Inform : Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi
Vol.7 No.1 January 2022, P-ISSN : 2502-3470, E-ISSN : 2581-0367

MVPA and GA Comparison for State Space Optimization at Classic Tetris Game Agent Problem

Hendrawan Armanto[1], Ronal Dwi Putra[2], C. Pickerling[3]

[1, 2, 3]*Computer Science Department, Institut Sains dan Teknologi Terpadu Surabaya, Indonesia*

[1]hendrawan@stts.edu (*)

[2]ronaldwiputra88@gmail.com

[3]pickerling@stts.edu

Received: 2021-12-17; Accepted: 2022-01-21; Published: 2022-01-24

Abstract— Tetris is one of those games that looks simple and easy to play. Although it seems simple, this game requires strategy and continuous practice to get the best score. This is also what makes Tetris often used as research material, especially research in artificial intelligence. These various studies have been carried out. Starting from applying state-space to reinforcement learning, one of the biggest obstacles of these studies is time. It takes a long to train artificial intelligence to play like a Tetris game expert. Seeing this, in this study, apply the Genetic Algorithms (GA) and the most valuable player (MVPA) algorithm to optimize state-space training so that artificial intelligence (agents) can play like an expert. The optimization means in this research is to find the best weight in the state space with the minimum possible training time to play Tetris with the highest possible value. The experiment results show that GAs and MVPA are very effective in optimizing the state space in the Tetris game. The MVPA algorithm is also faster in finding solutions. The resulting state space weight can also get a higher value than the GA (MVPA value is 249 million, while the GA value is 68 million).

Keywords— Tetris, Artificial Intelligence, GA, Most Valuable Player Algorithm.

I. INTRODUCTION

Tetris is one of the most popular games (not infrequently addictive [1], [2]) with relatively simple game rules. Players are asked to rotate or slide the blocks and put them in the right position to form a horizontal line without gaps. Although it seems simple, this game requires the right strategy and continuous practice to play well and get the highest possible score [3]. Apart from the player side. The difficulty level of this game from the number of possibilities that can occur. According to initial estimates, there are approximately 1060 possibilities. This game can also be categorized into NP-Complete problems regardless of the size of the rows and columns have given [4]. This makes Tetris an interesting game to study, especially in artificial intelligence. Like the way humans play, the strategy in this game is a challenge for artificial intelligence to play stably and continuously. There is still no artificial intelligence that can play Tetris without losing.

II. LITERATURE REVIEW

In this study, further learning is needed regarding the Tetris game itself and previous studies that have been carried out so that this research can provide the right and good solution.

A. Tetris Game [5], [6]

In 1984, Alexey Pajitnov inspired the classic game Roman Pentomino Puzzle to develop the Tetris game. There are two basic and important components in this game: blocks or game pieces called tetrominoes and game boards of various sizes. The standard size in this game is 20 (t) x10 (l) cells, but there are also many game boards measuring 16 (t) x 10 (l) cells or 24 (t) x 10 (l) cells.

The tetromino is appropriately positioned and forms a horizontal row without gaps, and the game will automatically drop it slowly, allowing the user to rotate or shift it as needed. Each tetromino will be packaged with a game board that will have a variety of orientations and colors based on the shape of the tetromino in question. There are seven basic tetromino forms used in this study, namely I, O, T, S, Z, J, and L (Figure 1). In contrast, the form of the tetromino variation was not part of this study because the focus of the research remained on classic Tetris.

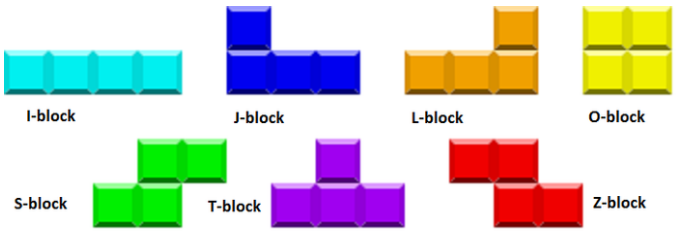


Figure 1. Tetromino Shape

In the Tetris game, the tetromino can be rotated by 90 degrees, depending on the available space on the game board. If there is not enough space, the tetromino cannot be rotated, so it will continue to fall. Players can only slide the tetromino left or right.

Tetris does not have a win condition like the game in general. The game will continue without stopping as long as no tetrominos are piled up until they reach the upper limit of the game board. The game's ultimate goal is to clear as many lines as possible and get as many points as possible. In some versions of Tetris, if you can clear rows with one tetromino, it can produce combos, and the value obtained will be very high.

