Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

Using Genetic Algorithm to Solve Puzzle Games: A Review

Iksan Bukhori[1)], Jason Felix[2)], Saddam Ali[3)]
[1)2)3)]Study Program of Electrical Engineering, President University, Bekasi 17550, Indonesia
[1)]iksan.bukhori@president.ac.id, [2)]jason.felix@studen.president.ac.id, [3)]saddam.ali@student.president.ac.id

ABSTRACT

Puzzles have been recognized for their development as a popular form of entertainment due to their ability to intricately challenge the mind while engendering creativity in the player. The development of puzzle games has given rise to a new generation of puzzle games characterized by diverse sequences and different image variations. With the rapid development of puzzle games, we looked at solving approaches using Genetic Algorithms (GA). In this paper, we try to analyze several puzzle games such as Sliding Blocks, Sudoku, Tic-Tac-Toe, and Jigsaw that can be solved using GA. We found that 120 papers have examined the use of GA for puzzle games, and eliminated into 14 papers. We evaluated these 14 papers for each puzzle game we selected by comparing the chromosome representation, GA operator, GA parameters, and the results. Based on the discussion, the application of GA to solve puzzle games can be effectively executed with a high degree of accuracy. Puzzle games that use measurement methods such as Sliding Block, Sudoku, and Jigsaw run in a similar pattern. What is common to all of them is that the chromosomes are represented as matrices or arrays in all cases, and standard genetic operators such as selection, crossover, and mutation are used. The population size is large, often 1000 chromosomes, and parameters such as mutation rate are kept low, around 5%. On the other hand, the performance of GA for solving Tetris and Tic-Tac-Toe from each publication cannot be compared due to different measurement methods and metrics.

Keywords: Genetic Algorithms, Jigsaw, Puzzle Games, Sliding Block, Sudoku, Tetris, Tic-Tac-Toe

INTRODUCTION

Puzzle games have long been a type of entertainment that challenges the mind while stimulating creativity. The term "puzzle" was derived from the old French language, namely "Aposer," which was then transformed to “Pose” or “Pusle” in ancient English, which was then altered to puzzle. A jigsaw puzzle game was discovered in 1766 as the first puzzle game. That year, John Spilsbury, an excellent cartographer, made a map on a sheet of wood and chopped it into parts based on nation lines. In the 1800s, jigsaw puzzles, which were first solely used to aid in geography learning, evolved into puzzles utilized by all groups in formats like paintings or transportation imagery.

The development of puzzle games reflects that, over time the interest in games continues to grow. Along with the growing appeal, puzzle games are also adaptive to the time. Puzzles, which were first recognized as learning education aids, have become broad entertainment activities. In 2020, Puzzle games took first place as mobile games played in the U.S., U.K., Japan, and South Korea. Apart from that, puzzle games also generate, generating $6.9 billion in revenue in one year (Gu, 2021). These numbers show that puzzle games are still trending and becoming a challenge for developers to find innovative methods to create exciting playing experiences.

With the increase in intricacies and sophisticated strategy required to win the game, puzzle games also attract the attention of researchers to develop machine learning algorithms to win the game without human involvement. Genetic Algorithm is one of the algorithms that take such a challenge.

The genetic algorithm is a computational technique utilized for addressing optimization issues, including both restricted and unconstrained scenarios. This approach draws inspiration from the principles of natural selection, which govern the evolutionary processes observed in biological systems (Nayyar, Le, & Nguyen, 2018). The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution (Sharma,2022).

In this research, we will explore the integration of genetic algorithms as answers in developing increasingly complex puzzles. We hope to make a meaningful contribution to the understanding and development of puzzle games and genetic algorithms, as well as unlock the potential for innovation in the development of puzzle games that continue to grow. Therefore, this study will offer significant insights for those in the gaming industry, including gamers,

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

developers, and researchers. With an overall objective of enhancing the level of engagement and complexity within the puzzle player community, the primary aim is to develop an exciting gaming experience.

METHOD

This review paper utilized the Publish or Perish software to narrow the scope of the search by entering "Genetic Algorithm" in the Title Words box and "Puzzle Game" in the Keywords box. One hundred and twenty papers were gathered for reference using this parameter. Following a filtering process, 14 publications were eventually accepted for discussion as genetic algorithm approaches to puzzle game solutions in this review paper. The reason for the decrease from 120 to 14 publications was due to the accessibility to the previous publications and the relevancy of the main topic that the author discussed.

RESULTS

Researchers have chosen puzzle games as experimental problems to explore and develop optimal solutions and strategies using genetic algorithms. When approaching a puzzle game using a genetic algorithm, several things must be considered, such as encoding, accuracy, and uniqueness. Considering the many puzzle games that have been developed to date. In this research, the author chose the following puzzle games which could use a genetic algorithm approach for solving.

Genetic algorithms are based on the principles of genetics and evolution; in 1975, John Holland and one of his students at the University of Michigan developed this idea in his book “Adaptation in Natural and Artificial Systems.” He described how to apply the principles of natural evolution to optimization problems and built the Genetic Algorithms. Holland’s theory has been further developed, and now, Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization problems (Sivanandam & Deepa, 2008).

