Safe Robot Path Planning and Obstacle Avoidance using Efficient Genetic Algorithm

Oyas Wahyunggoro[1], Hendri Himawan Triharminto[2] and Adha Imam Cahyadi[3]

[1,3]Department of Electrical Engineering and Information Technology, Faculty of Engineering, Universitas Gadjah Mada, Jl. Grafika No 2 Yogyakarta 55281, Indonesia [2]Indonesian Airforce Academy, Jalan Solo-Yogyakarta 55281, Indonesia
oyas@ugm.ac.id, adha.imam@ugm.ac.id, hendri@aau.mil.id

Abstract: One of the major drawbacks of the Genetic Algorithm (GA) is the computational complexity due to the random process at each step. A new initial population scheme integrated with a new crossover operator strategy is proposed to overcome this drawback. Before employing the crossover operation, permissible paths based on the c-obstacle concept were generated. To accelerate the convergence, the initial population was divided into two parents, i.e., the parent's chromosome containing the initial and goal positions and the parents composed of nodes from each extracted c-obstacle. Before applying the crossover operator, a filtering algorithm was performed to remove the uncorrelated offspring. A further c-obstacle inclusion made it more efficient; thus, only possible hoping nodes were considered. The random populations and random operations could be reduced efficiently using these steps. Finally, the numerical study method was tested. It is seen that the modified GA is faster and can reduce the total generation, and significantly yields an adaptive generation number.
Keywords: Robot path planning, obstacle avoidance, genetic algorithm, initial population algorithm, crossover operator

1. Introduction

Research on autonomous vehicles to assist human beings in daily activities is getting more accessible and advanced recently. One of the implications is the rapid development of autonomous vehicles as well as their hardware and software. In selecting appropriate supporting hardware, nano and micro-material technologies are developed to compensate for the size limitation with high performance [1]. On the other hand, software compatibility is inseparable from a hardware device. A control and intelligence system must be provided for this autonomous vehicle system.

Path planning is one of the intelligence system parts guiding the robot to find a path from the start to the goal point. The main issues of path planning are its feasibility, computational complexity, global optima, and adaptability. Adaptability relates to dynamic and static environments.

Many researchers used a specific approach to solve the path-planning problem. The main problem is the solution for collision-free path computation in dynamic environments [1]. The first approach is a grid-based algorithm. Some methods of the grid-based algorithm are the A* and the Greedy algorithms [2]. The approach employed for these methods was the global method to find feasible paths in the workspace [3]. Although the algorithm constructed a feasible path, it had a long computation time [4]. Therefore, normally, the approaches deal with the static environment.

The Artificial Potential Field is one of the well-known approaches based on obstacles as repelling force sources and the goals as attracting force sources [5]. The algorithm is appropriate for real-time implementation, for it only requires local gradient information without global information.

The main disadvantage of the potential field method is the local optima obtained from the total potential of repulsive and attractive forces. The superposition result cannot ensure the shape and direction of the total potential field [6]. A new approach to this algorithm is a superquadric artificial potential function proposed by Volpe and Khosla, applicable only to two obstacles at

Received: February 1st, 2022. Accepted: August 29th, 2023
DOI: 10.15676/ijeei.2023.15.3.2

most [6]. Another proposed method, which is based on a probabilistic path planner (PPP) with artificial potential fields (APF), can validate the theoretical concepts into simulations using fuzzy rules for obstacle avoidance [7]. However, the solution requires a few moments of computation time.

