Playing NES Tetris with No Piece Rotations

Adler A. L. Soster
Joinville Technological Center
Federal University of Santa Catarina
Joinville, SC, Brazil
soster.adler@gmail.com

Michael Birken
meatfighter.com
o__1@hotmail.com

Pablo A. Jaskowiak
Joinville Technological Center
Federal University of Santa Catarina
Joinville, SC, Brazil
pablo.andretta@ufsc.br

***Abstract*—Tetris is one of the highest-grossing video games in all history and, despite of its age, remains quite popular. One of its most acclaimed versions was released in 1989 for the Nintendo Entertainment System (NES) and is often referred to as NES Tetris. This particular version of the game has led to the creation of the Classic Tetris World Championship (CTWC), resulting in growing popularity and alternative modes of gameplay. In one of such variants, players aim to clear as many lines as possible, with an additional constraint: piece rotations are not allowed. In this work we build and evaluate agents to play this particular variant of the game based on different metrics that grade board configurations. The relative importance of metrics is determined with the Particle Swarm Optimization. Our best results match those of top performing human players, even though the metrics we employ were not specifically developed for this game variant.**

***Index Terms*—Tetris, NES Tetris, Heuristics, Metrics, PSO.**

I. Introduction

Tetris is a computer/video game created in 1984 by Alexey Pajitnov. The game is composed of a board with 20 lines $\times$ 10 columns and seven different pieces, the so-called tetrominoes, which fall, one at a time, from the top of the board in a particular order (in most implementations randomly). As a tetromino falls, the player has to rotate and position it in the board, aiming to completely fill lines and making them disappear, that is, clearing them. Line clears can happen in one, two, three or four (a so-called Tetris). The ultimate goal is to maximize the number of points, which are awarded to the player as lines are cleared. The more lines cleared at once, the more points awarded, with higher levels awarding more points per lines cleared, given that pieces fall at a faster pace. The game is over when no more pieces can be fitted in the board.

Despite of its age and overall simplicity, Tetris is a quite popular game, with new implementations released recently (e.g., Tetris Effect [1]). Other than that, the game has driven the creation of fandoms and championships, among which the Classic Tetris World Championship (CTWC) stands out[1]. In the CTWC players compete against each other in the Nintendo Entertainment System (NES) version of the game, NES Tetris for short. In this version of the game the player has to deal with: (i) a limited look-ahead, that is, he/she knows only the current and next piece and; (ii) increasing fall speeds, as the number of lines cleared increase and levels progress.

[1]CTWC Official Website: https://thectwc.com/

Player scores in CTWC have increased considerably in last couple years, pushing the limits of what was though to be possible in the game. In recent editions of the tournament several players were capable of scoring over 999,999 points, which is called a *max out*, given that the original implementation of NES Tetris don't even register scores higher than that. Such scores are remarkable given that in NES Tetris playing above level 29 is nearly impossible (given that pieces fall at an alarming rate) and games in CTWC start at level 18.

In a recent work, M. Birken [2] employed Particle Swarm Optimization (PSO) [3] in order to train a NES Tetris playing agent and obtained results comparable or even better than the ones observed on CTWC. On average, the approach obtained a score of 1,036,706 and a standard deviation of 149,458. It is important to note that during evaluation all games started on level 19 and were played at most until level 29, in order to make results comparable to those observed on CTWC.

In this paper we investigate how the Particle Swarm Optimizations (PSO) and existing metrics perform when playing NES Tetris with an additional "twist" or constraint: *no piece rotations are allowed*. Note that this is not a particular mode of the NES Tetris *per se*, but an informal variant, that is, it depends on the user not pressing the buttons responsible for rotations. In this variant a player seeks to maximize the number of lines cleared, instead of the overall observed score.

By the year of 2019, human player recordings featuring this particular variation on NES Tetris started to emerge on YouTube. The first registers show top performing players clearing at most 17 lines. From then, these numbers have been increasing consistently, with a complete account of the timeline given by [4]. To the best of our knowledge, as the writing of this paper, the record for the no rotation variant of NES Tetris by a human player is of 44 cleared lines [5].

The remainder of this paper is organized as follows. In Section II we discuss related work, focusing on metrics/features previously developed/employed by agents to play Tetris. In Section III we provide an overview of the approach adopted to build playing agents, reviewing all the metrics employed and the PSO algorithm, which was used to tune metrics' weights. In Section IV we discuss the experimental setup of our approach. Section V presents the main results of our work and discussions. In Section VI we draw the main conclusions of the paper and discuss possible future research directions.