Since the goal of a genetic algorithm is to look like natural environments, most of the necessary vocabulary comes from the life sciences. Although this language is used to describe biological entities, the entities in genetic algorithms are much simpler to understand (Mitchell, M. 1995).

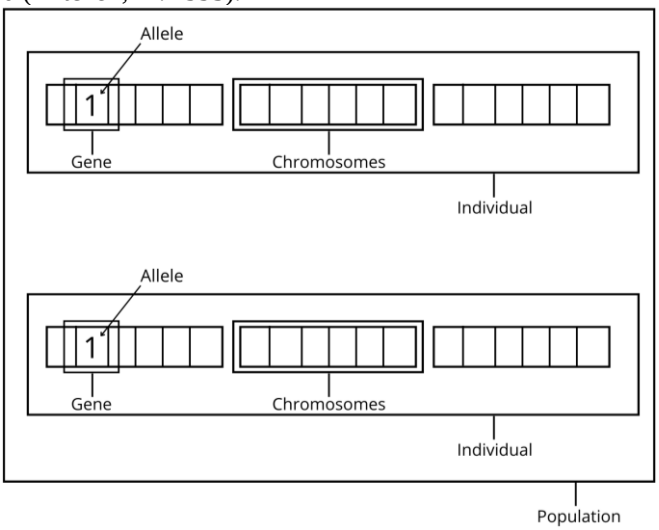


Fig. 1 Principles of Genetics and Evolution

As seen above, a gene contains an allele, the gene's value (represented by the number 1). At the same time, a chromosome is a sequence of genes, an individual is a collection of chromosomes, and a population is a collection of individuals. In its basic principles, a genetic algorithm has a cycle that can be explained and depicted through the following flowchart. In the flowchart, there are basic stages of how genetic algorithms work to solve various problems.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

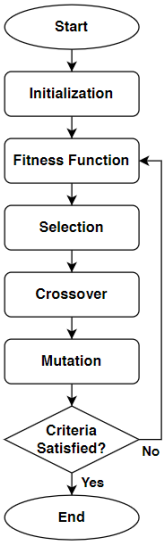


Fig. 2 Genetic Algorithm flowchart

This concludes our explanation of genetic algorithms, the preceding tables and flowcharts are our attempts to simplify the explanation of genetic algorithms. In this explanation, we focus more on explaining the application of genetic algorithms in various puzzle games, such as sliding blocks, sudoku, tetris, tic-tac-toe, and jigsaw. We will outline how this genetic algorithm approach is used in the context of these games to achieve efficient and optimal solutions.

Sliding Block

Slide Box Puzzle, is a puzzle that generally consists of 15 boxes, numbered 1 to 15, that can be shifted horizontally or vertically in a four-by-four grid that has one space between its 16 locations. The puzzle aims to arrange the boxes in numerical order using only the extra space in the box to slide the numbered titles. These pieces can be simple shapes or have various features such as colors, patterns, sections of a larger picture (similar to a jigsaw puzzle), numbers, or letters (Britannica, T. Editors of Encyclopaedia, 2009).

Suh and Lee (Suh et al., 1989) proposed GA to solve the sliding block case. Different from traditional methods, Suh's algorithm uses an operator-oriented representation scheme, where each move in the solution sequence is represented as ((x,y),o), where "(x,y)" denotes the location of its initial state and "o" represents as 0,1,2,3 denote moves (L, U, R, D). The example of this algorithm is ((2,1),0), ((2,0),1), ((1,0), 2), ((1,1), 3), ((2,1), 2), ((2,2), 1). This approach uses two genetic operators which are crossover and directed mutation. Parameters to adjust the algorithm are initial length, maximum length, minimum length, maximum generation, population, crossover rate, and mutation range. The result for 3x3 puzzles, the moves needed are between 5 to 40 while for 4x4 puzzles, the moves needed are between 20 to 40. However, Suh stated that their algorithm had trouble solving sliding block puzzles that require quite a long solution sequence/move.

Sha'ban (Sha'ban et al., 2009) proposed a GA to solve the sliding tile 8-puzzle which is the same as a 3x3 sliding block puzzle. The algorithm operates through a cycle of stages, including tournament selection, fitness evaluations, crossovers, and mutations. The chromosome is represented as an array of 9 numbers which denotes. The number of solutions depends on the number of generations that are inputted. They concluded that the proposed GA to solve the sliding tile 8-puzzle is a feasible plan because the proposed heuristic GA can find the solution in a large search space.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

Table 1. The Comparison of Sliding Block Genetic Algorithm

Comparison	Suh's GA	Shaban's GA
Chromosome Representative	a sequence consists of ((x,y),o) where (x,y) is initial position and o is the move L, U, R, D	a 3x3 matrix with 1-8 numbers and a blank
Operator	selection, crossover, mutation	selection, crossover, mutation
Parameter	initial length = 12, max generation = 36, population = 20, crossover rate = 0.7, mutation range = 3	population = 5, crossover rate: 0.7, mutation rate: 0.011
Result	3x3 puzzle solved only once in 5-20 generation	3x3 puzzle solved several times in 5-25 generation

Sudoku

Sudoku is a popular form of number game. In its simplest and most common configuration, sudoku consists of a 9 × 9 grid with numbers appearing in some of the squares. The object of the puzzle is to fill the remaining squares, using all the numbers 1–9 exactly once in each row, column, and the nine 3 × 3 subgrids. Sudoku is based entirely on logic, without any arithmetic involved, and the level of difficulty is determined by the quantity and positions of the original numbers (Wilson R, 2023).

