Comparative Analysis of the Performance Testing Results of the Backtracking and Genetics Algorithm in Solving Sudoku Games

Bonifacius Vicky Indriyono[1]*, Natalinda Pamungkas[2], Zudha Pratama[3],Ery Mintorini[4], Imelda Dimentieva[5], Pita Mellati[6]
[1,2,5,6]Information System Department, Universitas Dian Nuswantoro, Indonesia
[3]Informatics Department, Universitas Dian Nuswantoro, Indonesia
[4]Visual Communication Design, Universitas Dian Nuswantoro, Indonesia
[1]bonifacius.vicky.indriyono@dsn.dinus.ac.id*; [2]natalinda.pamungkas@dsn.dinus.ac.id; [3]zudhapratama@dsn.dinus.ac.id;
[4]ery.mintorini@dsn.dinus.ac.id; [5]612202200069@mhs.dinus.ac.id; [6]612202200061@mhs.dinus.ac.id
*corresponding author

ABSTRACT

Games that hone thinking skills and logical accuracy have recently been very popular. One of them is the game Sudoku. Sudoku is a game that hones logic through puzzles arranged in rows and columns. Sudoku is also defined as a puzzle game that aims to arrange several numbers in a grid from one to nine on a grid consisting of 9x9 squares. The concept of this Sudoku game is to enter numbers into the rows and columns provided. The rule of this game is that the numbers arranged on the board cannot be the same in every row, column, and 3x3 square in the grid. In another sense, each number entered must appear once in each row and column. When running Sudoku, several numbers are already instructions for players to fill in the next boxes. The number of clues at the beginning of the game determines the difficulty level players face. The fewer clues, the more difficult the Sudoku is to solve. This study aims to compare how to solve Sudoku using genetic algorithms, backtracking, and the completion time needed. The tests' results show that the genetic and backtracking algorithms can solve Sudoku games quickly. Still, the backtracking algorithm has the advantage of being relatively shorter, and the process is not so complicated that the backtracking algorithm can be an alternative solution to solving Sudoku logic games.

Keywords : Puzzle Games; Sudoku Games; Backtracking Algorithm; Genetic Algorithm.

Article History

Received : June, 21st 2023	Accepted : July, 15th 2023	Published : July, 25th 2023

I. INTRODUCTION

A puzzle game is a game that requires logic and proper reasoning to solve problems involving logic. Currently, there are many types of puzzle games and riddles. Of the many puzzle games, one of them is a number puzzle. A well-known number puzzle game is Sudoku. Sudoku itself is a number puzzle game originating in Japan [1]. How to use Sudoku is played on a board consisting of 9 squares where each box has a subgrid with a size of 3×3. Sudoku is a numbers game where players must fill in several available empty boxes using random numbers from 1 to 9 [2]. At the beginning of this Sudoku game, several random numbers from 1 to 9 and boxes serve as clues to solve Sudoku puzzles. The player's task is to complete other empty or unfilled squares to fill the entire Sudoku board with numbers [3]. When viewed from the game's rules, this Sudoku has simple rules, requiring players to fill in an empty box with random numbers. However, even though it is simple in terms of the rules, it turns out that many players cannot complete it.

Several algorithms are used to solve Sudoku games, including genetics and backtracking. A genetic algorithm is an algorithm for searching based on the mechanism of natural selection and genetics. The genetic algorithm can be used as a very precise algorithm to solve complex optimization problems that cannot be solved using conventional methods [4]. While the backtracking algorithm is a revision of the brute-force algorithm and the exhaustive-search algorithm, in which this algorithm builds a state-space tree to find a solution [5]. The concept of this backtracking algorithm is to work recursively in finding a solution to a problem with several possible solutions [6].

