Dynamic path planning using a modified genetic algorithm

Awang Hendrianto Pratomo [a,1,*], Oyas Wahyunggoro [b,2], Hendri Himawan Triharminto [c,3]

[a] Department of Informatics Engineering, Faculty of Industrial Engineering, Universitas Pembangunan Nasional "Veteran" Yogyakarta, Jl. Babarsari No. 2, Tambakbayan, Depok, Sleman, Yogyakarta, 55281, Indonesia
[b] Department of Electrical Engineering and Information Technology, Engineering Faculty, Universitas Gadjah Mada, Jl. Grafika No. 2, Yogyakarta, 55281, Indonesia
[c] Department of Electrical Engineering, Indonesian Air Force Academy, Jl. Laksda Adisucipto, Yogyakarta, Indonesia
[1] awang@upnyk.ac.id; [2] hhtriharminto@aau.ac.id; [3] kanghimawan@gmail.com
* corresponding author

ARTICLE INFO

Article history
Received August 10, 2021
Revised June 24, 2023
Accepted July 01, 2023
Available online August 31, 2024

Keywords
Path planning
Genetic algorithm
Initial population
C-Obstacle
Crossover operator

ABSTRACT

The genetic algorithm (GA) is a well-known algorithm that finds feasible path planning, which can be defined as a global optimum problem. The drawback of GA is the high computation due to random processes on each operator. This research proposed that the new initial population should be integrated with a new crossover operator strategy. The parameter is the length of distance traveled by the robot. Before employing the crossover operator, a c-obstacle was generated. The c-obstacle is used as a filter to reduce unnecessary nodes and decrease time computation. After that, the initial population has been determined. The initial population is divided into two parents whose parent's chromosome contains an initial and goal position. The second parents are fulfilled with nodes from each obstacle. The genes of chromosomes will be added with c-obstacle nodes. The Crossover operator is applied after filtering, and the c-obstacle of possible hopping is determined. The filtering method is used to remove unnecessary nodes that are part of the c-obstacle. The fitness function considers the distance from the last to the next position. The optimum value is the shortest distance of path planning, which avoids obstacles in front. The aim of the proposed method is to reduce the random population and random operation in GA. By using a similar data set from previous research, the modified GA can reduce the total generation and yield an adaptive generation number. This means that the modified GA converges faster than the other GA methods.

1. Introduction

Nowadays, an autonomous vehicle reminds us of open research for helping human beings in daily activities. Moreover, autonomous vehicles can be used to help people mitigate natural disasters such as landslides. Thus, it has advantages for reducing human risk. Therefore, the development of autonomous vehicles has been interesting for many researchers, including in hardware and software areas. Nano and micro-material technology used for supporting hardware requirements are grown to compensate for size limitations with a high performance [1]. On the other hand, software capability cannot be separated from hardware devices. Control systems and intelligence systems must be provided for the autonomous vehicle.

Path planning is one of the parts of an intelligence system that guides a robot to reach its goal. The main issues of path planning are feasibility, computational complexity, global optima, and adaptability.

Adaptability relates to dynamic and static environments. Feasibility means that the path planning must fulfill kinematic constraints and reach the global optima with minimum computational complexity. High computational complexity will affect hardware requirements.

Many researchers used a certain approach to solve path-planning problems. The first approach was a grid-based algorithm. Some methods of the grid-based algorithm were A*, D-star, and Greedy algorithm [2]–[4]. These methods were approached using global methods to find feasible paths in the workspace [5]. Although the algorithm constructed a feasible path, it had a very expensive computation [6]. Therefore, normally, the approaches dealt with static environments.

Artificial Potential Field (APF) is one of the well-known approaches based on obstacles such as repelling force sources and goals such as attracting force sources [7], [8]. The algorithm is appropriate for real-time implementation, for it only requires local gradient information without requiring global information [9]. The main disadvantage of the potential field method is the local optima obtained from the total potential of repulsive and attractive forces.

The other solution to the path planning problem is evolutionary computation. One of the examples is particle swarm optimization (PSO) [10]. The inspiration for the social behavior of flocking birds seeking food is that the solutions to the optimization problem are the birds that are called particles. The optimization process is the progressive movement of the particles while they seek food. Each particle has its velocity and is computed by the fitness function. All the particles will move until the optimal or near-optimal solutions are obtained. Although the traditional PSO algorithm has the ability to solve the optimization problem, it encounters some disadvantages, such as premature convergence and stochastic stagnation [11]. The other approach to evolutionary computation is ant colony optimization (ACO) [12]–[14]. The ant's behavior within its pheromones when searching for a food source is the idea behind the method. In the initial stages of the absence of pheromone guidelines, the path is in random search which ants will have the same probability to all paths. Regarding the distance of the path, the shortest path will have higher concentrations on the pheromone than long path. Due to the search space is very wide, the disadvantages of ACO are that ACO converges into local best solution and needs long search time to solve a problem [15].