Mantere (Mantere et al, 2006) proposed GA to solve and rate sudoku puzzles. The chromosome of the algorithm as the solution candidate is in the form of an integer array with 81 numbers divided into 9 sub-blocks of 9 numbers. Each number in the array represents a possible value for a cell in the sudoku grid. The algorithm applied uniform crossover only in the same sub-block and the swap, 3-swap, and insertion mutation with the ratio 50:30:20 respectively. The population size is 100 and the algorithm runs until it finds the optimal solution in between 100,000 generations. Mantere stated that there are many algorithms that perform more effectively than GA. However, this algorithm succeeded in determining the difficulty rating of sudoku puzzles from the newspaper at that time.

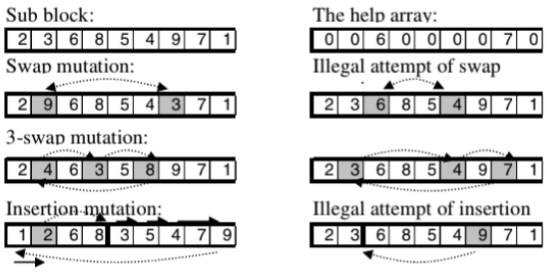


Fig. 3 Mutations in Mantere's GA

In Afriyudi et al (Afriyudi et al,2008) research, they proposed 3 variants of GA. The first variant's encoding method was using roulette-wheel selection, one-point crossover, and mutation. It didn't fit the problem constraints, resulting in suboptimal solutions. The second variant improves this by changing the encoding of the chromosomes to 1-9 integer array and multi-point crossover, but it's limited to scenarios with few cells to fill. The third variant combines the second variant's encoding method with selection and mutation without crossover processes to overcome these limitations. The fitness values are calculated by counting the number of occurrences of the same number in each row, column, and region of the Sudoku puzzle. The higher the fitness value, the better the gene is considered. The third variant could solve sudoku effectively from very easy, easy, medium, hard, and fiendish difficulty rating.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

Kazemi et al (Kazemi et al, 2014) proposed GA to compare their algorithm with Mantere's algorithm. The chromosome is represented as a two-dimensional integer array with 9×9 size with 3 basic rules which are predefined numbers that cannot be changed, each row and column contain 1-9 numbers only once, and each 3x3 sub-squares contains 1-9 numbers only once. To calculate the fitness function, they formulate it by using the maximum possible number of mismatches in a chromosome (max: 81) minus the total number of mismatches in the given chromosome. The process of the algorithm will be stopped if the chromosome with the fitness of 81 is found. The best population size for this algorithm is 20 and they are also limited to 100,000 generations to compare with Mantere's GA. Both tests are given 100 sudoku puzzles and solved sudoku can be seen in the 'Count' column in their publications.

Table 2. The Comparison of Sudoku Genetic Algorithm

Comparison		Mantere's GA	Afriyudi's GA	Kazemi's GA
Chromosome Representative		an integer array of 81 numbers divided to 9 sub-blocks	an integer array of only missing numbers	a 2 dimensional integer array (9x9)
Operator		uniform crossover, mutations (swap, 3-swap, insertion)	selection, mutation	crossover, mutation
Probability Rate		swap = 0.5; 3-swap = 0.3; insertion = 0.2	-	crossover = 0.65; mutation 1 = 0.15; mutation 2 = 0.04
Difficulty Rating	1 star	100	-	100
	2 stars	69	-	100
	3 stars	46	-	100
	4 stars	26	-	97
	5 stars	23	-	79
	Easy	100	-	100
	Challenging	30	-	51
	Difficult	4	-	17
	Super difficult	6	-	20
	Very easy to Fiendish	-	solved effectively	-

Tetris

Tetris has been released for virtually every computer and electronic gaming system, and it is often revered as a classic. Though numerous sequels have been spawned, Tetris games almost always have the same play mechanics: differently shaped blocks drop at varying speeds, and, as the blocks descend, the player must rotate and arrange them to create an uninterrupted horizontal row on the screen. When the player forms one or more solid rows, the completed rows disappear. The goal of the game is to prevent the blocks from stacking up to the top of the screen for as long as possible. Subsequent versions of the game included different modes of play and unique twists, but the overall gameplay usually mirrored the original Tetris quite closely (Britannica, T. Editors of Encyclopaedia, 2023).