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

The human desire to beat the value of friends or foes makes Tetris still alive today. It has even become a game played in the international arena.

B. Last Research

Research on artificial intelligence for the Tetris game has been done for a long time. However, in this study, only research over the year 2000 or research in the last five years was used in detail. This is because the longer the research, the more lagging the method used in the study. Since 2000, various artificial intelligence algorithms have been studied, including:

1) Approximate Dynamic Programming [7]: This research focuses on finding the policy of the Tetris game using approximate dynamic programming. The trials carried out here included two board models, namely 10x10 and 10x20 and on average managed to get a score of 51,000,000 for each game.

2) Reinforcement Learning [8]: The same is true of previous research. Reinforcement learning also aims to maximize rewards and find the best strategy in playing Tetris. The weakness that arises from this research is the length of time it takes to find a strategy to play Tetris well. Researchers state that it takes approximately 40 hours in training.

3) Optimization of State Space Using Evolutionary Algorithm: In contrast to the two previous studies. This research focuses on optimizing state space that was previously used but is now abandoned because of the difficulty level in searching heuristically. Researchers use evolutionary algorithms [9][10], such as GA [11][12], Covariance Matrix Adaptation Evolution Strategy [13], and Harmony Search Algorithm [14].

This research focuses on state-space optimization, but the evolutionary algorithm used is the Most Valuable Player Algorithm (MVPA) and GA to compare MVPA optimization quality. GA [15] is used as a comparison in this study because this algorithm is a basic evolutionary algorithm and has been successfully applied and developed for various types of research [16] [17].

C. Most Valuable Player Algorithm (MVPA) [18]

MVPA is one of the evolutionary algorithms published in 2020. A collection of candidate solutions (or population in GA terms) is called a team. In contrast, the candidate solutions (or individuals in the GA) are referred to as players. Each player will have his skill (or cell in the GA) to determine how good a player is. A description of a player's skills can be seen in Figure 2.

The fitness value or how good the MVPA algorithm generates a candidate solution is called Efficiency or Rating. Similar to the fitness value of other evolutionary algorithms, in MVPA, the higher the efficiency value, the better and is a good solution. In addition to these essential terms, there are still some terms that must be known to make it easier to understand the MVPA algorithm, including:

- Franchise Player: The best player in a team.
- Most Valuable Player: The best player overall or in sports is the best player in that season.
- League: Matches are held to get points to determine who is the best team.
- Championship: A competition held to get the best team and players.
- Fixture: The time of the match is generally determined by the date and place.

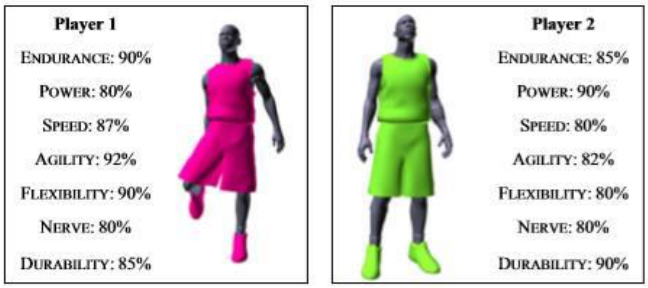


Figure 2. Player Skill Illustration

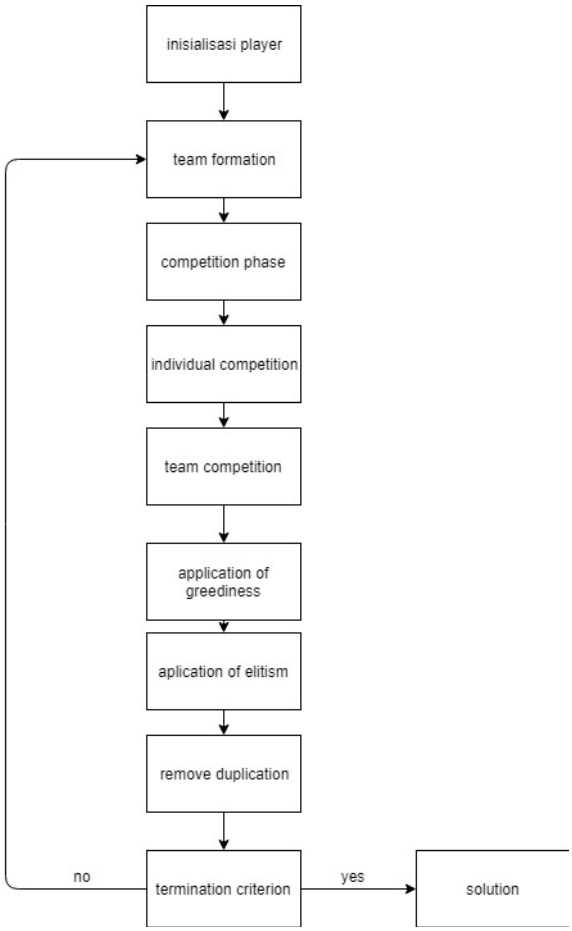


Figure 3. MVPA Architecture.