The research of Ajmal Deen Ali et al. found the effectiveness of genetic algorithms (GAs) on the study of collision free path planning that compares with conventional A* [16]. The result yields GA has a better performance both in the distance traveled and in computation time [17], [18]. Chen and Zalzala compared the GA with the modified A* in mobile robot path planning [19]. The result showed that the modified A* method obtained less time complexity than the GA but fall into some local optima. The other disadvantage is that A* algorithm has difficulty solving problems with multiple constraints. Moreover, this algorithm is based on the cost map, which is a very time-consuming task and effect to slow execution [20]. On the other hands, probabilistic optimization approach based on GA always generates global optimum or near global optimum solutions [21]. Although some of the researches proposed techniques in GA [22]–[24], but GA have lack of disadvantages, i.e. computationally expensive, requires large memory spaces when dealing with dynamic and large sized environments, and time consuming [25]. Lee and Kim use DAG (Directed Acyclic Graph) to generate multiple path as offspring candidates of feasible path. The lack of GA occurs because of the random selection of population and operators. Nazarahi et al. uses APF to determine all feasible path as initial population. This hybrid method increase the time complexity due to the application of APF algorithm [12]. Thus, this research has an objective to reduce the random process of GA and proposed several operators to increase time convergent significantly.

In order to elaborate the proposed method, the modelling of environment including GA representation and the main contribution of GA improvement are derived and presented in second section. The next part brings how to proof the concept and how do the results can be obtained. The comparison with previous method is tested as well. The last section gives a conclusion and opportunity for the future work.

2. Method

2.1. Material

Actually, path planning algorithm aims to set the equilibrium point of control system while considering safety and distance factor. Path planning algorithm constructs feasible path which means the shortest path and avoids the obstacle in front during movement.

The research was conducted using simulation. Thus, a computer or laptop with supported software is the main research material to obtain the simulation data. The material, which is a fundamental part of the research methodology, is explained below.

A grid-based model representation of the environment space makes it easier to calculate the distance and determine a position [26]. The grid-based assumption can be represented in two ways, i.e., by an orderly number or by coordinates that are based on the number of dimensions of space. For instance, for two dimensions of space, the grid-based environment is divided into the x and y-axis. Due to the simplicity of the grid-based method, this research uses the method to model the environment, as seen in Fig. 1.

Based on Fig. 1, in the scenario, the robot has two poses, x and y, that determine the position in the space. The area space has a certain size that is assumed to be the size of the robot environment. Initial and goal positions are set based on the scenario. Some of the obstacles occupied certain areas of the environment. The robot moves along the environment from the initial to the goal position. While the robot was traveling in the environment, it identified and avoided the obstacles in front of it. In this research, for simplicity, the obstacle has a structured shape and has been set in static conditions. The robot has been given prior knowledge of the environment, which means all the information is global.

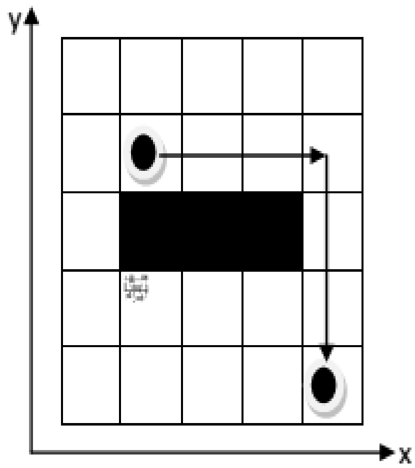


Fig. 1. Example model of the environment

The main components of GA are a genetic representation of the solution domain, an operator to generate a new solution in a search space, and a fitness function to evaluate the solution domain. The main components are explained as follows:

2.1.1. Chromosome of GA

A chromosome of the GA is one of the candidate solutions for path planning problems. The chromosome is divided into two types. The first type is filled with genes that consist of a starting node, a goal node, and one or more hoping nodes. The hoping nodes are waypoints in between the starting and goal nodes. The second type consists of obstacle points. Generally, although binary code is used by some research in the path planning area, based on a decimal-coded string, it has less memory and less space in optimization [27]. Thus, this research involves decimal codes to represent the chromosome of GA, which can be depicted in Fig. 2.

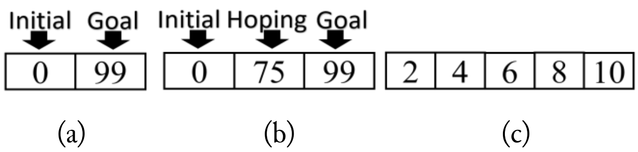


Fig. 2. 2a: The example of the parent's chromosome that contains the initial and goal node, 2b: The example of the parent's chromosome that contains the initial, hoping, and goal node, 2c: The example of supporting parent that contains the obstacle's position

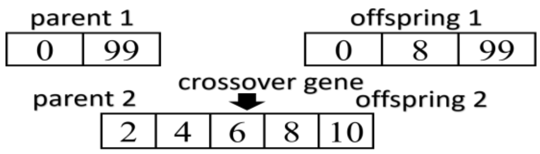
Fig. 2a shows an example of the chromosome with 0 as the initial position and 99 as the goal position. The waypoint is shown in Fig. 2b, as shown in (75). The other chromosome that fulfilled with obstacle position as in Fig. 2c. Fig. 2c explains the obstacles appear at positions 2, 4, 6, 8, and 10, which means the robot must avoid those positions in robot space