In 2004, Flom (Flom et al, 2004) proposed research to show that GA can be used to weight an evaluation function for a Tetris game. The process begins with representing the chromosomes of the agents as arrays of 32-bit IEEE float numbers. These individuals then run the Tetris game and their performance determines their fitness by the total number of lines each agent made. After testing 30 times of 750 generations with 4 types of crossover (uniform, one-point, HUX, and reduced surrogate one-point), Flom chose uniform crossover because it has the best result among other crossovers by the graph. The result tests are from two different experiments and show that Figure *left* is total lines made while Figure *right* is pieces-per-line ratio from total average of best individual.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

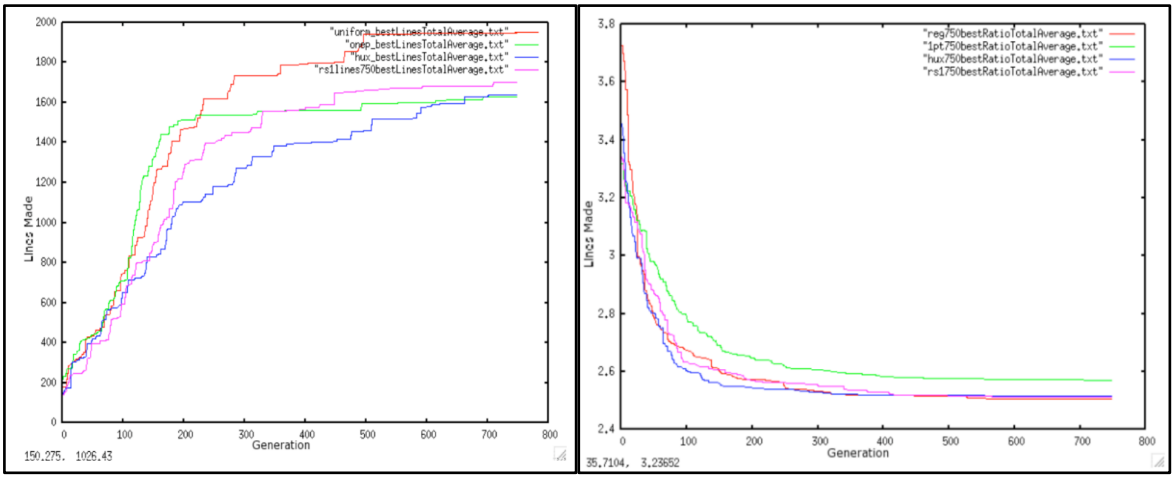


Fig. 4 Flom's GA result by total lines (left) and pieces-per-line ratio (right)

Jason Lewis (Lewis, 2015) proposed a GA to solve an optimization problem for Tetris in 2015. The algorithm starts with an initial population of candidate players, each candidate represented by a fixed weight vector in the matrix. These weights determine how the player evaluates and selects moves in the game. Tetris that he implemented in this paper has the same scoring mechanism as Nintendo's version of Tetris. The score increases by 2 points for 1 line, 5 points for 2 lines, 15 points for 3 lines, and 60 points for 4 lines (also known as tetris).

It is not explicitly mentioned about how the fitness function evaluates each generation. Even so, the result is quite surprising. Lewis' algorithm succeeded to achieve a minimum average score 2.4 per move and equivalent to a minimum efficiency of 40% after run through 30 generations with 1000 population and the rate of selection, crossover, and mutation being 20%, 40%, and 40% respectively. The result, which was able to make 179,531 moves in 3.9 minutes, has already demonstrated the effectiveness of the algorithm while a human with one move per second can achieve this result in 2 days.

Table 3. The Comparison of Tetris Genetic Algorithm

Comparison	Flom's GA	Lewis's GA
Chromosome Representative	arrays of 32-bit IEEE float numbers	*not mentioned*
Operator(s)	uniform crossover	selection, crossover, mutation
Parameter	5 (lines made, pile height, bumpiness, closed holes, wells)	10 (weighted blocks, connected holes, lines cleared, roughness, tetrises, pit hole percent, clearable line, deepest well, blocks, column holes)
Result	4.75091E33 lines made	2.5 points per move 179,531 moves (in 3.9 minutes)

Tic-Tac-Toe

Tic-tac-toe, an ancient game with origins dating back to 1300 BC in Egypt, is a classic two-player contest played on a 3x3 grid(Choi, A. S. (2021) the objective is for each player to strategically place their designated markings, typically represented as "X" and "O," with the goal of achieving a line of three of their markings in a row, column, or diagonal.