Several previous studies that discussed using genetic algorithms and backtracking in artificial intelligence problems, especially Sudoku games, included [7] research on how to solve Sudoku using brute force and backtracking algorithms. Then [8] describes solving Android-based Sudoku puzzles with the backtracking algorithm. Research conducted by [9] discusses the steps for completing the 9x9 Su Doku Pattern using backtracking. Research from [10] is related to the discussion of how genetic algorithms can help solve Sudoku puzzle logic problems.

This study aims to analyze the results of comparing how to solve logical Sudoku puzzles between genetic algorithms and backtracking. The analysis was carried out regarding the complexity of the processing speed between the two algorithms used. This comparison aims to produce the best algorithmic solution for solving Sudoku games based on the solving steps' performance, speed, and complexity.

II. METHOD

A. Data Collection Techniques

Data collection techniques are part of finding reference sources needed to support research. In this study, a literature study is used to find reference sources. A literature study is a data collection technique by conducting surveys and investigations on existing documents and journals regarding potato plants. Reference books can also be used as a guide in the system development process.

B. Backtracking Algorithm

The backtracking algorithm is one of the algorithms in dynamically finding solutions to the state space tree based on the Depth First Search (DFS) algorithm. The backtracking algorithm will trace possible numbers that only lead to a solution or settlement. A settlement or solution can be found with a few traces and is efficient because it does not examine all possible existing settlement numbers [11]. Fig. 1 shows an example of a state space tree of the backtracking algorithm [12].

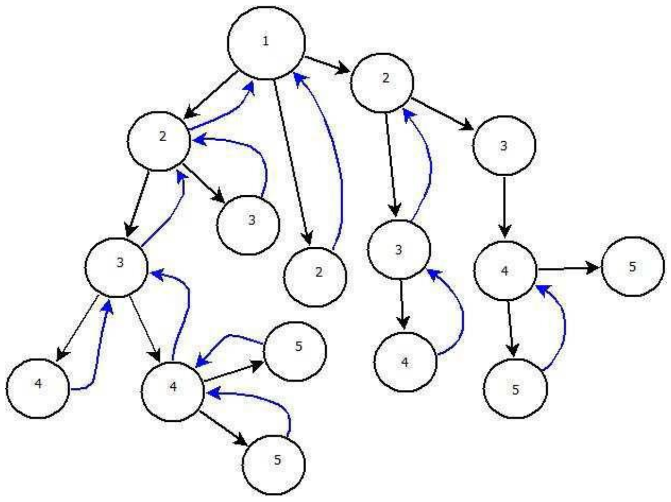


Fig.1. Example of a Backtracking Algorithm State Space Tree

Based on Fig. 1, the search process begins by visiting node 1 to node 4. If the destination to be searched has not been found, the next process is to continue the search to one of the previous branches (node 2 to node 3) and go down if there are still other branches. The process is carried out iteratively until the final destination is obtained (node 1 to node 5). With this backtracking method, the search process leads to only one solution, which does not lead to all possible solutions. This situation makes the search time with backtracking less so that the time needed is effective and efficient. In general, the completion steps in this backtracking algorithm (1) [13] are as follows [5] :

a) The form of finding a solution is a path from roots to leaves. The resulting node is called a live node, and the expanded live node is called an E-node (Expand node).
b) If the expansion of node E forms a path that does not lead to a solution, then that node will become a dead node that cannot be expanded anymore.
c) If the last position is located at a dead node, then the search process is carried out by generating other child nodes, and if there are no child nodes, then tracing back to the parent node.
d) The search process will stop if a solution has been found or no live nodes can be found.

Algorithm (1). Backtracking Algorithm

```
Backtrack procedure (input k:integer)
{
   Looking for all solutions with backtracking method
   Input : k, that is index of component vector x[k]
   Output : solution
   X=(x[1].x[2],...x[n])
}
Algorithm :
  for each untested x[k] such that (x[k]=T(k)) and B(x[1].x[2],...x[k])=true
```

Algorithm (1). Backtracking Algorithm

```
  do
    if (x[1],x[2],...x[k]) is the path from root to leaf then PrintSolution(x)
    endif
    BacktrackR(k+1)
Endforeach
```

C. Genetic Algorithm

A genetic algorithm is a method for solving optimization problems based on natural selection, namely techniques that follow the evolution and development of biology [14]. Simultaneously the genetic algorithm evaluates several points in space. The genetic algorithm has a natural evolutionary process includes population, inheritance, mutation, crossover, and selection [15].