2.1.2. Parent Selection

Roulette wheel is a conventional GA method to select parents from a population [19], and the population is generated randomly. In the roulette wheel, each individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, individuals are chosen using stochastic sampling, and individuals with bigger fitness functions will be selected more times. The random process of the population obtains two possible conditions, i.e., feasible and infeasible solution. The feasible and infeasible solution can be improved by using a GA operator. For an infeasible solution, sometimes the solution is far away from the global optimum, and then it will take more time complexity to find the optimum solution [28]. On the other hand, the random process reduces the performance of GA in finding the global optimum solution. One of the contributions of this research is a new selection parent, which avoids the random process. The parent is determined from the initial condition.

2.1.3. Crossover and mutation operator

The idea behind the crossover operator is a combination of two parents to obtain two offspring. Normally, the probability of crossover is [0,1] [19]. This study applied single-node crossover, which swapped the genes of two chromosomes, as seen in Fig. 3. It should be noted that the crossover operation is for one parent and not for both parents. Before the crossover operator was applied, a filtering method was generated to reduce the possibility of obtaining a worse result than the parents. The filtering method will delete one or more chromosome genes that fulfill an infeasible path



(a) (b)

Fig. 3. (a) The example of crossover gene (b) The result of crossover operator

Generally, mutation is a process of random small changes in a gene. Mutation has a smaller probability than crossover to avoid significantly obtaining a change in an individual. The probability of mutation operator is one percent [19]. The mutation operator aims to increase diversity and avoid the local optima.

Regarding the mutation operator, in this research, only one parent has a possibility for the mutation operation. The chromosome had to undergo mutation to obtain a better individual. This process implemented the idea of a mutation operator [5]. That simple idea considered the genes, which are values from a certain gene's surrounding position, and calculated the fitness function.

2.1.4. Fitness Function

The fitness function is the key of GA to finding an optimum solution for path planning from the start to the target node. The optimal path should be the shortest and safest path. In the path planning problem, the fitness function is a formula to find the shortest path that can be modeled as (1).

$$f = \sum_{i=1}^{n-1} d(p_i, p_{i+1}), \tag{1}$$

Where

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{2}$$

Variable p_i is the ith genes of the chromosome, n is the length of the chromosome, d is the distance between two points, x_i and y_i are robot current pose in x and y axis, x_{i+1} and y_{i+1} are the next robot pose in x and y axis. The direction of the robot can be obtained by using the formula in (3).

$$\propto = tan^{-1} \frac{(y_{goal} - y_i)}{(x_{goal} - x_i)} \tag{3}$$

Pose of y_{goal} and x_{goal} is the destination position. From (1), it is shown the objective function is defined as the sum of distances between each node in a path. The shortest distance path with obstacle avoidance is the best individual for a global optima solution.

2.2. Method

The modified GA is applied to all the operators of basic GA, but it is added using the c-obstacle and filtering method. The proposed GA is depicted in Fig. 4. The modified GA starts with a common method, i.e., generating population and selecting parents.

The c-obstacle step generated the safety distance surrounding the obstacle that considers the degree of freedom of the robot. The filtering step is used to reduce the probability of obtaining an infeasible gene. The fitness function is implemented before the crossover operator. One of the differences between this proposed method and basic GA is in the fitness function. The fitness function is applied after the filtering method, or it can be said that the fitness function evaluates and selects the best gene that will

Vol. 10, No. 3, August 2024, pp. 397-409

be used for crossover operation. The Crossover operator is then used to fill the hoping node for the main parents. The last task of the GA is mutation, which refers to [3]. The result of mutation is a new offspring that will be used as one of the parents. The looping condition arises until the robot reaches its goal.