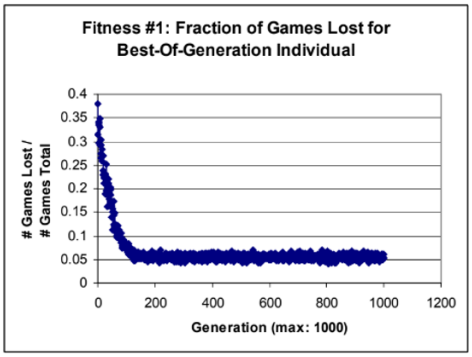
Hochmuth (Hochmuth, G, 2003) proposed a successful GA to find a perfect tic-tac-toe strategy in 2003. Hochmuth

* Corresponding author

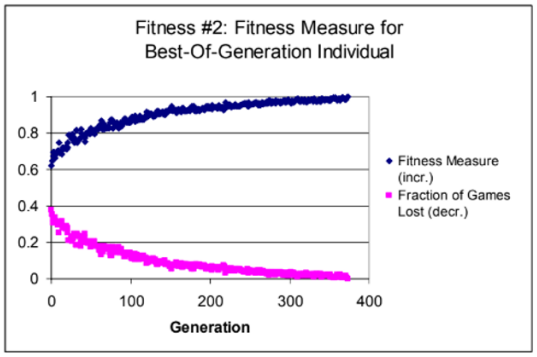
Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

defined each game states table refers to its genome, which represents 827 distinct genes. The operators are replication, mutation, and crossover. Replication in this context means copying one individual into the next generation with defined probability and without modification. The mutation will replace each gene with the new random value with its probability. The crossover operator is modified, and the number of crosses is chosen so it is represented as a variable and evaluated by its probability too. The first experiment failed because of the insufficient population size and different values that were performed. After the evaluation, his second experiment succeeded in finding the best solution within 373 generations after 1688 seconds (28 minutes). From this experiment, Hochmuth concludes that the higher crossover probability was likely to cause excessive mutation in the offspring and may therefore have prevented the populations from reaching optimality in the end.



Population Size	10,000
Generations	1000
$p_{crossover}$	0.15
$p_{replicate}$	0.10
p_{mutate}	0.001



Population Size	500
Generations	termination at 373 (max: 500)
$p_{crossover}$	0.15
$p_{replicate}$	0.10
p_{mutate}	0.001

Fig. 5 Hochmuth's GA result from first experiment (failed) and second experiment (succeed)

Anurag et al (Anurag et al, 2008) implemented customized GA to search the no-loss strategy for tic-tac-toe. Different from Hochmuth's GA, they only used 765 game states. It is a 765x10 matrix which means 765 individuals and 10 numbers that consist of 9 numbers as the game-state (0 = none, 1 = "X", 2 = "O"), and the last number as the total move. In the GA process, the initial step involves creating a random population of solutions or strategies, followed by the design of GA operators like selection, recombination, and mutation to address complex game-playing problems. The efficient problem-specific representation allows for a straightforward recombination operator choice, while an innovative niching mechanism, the controlled elite-preserving operator, maintains population diversity for better solutions. Furthermore, their approach employs a two-tier GA strategy in which both of the players use GA to find the strategy of each move. It returned 117 solutions for the first player and 621 solutions for the second player, and there are 72,657 no-loss strategies in total from the combination of the first and second player which enhances its effectiveness.

The algorithm proposed by Ling et al (Ling et al, 2011) is learned by a neural network with a double transfer function (NNDTF) and trained by a genetic algorithm. It has the purpose of proving that NNDTF can perform better than traditional neural networks in playing tic-tac-toe. For the experiment session, a 9-input-1-output neural GA network with 8 hidden nodes of NNDTF was trained using 100 patterns over 50,000 iterations, employing a GA with a population size of 10, a 0.5 selection probability, and a weight factor of 0.1. The algorithm achieved a fitness value of 0.9605, demonstrating its efficiency in modeling the data.

Mohammadi (Mohammadi et al, 2013) proposed genetic programming, coevolution, and interactive fitness in purpose to develop a human-competitive algorithm for playing tic-tac-toe. The algorithm used a 3x3 matrix to represent the chromosome and evaluated by traditional GA operators such as selection, crossover, and mutation.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

Table 4. The Comparison of Tic-tac-toe Genetic Algorithm

Comparison	Hochmuth's GA	Anurag's GA	Ling's GA	Mohammadi's GA
Chromosome Representative	a mapping table for 827 distinct game states	an array of 10 numbers from 765 game states	i, j, l of each NNDTF outputs	a 3x3 matrix
Operator	replication, mutation, crossover	selection, recombination, mutation	crossover, mutation	crossover, mutation, reproduction
Parameter	population = 500, crossover = 0.15, replication = 0.10, mutation = 0.001	population = 100	population = 10, selection prob. = 0.5, weight factor = 0.1	population = 128, depth size = 20, crossover = 0.8, mutation = 0.5, replication = 0.2, coevolution and interactive fitness
Result	fittest solution at 373rd generation within 1688 seconds (28 minutes)	chance of no-loss strategies for the first player is higher than the second player	NNDTF trained with GA performs better than traditional NN	achieved a human-competitive algorithm

Jigsaw

A jigsaw puzzle is a collection of varied and irregularly shaped pieces that, when arranged correctly, form a picture or map. These puzzles are so named because the image, which is first pasted on wood and then on cardboard, is cut into pieces with a jigsaw, which cuts intricate lines and curves (Britannica, T. Editors of Encyclopaedia, 2021).