In designing a Sudoku puzzle game with a genetic algorithm, several candidate solutions for each puzzle box are first sought before entering the process. The goal of this process is to eliminate the unlikely solution candidates. For example, if a box is filled with the number 9, then the number 9 is eliminated from the candidates in the boxes in the same row, column, or region. This step aims to make the genetic algorithm have good convergence. The solution to a puzzle is obtained when a chromosome has a perfect fitness value because a chromosome with a fitness value below the perfect fitness value is not a solution to a Sudoku puzzle. In the genetic algorithm, one chromosome is a candidate solution to the puzzle. Because the possible values in the solution are between 1 and 9, the chromosome representation is a series of integer numbers with the chromosome length according to the number of empty squares in the puzzle. The fitness value is obtained from the chromosome length minus the number of error genes. An error gene is a gene that contains numbers that appear multiple times in a row, column, or region.

D. Definition of Puzzle

Puzzles are pieces of images that can be shaped or arranged to form one unified picture. From its shape, Puzzle media can be categorized as picture-learning media. Puzzles can be played by all groups, from children to adults. By playing puzzles, children can learn to be thorough and train children's memory. This puzzle game can be played anywhere at school, home, or elsewhere [16].

A puzzle game is a game that has pieces of images with a level of difficulty that adjusts the child's development. Puzzles are a form of play that requires precision and trains children to concentrate because they have to concentrate when assembling puzzle pieces to form a complete and complete picture as a whole [17].

E. Sudoku puzzles

Sudoku Puzzle is a fun and simple educational game that can attract students' attention in the learning process in the classroom. Sudoku has advantages, including developing ways of reasoning and thinking correctly according to logic. Sudoku generally uses numbers one through nine, arranged in a 9 × 9 grid divided into 3 × 3 squares called subgrids. The game Sudoku Puzzles requires players to use brain power and concentration in problem-solving strategies [18].

Sudoku is a puzzle game consisting of a grid of 81 squares arranged in 9 rows and 9 columns. In the Sudoku grid itself, some subgrids are 3x3 in size. In the Sudoku game, players are asked to fill in each box with one number, namely the numbers 1 to 9, so that each box has a unique number for each row, column, and subgrid on the board. The difficulty level of the Sudoku game is measured by the number of clues on the grid at the start of the game. The fewer clues on the grid, the more difficult the Sudoku will be to solve.

III. RESULT AND DISCUSSION

This section will explain how to solve the Sudoku math puzzle game using backtracking and genetic algorithms. The Sudoku puzzle model used is a 9x9 grid size. The pseudocode for the Sudoku solution is shown in the algorithm (2).

A. Analysis of Solving Sudoku with the Backtracking Algorithm

In general, solving Sudoku puzzles can be done manually using the following techniques :

1) Scan. Performs a row or column scan to identify which row contains certain numbers. This process is then repeated for each column (or row) systematically.
2) Marking. It is a process of logical analysis by marking candidate numbers that can be put in a box.
3) Analysis. The process of eliminating candidates, where progress is made by eliminating or deleting candidate numbers consecutively until one box contains only one candidate.