II. Related Work

In this section we discuss related work. We limit the discussion to works that employed an evaluation function composed of different metrics/features to grade board configurations and guide the decisions of the playing agent. In order to keep the discussion as brief as possible, the description of the actual metrics is performed later, on Section III, where only the metrics considered in our work are discussed in detail.

Several works have developed metrics/features that aim to grade[2] a given state of the Tetris board. By considering different aspects of the board (e.g., number of wells and height of each column) different metrics can be developed. Once a set of metrics is defined, it can be employed not only to grade the current board, but also to grade all possible placements of the current piece and possibly the next one. If the metrics are meaningful enough, one has only to tune their weights, that is, determine their individual influence in the final grade.

The work performed by Pierre Dellacherie is often cited as influential regarding the development of metrics for playing Tetris [6], [7]. Dellacherie proposed six new metrics and tuned their weights by trial and error, obtaining impressive results: on average his approach cleared 660,000 lines [7], [8]. Thiery and Scherrer [8] developed two new metrics, which were then combined with the ones employed by Dellacherie. In order to tune metric's weights the authors employed the cross-entropy method, obtaining even better results than previously observed originally by Dellacherie. The authors report that on simplified versions of the game their approach could clear, on average, 35,000,000 lines, within a margin of 20% per game.

Another influential work regarding metrics was presented Böhm *et al.* [9]. The authors introduce some new metrics (which are employed alongside existing ones) and use a Genetic Algorithm in order to tune their weights, achieving competitive results to the ones reported in the literature.

A good review of the metrics previously cited is provided by Algorta and Simsek [7]. Given that we adopt/evaluate a number of these metrics for playing Tetris with no rotation in the current work, we choose to detail them in the next section of the paper. It is important to note, however, that the aforementioned works consider, in their vast majority, a simplified version of the Tetris game and not the NES Tetris version of the game. Moreover, in all cases, piece rotations were allowed during gameplay, differently from our work.

It is worth recalling that in this paper we are mostly interested in the NES Tetris version of the game, given that: (i) it is the version of the game played in the CTWC (Classic Tetris World Championship), and (ii) due to its use in CTWC, a number of human players are attempting to break records considering the no rotation variation, in this very same version of the game. Although we are not aware of any works aiming to specifically develop metrics to play with no piece rotations in the NES Tetris version of the game (nor in other versions), recent developments have been made concerning agents that play NES Tetris in a "traditional" manner, as we describe next.

The previously discussed numbers of cleared lines are beyond human capabilities. In the case of NES Tetris, humans can play *comfortably*[3] up to level 29. Even though some outstanding players can go beyond level 29, these are rare events worth of register (which can be found in YouTube, see, for instance [10]). Indeed, the transition between levels 28 and 29 is often referred to as *Kill Screen* in the tournament.

Taking this into account, M. Birken [2], programmed and evaluated whether an agent could play NES Tetris and score as high as a human player within the limit of the Kill Screen. All games started at level 19 (similarly to CTWC) and were terminated either with a top out or by reaching level 29. The author employed Particle Swarm Optimization (PSO) [3] to tune the weights of a set of 17 metrics. As a result he obtained a score of 1,036,706±149,458, a solid one in comparison to those achieved at CTWC. Given the results observed by M. Birken with NES Tetris, we follow his approach in this work.

III. Materials and Methods

Our goal is to optimize the performance of an agent that plays NES Tetris with no piece rotations. We measure the performance of an agent by the number of lines it clears in a given game, selecting as best agent the one that, on average, clears the highest number of lines. This selection is carried out by considering a set of validation games, as we discuss later. The playing agent is based on an evaluation function that combines distinct board metrics linearly, by weighting their importance. At a given game state all possible piece placements, considering the current and next piece, are graded by the agent's evaluation function, that is, each possible board is graded by the agent's evaluation function. The board configuration yielding the best evaluation is selected, the pieces are placed/locked in the board accordingly, and the process is repeated until the game is finally over.

Different agents can be obtained either by modifying the set of metrics employed in the evaluation function or their corresponding weights. An overview of the approach we adopt in our experimental evaluation is provided in Fig 1.