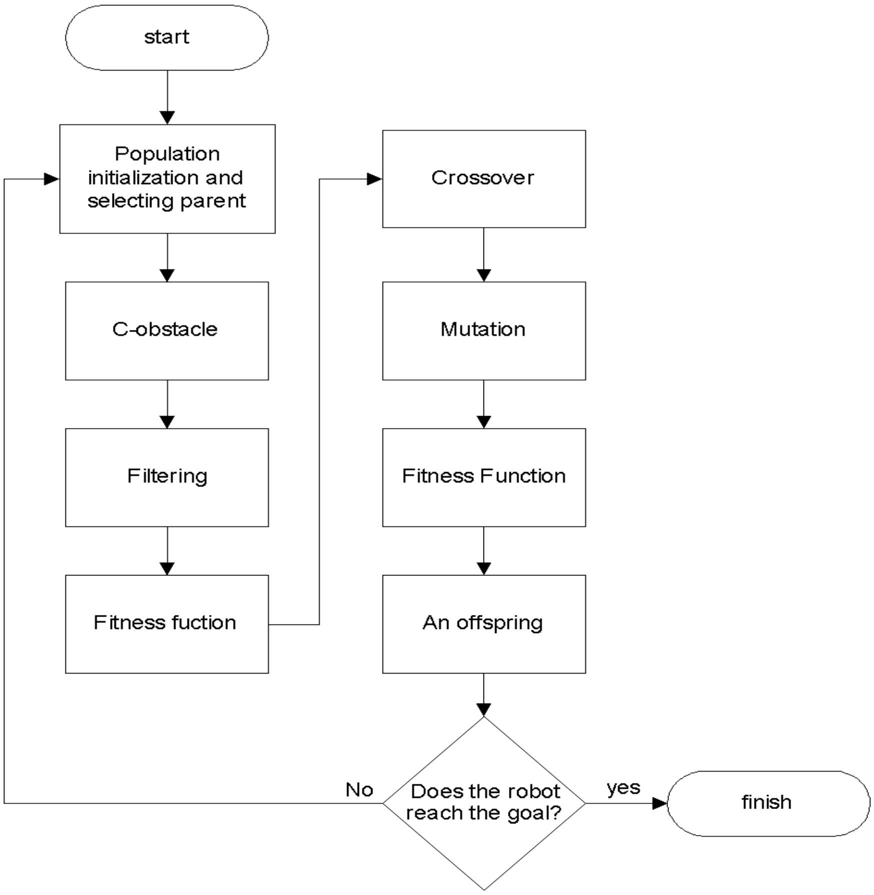


Fig. 4. Diagram block of modified GA

2.2.1. Population initialization

Normally, the population is generated randomly. In this research, the population is determined exactly. Population yields two individuals as parents. The difference between the basic GA and the population is that the population is used as the parent itself. This means that this proposed method merely generates one population without selecting parents. The member of the population equals the member of the parent, as seen in (4). The individual is divided into two parents. The first parent is called the main parent, and the second is the supporting parent. The first parent is filled with the initial node and goal node. The second parent is a flyable node surrounding the obstacle. A flyable node is a node that the robot can traverse and is obviously not part of the obstacle. The main parent is illustrated in Fig. 2a, and the supporting parent is depicted in Fig. 2c. In Fig. 2, e.g., the initial main parent, 0 is an initial node, and 99 is the goal node. On the other hand, nodes 2,4,6,8, and 10 are obstacles nodes in the space.

$$if\ t = 0\ then\ \{n_{par}\} \approx \{n_{pop}\}$$

$$if(t > 0\ \wedge (next\ node = goal\ node))\ \ then\ \{n_{par}\} \approx \{n_{pop}\}$$

$$if\,(t > 0\ \wedge (next\ node = goal\ node))\ \ then\ \{n_{par}\} \not\approx \{n_{pop}\} \quad (4)$$

Let assume n_{par} is a set of members of parent and npop is a set of members of the population. When the t equals 0, the member of a parent is similar to the member of the population. If the t is more than 0, then two conditions can be happened. The first condition occurs if the robot already meets the goal, and then the member of the parent is the same as the member of the population. This condition is called non-obstacle condition which there is no obstacle in between the robot and the goal. The second condition occurs if the robot avoids an obstacle, and then the offspring will replace the parents.

It can be concluded that the main parent is the optimum path when the robot meets a non-obstacle environment. The optimum path of a non-obstacle environment is a linear function. Thus, the global optimum solution is the main parents. For one or more environmental obstacles, the modified GA continues with the next steps, such as c-obstacle, filtering, etc.

2.2.2. C-obstacle

The c-obstacle is the simplest way to find safety nodes surrounding an obstacle. The safety node is obtained by considering the dimension of the robot, whereas in the two-dimensional case, by adding the radius of the robot dimension to the obstacle. This means that if the robot's distance against the obstacle is equal to or bigger than the radius, it will be safe. In order to generate the c-obstacle, the outer nodes of the obstacle have to expand at a certain distance. The determination of the c-obstacle is shown in the Fig. 5.

n1	n2	n3
n4	n	n5
n6	n7	n8

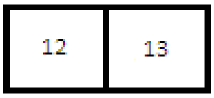
Fig. 5. C-obstacle construction

Fig. 5 shows that n is the node and n1 until n8 is the expandable node that can be determined as the c-obstacle. It was assumed that the distance of the c-obstacle was 1 unit. The formulation is depicted in (5).

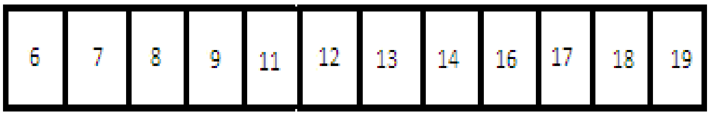
$$n1 = (n-1) - axis;\ n2 = n - axis;\ n3 = (n+1) - axis;\ n4 = n - 1;$$

$$n5 = n + 1;\ n6 = n - 1;\ n7 = n + axis;\ n8 = (n+1) + axis; \quad (5)$$

The axis is the length dimension of air space. The example of the calculation can be explained as follows. For instance, if $n = 11$ and the axis has 9 of value. In Fig. 1, by using (4), then $n1 = 1, n2 = 2, n3 = 3, n4 = 10, n5 = 12, n6 = 19, n7 = 20,$ and $n8 = 21$. The c-obstacle operation yields an expanded node for each gene. Therefore, the chromosome of the supporting parent will be combined with the original node and expanded node. The process is illustrated in Fig. 6.