In particular, the application of the Backtracking algorithm in the Sudoku logic game is as follows:

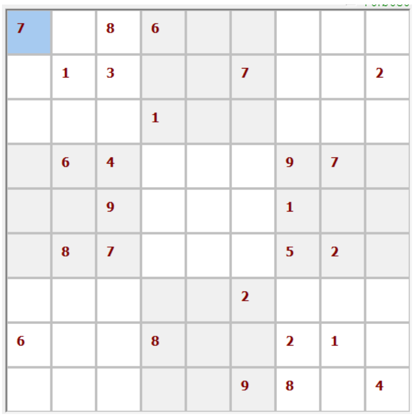
a) The process starts from the first row and first column matrix elements
b) Check all possible numbers that the row can own
c) If there is a valid number with the constraints of the Sudoku game, proceed with checking the next element of the matrix.
d) If no valid numbers are found, backtrack to the previous matrix elements.
e) The process stops when it has found a solution or if there is no possibility of a solution.

Algorithm (2). The Sudoku Algorithm

```
Procedure SolveSudoku(inputarray:matriks)
```

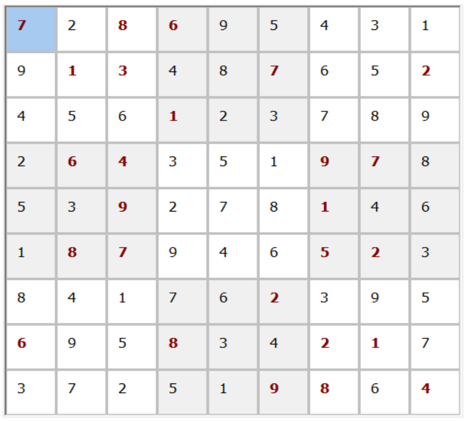
```
Solution : boolean, i,j:integer
Algorithm1.solution=false;
I=1;
J=1;
While (!solution&&i>0)
{
Array[i][j].SetValues(array[i][j].GetValue()+1);}
While (array[i][j].GetValue()<=9&&!place(i,j))
{
array [i][j].SetValue(array[i][j].GetValue()+1);
}
If (array[i][j].GetValue()<=max)
{
If(i==9&&j==9)
{
Solution=true
}
Else
{
j=j+1;
if (j>9)
{
i=i+1;
j=1;}}}
else
array[i][j].SetValue(0);
j- -;
if (j==0)
{j=column
i=i-1;}}}
```

We got some interesting results when putting this backtracking method and the programming language Delphi through their paces to solve a Sudoku puzzle. Fig. 3 shows the unfinished initial board or puzzle sudoku, which results from selecting the puzzle's difficulty level. The logic and pseudocode of the backtracking algorithm described above, the Sudoku puzzle problem in Fig. 3, can be solved as shown in Fig. 4.



7		8	6					
	1	3			7			2
			1					
	6	4				9	7	
		9				1		
	8	7				5	2	
					2			
6			8			2	1	
					9	8		4

Fig. 3. Example of a 9x9 Sudoku Puzzle



7	2	8	6	9	5	4	3	1
9	1	3	4	8	7	6	5	2
4	5	6	1	2	3	7	8	9
2	6	4	3	5	1	9	7	8
5	3	9	2	7	8	1	4	6
1	8	7	9	4	6	5	2	3
8	4	1	7	6	2	3	9	5
6	9	5	8	3	4	2	1	7
3	7	2	5	1	9	8	6	4

Fig. 4. 9x9 Sudoku Puzzle Solving

The process of testing the backtracking algorithm in solving this puzzle is carried out in several 10 Sudoku puzzles with different levels of difficulty or game level. The test results are shown in Table I.

TABLE I
TESTING RESULTS OF 10 SUDOKU PUZZLES USING THE BACKTRACKING ALGORITHM

Process Testing to	LOTS EMPTY BOX	Time Required	Number of Processes Backtracking
1	43	71230	628
2	43	80986	925
3	43	84012	818
4	43	107202	1165
5	44	81048	855
6	44	130791	1305
7	44	140675	1276
8	44	120871	1143
9	45	300986	2835
10	45	116479	1179

Table I shows the results of the trial process of the sudoku program on 10 puzzles that have different levels of difficulty. The number of empty boxes in each test is set from 43–45 empty boxes. In this experiment, the analysis was carried out on the side of the completion time and the number of backtracking processes. Based on the results in Table 1, it can be concluded that if there are many empty squares in the puzzle, not all are directly proportional to the increase in time and the amount of backtracking. Fig. 5 shows a graph of the backtracking test according to the data in Table I.