Evolutionary computation is another solution to the path planning problem. One of the examples is particle swarm optimization (PSO). It is inspired by the social behavior of flocking birds seeking food; the solutions to the optimization problem are the birds seeking out food, called particles. The process is as the particle's progressive moves while seeking out food. Each particle has its velocity and is computed by a fitness function. All particles would move until the optimal or near-optimal solutions are obtained. Although the traditional PSO can solve the optimization problem, it encounters some disadvantages, such as premature convergence and stochastic stagnation [5]. Another approach to evolutionary computation is ant colony optimization (ACO) [7][8][9]. The idea underlying this method is the ant's behavior within its pheromones for seeking out food. In the initial stages of the absence of pheromone guidelines, the path is in random search where ants would have the same probability to all paths. Regarding the path distance, the shortest path would have higher concentrations on the pheromone than the longest. On the other hand, this algorithm has been modified in submarine path planning [8] and hair-insertion robot arm path planning [9]. However, due to the vast search space, each individual in the ACO can converge on the local best solution, and ACO needs a lengthy search time to solve a problem [10].

The research conducted by Ali has found the effectiveness of genetic algorithms (GAs) in the study of collision-free path planning compared to a conventional A*[11]. The result indicates that GA has a better performance both in the distance traveled and in computation time. Chen and Zalzala compared the GA with the modified A* in mobile robot path planning. The result shows that the modified A* method obtains less time complexity than the GA but falls into some local optima [10]. On the other hand, the probabilistic optimization approach based on GA always generates global optimum or near global optimum solutions [11]. Long has modified the Genetic Algorithm into the simulation of path planning for an unmanned surface vehicle by manipulating fitness function, crossover probability, and mutation probability to make the convergence in the algorithm [12]. However, due to the random selection of populations and operators, GA lacks disadvantages, i.e., computationally expensive, requires large memory spaces when dealing with dynamic and large-sized environments, and is time-consuming [13].
Another work done by Kwaśniewski et al. [14] proposed GA with obstacle avoidance. Because they worked for space exploration, the obstacle has to be arbitrary. However, it needs a lot of computing power due to the nature of the GA and the necessity for high-level decision-making. Mane et al. [15] claimed to develop a better GA Algorithm. However, their methods lack the theoretical concept and proof of their effectiveness. Finally, Rahmaniar and Rakhmania [16] proposed another method based on GA for a dynamic environment. Apart from their method, which lacks a proven concept, the methods used much computing power by dividing their scenario into several steps, i.e., local search, local optimization, smoothing, etc.

Thus, this research aims to reduce the GA random process to reduce the computation time and complexity. In order to elaborate the proposed method, the environment modeling, including GA representation and the main contribution of GA improvement, is derived and presented in the second section. The next part discusses how to prove the concept and how the results can be obtained. The comparison with the previous method is tested as well. The last section gives a conclusion and opportunity for future work.

2. Proposed Method

This work focuses on making the path planning algorithm that converges to a static goal while considering safety and distance factors. In brief, the proposed path planning algorithm is built to construct a feasible path while recursively choosing the shortest path and avoiding obstacles during movement toward the goal point.

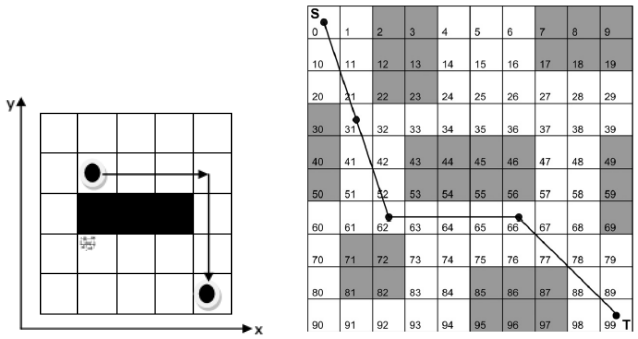
A. Environment Representation

In this paper, the environment representation was established by using a grid-based model. This representation is easy to calculate, especially when the distance or position is needed instantly. This scheme has been used extensively using many works, such as in [3]. In literature, the grid-based representation can be represented in two ways, i.e., by an orderly numbered or by a coordinates system. These choices should consider how many parameters are needed to model the robot and environment uniquely. For instance, in two dimensions of space, the grid-based environment is divided into the x and y axis only, and it neglects the information about the robot's orientation.