(a)



(b)

Fig. 6. Original node (a) Expanded node (b)

Fig. 6a is the example of the original chromosome of the supporting parent filled with nodes 12 and 13. After the c-obstacle operation, the result in Fig. 6b obtained a chromosome that fulfills the original and extended nodes. The extended node is derived from (5).

2.2.3. Filtering

The filtering method is the main part of eliminating the possibility of obtaining the infeasible path. Before the crossover process, the modified GA will make sure that the candidate of the supporting parent is the gene that avoided an obstacle. The linear function method, which is set as the dependent variable in a certain value, evaluates the candidate genes in the supporting parent. The function can be formulated in (6) and (7).

$$x_t = x_{t-1} \pm (\cos(\propto) * \mathrm{k}) \tag{6}$$

$$y_t = y_{t-1} \pm (\sin(\propto) * \mathrm{k}) \tag{7}$$

where k was a constant that determined the desired distance at a certain time, and α was derived from (3). It was shown that if the extrapolation node equals one of the points of the obstacle, then the node is not the candidate of genes. Otherwise, if the node is not a member of an obstacle, the value is one of the candidate genes that the crossover operator will apply. The filtering method will reduce candidate genes significantly. The genes that can be chosen will have a probability of 1, and the genes with a probability equal to 0 will not be selected.

2.2.4. Crossover operator

The crossover operator in the experiment fulfilled the main parent chromosome with the candidate genes in supporting parents. The crossover is operated when the airspace is in an obstructed environment. Before the crossover operator is employed, the GA calculates the fitness objectives of each gene as modeled in (8). Therefore, the operator will select the gene with the best fitness. Since the main parent contains two genes, the crossover operator will take one gene from the supporting parent as a hoping node, as seen in Fig. 3b.

$$\{selected\ gene || \max(fitness)\} = hoping\ node \tag{8}$$

2.2.5. Mutation operator

In this study, the mutation operator implemented the method that was introduced by [5]. The mathematical model of the method is described as,

$$\{mutation\ gene || \max(fitness\ of\ n_i)\}, \tag{9}$$

where $i = 1$ to 8 and n is adjacent surrounding the node. As in (9), the concept of the mutation method simultaneously considers all the free nodes adjacent to the mutation node instead of randomly selecting a node one by one. This means that this proposed method neglects the node that is far away from the mutation node. The concept is to avoid the method not convergent at the global optimum and to increase the computational cost due to unselected random nodes. The mutation method evaluates the node according to the fitness value of the total path instead of the direction of movement through the mutated node

3. Results and Discussion

The experiments are performed to test the performance of modified GA. Parameter k on (3) is set to 1 unit. The scenario of the environment is divided into two types. The non-obstacle environment is the first type of environment, and the second type is the environment similar to the previously improved GA studied in the literature [16]. In the non-obstacle environment, the robot moves from the initial to the final destination with no obstacle. A simple non-obstacle environment is conducted, as seen in Fig. 7.

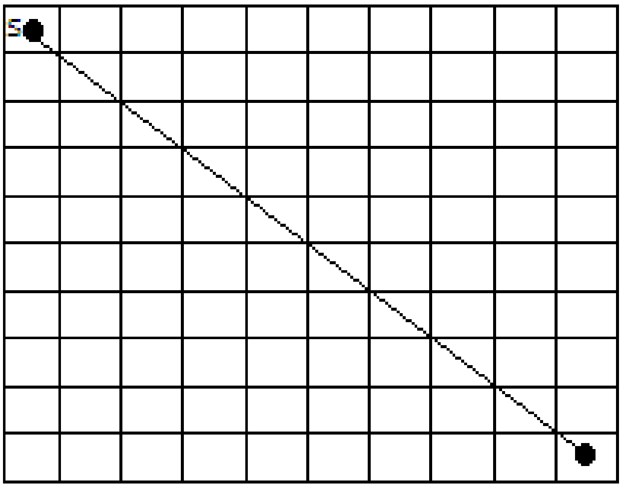
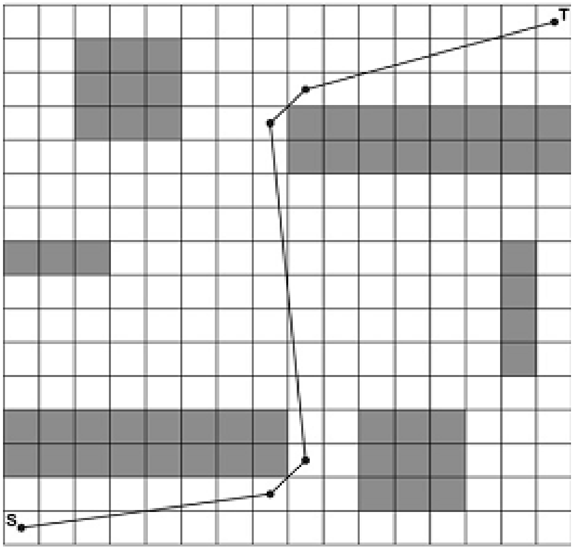