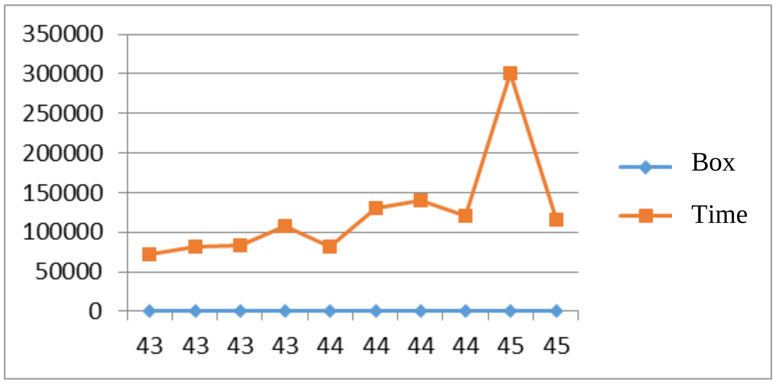


Fig. 5. Graph of Backtracking Test Results

To calculate the average Sudoku completion time with the backtracking algorithm, you can use the following Equation (1).

$$y = \frac{\sum x}{s} \tag{1}$$

Where the *y* variable is the average completion time, the *x* variable is the time required (*ms*), and the *s* variable is the number of trials, so from the above Equation, the average value can be calculated using Equation (2).

$$y = \frac{1234280}{10} = 123428\,ms \tag{2}$$

The graph in Fig. 5 shows the time the program needs to complete the Sudoku Puzzle from the tests carried out in this study. The time parameter in this test uses units of microseconds (ms). This was done to see the difference in time between tests with empty boxes and different numbers. The graphs and test data show that the 7th and 9th tests have a more prominent completion time than the others. This happened because the two tests had a higher difficulty than the other 8 tests.

B. Analysis of Solving Sudoku with Genetic Algorithms

The following procedures need to be carried out to use a genetic algorithm to solve a Sudoku puzzle:

1. Convert the values in the cell into the genes that will form the chromosomes. The conversion process uses decimal encoding because it resembles the values in a cell where an empty cell will be converted to gene 0, and a cell with a value will immediately take its value as a gene. An example is based on Fig.6.

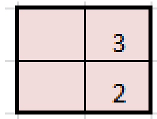


Fig. 6. Chromosome Encoding Decimal 0320

2. The results of the formation of these chromosomes represent each row, column, and region so that the genes on the chromosomes are more directed when optimized. For example, based on Fig.7.

E - ISSN : 2686-6269

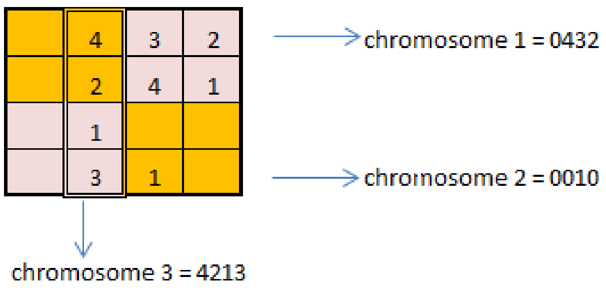


Fig. 7. Chromosome Representation Picture

3. After the chromosomes are generated, assigning the initial value of the gene to the chromosome with a value of 0 is carried out by randomizing the gene. For example, a chromosome has the 0230 gene. After initialization, it produces 1234.
4. Next is to calculate the fitness value for each chromosome using Equation (2).