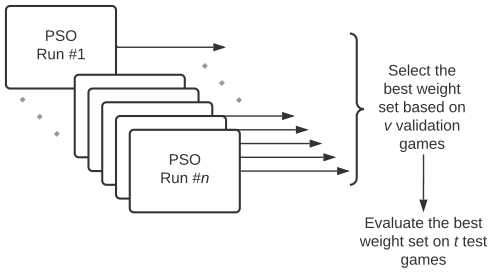


Fig. 1. Overview of the approach employed to train, validate and test agents.

An evaluation function is defined by a set of metrics, and their weights are optimized with the Particle Swarm

[2]In order to avoid confusion, we employ the term grade when referring to the quality of a given Tetris board, as evaluated by a set of metrics. The term score is used only in the context of the total of points awarded to a player.

[3]This applies mostly to CTWC contestants.

Optimization (PSO) algorithm [3], for which different runs are performed. At the end of each run, the resulting agent is evaluated on a set of v validation games. The agent that produces the best validation results is then tested on t test games, for which we report results. In the following we discuss the metrics considered in our evaluation and how PSO was employed to adjust their relative weight/importance.

A. Metrics

We considered a total of 19 metrics in our investigation. Below we provide a brief description of each one.

1) Total lines cleared: the number of cleared lines obtained after locking in place current and next piece.
2) Total lock height: the sum of the heights at which the current and next tetromino are locked in position.
3) Total well cells: Number of cells across all wells. A well is an empty cell surrounded by filled ones, except for its top. The board walls are treated as filled cells.
4) Total deep wells: Number of wells with 3 or more cells.
5) Total column holes: Number of empty cells with filled cells immediately above them (holes). By definition empty columns don't contain any holes.
6) Total weighted column holes: Weighted sum of holes in the board. Weights are assigned according to the row at which the hole appears, from top (1) to bottom (20).
7) Total column hole depths: The sum of the height difference between each hole and it's column top.
8) Min column hole depth: The smallest column hole depth. If there is no holes, its default value is 20 (board depth).
9) Max column hole depth: The largest column hole depth. If there is no holes, its default value is set to 0.
10) Total column transitions: The sum of transitions between filled/empty cell and empty/filled cell (by column). Transitions at the edges (top and bottom) are not accounted.
11) Total row transitions: The sum of transitions between filled/empty cell and empty/filled cell (by row). Board walls are regarded as filled. Empty rows are ignored.
12) Total column heights: The sum of the heights (distance between board floor and highest filled cell) of all columns. Empty columns have a height of 0.
13) Pile Height: The height of the highest column.
14) Column height spread: The height difference between the highest and the shortest columns of the board.
15) Total solid cells: The number of filled cells in the board.
16) Total weighted solid cells: The weighted sum of all filled cells. Filled cells are weighted by their height, from bottom (1) to top (20) of the board.
17) Column height variance: The sum of the absolute height differences between all adjacent columns of the board.
18) Removed lines: The number of cleared lines obtained after the lock of the last piece only.
19) Max well depth: The depth of the deepest well.

Metrics 1 through 17 were proposed/described by M. Birken [2]. Metrics 3, 5, 10, 11, 13, 14, 16, 18, 19 were introduced by Böhm *et al.* [11]. We developed three distinct playing agents by selecting three different subsets of the aforementioned metrics. The evaluation function (objective function) of each playing agent is built on the basis of a linear combination of the metrics, which are weighted by their importance (weights are determined by PSO in the training phase). Henceforth we refer to each one of these agents by its Metric Set, as follows: MS1 is built with the metrics from Birken; MS2 is built with the metrics from Böhm and; MS3 is built with the metrics 2, 3, 5, 10, 11, 12, 14, which we defined by empirical experimentation/observation.

B. Particle Swarm Optimization

In order to tune the weights of the metrics that compose an agent's evaluation function we adopt Particle Swarm Optimization (PSO), a method aimed to optimize continuous, non-linear functions [3]. It is inspired on the social behaviour of animals, such as birds. In PSO, each solution is expressed as a particle in the search space (initially, randomly positioned). In brief, in a given step of PSO, for each particle: its evaluation is obtained, considering its current position; the personal best solution is updated, alongside the global best solution, if needed; the distances of the particle to its own best and the global best solutions are computed; a new velocity of movement and new position are estimated considering its personal best and the global best positions. The procedure goes on until convergence is observed or a predetermined number of steps is reached. Given its non deterministic nature, several runs (with a different seed each) are usually performed.