The architecture of the MVPA algorithm has several stages that must be passed, as shown in Figure 3, namely:

1) Initialization: Initialization is the stage where

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

determine the number of existing players and the ability of each player (generate candidate solutions).

2) Team Formation: After all the players are formed, at this stage. Determine the number of teams and divide the players evenly throughout the team. For example, the number of players is ten, and the number of teams is 3. The players will be divided equally among the entire team into 4-3-3.

3) Individual Competition: Each player in a team will try to develop their skills based on the existing franchise players and MVP.

4) Team Competition: At this stage, two different teams will be chosen randomly and competed with one another, where the match results may only win or lose (there can be no draw in this match). The mechanism to determine which team wins will use probability.

5) Application of Greediness: After going through various competitions, this stage compares the player's skills before and after the competition. If it is better, it will be used for the next iteration.

6) Application of Elitism: 1/3 of all players with the worst skills will be replaced by 1/3 of the players with the best skills.

7) Remove Duplicates [12]: After going through the elitism process, this stage will see individuals who are twins and next to each other. If there is one of these individuals will be updated skills.

After going through these various stages, a determination is made whether the stopping conditions have been met. The stopping conditions are whether the data has converged or the number of iterations has been met. When it has been met, then the iteration is stopped, and the best player is the resulting solution. But if it has not been fulfilled, it will return to the team formation stage.

Although MVPA is a new evolutionary algorithm (published in 2020), this algorithm has been used several times to solve other special problems. For example, problems such as Partially Shaded PV Generation System [19], Energy Control Center for Energy System Security [20], and Optimal Antenna Network Positioning [21]. This is one of our bases in deciding to use MVPA in this study.

III. Research Methodology

This research has several critical stages of running well. These stages include determining the scope of the Tetris problem, the representation of Tetris in MVPA and GAs, how to calculate the efficiency or fitness value of a candidate solution, and the AI architecture.

A. Tetris Problem Scope

This study used 10×20 Tetris cells and seven basic tetromino shapes. This aims to prove the comparison of the performance of the two algorithms. There are also restrictions on spawns, including 100 spawns, 300 spawns, 500 spawns, and 1,000 spawns. Determine the number of line clears obtained in a Tetris game with the spawn limit. Table I is the ratio of spawn pieces and lines clear where the spawned piece is the maximum number of tetrominoes that drop in a game, while lines clear is the maximum number of lines that the player can complete. The ratio between spawn pieces and lines clear is certainly in the range of 2.5.

TABLE I
MAXIMUM SPAWN PIECE AND LINES CLEAR

Spawn Piece	Lines Clear	Ratio
100	39	2.564
300	119	2.521
500	199	2.512
1000	399	2.506

Even so, at the end of this study to release these limitations. Assuming that if the artificial intelligence manages to resolve the four limitations well, then the artificial intelligence will also play Tetris without the spawn piece limitation.

B. Tetris Representation in Evolutionary Algorithm

For a problem to be solved by an evolutionary algorithm, anything including MVPA and GAs requires an exact representation. In this study, the movement of the tetromino in Tetris may be represented by state space, which can be either one level or two levels deep, with each state representing the tetromino's location and rotation. If the state space has one state-level, then the state space only considers where the tetromino's laying position and rotation conditions are currently going down. Still, if the state space has two state levels, then the state space also considers the next tetromino that will drop (this can be done considering in the game of Tetris can see the next tetromino. Figure 4 is an example of 1 level state space for one tetrominoes.