$$f(x) = \frac{1}{\sum_{i=1}^{n} x_i} \tag{2}$$

5. After calculating the fitness value, the chromosomes will be sorted starting from the fitness with the smallest value because the smaller the fitness value, the worse the genes in the chromosome.
6. After sorting, mutations are carried out on the chromosomes according to that sequence. There are two types of mutations, Gene mutations and Swap mutations. Gene mutations are carried out as long as they are not stuck, while swap mutations are carried out on the gene of a chromosome with the highest mutation frequency. The gene will be exchanged with another randomly selected gene.
7. When the mutation is complete, the fitness calculation will repeat until it produces a solution. That is, each gene in the chromosome does not have the same gene in the chromosome. The pseudocode of the genetic algorithm in the Sudoku game is shown as algorithm (3).

Algorithm (3). The Genetic Algorithm In The Sudoku Game

```
Start
//determine the chromosomes
   Call InitiateRandomChromosom
Run
//Find the fitness value of each
Fitnessvalueprocess
//start chromosomes based on fitness
   Call
If stuck then
//Run mutation swap on all stuck genes
SwapMutationProcess
If not
//Do Gene Mutation On All Bad Genes
GeneMutationProcess
Done if
Repeat until a solution is
Finish
```

Based on the pseudocode shown above, it will be used to test the algorithm in a Sudoku game, which can only be finished when all the cells contain nothing. The blank cell test was performed 10 times sequentially on a 9x9 Sudoku model. The test results were carried out in mild and severe case conditions. The test results in the form of information on the length of time used are shown in Table II. From the two examples of tests carried out, it can be concluded that the completion process using this genetic algorithm goes through several stages to obtain the maximum result or solution with the least number of genes.

TABLE II
RESULTS OF SUDOKU TESTING USING GENETIC ALGORITHM

Process Testing to	Degree of Difficulty			
	Light Case		Severe Case	
	Time Required	Gens	Time Required	Generation
1	1,503	21	0,334	8
2	2,012	26	0,543	14
3	2,246	30	3,015	83
4	3,041	43	6,592	175
5	4,356	57	9,123	244
6	8,271	117	11,827	330

Process Testing to	Degree of Difficulty			
	Light Case		Severe Case	
	Time Required	Gens	Time Required	Generation
7	12,504	175	17,834	487
8	18,653	260	25,838	740
9	21	341	26,89	732
10	48,042	680	117,671	3004

IV. CONCLUSION

Based on the tests conducted to compare the work processes between genetic algorithms and backtracking, it can be concluded that both algorithms can optimally solve Sudoku game logic problems, especially in the 9x9 model. Of the 10 trials carried out, there is a difference in completion time where at a difficult level, the backtracking algorithm has a relatively faster time than the genetic algorithm. Even from a process perspective, the backtracking algorithm has shorter steps than the genetic algorithm. The backtracking algorithm can be used as an alternative Sudoku game-solving algorithm.