Fig. 7. The proposed method for a non-obstacle environment

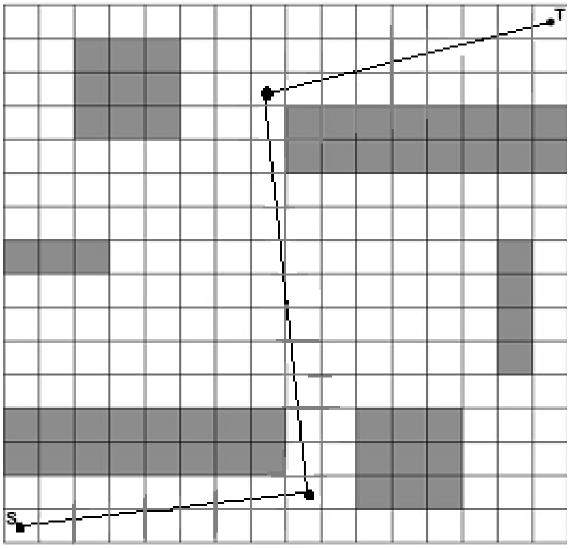
For the non-obstacle environment, since the global optimum solution of the non-obstacle environment is in linear function and the main parent can solve it, the modified GA merely needs one generation to solve the global optimal problem. In Fig. 7, the parents contain genes 1 and 100, which indicate the initial and goal position, respectively.

Fig. 8 shows the environment consists of 16x16 grids. The obstacles occur in six areas, which are shown in shaded areas. The initial node is set to 0, and the goal node is set to 15. GA runs with the proposed filtering and crossover operator. The mutation operator applied refers to [5]. The experiments give results as shown in Fig. 8a and Fig. 8b. In Fig. 8a is the method introduced by [5]. As shown in Fig. 8b, the proposed method yields a different path from the previous research. The length of the path is 28.35, which is longer than 27.82 in the previous research. Despite the path length, the generation results different number from the previous method. The comparison is shown in Table I, and it is proven that the modified GA reduces the average generation number significantly.

Conversely, the proposed method gives a more stable system according to a 100 percent optimal solution. This means that the proposed method minimizes the random process in GA and yields precise results. Therefore, it is possible to use it for online and dynamic paths.



(a)



(b)

Fig. 8. The proposed method for non-obstacle environment

Table 1. Comparison to other research works

Research Work	Indicators				*Generation Number*
	# of optimal solution	*# of near optimal solution*	*# of infeasible solution*	*Fitness value*	
Whitley [29]	3	69	28	29.25	23
Yao and Ma [30]	2	69	29	29.91	22
Tuncer and Yildirim [5]	54	44	2	27.82	11
This work	**100**	**0**	**0**	**28.35**	**2**

Although the proposed method can solve the global optimum problem, the method neglects non-holonomic constraints. Fig. 8 shows that the result is sharp and cannot be implemented for non-holonomic robots. Thus, the method needs a curve algorithm, such as a B-spline curve, to make a smooth path. The other thing is due to the grid-based method, which merely solves discrete problems and cannot solve continuous problems. Generally, in the real-time system, the problem is in the time domain, meaning it must be a continuous problem naturally.

Fig. 9 contains the generation number of optimal solutions versus the total obstacle avoidance. It can be concluded that the generation number is influenced by the obstacle that the robot should pass. It doesn't mean that the generation equals the number of obstacles; it merely depends on the obstacles that should be passed. The modified GA has an adaptive generation number compared with the other methods.

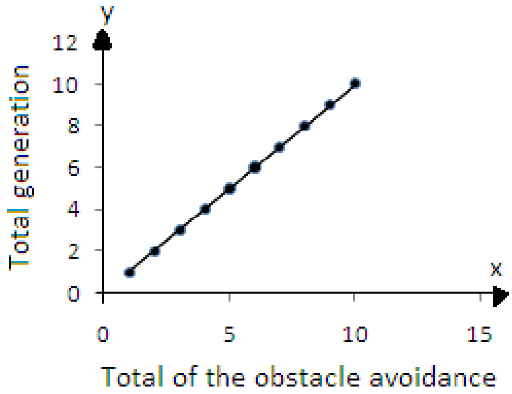


Fig. 9. Total generation vs total of the obstacle avoidance

One of the drawbacks in Fig. 9 is that the method will have high computational costs when the robot encounters many obstacles. Otherwise, this method is very effective for a small number of obstacles. Therefore, the method has the opportunity to be used in a real-time system platform.