In this work, the grids were generated for later use when determining the feasible paths so that the numbering carried the coordinate information and served as a flag for obstacles. For that reason, the algorithm was constructed as follows:

Algorithm 1 (Grid generation)
Step 1: determine the number of grids in the horizontal and vertical directions: =(m,n)
Step 2: compute the distance between the grid in the horizontal and vertical directions: =(d_x/m, d_y/n)
Step 3: compute the center of each grid as follows For i=1 to m For j=1 to n C_x(i,j) = d_x/(2m)+ (d_x/n)*i C_y(i,j) = d_y/(2n)+ (d_y/n)*j
Step 4: generate the grid. The grids are square whose centers are (C_x(i,j) , C_yi,j)) $\forall i \in \{1, \cdots m\}, j \in \{1, \cdots, n\}$
Step 5: number the grid as follows For i=0 to n For j=0 to m-1 N(i,j)=i+j It would generate the number 0 for the grid in the upper left corner to the number n+m-1 in the grid in the lower right corner.

The illustration of Algorithm 1 when m=n=10 is presented in Fig. 1a and 1b. In Fig. 1a, the robot is assumed to be able to move in all directions, either x or y. The robot's position could be determined by its grid number. In the figure, the robot's initial and goal coordinates have been set to be the first the last grids. Moreover, obstacles in the environment are denoted by darker grids. The robot was equipped with sensors to detect obstacles as it traversed the environment between the initial and the goal positions. This work focuses on offline path planning; thus, the robot has been given prior knowledge of the environment model.



(a) (b)

Figure 1. Example model of the environment

The following algorithms were proposed to determine the grid occupied by obstacles and the permissible cell to explore.

Definition 1. For current robot position $p \in \{0, \cdots (m+n-1)\}$, the surrounding cells are defined as $p_s = \{p-m-1, p-m, p-m+1, p-1, p+1, p+m-1, p+m, p+m+1\}$. It should be noted that the surrounding cells for $p < m$ robot position were less than eight cells because some cells might lie outside the environment.

Algorithm 2 (Obstacle flag)
Step 1: Determine the initial position of the robot in the environment $p \in \{0, \cdots (m+n-1)\}$
Step 2: Mark the grid with the obstacle For i=p to (m+n-1) If Grid(i) has the obstacle Flag(p)=1;

Definition 2. The surrounding cell is considered free if and only if it does not belong to the obstacle, i.e., the flag is zero.

The algorithm was developed from Definitions 1 and 2 to mark all permissible flags. Later, this algorithm was used to choose the offspring to efficiently reduce the computational burden.

B. Path optimization using Genetic Algorithm

In this sub-section, the main components of GA, i.e., GA formalism of the solution space, GA operator to generate a new and possibly better solution from solution space, and fitness function, were applied to evaluate the solution domain to obtain better offspring. The main components are explained as follows. The binary coded chromosome, as used by literature such as in [17], [18], was not used.

Definition 3. Chromosome type 1 contains an initial cell, at most one hoping cell, and a goal cell. The chromosome is coded in number as mentioned in Algorithm 1, step 5. The hoping cell is used as a waypoint between the initial and goal nodes necessarily included in the path.

Definition 4. Chromosome type 2 contains obstacle positions.
An example of the parents' chromosome that contains the initial position and the goal node is shown in Figure 2.

Fig. 2a. shows an example of the parent's chromosome that contains 0 as the initial position and 99 as the goal position, categorized as a type 1 chromosome. It still belongs to type 1 but has 75 as the hoping point, as shown in Fig. 2b. In Fig. 2c.; the obstacle information addition can be added as supplementary genes. The other is seen that the obstacles appear at the positions of 2, 4, 6, 8, and 10.