The position of each particle is updated according to Equation (1), where $\mathbf{s}$ and $\mathbf{v}$ account for the particle's position and velocity at a given step i. Note that both $\mathbf{s}$ and $\mathbf{v}$ are d-dimensional vectors, where d is the number of problem variables that are being optimized, in our case, metric weights.

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}_i \tag{1}$$

Equation (2) is responsible for updating the velocity of each particle. The new velocity ($\mathbf{v}_{i+1}$) is a sum of three terms which take into account the particle's current velocity, a cognitive component and a social component. The current velocity ($\mathbf{v}_i$) is multiplied by an inertial coefficient (w_i), which aims to keep the particle moving in the current direction. The cognitive component moves the particle towards the personal best solution it found so far ($\mathbf{s}_i^p$), whereas the social component moves the particle towards the best global solution ($\mathbf{s}_i^g$) found until the current iteration of the procedure. The terms c_i^1 and c_i^2 are the cognitive and social coefficients, respectively, which determine how much exploration and exploitation is performed [12]. These are multiplied by two random values, r_i^1 and r_i^2, generated in the $[0, 1]$ interval at each update.

$$\mathbf{v}_{i+1} = w_i \mathbf{v}_i + c_i^1 r_i^1 \left(\mathbf{s}_i^p - \mathbf{s}_i\right) + c_i^2 r_i^2 \left(\mathbf{s}_i^g - \mathbf{s}_i\right) \tag{2}$$

Note that all coefficients in Equation (2) are iteration indexed (i). Following [13], the inertial coefficient and acceleration coefficients (cognitive and social components) are updated according to Equations (3) and (4), respectively. In Equation (3), w_{min} and w_{max} account for the minimum

and maximum desired inertial values. In Equation (4), the superscript indicates that the update applies to both c^1 and c^2 components, each of which has an initial ($c^{\{1,2\}}_{initial}$) and final value ($c^{\{1,2\}}_{final}$). In both equations i accounts for the current iteration and i_{max} for the maximum number of iterations.

$$w_i = w_{min} + (w_{max} - w_{min}) \frac{i_{max} - i}{i_{max}} \quad (3)$$

$$c_i^{\{1,2\}} = \left(c^{\{1,2\}}_{final} - c^{\{1,2\}}_{initial}\right) \frac{i}{i_{max}} + c^{\{1,2\}}_{initial} \quad (4)$$

After update, the absolute value of each particle's velocity is limited (in all dimensions) to a maximum value (v_{max}), preventing particles from "flying" out of the solution space [13].

IV. Experimental Setup

We run experiments to compare how the three sets of metrics we defined in Section III (MS1, MS2, and MS3) perform when playing NES Tetris with no rotations. In order to run experiments for each set of metrics, we considered the hyperparameters defined by Table I. It is important to highlight that the parameter *number of training games* defines how many games are played by a particle at each step of the optimization procedure to yield its fitness (evaluation), which is given by the mean number of cleared lines across the training games. It is also important to note that experiments were performed in such a way that the training, validation and test games played during the evaluation of each one of metric sets are the very same, allowing for a fair comparison of their final results.

TABLE I
Parameters adopted in the experimental evaluation

Parameter	Value
Number of particles	30
Maximum number of iterations	200
Weights search interval	[-1,1]
Particles' maximum velocity	0.01
Number of PSO runs	20
Number of training games	50
Number of validation games	500
Number of test games	1,000,000
Inertia - max / min	0.9 / 0.4
cognitive coefficient range - $c^1_{initial}$ / c^1_{final}	1.5 / 0.5
social coefficient range - $c^2_{initial}$ / c^2_{final}	1.0 / 4.0

Last but not least, we have to define how Tetris was actually played during all experiments. In order to play each game we employed an implementation that is able to simulate all aspects of NES Tetris, including possible movements and most importantly its RNG (Random Number Generator), which defines the sequence of pieces spawn at each game. The implementation we employ is made available by M. Birken [2]. It is important to note that this implementation is way faster than running a NES Tetris emulator, allowing for a less computationally intensive experimental evaluation. The function that performs the search for the best lock position of each piece considering the current evaluation function and most of the metrics were also implemented and made available by [2]. We used such code in our experiments adding for the metrics that were not initially available and our own implementation of the optimization procedure, that is, the PSO method.

