Available *online* at : http://bit.ly/InfoTekJar

InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan

ISSN (Print) 2540-7597 | ISSN (Online) 2540-7600

Algorithm Strategy

Improving Performance of The Genetic Algorithm on NP-Complete Problem

Herimanto [1], Muhammad Zarlis [2], Syahril Efendi [2]

[1]*Master Programme in Informatics, Universitas Sumatera Utara*
[2]*Faculty of Computer Science and Information Technology, Universitas Sumatera Utara*

KEYWORDS

Algorithm Strategy, Genetic, Optimization

CORRESPONDENCE

Phone: +62 852 65124 310

E-mail: m.zarlis@usu.ac.id

A B S T R A C T

Non-Deterministic Polynomial Complete Problem is the most challenging problem and also engaging in algorithm strategy. One representation of this problem is the sudoku numbers game. To fill an empty sudoku puzzle, a specific formula does not apply, but filling in sudoku is a matter of decision. So it takes a special algorithm and strategy to solve it. As such, the case of the sudoku numbers game has been widely praised as the topic of finding the best results. One of the methods used is a genetic algorithm. However, due to many processes and data used in the implementation of genetic algorithms, the results obtained are often not optimal. This research will introduce a special strategy in implementing genetic algorithms in NP-Complete problems, namely by optimizing the genetic algorithm in the process of population formation. From the test results, it is found that the application of the genetic algorithm with optimization results in smaller time data and test data compared to the algorithm without optimization.

INTRODUCTION

One thing that is quite interesting in computer science is that there are so many cases in research in this field. There are approximately 14 research areas in this field. One of them is the case in the algorithm strategy, namely the Non-Deterministic Polynomial Complete Problem or commonly known as NP-Complete. This case has become very hot because there have been many studies that have raised this topic, and many researchers have proposed a solution method but so far have not found optimal results. [1]

Briefly, cases classified as NP-Complete are cases that cannot be solved mathematically or with exact calculations. According to Kanneth in the journal article Proceedings Of the Third International Conference On Genetic Algorithms entitled "Using Genetic Algorithms to Solve NP-Complete Problems", Kanneth openly admits that the NP-Complete problem is a complex and challenging problem to solve in algorithmic strategies. NP-Complete issues are considered "difficult" because they cannot be solved in polynomial time. [2]

An example of a case in an NP-Complete is graph coloring problems, n-puzzle, Knapsack problem, Subgraph isomorphism problem, and many more. One of the best examples to represent the NP-Complete case is the number game sudoku. This puzzle game is quite complicated and is one of the games in artificial intelligence because finding the solution of numbers entered into the sudoku matrix cannot only be done by mathematical calculations or specific formulas. There needs to be a special strategy and efficient algorithm to find a solution to a sudoku puzzle. To solve the sudoku case which is an NP-Complete problem, a suitable algorithm is needed. So far, several algorithms have been widely applied in finding the best solution, such as brute force algorithms, backtracking, ant colony optimization to genetic algorithms. The genetic algorithm has advantages that distinguish it from other optimization methods, namely this algorithm can perform optimization with complex problems and vast search space, making it suitable for complex issues such as NP-Complete. [3]

In this study, the authors offer a genetic method with limitations. Each population formation process applies elimination, namely numbers that already exist in each row, column, and block are automatically eliminated from the initial step and are not included in the candidate population. With this optimization method, it is expected that in subsequent processes such as chromosome formation, selection, crossover to mutation, fewer numbers will be processed to minimize the number of generations and reduce computation time.

https://doi.org/10.30743/infotekjar.v6i1.3909

LITERATURE REVIEW

NP-Complete Problem

NP-Complete Problem is the most challenging and exciting problem in algorithmic strategy. This problem has been widely praised as a topic in research, both in finding the best algorithm to solve it and comparing several algorithms. The NP-Complete Problem or known as the Non-Deterministic Polynomial Problem is a problem that mathematical calculations or exact calculations cannot prove. There needs to be a special algorithm to solve this problem, so it takes accuracy and an efficient algorithm. [3]

Genetic Algorithm

A Genetic Algorithm (Genetic Algorithm) is a search algorithm used in computing to find exact solutions or approximations to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a special class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biologies such as inheritance, mutation, selection, and crossover (also called recombination). [4] [5]