REFERENCES

[1] K.A Putrilani et al, “Efektivitas Media Permainan Sudoku Dalam Menghafal Huruf Kana (Menggunakan Metode Eksperimen Quasi Terhadap Siswa Japanese Club SMP Laboratorium Percontohan UPI)”, JAPANEDU, Vol. 1, No. 3, pp. 34-43, 2016.
[2] F.A. Rahman; D. Anubhakti, “Implementasi Algoritma Backtracking Pada Permainan Sudoku”, MEANS (Media Informasi Analisa dan Sistem), Vol. 5, No.1, pp. 67-71, 2020.
[3] D.S.Rahayu et al, “Evaluasi Algoritma Runut Balik dan Simulated Annealing pada Permainan Sudoku”, Jurnal Teknik Informatika dan Sistem Informasi, Vol. 3, No. 1, pp. 169-178, 2017.
[4] K. Krisnandi; H. Agung, “Implementasi Algoritma Genetika untuk Memprediksi Waktu dan Biaya Pengerjaan Proyek Konstruksi”, Jurnal Ilmiah Fifo, Vol. 9, No. 2, pp. 90-97, 2017.
[5] V.S. Widjaja; D.Z. Sudirman, “Implementasi Algoritma Backtracking Dengan Optimasi Menggunakan Teknik Hidden Single Pada Penyelesaian Permainan Sudoku”, Seminar Nasional Aplikasi Teknologi Informasi (SNATI), Yogyakarta, 15 Juni 2013, pp. 1-5, 2013.
[6] C. Danuputri; N. Santosa, ““Aplikasi Pemecahan Soal Sudoku dengan Metode Backtracking” , Jurnal Informatika Universitas Pamulang, Vol. 6, No. 3, pp. 506-511, 2021.
[7] A. Yusuf; Hendra, “Penyelesaian Puzzle Sudoku Menggunakan Algoritma Brute Force Dan Backtracking”, Jurnal Techno Nusa Mandiri, Vol.X, No.1, pp. 207-215, 2013.
[8] Herimanto et al, "An Implementation of Backtracking Algorithm for Solving A Sudoku-Puzzle Based on Android", Journal of Physics: Conference Series. https://doi.org/10.1088/1742-6596/1566/1/012038, pp. 1-6, 2019.
[9] F. Utama, et al , “Implementasi Backtracking Algorithm Untuk Penyelesaian Permainan Su Doku Pola 9x9”, Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer, Vol. 11, No. 1,pp. 29–36. https://doi.org/10.30872/jim.v11i1.200, 2017.
[10] Afriyudi et al, “Penyelesaian Puzzle Sudoku menggunakan Algoritma Genetik”, Prosiding SNASTI, ISBN 978-979-89683-31-0.
[11] A.A. Hanafi et al, ”Penyelesaian Permainan Sudoku Menggunakan Algoritma Backtracking Berbasis Artificial Intelligence”. Jurnal ICTEE, Vol. 2, No. 2, E-ISSN : 2746-7481, Hal. 50-57, 2021.
[12] Teneng et al, “Penerapan Algoritma Backtracking Pada Permainan Math Maze”, Jurnal Informatika, Vol. 6, No 1, pp. 56-67, 2010.
[13] H.S. Sulun; R. Muni, “Pembangkit Teka-Teki Silang Dengan Algoritma Backtracking Serta Aplikasi Permainannya Yang Berbasis Web”, Jurnal Informatika, Vol.4 No.2, pp. 457-466, 2010.
[14] H.K. Tupan et al, “Optimasi Penempatan Load Break Switch (LBS) pada Penyulang Karpan 2 Ambon menggunakan Metode Algoritma Genetika”, Jurnal EECCIS, Vol. 11, No. 1,pp. 1-8, 2017.
[15] S.C. Sumarta, “Pengaruh Pengaturan Individu Proses Crossover Dan Mutasi Algoritme Genetika Pada Kasus Traveling Salesman Problem”, Jurnal Tematika, Vol. 4, No. 2, pp. 83-89, 2016.
[16] M.E. Riadi, “Penggunaan Media Puzzle Untuk Meningkatkan Hasil Belajar Siswa Pada Tema Lingkungan Kelas Ii Sdn Jajartunggal III Surabaya”, JPGSD, Vol. 02, No. 02, pp. 1-11, 2014.
[17] N. Nari et al, “Penerapan Permainan Puzzle Untuk Meningkatkan Kemampuan Membilang”, Jurnal Pembangunan dan Pendidikan: Fondasi dan Aplikasi, Vol.7, No. 1, pp. 44-52, 2019.
[18] S. Sauri et al, “Pembuatan Media Pembelajaran Sudoku Puzzles Pada Materi Sistem Periodik Unsur” Gunung Djati Conference Series Seminar Nasional Tadris Kimiya 2020, Vol. 2, pp. 172-186, ISSN: 2774-6585, 2021.