V. Results

In this section we present the results obtained during the evaluation of the best agent, considering each one of the three metric sets previously defined. The evaluation of each agent was performed on the very same set of 1,000,000 test games. It is important to note that although we limited PSO to a maximum of 200 iterations/steps, in the vast majority of its runs convergence was observed. Having made such considerations, Table II provides some statistics for each metric set, concerning the test games. The difference within the three metric sets results is quite small. Overall, MS3 obtained the best results, followed by MS2, and MS1. Interestingly, the smaller the number of metrics in a feature set, the better its results. This may arise due to the fact that a smaller metric set translates into a reduced search space, allowing for better exploration by PSO. Regarding the slightly better results observed with MS3, we highlight that this particular metric set was intuitively selected/filtered from the remaining ones, based on prior experimentation. In this sense, even though none of the metrics that compose it was specifically designed to play NES Tetris with no piece rotations, we carefully selected metrics that seemed to make sense from previous experiments. We applied Student's t-test to the results from Table II and observed no significant differences among them.

TABLE II
Summary of the results considering the 1,000,000 test games

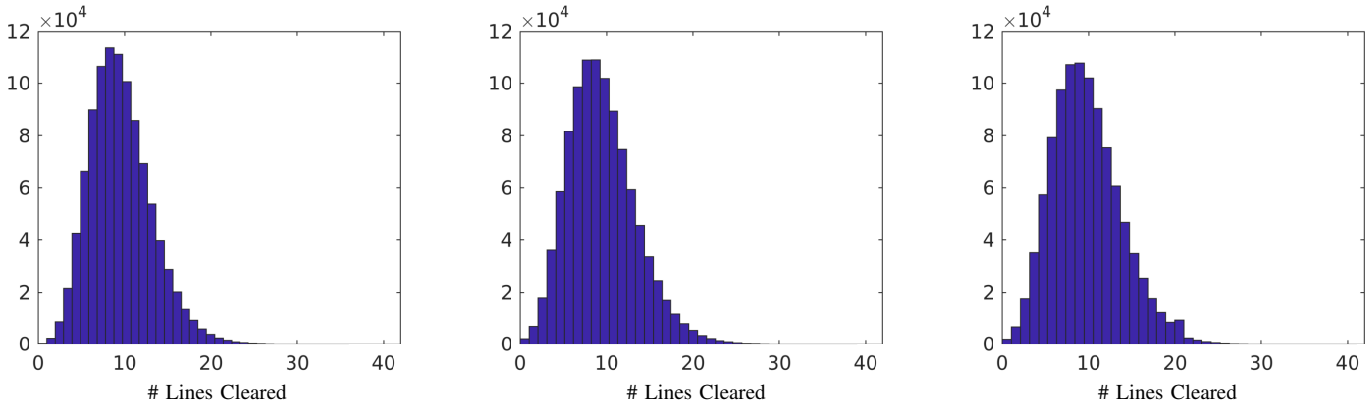
Metric Set	Mean	Std.	Median	Max
MS1	9.29	3.67	9	36
MS2	9.70	3.81	9	39
MS3	9.79	3.84	9	41

In Fig. 2 we depict histograms of the 1,000,000 game results for each one of the metric sets, complementing the results from the previous table. The distributions of game results (#lines cleared) for the three metric sets is quite similar. In order to highlight the differences among the metric sets, in Fig 3, we plot in the y axis the probability of observing a game with a total of cleared lines equal or higher than that from the x axis. This figure also allows for a better examination of the overall advantage of MS3 w.r.t. the remaining metric sets.

In order to provide some insight on how the best agent (built with MS3) is actually playing, we allowed it to play a NES Tetris Game ROM with the Nintaco emulator [14], recording several games (all starting on Level 19). The best game (41 line clears) can be watched here: https://youtu.be/3fjMeww7GpY. Another, quite long video with 2 hours, showcasing the Top 100 scoring games (w.r.t. lines cleared) played by the same agent can be watched here: https://youtu.be/yrJgQ4kCc68.

VI. Conclusions

We presented an empirical evaluation of PSO for playing NES Tetris with no rotations. For that we considered three



(a) MS1 (b) MS2 (c) MS3

Fig. 2. Histograms of the results for the three metric sets considering the same 1,000,000 test games.

