino placements in this set as a *scenario* denoted where ,

.

Note

A Tetris game can be represented as a sequence of “currently falling” Tetromino placements, . The problem of playing Tetris is then selecting a “best” (and therefore a). The following eight heuristics were considered in the creation of MKB to estimate the desirability for each

The sum of the total ‘holes’ underneath a placement  
 The sum of the number of new ‘holes’ underneath a placement  
 The sum of the number of lines cleared by a particular placement  
 The width of the tetromino’s orientation divided by its height  
 The sum of the height of all blocks of the tetromino (counted from top)   
 The sum of the number of blocks being touched laterally by other blocks  
 The sum of the number of blocks being touched laterally by edge of the grid  
 The sum of the blocks of the tetromino that land on a solid (block or floor).

(For a list of popular features in other Tetris players see [1])

For a particular point in time , note the number of possibilities for is (the number of all possible secondary Tetrimino placements). Let

.

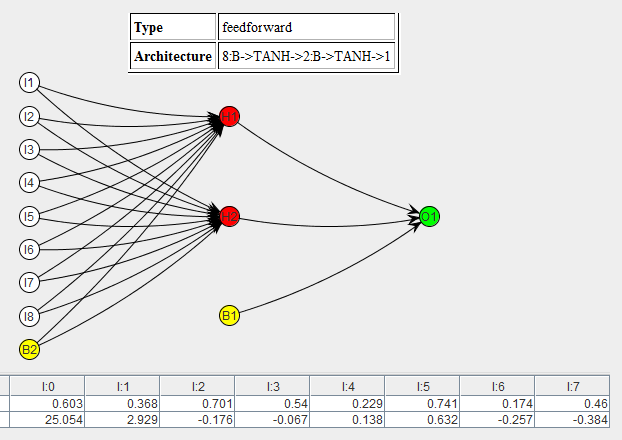
This introduces a set of weights, where each corresponds to an . Selecting these weights is accomplished by a genetic algorithm (GA).

The GA to create MKB started with a population of 300 sets of randomly selected weights with the condition . Each set of weights in the population was then used to play a game of (simplified [1]) Tetris in the method described above. The set of weights in the population that resulted in the highest score (number of tetrominos placed before game over) was selected to parent generation two by producing 300 sets of slight random mutations of its weights[[1]](#footnote-2). The process is continued in the manner described until so desired[[2]](#footnote-3). After quite a few generations and several hundred thousand games of Tetris, the observed best game scored 4,218,350 tetrominos[[3]](#footnote-4). However, the Tetris player from time to time made tetromino placements that “would make a human player cringe.” These “cringe” placements occurred about 9% of the time[[4]](#footnote-5). To correct these “cringe” placements and improve upon overall score, an artificial neural network was introduced. The method of using a set of weights to determine the “desirability” of a scenario will from here on be referred to as the GA.

**Artificial Neural Network:**

Artificial neural networks (ANNs) are good at classification, prediction, pattern recognition, and optimization [3]. Therefore a good ANN builds on these strengths. The ANN in MKB builds on the strengths of classification and prediction. The ANN’s goal is not to find an optimum placement of a tetromino (the GA does that well 91% of the time), but to simply rate a particular tetromino placement on a scale of “how human” the placement was (1 being the most human a placement could be, -1 being the least), and adjust the accordingly to a more “human-like” placement. Note we are making the assumption that human play is more desirable than the GA’s choice and will positively affect the score of the game.