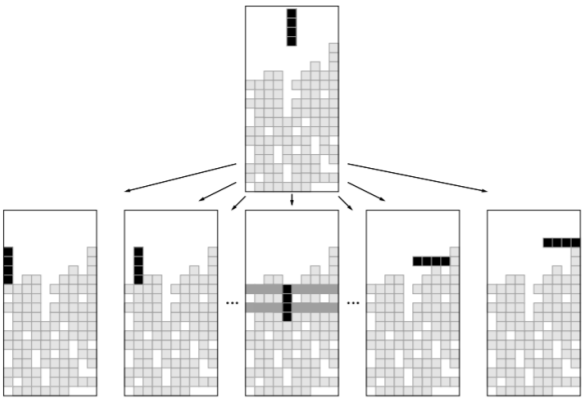


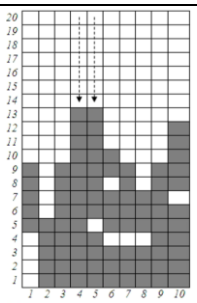
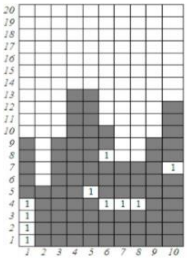
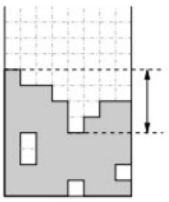
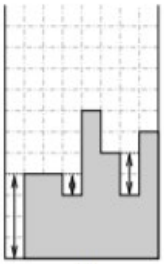
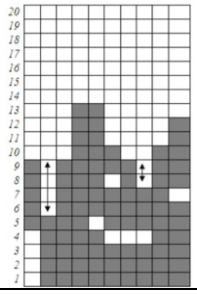
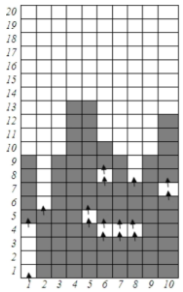
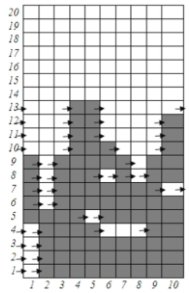
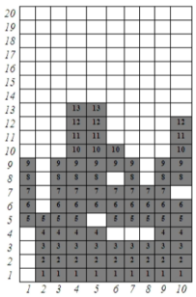
Figure 4. Tetris State Space (1 level)

This study will apply two levels of state-space, assuming that artificial intelligence will be better if it can take several future steps compared to only taking into account current steps. So that artificial intelligence can determine what state should be taken at this time. For each goal state that is formed, a feature calculation is carried out related to the condition of the Tetris board when the state is executed. Table II contains ten features used in this study. These ten features are used to measure how well the board condition after a state is executed. The state path that produces the best value will be taken as the

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

current step. This is done continuously every time the tetromino is about to drop until the game ends.

TABLE II
LIST FEATURE OF TETRIS BOARD [15]

No	Feature Name	Illustration	Description
1	Pile Height		The highest block in all row
2	Holes		Count of cells that empty and closed by another block
3	Lines Clear	-	Count of lines that successfully cleared
4	Altitude Difference		The height difference between the highest and the lowest pile
5	Maximum Wells Depth		The deepest wells in the board
6	Sum of Wells		Sum of well in the board
7	Column Transition		Count of vertical transition from filled cell to an empty cell
8	Row Transition		Count of horizontal transition from filled cell to an empty cell
9	Block Weight		Sum of block weight for all filled cells in the board. The higher the block will get, the bigger weight
10	Smoothness	-	Amount of the absolute difference between every column

Not all features in table II have the same importance. Some features may be more important than others. A problem arises where we cannot know the importance of each feature. If we try one by one, it will take a very long time. This is where the role of MVPA and GAs is to find the weight or importance of each feature so that artificial intelligence can choose the correct state.

1.53	0.21	0.88	2.17	0.12	0.55	0.18	1.78	1.55	2.85

Figure 5. MVPA and GA Representation

Figure 5 is an example of MPVA representation and GA. The length of the player or individual is 10 (according to the number of features), where each skill or cell represents the weight of each feature. The higher the value of a particular cell, the more important the feature represented by that cell. The first cell represents the first feature, i.e., pile height and so on.

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

C. Calculate Efficiency or Fitness Value

A candidate solution in the evolutionary algorithm requires a value of goodness (efficiency in MVPA and fitness in the GA) to determine which candidate solution is feasible to use as the final solution. In this study, we use the same method of calculating fitness values as previous studies [9][10][11][12][13][14], namely the number of lines that have been completed in a game. Adjust to the number of spawn pieces (Table I) used, the maximum value of lines clear is formulated as in equation (1). This maximum value aims to determine whether the fitness value formed from a candidate solution is good or not. For example, 100 spawn pieces are used. If a candidate solution obtains a fitness value of 39, then the potential solution is very good, considering the maximum lines clear for 100 spawn pieces is 39. But if we use 300 spawn pieces and a candidate solution manages to obtain a fitness value of 39, then the potential solution can be said that it is not good considering that 300 spawn pieces have maximum lines clear of 119.