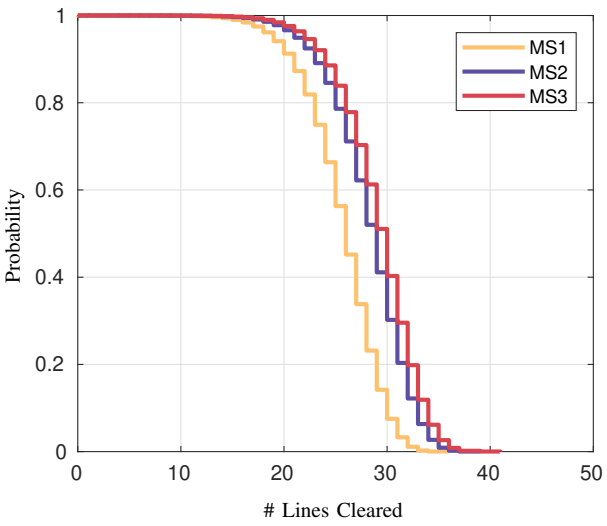


Fig. 3. Probability (y axis) of observing a game with a total number of cleared lines equal or higher than that from the x axis, considering each metric set.

different metric sets and observed that the smallest one, which consists of a subset of existing metrics presents the overall best results. This particular set obtained a record of 41 lines cleared. These results are quite impressive, if we consider that: (i) none of the metrics evaluated in this work was developed with the no rotation version of the game in mind; and (ii) until recently, human players were unable to clear more than 30 lines in the no rotation version of the game. Indeed, video registers of 30 line clears or more in this variation only appeared after August, 2020, with a current registered record of 44 lines [5].

It is important to highlight that this is a first exploration on the topic. As future work we plan to develop custom metrics for playing the no rotation version of the game (to the best of our knowledge no work has done that to this date). We believe that such an approach may succeed given that: (i) PSO was already successfully employed to play NES Tetris in its traditional version with custom made metrics; and (ii) Artificial Neural Networks (ANNs) were recently employed to play the no rotation version of the game, clearing at most 55 lines [15], showing that there is still room for improvement.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive feedback.

References

[1] Wikipedia Contributors, "Tetris Effect." Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Tetris_Effect (Accessed Jul. 10, 2021).
[2] M. Birken, "Applying artificial intelligence to handicapping NES Tetris." Meat Fighter, Aug. 17, 2019. https://meatfighter.com/handicappedtetris/ (Accessed Jul. 10, 2021).
[3] J. Kennedy and R. Eberhart, "Particle swarm optimization." in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.
[4] D. Macdonald, "World Record Progression: NES Tetris No Piece Rotation." (Sep. 24, 2020), Accessed on Jul. 10, 2021. [Online Video]. Available: https://youtu.be/yHqIdICC5wQ.
[5] Z. Bereczki, "NES Tetris no rotation lvl0 start 44 lines [World Record]." (Jan. 07, 2021), Accessed on Jul. 10, 2021. [Online Video]. Available: https://youtu.be/4ua79-J4D5E.
[6] F. Colin, "Tetris." Colin Fahey. https://colinfahey.com/tetris/tetris.html (Acessed Jul. 10, 2021).
[7] S. Algorta and Ö. Simsek, "The game of tetris in machine learning," pp. 1–7, 2019. [Online]. Available: http://arxiv.org/abs/1905.01652.
[8] C. Thiery and B. Scherrer, "Improvements on learning tetris with cross entropy," *International Computer Games Association Journal*, vol. 32, pp. 23 – 33, 2009.
[9] N. Böhm, G. Kókai, and S. Mandl, "An evolutionary approach to tetris," in *The Sixth Metaheuristics International Conference*, 2005, pp. 1–6.
[10] J. Saelee, "First Ever Level 34 in NES Tetris." (Feb. 16, 2020), Accessed: Jul. 10, 2021. [Online Video]. Available: https://youtu.be/rWMUYBinriw.
[11] G. Da Col and E. C. Teppan, "Heuristic Search for Tetris: A Case Study." in *Intelligent Computing*, K. Arai, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2019, pp. 410–423.
[12] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, pp. 33–54, 2007.
[13] H. ling Chen, B. Yang, S. jing Wang, G. Wang, D. you Liu, H. zhong Li, and W. bin Liu, "Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy," *Applied Mathematics and Computation*, vol. 239, pp. 180–197, 2014.
[14] *Nintaco*. (2020). Michael Birken. Acessed: Jul. 10, 2021. [Online]. Available: https://nintaco.com/.
[15] A. Wu, "No-Rotation Tetris AI." 2020, Github: Adrien Wu, 2020. https://github.com/adrien1018/noro-tetris-ai. (Accessed Jul. 10, 2021).