Genetic algorithms are stochastic meta-heuristics that mimic some features of natural evolution, or in another sense mimic the evolutionary process and the idea of survival of the fittest. Starting with a randomly generated population of solutions the better individuals are more likely to be selected for recombination into the new solution. The more fit a solution, the more likely it is to pass the information on to the next generation of solutions. In addition to recombining completions, new solutions can be formed by randomly mutating or swapping old solutions. Some of the best solutions from each generation are saved while others are replaced by newly formed solutions. This process is repeated until the stopping criteria are met. [6] [7]

In each generation, the fitness of each individual in the population is evaluated, several individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Generally, the algorithm stops when the maximum number of generations has been generated, or a fitness level for the population has been reached. If the algorithm has stopped because the maximum number of generations has been reached, a satisfactory solution may be reached and may not have been reached. [8] [9]

Evolution Cycle Of Genetic Algorithm

1. *Initial Population*
   The process starts with a set of individuals, where each individual is a solution to the problem to be solved. Each individual is characterized by a set of parameters (variables) called genes. Genes combine into a string to form a chromosome.
2. *Fitness Function*
   Fitness function shows how to fit an individual is (fit: individual's ability to compete with other individuals). In this stage, each individual is given a fitness score. The probability that an individual will be selected for reproduction is based on its fitness value.
3. *Selection*
   This stage is to select individuals who have the best fitness to pass their genes on to the next generation. Two individuals (who are considered as parents) will be selected by looking at their respective fitness values. Individuals with high fitness values have more room to reproduce.
4. *Crossover*
   Crossover is a phase where two individuals will be determined by the same cut-off point, then exchange some of the gene/allele values of individual 1 with the value of gene/allele on individual 2 with a predetermined crossover point limit to produce individuals with new genes and fitness values.
5. *Mutation*
   The mutation process is carried out to replace the values of several selected genes in the population to produce a better fitness value. The rules in this mutation process are determined by the probability mutation, which is 10% of the total number of genes.

RELATED WORKS

A. B. H. Arabi, (2016) in his journal article entitled "Solving NP-Complete Problems Using Genetic Algorithms" found that genetic algorithms are suitable for use in NP-Complete problems. From 3 tests it was found that the genetic algorithm is faster with lower complexity than the Exhaustive Search Method (ESM) algorithm. [10]

B. F. Gerges, G. Zouein, and D. Azar, (2018) in a journal entitled "Genetic Algorithm with Local Optima Handling to Solve Sudoku Puzzles" conducted a new method in genetic algorithms by modifying crossover and mutation operators. They call this approach the “purge approach”. This modification is claimed to optimize the performance of the genetic algorithm because it can avoid local optima. [11]

C. Afriyudi (2008) in a journal entitled "Solving Sudoku Puzzles using Genetic Algorithms" found that genetic algorithms to solve sudoku puzzles require a lot of time and iterations (generations). This is because the sudoku problem requires the genetic algorithm to achieve perfect fitness in finding a solution. [12]

METHOD

The methodology is a part of writing that explains the steps and stages that will be carried out in this research.

Data Used

The data used in this study are numbers consisting of numbers 1 to 9 which are generated randomly in the form of genes, where these genes are components of chromosomes whose length is equal to the number of empty puzzles in sudoku.

1	2	3	4	5	6	7	8	9

Figure 1. Data Used

Population Formation Process

The population is a series of random numbers consisting of the numbers 1 to 9 as a candidate solution for the sudoku number game and its length is based on the number of empty puzzles in the sudoku number game. For more details can be seen in figure below.

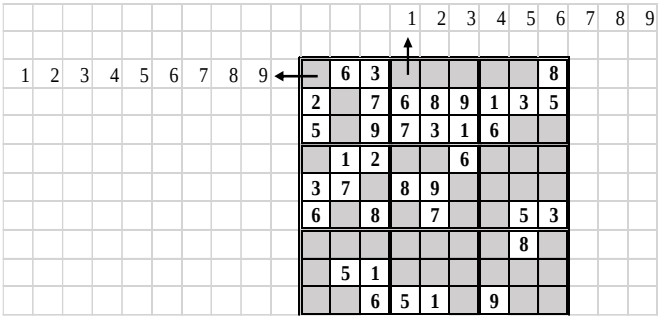

Figure 2. Population Formulation Process

The population is obtained from all possible numbers that can be filled in each empty puzzle. For example puzzle (1,1) in general the numbers that can be filled are 1 to 9, and so on.