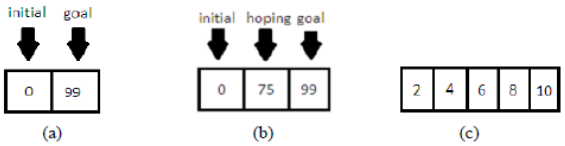


Figure 2. The example of a parent's chromosome

C. Population Initialization

Normally, the population is generated randomly. However, in this research, the proposed method was generated differently using a deterministic method for population initialization. The population was initiated by two individuals as parents. It differs from the conventional GA, where the population is used as the parent itself. Therefore, it means that the population was generated without selecting parents.

Algorithm 3 (population initialization)
Step 0: initialization p=0, initial time t=tinit, maximum time=tmax, determine start position s and goal position g, set a single initial populationn_{pop} as seen in Fig. 2a.
Step 1: Determine the initial position of the robot in the environment $p \in \{0, \cdots (m+n-1)\}$
Step 2: for p=0 to (m+n -1) If $s = g$ $then$ $break$, if flag(p)=0 then grows typ 3e 1 individual as next hoping genes by adding p into the original chromosome else grows type 2 individual by adding p as the next obstacle gene Add a new individual into$\{n_{pop}\}$ $\{n_{par}\} = \{n_{pop}\}$
Step 3: jump to step 2 until t=tmax

The used set notation denoted as {·} means the collection of individuals.

Algorithm 3 can generate all possible individuals with the type 1 chromosome as the initial population and those with the type 2 chromosome as a set of possible obstacles. However, some type 1 individuals are not suitable to be chosen as the path. It is the challenge of the developed algorithm to choose the proper ones and to decide the best path.

In Algorithm 3, $\{n_{par}\}$ is a set of parent's members while$\{n_{pop}\}$ is a set of population's members. Initially, it only has one population, as presented in Fig 2a. As computation occurs, the population number grows until the maximum allowable time is reached.

For practical consideration, when the robots and obstacles dimension is the matter and when the robot's maneuverability cannot be neglected, the concepts of c-obstacle and filtering are introduced in the following sections.

D. Adoption of C-obstacle for safe path generation

The c-obstacles concept is used in literature to make the abstraction of point mass robots possible. It enables them to concentrate the work on path planning design. Considering the degree of the robot's freedom, the safety nodes surrounding the obstacle are generated to ensure the robot's safety for not hitting the obstacles. For curious readers, please refer to other works such as in [19], [20]. The safety node was obtained by considering the two dimensions of the robot, such as adding the robot dimension radius to the obstacle. The robot would be safe if the distance between the robot and the obstacle is equal to or bigger than the radius. In order to generate the c-obstacle, the outer nodes of the obstacle had to expand at a certain distance.

Definition 5. Surrounding cells are considered the c-obstacle if at least one flag in Algorithm 2 contains an obstacle flag.

The construction of c-obstacles is shown in Algorithm 4.
Fig. 3 shows an example of the determination of the c-obstacle. If n is the obstacle, the surrounding cells are considered c-obstacle if at least 1 cell contains 1 valued flag.

Algorithm 4 (Construction of c-obstacles)
Step 1: Position of obstacle in the environment $o \in \{0, \cdots (m+n-1)\}$
Step 2: For i in 8 surrounding cells If Grid(i) has an obstacle Flag(i)=1; surrounding cells are c-obstacle Else surrounding cells are not an obstacle

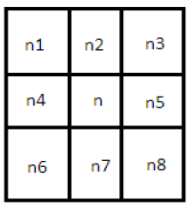


Figure 3. C-obstacle construction

When a c-obstacle was applied, a parent with a type 2 chromosome would grow in size. The example of the calculation can be explained as follows. For instance, if n=11 and the axis value is 9 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 yielded an expanded node for each gene. Therefore, the chromosome of the supporting parent would be combined with the original node and expanded node. The process is illustrated in Fig. 4.

Fig. 4a is an example of the supporting parent's original chromosome filled with nodes 12 and 13. After the c-obstacle operation, the result in Fig. 4b obtains a chromosome fulfilling the original and extended nodes. The generated c-obstacle was obtained from Algorithm 3.