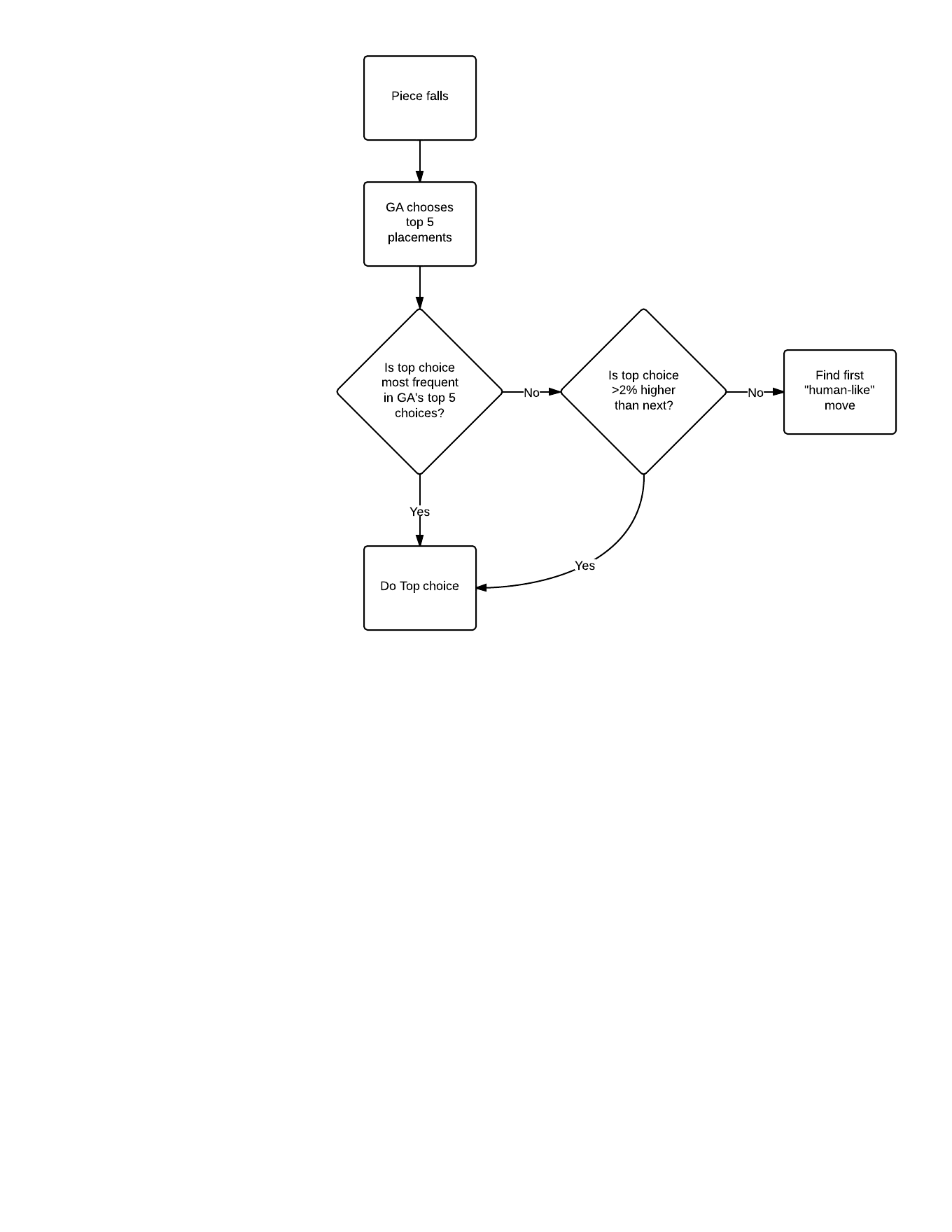
The ANN was trained on a total of 3,144 corrections to the GA’s placements made by a local Tetris expert (my mother Shirley Kline). The optimum inputs for the ANN were found to be the eight heuristics mentioned above. Further description of the highest performing ANN is described below.



Creation, training, and code were completed using Encog Workbench 3.2.0 [3]. For an in depth explanation of ANN’s, see <http://www.colinfahey.com/neural_network_with_back_propagation_learning>.

**Connecting the GA’s weights and the ANN:**

Simply put the GA is used to come up with the top five placements for a falling tetromino. Then based on a metric, whether or not the GA’s top placement should be questioned is determined. The best metric found for determining when the placement chosen by the GA’s weights should be put in question, is a simple frequency count in the GA’s top five choices to place a falling tetromino (recall that the “desirability” of a placement is determined by exploring the placement of the current piece in combination with the next piece. For this reason, identical first piece placements are not uncommon). If the top choice is not the most frequently occurring placement (or ties) in the top five placements, the top five placements are sent to the ANN to receive a “how human” rating. Experimentally it has been found that the best results occur when evaluating the top choices from highest rated to lowest and taking the first positive “human” rating. 255 games were played using this ANN in tandem with the GA and compared with the results of playing 255 games using the same weights without an ANN. Even though the corrections made by the ANN to the GA appear to be beneficial, the result showed performance almost exactly the same as the version without the ANN. The version implementing the ANN even performed slightly worse. We concluded the ANN must be making “unwarranted” corrections. Therefore the final version of MKB includes a “back out” feature described as follows. If the GA’s top choice’s “desirability” is rated as to the GA’s second choice, go with the GA’s first choice.