In Sholomon et al.'s work (Sholomon et al, 2013), they demonstrated the GA's ability to solve incredibly large jigsaw puzzles, scaling up to 22,834 pieces, a remarkable leap from the previous limit of 3,000 pieces achieved by Pomeranz et al (Pomeranz et al, 2011) Their approach, by employing a traditional genetic algorithm and representing each puzzle as a (N x M) matrix with initial piece numbers. There are 1,000 chromosomes in the population and preserve the top 4 chromosomes in each generation. The crossover operator generates the remaining population with a 5% mutation rate. The roulette wheel selection of parent chromosomes results in a single child in each crossover. The GA always runs for 100 generations in total. The results from using the GA to solve the jigsaw puzzles ten times are exceptional; the average scores are all above 80%, and the best solution achieved a score of 97.12% while finishing the puzzle with 10,375 pieces.

$$D_h(x_i, x_j) = \sqrt{\sum_{i=1}^{L} \quad \sum_{k=1}^{3} \quad (x_i(l, L, c) - x_j(l, 1, c))^2} \qquad (1)$$

$$D_v(x_i, x_j) = \sqrt{\sum_{i=1}^{L} \quad \sum_{k=1}^{3} \quad (x_i(L, l, c) - x_j(1, l, c))^2} \qquad (2)$$

$$f(a) = \frac{1}{1+\sum_{i=1}^{N} \quad \sum_{j=1}^{M-1} \quad D_h(c_{i,j}, c_{i,j+1}) + \sum_{i=1}^{N-1} \quad \sum_{j=1}^{M} \quad D_v(c_{i,j}, c_{i+1,j})} \qquad (3)$$

A year later, Hynek (Hynek, 2014) employed GA to solve jigsaw puzzles with a small piece count, utilizing pixel border dissimilarity in 24-bit RGB images, in contrast to Toyama's GA, (Toyama et al, 2002) black and white images. However, aligning pieces based on pixel values along borders proved challenging due to potential image irregularities. This algorithm assesses puzzle solutions as NxM matrices, with puzzle pieces represented as LxLx3 color intensity grids. It computes fitness (3) using horizontal dissimilarity (1) and vertical dissimilarity (2) measures with L representing the width and the length while c refers to the color of each piece. The algorithm carried 1000 individuals for 50 generations. This algorithm has two genetic operators. First, a partially matched crossover makes it simple to divide a series of perfectly built parts since crossing sites are chosen at random. Secondly, decimation is a secondary genetic operation used to enhance the fraction of fitter individuals in the population. It is often carried out at the start of a run. The best result is completing the 225-piece puzzle with 98% accuracy.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

In 2020, Guo et al (Guo et al, 2020) proposed a GA solver for small-scale jigsaw puzzles. A (N x M) matrix is again used as the chromosome representative. They modified the crossover so that the matched pieces from the parent's chromosome would seek to match pieces within the boundary and add them to the child's chromosome. If the crossover phase can't get the best-matching piece anymore then the mutation will give the random remaining pieces. The population has 1000 chromosomes and maintains the top 2 chromosomes in each generation. The rest of the population was generated through crossover, with a 5% mutation rate. Parent chromosomes were selected using the roulette wheel method and would be run for 100 generations in each experiment. The proposed algorithm achieved 98.45% on average for a 626-piece jigsaw puzzle.

Table 5. The Comparison of Jigsaw Genetic Algorithm

Comparison	Sholomon's GA	Hynek's GA	Guo's GA
Chromosome Representative	a NxM matrix	a NxM matrix	a NxM matrix
Operator	selection, crossover, mutation	selection, partially-matched crossover, decimation	selection, modified crossover, mutation
Parameter	population = 1000, mutation = 0.05	population = 1000, selection = 0.1, hor. dissimilarity, ver. dissimilarity	population = 1000, mutation = 0.05
Result	97.12% on a 10,375 pieces puzzle	98% on a 225 pieces puzzle	98.45% on a 626 pieces puzzle

DISCUSSION

Genetic algorithms work well in solving puzzle games. After years of development, genetic algorithm research has become more accurate and effective than previous research. On this occasion, we will try to summarize the conclusions from each puzzle game using a genetic algorithm. The conclusions we draw are based on the algorithm approach and performance matrix.

Sliding Block

Suh's and Shaban's GA, each employing unique approaches in solving the 3x3 puzzle. Suh's GA represents its chromosomes as sequences in the format ((x, y), o), where (x, y) denotes the initial position and o represents moves (L, U, R, D). On the other hand, Shaban's GA adopts a 3x3 matrix representation with numbers 1-8 and a blank space. Both algorithms use common genetic operators, including selection, crossover, and mutation. Suh's GA incorporates additional parameters such as initial length and mutation range.