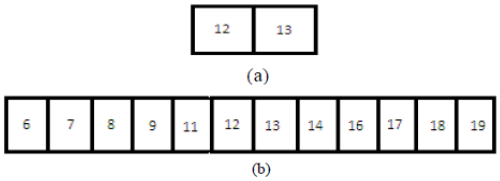


Figure 4. Original node (a) Expanded node (b)

E. Filtering

The filtering method is the main part to eliminate the possibility of obtaining the infeasible path before GA operations, i.e., crossovers and mutation, as explained later. Before the crossover process, the proposed GA would ensure that the supporting parents' candidates have the only gene that avoids an obstacle. As explained later, this would enable the fitness function to be applied efficiently. After filtering is performed, several individuals can be removed. Thus, only the ones with the best genes are valid for crossover and mutation operations. The filtering is described as a mathematical equation as follows

$$x_t = x_{t-1} \pm (\cos(\alpha) * k)$$
$$y_t = y_{t-1} \pm (\sin(\alpha) * k). \quad (1)$$

where k is a constant determining the desired distance in a specific time, and alpha is derived from (4). It is shown that if the extrapolation node equals to one of the points of the obstacle, the node is not the gene candidate. Otherwise, if the node is not a member of an obstacle, then the value is one of the candidate genes that the crossover operator would apply. The filtering method would reduce candidate genes significantly. The possibly chosen genes would have a probability of 1, and the genes with a probability equal to 0 would not be selected.

After going through processes, the last task of GA is a mutation which refers to [3]. The result is a new offspring used as one of the temporary parents, which iterates until reaching the goal position. The operation details are described in the next sections.

F. GA Operators

A roulette wheel was used to select parents from a population to enable the formal operation used in the GA. This method is conventional but powerful as also demonstrated in [21]. In a roulette wheel, each individual was represented by a space proportionally corresponding to its fitness. By repeatedly spinning the roulette wheel, individuals were chosen using stochastic sampling, and the one with the bigger fitness function would be selected more. The random process of the population obtained two possible conditions, i.e., feasible and infeasible solutions. The feasible and infeasible solutions could be improved by using a GA operator. For the infeasible solution, sometimes the solution was far from the global optimum, and then it would spend more time finding the optimum solution [22] [23]. On the other hand, the wide range of solution space would make the random process reduces the performance of GA to find the global optimum solution. One of the contributions of this research is a new parent selection by narrowing the solution space to enable the random process to become efficient. Fig. 5 is an example of a crossover gene and operator.

Crossover and mutation operator: The idea behind the crossover operator is the combination of two parents to obtain two sets of offspring. Normally, the crossover probability is a crisp set, i.e., a set with an occurrence probability of either 0 or 1 [21]. This study applied single-node crossover, which swapped the genes of two chromosomes, as seen in Fig. 5. It should be noted that the crossover operation is only for one parent. As mentioned before, the crossover operator is applied after filtering is applied. It would reduce the possibility of obtaining a worse result than the parents by omitting one or more chromosome genes with an infeasible path.

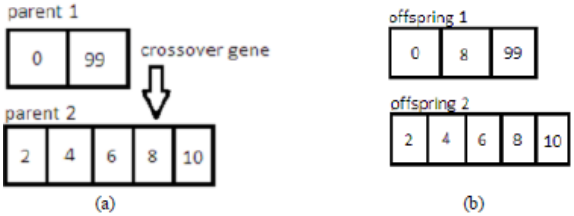


Figure 5. The example of crossover gene (a) The result of crossover operator (b)

1. Crossover operator

The crossover operator is necessary for any GA approach. In this work, the main parent chromosome with the candidate genes in supporting parents. The crossover is operated when the airspace is against obstacles.