$$MaximumLinesClear = \frac{SpawnPiece}{2.5} - 1 \quad (1)$$

Considering that the evolutionary algorithm is based on random, each candidate solution will be tried five times to provide accurate and reliable results. The fitness value formed is the average lines clear of the five experiments (equation (2)).

$$Efficiency / Fitness = \frac{\sum_{i=1}^{5} LinesClear(i)}{5} \quad (2)$$

D. Artificial Intelligence Architecture

The architecture used in this research can be seen in figure 6, where the evolutionary algorithm, in this case, is MVPA and the GA performs a continuous iteration process to produce a collection of candidate solutions. Each candidate solution is sent to the game simulator to be tried and calculated for the value of goodness (efficiency or fitness). The game simulator will play the game up to the specified spawn piece limit and return the line's clear value. This is done continuously until the MVPA stops and the GA is met. If the stopping conditions have been met, the algorithm will return a value to the simulator game to be used as a weight for the existing artificial intelligence movements.

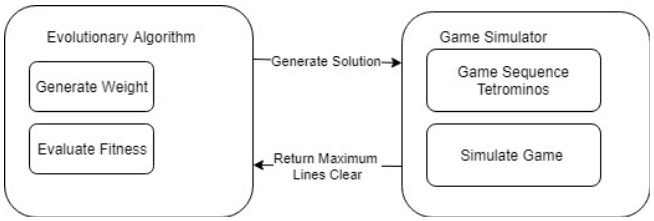


Figure 6. AI Architecture

IV. Result and Discussion

The experiment in this study was carried out in 3 stages. The first stage is a hyper-parameter experiment either on MVPA or on the GA. The second experiment is MVPA against a GA with the best parameters for a limited number of spawns. Finally, the third experiment is MVPA against a GA without a spawn piece limit. Each experiment was carried out five times, with the results obtained being the average of the five trials. This is done again because the evolutionary algorithm is based on random values.

A. Hyper Parameter Testing (MPVA)

In MPVA, a parameter must be tried, and the most appropriate value is sought, namely the combination of the number of players and the number of teams. This parameter is important because it can determine whether the algorithm can work well or not. As for the iteration parameters, no experiments will be carried out because the stopping condition of this research is not based on maximum iterations but whether the candidate solutions have converged or have found the highest clear lines based on the specified number of spawns pieces.

Figure 7 (spawn piece 1000) combines the number of players and teams that we tried. From this, it can be seen that we only tried a maximum of 15 players. This is done by considering the number of skills possessed is only 10, so it does not require a lot of combinations. In addition to the combination, the time aspect is also a consideration. A prospective solution takes a long time to calculate efficiency, so the more players there are, the longer it will take. Fifteen players are the maximum number when compared to the time it takes. Based on figure 7, in this study, we will apply ten players and two teams; choose this parameter because it can still produce perfect clear lines, but the time required is less due to the smaller number of players.

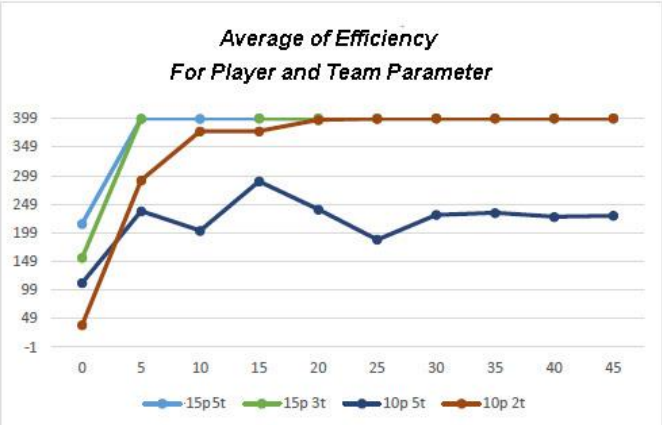


Figure 7. Player and Team Combination Testing

B. Hyper Parameter Testing (GA)

In the GA, we performed four hyperparameter experiments. Two are experimental methods for mutation and crossover,

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

and two are experiments with mutation rate parameters and the number of individuals in a population.

Figure 8 (spawn piece 1000) is an experiment of the mutation operator method in the GA. There are two methods we tried, including random resetting and bitflip. It can be seen in the figure that both methods are equally successful in obtaining perfect lines clear values, but the bitflip method is faster in terms of time. Although the bitflip method looks better, it is easier to get stuck in the local optima in this experiment. So for the mutation method, this study uses the random resetting method.