From Figure 2, there are 45 empty square sudoku puzzles, thus the length of the chromosome is 45. In this study, 1 population consists of 9 chromosomes, so the population can be formed as follows.

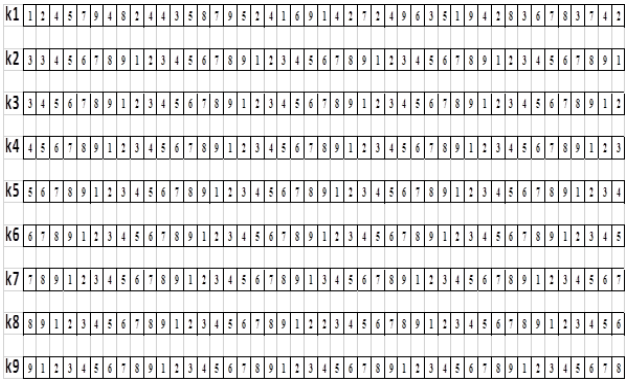

Figure 3. Chromosomes

One of the chromosomes will be selected with the most perfect fitness to be inserted into an empty sudoku puzzle as a solution to the puzzle.

Calculating the fitness value is obtained by taking the value of the genes or alleles of each chromosome to be inserted into an empty puzzle, then it will count how many numbers are repeated in each row, column and block.

For more details can be seen in the following image

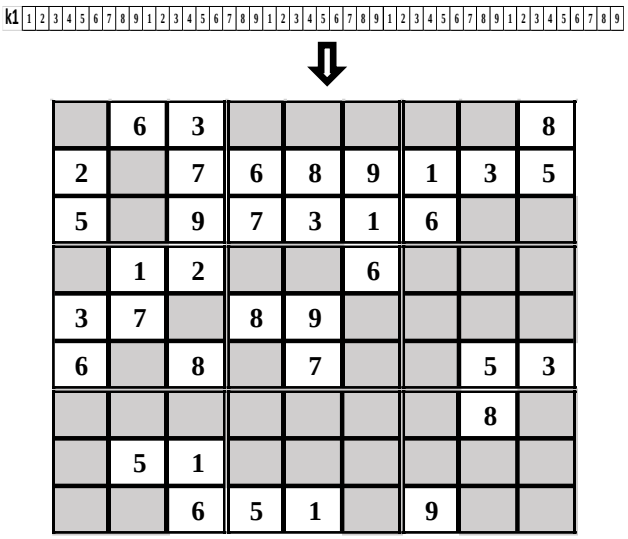

Figure 4. The Process Of Filling The Puzzle

The value of the chromosomes is entered into the blank puzzle in Figure 4, so that it becomes as follows:

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

Figure 5. Filled Puzzles

After the puzzle is completely filled with the values of the chromosome genes, the next step is to find and count the numbers that are repeated in each row, column and puzzle block. For more details shown in the following image.

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

Figure 6. The Complete Puzzle

https://doi.org/10.30743/infotekjar.v6i1.3909

On Figure. 6 shown numbers marked in red means that the number is repeated and not unique, because it has the same pair in one of the rows, columns or blocks. The next thing to do is to count the numbers in red, so that from chromosome 1 we get:

Repeated number = 33
Fitness value = number of chromosome lengths - number that repeats
= 45 – 33
= 12

So the fitness value for chromosome 1 is 12.

RESULT AND DISCUSSION

In this section, testing is carried out using conventional genetic algorithms and genetic algorithms using optimization.
The results obtained by both methods are as follows:

Table 1. Test Result

Method	Gene	Chromosome	Population	Generation
Konventional	405	9	1	50
Optimization	135	9	1	12

In table 1. the results of the comparison of the test data above are clear how the differences between the two methods are. The genetic method without limits on the number of genes and generations is far above the genetic method with limitations/optimization. For more details, see the following graphic illustration.