Algorithm 5 (Crossover operator)
Step 1: parent Initialization, let $i = 1$, i-th parent $n_{par}(i) \in \{n_{pop}\}$, k=number of populations
Step 2: remove non-feasible individuals using Filtering For i=1 to k If Filter $(n_{par}(i)) = 1$, $\{n_{par}\} = \{n_{pop}\} - n_{par}(i)$
Step 3: For i=1 to k For j=1 to k If $i \neq j$ $hoping\ node(i,j) = \{selected\ gene \mid\mid max(fitness)\}$ $n_{cross} = construct\ chromosome(hoping\ node(i,j)$ $\{n_{new}\} = \{n_{par}\} - n_{par}(i) - n_{par}(j) + n_{cross}$

In Algorithm 5 the selected gene is obtained by calculating the fitness function of each expanded node. The best fitness value of a particular gene would be selected as the hoping node in the offspring nodes.

2. Mutation operator: Usually, the mutation operator has less computational burden than the crossover operator. Thus, the common method is implemented in the literature. This method can be seen, for example, in [2].

Algorithm 6 (Mutation operator)
Step 1: parent Initialization, let $i = 1$, i-th parent $n_{par}(i) \in \{n_{pop}\}$, k=number of populations
Step 2: For i=random number from 1 to k For j in surround cells $n_{mutation} = \{mutation\ gene \mid\mid \max(fitness\ of\ n_j)\}$ $\{n_{new}\} = \{n_{par}\} - n_{par}(i) - n_{par}(j) + n_{mutation}$

It is seen *that* the mutation operator takes the best offspring by comparing the mutated cells with the best fitness of surrounding cells. The proposed mutation strategy considers all the free nodes adjacent to the mutation node instead of randomly selecting a node one by one. It indicates that this proposed method neglects the node far away from the mutation node. The concept is to avoid the divergent method 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 movement direction through the mutated node.

Generally, the mutation is a process of random small changes in a gene. Mutation has a smaller probability than crossover to avoid obtaining a change of individual significantly.
Regarding the mutation operator, in this research, only one parent in the total population is possible for the mutation operation. As the mutation aims to obtain a better individual, a proper fitness function is vital to speed up the convergence.

G. Fitness Function and offspring selection

As mentioned before, the fitness function is the success key for the GA to find the optimum solution of path planning from the start to the target node. In this work, the optimal path is defined to be the shortest and the safest path. This approach is very common in the path planning problem. The proposed fitness function is as follows

$$f(p) = \begin{cases} \sum_{i=1}^{n-1} d(p_i, p_{i+1}) \; for \; feasible \; paths \\ \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + penalty, otherwise \end{cases} \quad (2)$$

where $p \in \{n_{pop}\}$.
The flowchart of the modified GA is presented in Fig. 6.

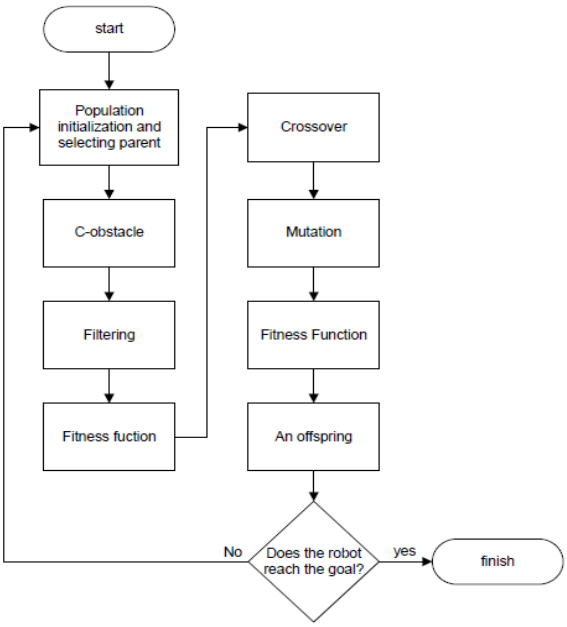


Figure 6. Flowchart of modified GA

The penalty added to the fitness function must be greater than the maximum path length of the environment. If this condition is met, the algorithm would be run iteratively until the penalty is omitted by searching for an appropriate chromosome. In the above function, the distance between nodes is formulated as