Suh's Genetic Algorithm involves a population of 20 individuals, an initial length of 12, a maximum of 36 generations, a crossover rate of 0.7, and a mutation range of 3. This algorithm successfully solves the 3x3 puzzle within 5-20 generations. On the other hand, Shaban's Genetic Algorithm operates with a smaller population of 5, a crossover rate of 0.7, and a mutation rate of 0.011. Despite its more concise parameter set, Shaban's GA achieves the goal of solving the 3x3 puzzle in 5-25 generations.

Sudoku

Mantere's GA focuses on solving and rating Sudoku puzzles using an integer array representation with 81 numbers divided into 9 sub-blocks. It employs uniform crossover and mutations such as swap, 3-swap, and insertion. Afriyudi et al. proposed three variants of GA with different encoding methods and variations in crossover and mutation strategies, addressing constraints and improving solution effectiveness for Sudoku puzzles. Kazemi et al. introduced a GA that utilizes a two-dimensional integer array with specific rules for encoding and fitness calculation, aiming to compare its performance with Mantere's GA.

Based on performance metrics, Mantere's GA successfully determines the difficulty rating of Sudoku puzzles, even though it may not outperform other algorithms in terms of solving efficiency. Afriyudi's third variant of GA effectively solves Sudoku puzzles with varying difficulty ratings, demonstrating adaptability to different challenge levels.

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

Kazemi's GA achieves specific difficulty ratings for solved Sudoku puzzles, and its performance is compared with Mantere's GA using 100 puzzles.

Tetris

Flom's GA for Tetris employs arrays of 32-bit IEEE float numbers as chromosome representatives. It utilizes the uniform crossover operator and focuses on optimizing parameters related to lines made, pile height, bumpiness, closed holes, and wells. Lewis's GA adopts a different approach, incorporating selection, crossover, and mutation operators. The specific chromosome representation is not mentioned, but the algorithm optimizes 10 parameters related to weighted blocks, connected holes, lines cleared, roughness, tetrises, pit hole percent, clearable line, deepest well, blocks, and column holes.

Based on performance metrics, Flom's GA achieves an impressive result of 4.75091E33 lines made, emphasizing the algorithm's effectiveness in optimizing Tetris gameplay for high-line production. Lewis's GA demonstrates performance with a result of 2.5 points per move, indicating its ability to optimize Tetris gameplay in terms of scoring efficiency. Additionally, it achieves 179,531 moves in 3.9 minutes, highlighting the algorithm's computational efficiency.

Tic-Tac-Toe

Hochmuth's GA makes use of crossover, mutation, and replication operators together with a mapping table that covers 827 different game states. Using a range of 10 values selected from 765 game states, Anurag's GA uses recombination, mutation, and selection to represent its chromosomes. Ling's GA uses crossover and mutation and uses the i, j, and l outputs from each NNDTF as its chromosomal representation. Mohammadi's GA uses crossover, mutation, and reproduction operations together with a 3x3 matrix as its chromosomal representation to achieve a human-competitive algorithm through interactive fitness and coevolution.

Based on performance metrics, Hochmuth's GA is notable for reaching its fittest solution in 1688 seconds at the 373rd generation. Anurag's GA, with a population of 100, shows that the first player has a greater probability of no-loss strategies than the second player. When trained using a Genetic Algorithm, Ling's proposed method performs better than conventional neural networks with a population of 10, a selection probability of 0.5, and a weight factor of 0.1. Using coevolution and interactive fitness, Mohammadi's GA achieves a human-competitive algorithm with a population of 128 and a depth size of 20, as well as specific crossover, mutation, and replication rates. Each method solves the Tic-Tac-Toe issue from a different angle, showcasing the versatility and flexibility of genetic algorithms in a variety of forms and tactics.

Jigsaw

The chromosome representative used in the Jigsaw Genetic Algorithms put out by Sholomon, Hynek, and Guo is an NxM matrix. Using a population of 1000, Sholomon's method makes use of crossover, mutation, and selection operators with a 0.05 mutation rate. Hynek's GA, which likewise makes use of a NxM matrix, has the following characteristics: a population size of 1000, a selection rate of 0.1, and considerations for both horizontal and vertical dissimilarity. It uses selection, partially-matched crossover, and decimation. Guo's GA uses a population of 1000 and a mutation rate of 0.05. It uses selection, modified crossover, and mutation. Its matrix representation is comparable to Guo's. Even while all algorithms share the same chromosomal representation, they bring distinct differences in their operators and parameters, which indicate different approaches to solving jigsaw puzzles.

On a large 10,375-piece puzzle, Sholomon's GA performed admirably, with a success rate of 97.12%. Hynek's GA performed better, using a mix of selection, partially-matched crossover, and decimation to achieve a high success rate of 98% on a smaller 225-piece puzzle. Using mutation, modified crossover, and selection, Guo's GA demonstrated an astounding 98.45% success rate on a 626-piece puzzle. These results demonstrate how well these Jigsaw Genetic Algorithms work to solve puzzles of different levels of difficulty.