Running the genetic algorithm using the above decision flowchart to find perhaps even more optimum weights resulted in the weights used in the final version of MKB.

**Results:** For a long time the best known two-piece Tetris player was that of Colin Fahey [4]. He reports a game of clearing 7,216,290 lines (~18,000,000 pieces) [5]. A group at Northeastern University claimed to have created game to rival Fahey’s [6]. In 2010 “Building Controllers for Tetris” was published reporting a Tetris player that bested all known Tetris players. Although never tested on non-simplified Tetris, they conservatively estimate a lower bound of their player to average 910,000 lines [1].

Using non-simplified Tetris, 23 games of Fahey’s, Glametris’s, and my own were played.

Using these 23 games with 95% confidence Fehey’s averaged 1,274708 pieces, Glametris averaged 1,212,500 pieces, and MKB averaged   
50,0171,556 pieces with the highest game placing ~120 million pieces. 50 million pieces converts to around 20 million lines cleared. To the best of my knowledge, MKB is the best two-piece Tetris player known. That being said, in December 2013, “Approximate Dynamic Programming Finally Performs Well in the Game of Tetris” was published announcing a one-piece (simplified) Tetris player that averaged 51 million lines (~127 million pieces) [7]. However “Building Controllers for Tetris” cites one particular one-piece Tetris player as averaging 5,200,000±20% lines in simplified Tetris, and 660,000±27% lines in non-simplified [1]. Assuming that this is a typical and linear transformation, the reduction from simplified to non-simplified is ~7.88 times lesser—or from 51 million to about 6.47 million lines. However, when comparing one-piece to two-piece Tetris players, the one-piece score is “severely underrated” [7]. While it is likely that MKB would be beaten by a two-piece implementation of the aforementioned Tetris player, there is no definitive “winner.”

# Bibliography

|  |  |
| --- | --- |
| [1] | C. Thiery and B. Scherrer, "Building Controllers for Tetris," *International Computer Games Association Journal,* pp. 32:3-11, 2009. |
| [2] | N. Böhm, G. Kókai and S. Mandl, "An Evolutionary Approach to Tetris," *The Sixth Metaheuristics International Conference,* pp. ??-1-6, 2005. |
| [3] | J. Heaton, "Heaton Research," Heaton Research, Inc, [Online]. Available: http://www.heatonresearch.com/. [Accessed 2014]. |
| [4] | S. J. Sarjant, *SmartAgent--Creating Reinforcement Learning Tetris AI,* 2008. |
| [5] | C. Fahey, "Tetris," 2012. [Online]. Available: http://www.colinfahey.com/tetris/tetris.html. |
| [6] | G. Coppersmith and J. Dennis, "Official site for Glamestris," Northeastern University College of Computer Information Science, 2004. [Online]. |
| [7] | V. Gabillon, M. Ghavamzadeh and B. Scherrer, "Approximate Dynamic Programming Finally Performs Well in the Game of Tetris," *Neural Information Processing Systems,* pp. 1754-1762, 2013. |

1. For each mutation of each weight. A population (of a generation) may contain duplicate weight sets, but it is unlikely given the domain size. [↑](#footnote-ref-2)
2. If no member of a population resulted in a game score greater than its parent, the size of the generation was simply doubled. If this generation still did not result in a game score higher than its seed (parent) weights’ score, the seed weights are abandoned. [↑](#footnote-ref-3)
3. The weights array for this game are as follows: [↑](#footnote-ref-4)
4. Observing 1000 placements, 93 were “cringe” moves. [↑](#footnote-ref-5)