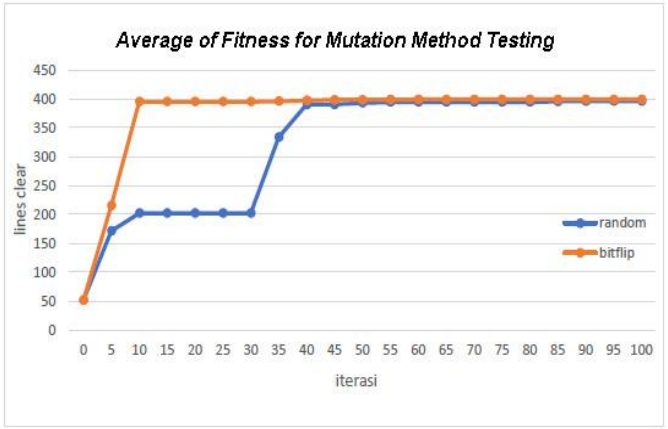


Figure 8. Mutation Method Testing

Figure 9 (spawn piece 1000) is an experiment on the crossover operator method in the GA. Two methods have been tried similar to the mutation operator, including one-point crossover and two-point crossover. From the figure, we can see that the two methods form a similar pattern. So we can conclude that there is no impact on selecting the crossover method, and in this study, we apply a two-point crossover.

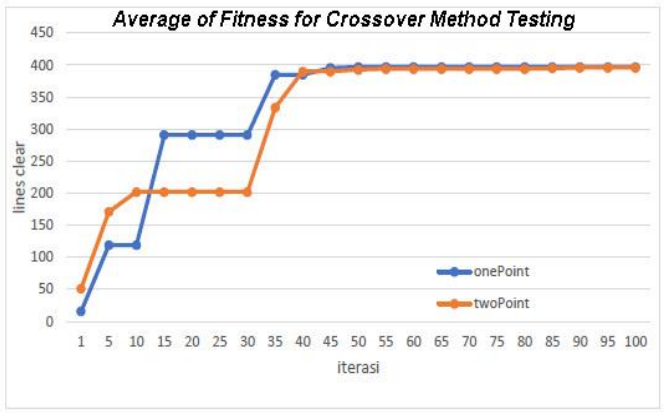


Figure 9. Crossover Method Testing

Figure 10 (spawn piece 1000) is an experiment on the mutation rate parameter. The higher the mutation rate, the more often the algorithm performs the mutation method, but it becomes less good at finding solutions if it is too high. Therefore, in this experiment, we limit the maximum to 20%. Based on the experimental results, it can be seen that a mutation rate above 10% helps speed up the search for solutions, so in this study, we will apply a mutation rate of 20%.

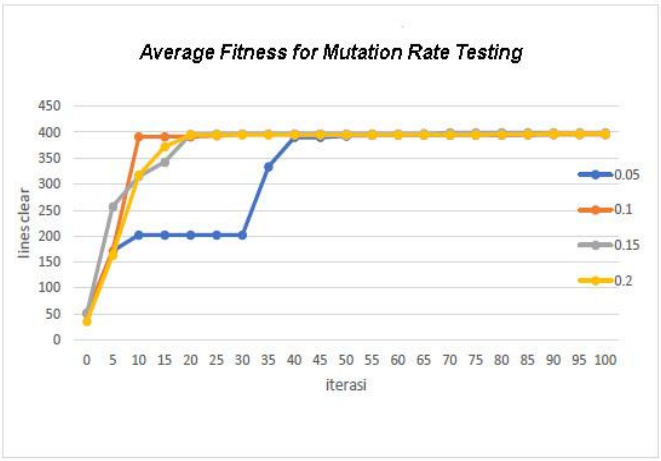


Figure 10. Mutation Rate Testing

Figure 11 (spawn piece 1000) is an experiment on the number of individuals parameter. Each problem has its own best number of individuals, and in the game of Tetris, the best score is ten based on this trial. The number of individuals five can give a perfect score, but it takes longer. This can be seen at the end of the graph, which shows signs of increase. While the increase for the number of individuals 15 can be very small or even non-existent. Whereas in terms of the time it takes, a larger number of individuals will take longer as well. Therefore, in this study, we will use ten individuals in a population.