4. Conclusion

This paper aims to improve the genetic algorithm in the path planning field. The contribution of this paper is explained as follows. Firstly, the initial population is divided into two individuals, i.e., the chromosome filled with the initial and goal node and the chromosome containing coordinates surrounding the obstacles. The coordinates surrounding the obstacles are based on the c-obstacle, which considers the robot's dimensions. The second contribution is in the filtering method. The method considers that the coordinates that yield an infeasible path will be eliminated. After that, the supporting parents will be added with the gene that can obtain the optimum path. The gene will fill the main parent as a hoping node. The supporting parents will have a constant gene in the chromosome in the method, but the main parent will change based on the needed hoping node. In conclusion, the modified GA merely yields one offspring. The mutation follows the method of [5] that checks all the free nodes surrounding the mutation node. Then, the method compares all the fitness function values and chooses the best one. The node that has the highest fitness function will be selected. In order to prove the concept, testing method is conducted by simulation using Matlab software with two kind of environments. The environments are non-obstacle environments that are similar to the previous improved GA studies in the literature. The method meets the optimum result for a non-obstacle environment due to a linear path. The result of the obstacle environment obtains a longer path for the environment than the previous GA. However, the total generation number of the modified GA is less than that of the previous method. The total generation number is influenced by the number of obstacles that the robot should pass. If the number of obstacles increases, then the total generation number will also increase. The GA has an adaptive generation number based on the total number of obstacles in the method. The drawback of the proposed method is the non-holonomic constraint. The method neglects the non-holonomic constraint and assumes the robot is a point mass and can move in any direction. Many robots cannot move without neglecting non-holonomic constraints. Thus, the opportunity that occurred from this research is to consider non-holonomic constraints and implement local information in a real-time scenario. Furthermore, this research can be implemented in a real autonomous vehicle, which is used to monitor environmental changes in disaster-prone areas, especially landslides.

Acknowledgment

We would like to thank Ministry of Research and Higher Education for supporting this work through Fundamental Research Grant schema, Universitas Gadjah Mada and Department of Informatics Engineering Universitas Pembangunan Nasional "Veteran" Yogyakarta for providing facilities.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding statement. This work is funded by Fundamental Research Grant schema from Ministry of Research and Higher Education No. 04/UN62.21/PT/IV/2019.

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] M. Nasrollahzadeh, S. M. Sajadi, M. Sajjadi, and Z. Issaabadi, “An Introduction to Nanotechnology,” in *Interface Science and Technology*, vol. 28, Elsevier, 2019, pp. 1–27, doi: 10.1016/B978-0-12-813586-0.00001-8.

[2] X. Wang, H. Zhang, S. Liu, J. Wang, Y. Wang, and D. Shangguan, “Path planning of scenic spots based on improved A* algorithm,” *Sci. Rep.*, vol. 12, no. 1, p. 1320, Jan. 2022, doi: 10.1038/s41598-022-05386-6.

[3] D. Li, L. Wang, S. Geng, and B. Jiang, “Path Planning of AS/RS Based on Cost Matrix and Improved Greedy Algorithm,” *Symmetry (Basel).*, vol. 13, no. 8, p. 1483, Aug. 2021, doi: 10.3390/sym13081483.

[4] S. Kadry, G. Alferov, and V. Fedorov, “D-Star Algorithm Modification,” *Int. J. Online Biomed. Eng.*, vol. 16, no. 08, pp. 108–113, Jul. 2020, doi: 10.3991/ijoe.v16i08.14243.

[5] A. Tuncer and M. Yildirim, “Dynamic path planning of mobile robots with improved genetic algorithm,” *Comput. Electr. Eng.*, vol. 38, no. 6, pp. 1564–1572, Nov. 2012, doi: 10.1016/j.compeleceng.2012.06.016.

[6] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” *Def. Technol.*, vol. 15, no. 4, pp. 582–606, Aug. 2019, doi: 10.1016/j.dt.2019.04.011.

[7] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and X. Liu, “Obstacle avoidance of mobile robots using modified artificial potential field algorithm,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2019, no. 1, p. 70, Dec. 2019, doi: 10.1186/s13638-019-1396-2.

[8] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, “Mobile robot path planning using membrane evolutionary artificial potential field,” *Appl. Soft Comput.*, vol. 77, pp. 236–251, Apr. 2019, doi: 10.1016/j.asoc.2019.01.036.

[9] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu, “Improved Artificial Potential Field Method Applied for AUV Path Planning,” *Math. Probl. Eng.*, vol. 2020, no. 1, pp. 1–21, Apr. 2020, doi: 10.1155/2020/6523158.

[10] M. D. Phung and Q. P. Ha, “Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization,” *Appl. Soft Comput.*, vol. 107, p. 107376, Aug. 2021, doi: 10.1016/j.asoc.2021.107376.

[11] R. Wang, K. Hao, L. Chen, T. Wang, and C. Jiang, “A novel hybrid particle swarm optimization using adaptive strategy,” *Inf. Sci. (Ny).*, vol. 579, pp. 231–250, Nov. 2021, doi: 10.1016/j.ins.2021.07.093.

[12] C. Miao, G. Chen, C. Yan, and Y. Wu, “Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm,” *Comput. Ind. Eng.*, vol. 156, p. 107230, Jun. 2021, doi: 10.1016/j.cie.2021.107230.