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (3)$$

where pi is the i[th] genes of the chromosome, n is the length of the chromosome, d is the distance between two points, x_i and y_i are the robot's current pose in the x and y axis, x_{i+1} and y_{i+1} are the next robot pose in x and y-axis. The robot's direction can be obtained by using the following formula

$$\alpha = \tan^{-1}\left(\frac{y_{goal} - y_i}{x_{goal} - x_i}\right) \quad (4)$$

The pose of y_{goal} and x_{goal} is the destination position. It is shown in (3) that 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 global optima solution.

H. Complete GA algorithm

The complete block diagram of the proposed path planning approach using GA is shown in Fig. 6. Generally, it is very similar to common GA. However, each block has unique features that enable efficient computation and guarantee convergence.

The algorithm started by initializing the population containing individuals with type 1 or type 2 chromosomes. Among those individuals, using c-obstacle and filtering, only a few were selected as eligible parents. It would enable efficient computational burden as well as speed up the convergence. Finally, the algorithm would undergo a similar process as common GA, such as crossovers, mutations, etc.

3. Results and Discussion

Some experiments were conducted to test the performance of the proposed GA. For simplicity, the parameter k is set to 1 unit. The experiment is divided into two scenarios according to the environment typ. First, the method was applied to a non-obstacle environment. Second, the environment that was similar to the previous improved GA proposed by Ali et al. was used as the benchmark. In the first setting, the robot moved from the initial to the final destination with no obstacle in between. Despite the simple and trivial solution, it shows that the algorithm functions well, as seen in Fig. 7.

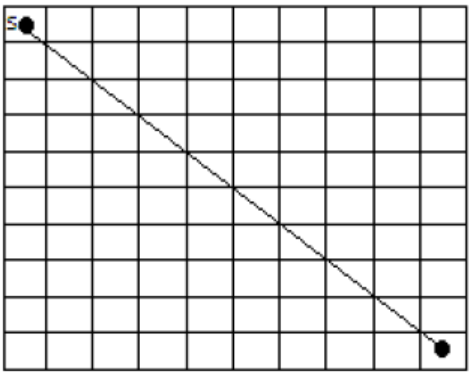


Figure 7. The proposed method for a non-obstacle environment

For the non-obstacle environment, since the global optimum solution of a non-obstacle environment was just a line path and the main parents could solve it, the modified GA only needed one generation to solve the global optimal problem. In Fig. 7, the parents contain gene 1 and 100, which indicates the initial and goal position, respectively.

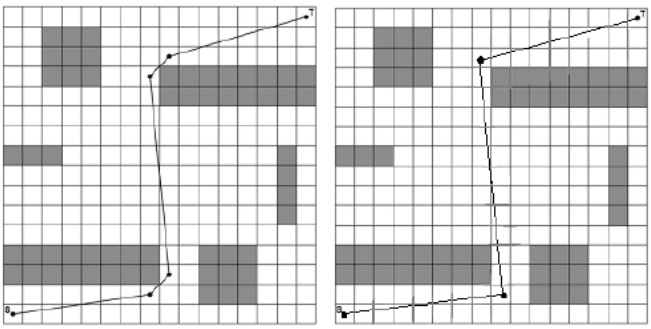
Fig. 8. shows the 16x16 grid environment with the obstacles depicted in shaded areas. The initial node is set to 0 and the goal node is set to 15. GA runs with the proposed filtering and crossover operator. The experiments give results as shown in Fig. 8a and 8b. In Fig. 8a, the method approached using global methods to find feasible paths in the workspace is applied. As shown in Fig 8b, the proposed method yields a different path from the previous research. The path length is 28.35, which is longer than the one with 27.82 in the previous research. Despite the path length, the generation results differently from the previous method.