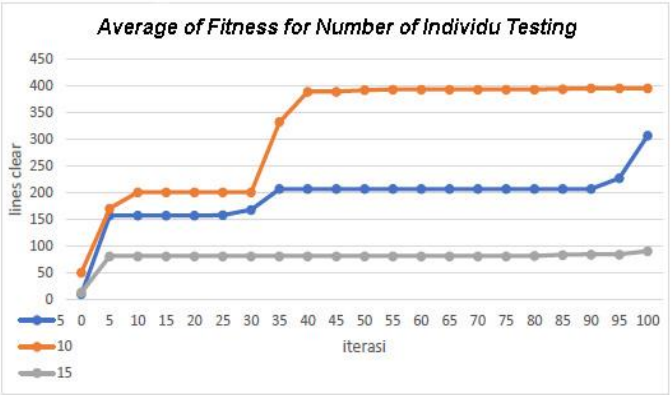


Figure 11. Number of Individu Testing

C. Limited Spawn Piece (MVPA vs. GA)

The Tetris game's scope we determined before. This experiment aims to examine the capabilities of the two methods when dealing with a limited number of spawn pieces.

Based on figure 12, we can conclude that both algorithms complete successfully in a limited problem (spawn piece), but the time required by MVPA is faster than the GA.

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

Figure 12. Limited Spawn Piece

D. Unlimited Spawn Piece (MVPA vs. GA)

Based on previous experiments, we assume that if MPVA can finish faster, then when the spawn piece limit is released, it also has a higher probability of success. Therefore, we carried out the final experiment in this study, namely MVPA vs. GA, without any limitations, using the best weights for each algorithm we found in previous experiments (table III). Each algorithm decides the importance of the 10 features used based on the table. The MVPA dictates that to play well requires clear lines, maximum wells depth, and good smoothness, while other features are of little importance. Meanwhile, according to the GA, the essential features are altitude difference, lines clear, and maximum wells depth.

TABEL III
THE BEST WEIGHT IN THIS RESEARCH

Weight for	MVPA	GA
Altitude Difference	-0.19599141394824904	0.6698807294925779
Lines Clear	0.2594041453984691	0.6841387500534917
Holes	-1	-0.37394472105813836
Sum Wells	-1	-0.6293119636901148
Pile Height	-0.2670627611891697	-0.7599808202133249
Coloumns	-0.9956868057081331	-0.9028331914982242
Transitions		
Row Transitions	-0.7084696281857331	-0.468797955591985
Block Weight	-0.37561818894044885	-0.09916144624084033
Maximum Wells	0.14278778535059133	0.10827643231026096
Depth		
Smoothness	0.44768976880987615	-0.10490860568298466

After going through 10x trials, MVPA beat the GA by an absolute 10:0. This proves that the MVPA feature selection is better than the GA. However, the GA also selected the two features selected by the MVPA (lines clear and maximum wells depth).

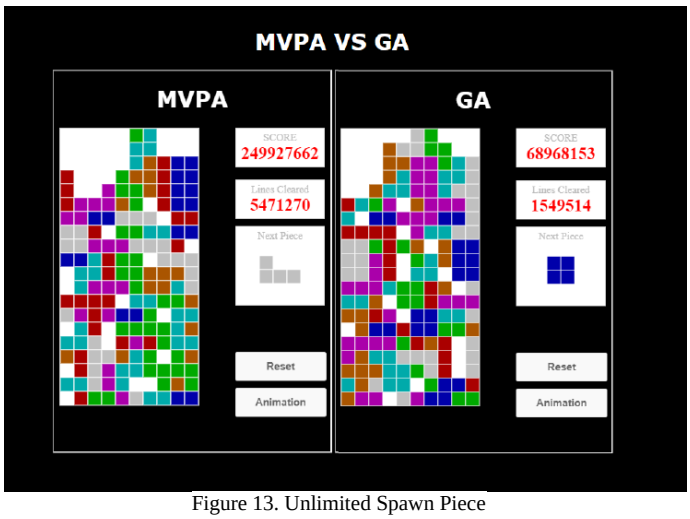


Figure 13. Unlimited Spawn Piece

Figure 13 is an example of one of the results of the MVPA vs. GA where the score calculation in this Tetris game is as follows:

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457

- Clear 1 Lines: Score 40
- Clear 2 Lines: Score 100
- Clear 3 Lines: Scored 300
- Clear 4 Lines: Scored 1,200

V. CONCLUSION

Evolutionary MVPA and GA algorithms work effectively in optimizing the state space of the Tetris game. Both trials have proven this using a limited spawn piece or a trial without a spawn piece limit. MPVA optimized the state space better than the GA in the Tetris game, which was proven during trials without a spawn piece limit. MPVA managed to reach a value of 249 million while the GA only reached a value of 68 million (Figure 13). In finding the optimal solution in the Tetris game, the MVPA algorithm managed to find a solution faster than the GA. This can be seen in the MVPA vs. GA trial, which has a limited number of spawn pieces (100, 300, 500, or 1000 spawn pieces). Determination of feature weights depends on how to play from Artificial Intelligence. Still, whether using MVPA or GAs, both algorithms agree that the lines clear and maximum wells depth are two essential features to play Tetris well.