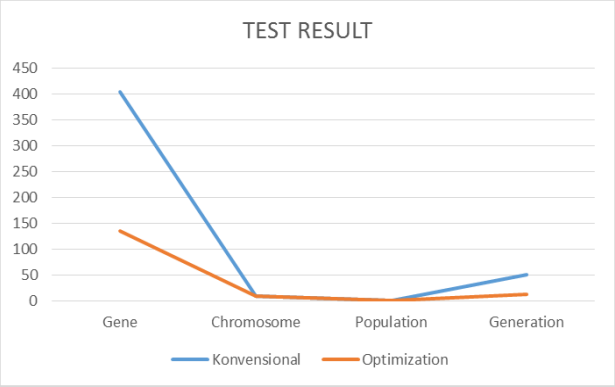

Figure 7. Test Result Graph

Based on the graph presented Conventional genetic algorithms require a much higher number of generations than genetic algorithms with optimization. This is caused by the process of formation of chromosomes and the large number of genes so that the process that is passed becomes very large and this requires a long time.

From table 1. which is presented shows the genetic algorithm conventionally requires up to 50 iterations to find the best fitness, while the application of the genetic algorithm with optimization is at least 12 iterations to find the best fitness

With the application of genetic algorithms using optimization, namely the limitations on the population formation process, the problems caused by the application of conventional algorithms have been resolved.

CONCLUSIONS

The application of constraints on the algorithm to reach the destination chromosome in the NP-Complete problem proved to be better in terms of time and computation. From the tests carried out to find perfect fitness it takes 50 iterations/generation in conventional genetic algorithms, while using the limit it takes 12 generations. The process carried out by each method is determined by the number of genes and chromosomes formed, the more genes the process will take longer, as well as the chromosomes.

But in this study the application of genetic algorithms to the NP-Complete case only solves the sudoku case that already has initial numbers and is not an empty puzzle or looking for a solution from the beginning. for the next research can consider the application of genetic algorithms on empty sudoku without initial numbers.

REFERENCES

[1] E. Utami, J. Istiyanto, and S. Raharjo, Metodologi Penelitian Pada Ilmu Komputer. Yogyakarta,: Seminar Nasional Teknologi 2007 (SNT 2007), 2007.
[2] K. De Jong, J. William, and W. Spears, “Using Genetic Algorithms to Solve NP-Complete Problems,” Proceedings Of The Third International Conference On Genetic Algorithms, vol. 3, pp. 124–132, Jul. 1998.
[3] D. I. R. Munir, “Teori P, NP, dan NP-Complete,” Bahan Kuliah IF2211 Strategi Algoritma Program Studi Teknik Informatika ITB, p. 36, 2018.
[4] A. Milton and C. Ortega-Sanchez, “Development and analysis of genetic algorithms: Sudoku case study,” in TENCON 2012 IEEE Region 10 Conference, Cebu, Philippines, Nov. 2012, pp. 1–6, doi: 10.1109/TENCON.2012.6412205.
[5] D. J. M. Weiss, “Genetic Algorithms and Sudoku,” Midwest Instruction and Computing Symposium, vol. 1, no. 4 April 2009, p. 9, 2009.
[6] N. Thirer, “About the FPGA implementation of a genetic algorithm for solving Sudoku puzzles,” in 2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel, Eilat, Israel, Nov. 2012, pp. 1–3, doi: 10.1109/EEEI.2012.6377058.
[7] T. Mantere and J. Koljonen, “Solving, rating and generating Sudoku puzzles with GA,” in 2007 IEEE Congress on Evolutionary Computation, Singapore, Sep. 2007, pp. 1382–1389, doi: 10.1109/CEC.2007.4424632.
[8] Y. Sato and H. Inoue, “Solving Sudoku with genetic operations that preserve building blocks,” in Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, Copenhagen, Denmark, Aug. 2010, pp. 23–29, doi: 10.1109/ITW.2010.5593375.
[9] M. Mitchell, An introduction to genetic algorithms, 7. print. Cambridge, Mass., 1999.
[10] B. H. Arabi, “Solving NP-complete Problems Using Genetic Algorithms,” in 2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation

(UKSim), Cambridge, United Kingdom, Apr. 2016, pp. 43–48, doi: 10.1109/UKSim.2016.65.

[11] F. Gerges, G. Zouein, and D. Azar, “Genetic Algorithms with Local Optima Handling to Solve Sudoku Puzzles,” in Proceedings of the 2018 International Conference on Computing and Artificial Intelligence - ICCAI 2018, Chengdu, China, 2018, pp. 19–22, doi: 10.1145/3194452.3194463.

[12] Afriyudi, Anggoro Suryo Pramudyo, and M. Akbar, “Penyelesaian Puzzle Sudoku menggunakan Algoritma Genetik,” 2008, doi: 10.13140/RG.2.1.4921.6726.