CONCLUSION

Based on the results and discussion section, the authors found that the application of GAs to solve puzzle games is executed effectively with a high level of accuracy. The authors concluded that there are puzzle games that already use the same measurement method for their performance such as Sliding Block, Sudoku, and Jigsaw. On the other hand, the performance of GAs to solve Tetris and Tic-Tac-Toe from each publication cannot be comparable due to different measurement methods and metrics. For future development, the authors suggest the existence of measuring

* Corresponding author

Journal of Computer Networks, Architecture and High Performance Computing
Volume 6, Number 1, January 2024
https://doi.org/10.47709/cnahpc.v6i1.3348

Submitted : Dec 9, 2023
Accepted : Jan 4, 2024
Published : Jan 5, 2024

parameters so that the performance metrics can be measured using the same method to make it easier to carry out further development.

REFERENCES

Gu, T. (2021, August 12). Newzoo Corporation. Retrieved from Newzoo Corporation Website: https://newzoo.com/resources/blog/puzzle-game-mobile-data-revenues-player-behavior-consumer-insights-us-japan-china-korea

Nayyar, A., Le, D.-N., & Nguyen, N. G. (2018). Advances in Swarm Intelligence for Optimizing Problems in Computer Science. In A. Nayyar, D.-N. Le, & N. G. Nguyen, Advances in Swarm Intelligence for Optimizing Problems in Computer Science (p. 28). CRC Press.

Sharma, H. (2022, November 07). SkillLync.Inc. Retrieved from LkillLync.Inc Website: https://skill-lync.com/student-projects/week-5-genetic-algorithm-104

Sivanandam, S., & Deepa, S. (2008). Introduction to Genetic Algorithms. In S. Sivanandam, & S. Deepa, Introduction to Genetic Algorithms (pp. 15-37). Berlin, Heidelberg: Springer.

Mitchell, M. (1995). Genetic Algorithms: An Overview. Complexity, 1(1), 31-39.

Britannica, T. Editors of Encyclopaedia (2009, July 21). Fifteen Puzzle. Encyclopedia Britannica. https://www.britannica.com/topic/Fifteen-Puzzle

Suh, J. Y., & Lee, C. D. (1989). Extending Distributed Genetic Algorithms to Problem Solving: The Case of the Sliding Block Puzzle. 1-56.

Sha'ban, R. Z., Alkallak, I. N., & Sulaiman, M. M. (2009). Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem. 1-13.

Wilson, R. (2023, May 5). sudoku. Encyclopedia Britannica. https://www.britannica.com/topic/sudoku

Mantere, T. (2010). Solving Rubik's Cube with Genetic Algorithm. 1-5.

Afriyudi. A, Pramudyo, A. S., & Akbar, M. (2008). Penyelesaian Puzzle Sudoku menggunakan Algoritma Genetik. 1-4.

Kazemi, S. M., & Fatemi, B. (2014). A Retrievable Genetic Algorithm for Efficient Solving. 1-5.

Britannica, T. Editors of Encyclopaedia (2023, October 29). Tetris. Encyclopedia Britannica. https://www.britannica.com/topic/Tetris

Flom, L., & Robinson, C. (2004). Using a Genetic Algorithm to Weight an Evaluation Function for Tetris. 1-5.

Lewis, J. (2015). Playing Tetris with Genetic Algorithms. 1-4.

Choi, A. S. (2021). Tic Tac Toe. 1-10

Hochmuth, G. (2003). On the Genetic Evolution of a Perfect Tic-Tac-Toe Strategy. 1-8.

Bhatt, A., Varshney, P., & Deb, K. (2008). In Search of No-loss Strategies for the Game of Tic-Tac-Toe using a Customiezd Genetic Algorithm. 1-8.

Ling, S. H., & Lam, H. K. (2011). Journal of Intelligent Learning System and Application . Playing Tic-Tac-Toe Using Genetic Neural Network with Double Transfer Functions, 37-44.

Mohammadi, H., Santos, M. V., & Borne, N. P. (2013). Evolving Tic-Tac-Toe Playing Algorithms Using Co-Evolution, Interactive. 1-6.

Britannica, T. Editors of Encyclopaedia (2021, July 26). jigsaw puzzle. Encyclopedia Britannica. https://www.britannica.com/topic/jigsaw-puzzle

Sholomon, D., David, O., & Netanyahu, N. S. (2013). A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles. 1-8.

Pomeranz, D., Shemesh, M., & Ben-Shahar, O. (2011). A Fully Automated Greedy Square Jigsaw Puzzle Solver. 1-8.

Hynek, J. (2014). Sequence Matching Genetic Algorithm for Square Jigsaw Puzzles. 1-8.

Toyama, F., Fujiki, Y., Shoji, K., Miyamichi, J.: Assembly of puzzles using a genetic algorithm. In Proc. of 16th International Conference on Pattern Recognition, Quebec, Canada, vol.4, pp.389-392 (2002).

Guo, W., Wei, W., Zhang, Y., & Fu, A. (2020). A Genetic Algorithm-Based Solver. 1-12.

* Corresponding author