[13] Q. Luo, H. Wang, Y. Zheng, and J. He, “Research on path planning of mobile robot based on improved ant colony algorithm,” *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1555–1566, Mar. 2020, doi: 10.1007/s00521-019-04172-2.

[14] Q. Song, Q. Zhao, S. Wang, Q. Liu, and X. Chen, "Dynamic Path Planning for Unmanned Vehicles Based on Fuzzy Logic and Improved Ant Colony Optimization," *IEEE Access*, vol. 8, pp. 62107–62115, 2020, doi: 10.1109/ACCESS.2020.2984695.

[15] X. Zhou, H. Ma, J. Gu, H. Chen, and W. Deng, "Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism," *Eng. Appl. Artif. Intell.*, vol. 114, p. 105139, Sep. 2022, doi: 10.1016/j.engappai.2022.105139.

[16] M. S. A. D. Ali, N. R. Babu, and K. Varghese, "Collision Free Path Planning of Cooperative Crane Manipulators Using Genetic Algorithm," *J. Comput. Civ. Eng.*, vol. 19, no. 2, pp. 182–193, Apr. 2005, doi: 10.1061/(ASCE)0887-3801(2005)19:2(182).

[17] A. H. Pratomo, N. H. Cahyana, and S. N. Indrawati, "Optimizing CNN hyperparameters with genetic algorithms for face mask usage classification," *Sci. Inf. Technol. Lett.*, vol. 4, no. 1, pp. 54–64, May 2023, doi: 10.31763/sitech.v4i1.1182.

[18] A. A. Zarir, S. Bashar, and A. R. Ismail, "Automated image captioning with deep neural networks," *Sci. Inf. Technol. Lett.*, vol. 1, no. 1, pp. 17–23, Apr. 2020, doi: 10.31763/sitech.v1i1.31.

[19] Mingwu Chen and A. M. S. Zalzala, "Safety considerations in the optimisation of paths for mobile robots using genetic algorithms," in *1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, 1995, vol. 1995, no. 414, pp. 299–306, doi: 10.1049/cp:19951066.

[20] X. Zhang and X. Chen, "Solving 'Limited' Task Allocation Problem for UAVs Based on Optimization Algorithms," *Wirel. Commun. Mob. Comput.*, vol. 2021, no. 1, p. 5530454, Jan. 2021, doi: 10.1155/2021/5530454.

[21] M. M. Riyahi, A. E. Bakhshipour, and A. Haghighi, "Probabilistic warm solutions-based multi-objective optimization algorithm, application in optimal design of water distribution networks," *Sustain. Cities Soc.*, vol. 91, p. 104424, Apr. 2023, doi: 10.1016/j.scs.2023.104424.

[22] P. Gia Luan and N. T. Thinh, "Real-Time Hybrid Navigation System-Based Path Planning and Obstacle Avoidance for Mobile Robots," *Appl. Sci.*, vol. 10, no. 10, p. 3355, May 2020, doi: 10.3390/app10103355.

[23] P. G. Luan and N. T. Thinh, "Hybrid genetic algorithm based smooth global-path planning for a mobile robot," *Mech. Based Des. Struct. Mach.*, vol. 51, no. 3, pp. 1758–1774, Mar. 2023, doi: 10.1080/15397734.2021.1876569.

[24] A. K. Rath, D. R. Parhi, H. C. Das, P. B. Kumar, M. K. Muni, and K. Salony, "Path optimization for navigation of a humanoid robot using hybridized fuzzy-genetic algorithm," *Int. J. Intell. Unmanned Syst.*, vol. 7, no. 3, pp. 112–119, Jul. 2019, doi: 10.1108/IJIUS-11-2018-0032.

[25] A. Vi, A. M. Kleinnijenhuis, and J. Doyne, "Qualities, challenges and future of genetic algorithms: a literature review," *arXiv*, pp. 1–48, 2021, [Online]. Available at: https://arxiv.org/abs/2011.05277.

[26] Z. Guo and L. Mo, "Robot path planning based on improved genetic algorithm," *Front. Comput. Intell. Syst.*, vol. 2, no. 3, pp. 78–81, Feb. 2023, doi: 10.54097/fcis.v2i3.5314.

[27] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019, doi: 10.1016/j.eswa.2018.08.008.

[28] P. Jiang, Y. Cheng, J. Yi, and J. Liu, "An efficient constrained global optimization algorithm with a clustering-assisted multiobjective infill criterion using Gaussian process regression for expensive problems," *Inf. Sci. (Ny).*, vol. 569, pp. 728–745, Aug. 2021, doi: 10.1016/j.ins.2021.05.015.

[29] D. Whitley, "A genetic algorithm tutorial," *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994, doi: 10.1007/BF00175354.

[30] Z. Yao and L. Ma, "A Static Environment-Based Path Planning Method by Using Genetic Algorithm," in *2010 International Conference on Computing, Control and Industrial Engineering*, 2010, vol. 2, pp. 405–407, doi: 10.1109/CCIE.2010.220.