REFERENCE

[1] D. Ackerman, *The Tetris Effect: The Cold War Battle for the World's Most Addictive Game*. Oneworld Publications, 2016.
[2] W. Books, *Summary, and Analysis of The Tetris Effect: The Game that Hypnotized the World: Based on the Book by Dan Ackerman*. Worth Books, 2017.
[3] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell, "Tetris is Hard, Even to Approximate," Oct. 2002.
[4] S. Asif *et al.*, "Tetris are NP-hard even with $O(1)$ rows or columns," Sep. 2020.
[5] G. Moore, *Tetris Puzzles*. Portable Press, 2019.
[6] B. Brown, *Tetris: The Games People Play*. First Second, 2016.
[7] V. Gabillon, M. Ghavamzadeh, and B. Scherrer, "Approximate Dynamic Programming Finally Perform Well in the Game of Tetris," in *NIPS*, 2013, pp. 1754–1762.
[8] G. Hendriks, "The effect of state representation in reinforcement learning applied to Tetris," University of Groningen, 2020.
[9] N. Bohm, G. Kokai, and S. Mandl, "MIC2005: The Sixth Metaheuristics International Conference - An Evolutionary Approach to Tetris," 2005.
[10] A. Boumaza, "On the evolution of artificial Tetris players," in *The IEEE Symposium on Computational Intelligence and Games*, Sep. 2009, pp. 387–393, [Online]. Available: https://hal.archives-ouvertes.fr/hal-00397045.
[11] S. Phon-Amnuaisuk, "Evolving and Discovering Tetris Gameplay Strategies," *Procedia Comput. Sci.*, vol. 60, pp. 458–467, 2015, doi: 10.1016/j.procs.2015.08.167.
[12] S. Algorta and Ö. Şimşek, "The Game of Tetris in Machine Learning," May 2019.
[13] A. Boumaza, "How to design good Tetris players," 2013. [Online]. Available: https://hal.inria.fr/hal-00926213.
[14] Ii Romero, M. Romero, L. Tomes, J. P. Yusiong, and T. Yusiong, "Tetris Agent Optimization Using Harmony Search Algorithm," *Int. J. Comput. Sci. Issues*, vol. 8, Nov. 2011.
[15] K. F. Man, K. S. Tang, and S. Kwong, "GAs: concepts and applications [in engineering design]," *IEEE Trans. Ind. Electron.*, vol. 43, no. 5, pp. 519–534, 1996, doi: 10.1109/41.538609.
[16] R. R. L. Haldurai T. Madhubala, "A Study on GA and its Applications," *Int. J. Comput. Sci. Eng.*, vol. 4, no. 10, pp. 139–143, Nov. 2016, [Online]. Available: https://www.ijcseonline.org/full_paper_view.php?paper_id=1092.
[17] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new GA for solving optimization problems," *Eng. Appl. Artif. Intell.*, vol. 27, pp. 57–69, Jan. 2014, doi: 10.1016/j.engappai.2013.09.013.
[18] H. R. E. H. Bouchekara, "Most Valuable Player Algorithm: a novel optimization algorithm inspired from sport," *Oper. Res.*, vol. 20, no. 1, pp. 139–195, Mar. 2020, doi: 10.1007/s12351-017-0320-y.
[19] I. Pervez, I. Shams, S. Mekhilef, A. Sarwar, M. Tariq, and B. Alamri, "Most Valuable Player Algorithm based Maximum Power Point Tracking for a Partially Shaded PV Generation System," *IEEE Trans. Sustain. Energy*, vol. 12, no. 4, pp. 1876–1890, 2021, doi: 10.1109/TSTE.2021.3069262.
[20] S. Shanmugapriya and D. Maharajan, "Most Valuable Player Algorithm Based State Estimation for Energy Systems," *Distrib. Gener. Altern. Energy J.*, Apr. 2021, doi: 10.13052/dgaej2156-3306.3543.
[21] H. R. E.-H. Bouchekara, A. Orlandi, M. Al-Qdah, and F. de Paulis, "Notice of Retraction: Most Valuable Player Algorithm for Circular Antenna Arrays Optimization to Maximum Sidelobe Levels Reduction," *IEEE Trans. Electromagn. Compatibility.*, vol. 60, no. 6, pp. 1655–1661, 2018, doi: 10.1109/TEMC.2018.2800774.

DOI : http://dx.doi.org/10.25139/inform.v7i1.4457