The comparison is shown in Table I, which proves that the modified GA reduces the average generation number significantly. So, the proposed modified GA is even better than the method proposed by Tuncer and Yildirim in reducing the average generation number. On the other hand, the proposed method gives more stable results according to a 100 percent optimal solution. Moreover, it minimizes the computation used in GA and improves precision.

Table 1. Experimental Result in Fig. 8.

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	Generation number
Ref [20]	3	69	28	29.25	23
Ref [21]	2	69	29	29.91	22
Ref [3]	54	44	2	27.82	11
This research	100	0	0	28.35	2

Although the proposed method can solve the global optimum problem, it neglects non-holonomic constraints. Fig. 8 shows that the result is in a sharp pattern, and cannot be implemented for the non-holonomic robot. Thus, the method continues with a smoothing curve algorithm such as a B-spline curve. On the other hand, the proposed method based on a grid environment merely solves a discrete problem. The real problems are that all the systems must be computed continuously.



(a) (b)

Figure 8. The proposed method for a non-obstacle environment

Fig. 9 shows the generation number of optimal solutions against the total of the obstacle avoidance. It can be concluded that the generation number is affected by the number of obstacles the robot passes. The generation number merely depends on the obstacles that should be passed instead of the obstacle number in the environment. Compared to other methods, the modified GA has an adaptive generation number until it converges to an optimal solution.

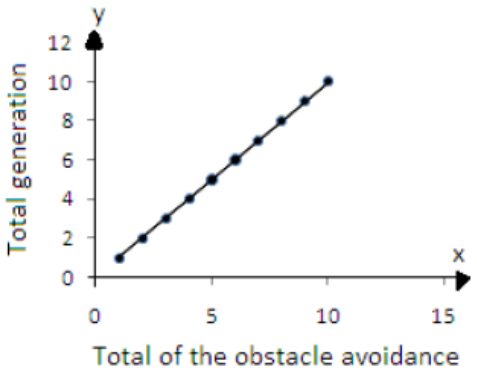


Figure 9. Total generation vs. total obstacle avoidance

One of the drawbacks arising in Fig. 9 is that the method would have a high computational cost when the robot encounters a large obstacle number. Otherwise, this method is very effective

for a small obstacle number. Therefore, the method has an opportunity to be used in the real-time system platform.

4. Conclusion

This research has three contributions related to the modified GA. The first is the population initialization, the second is the filtering method, and the third is the crossover operator. The initial population is divided into two individuals, i.e., the chromosome filled with the initial and goal nodes and the chromosome containing coordinates surrounding the obstacles. The coordinates surrounding the obstacles are based on the c-obstacle to compensate for the robot's dimension. Then filtering is used to choose only the feasible path. After that, the supporting parents would be added as a hoping node with the gene to obtain the optimum path. The proposed method in this work is that the supporting parents need a constant gene in the chromosome, but the main parent would change in length based on the generated hoping node.

The mutation is set using the efficient roulette algorithm combined with the c-obstacle concept that checks all the free nodes surrounding the mutation node. The method compares all the fitness function values and chooses the best solution. The node that has the highest fitness function is selected. In order to prove the concept, the simulation experiment is conducted using *Matlab* software with two kinds of environments: The environment that is non-obstacle and the environment similar to the previous improved GA studies in the literature. For a non-obstacle environment, the method meets optimum results due to the linear path. The result with 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 the previous method. The total generation number is influenced by the number of obstacles the robot should pass. If the total number of obstacles the robot passes is increased, then the total generation number would also increase. The GA has an adaptive generation number based on the total obstacles in this paper.

5. References

[1]. H. SHIN and J. CHAE, "A Performance Review of Collision-Free Path Planning Algorithms", *Electronics*, 9(2), 316. 2020.

[2]. J.M. Font and J. Kovecses, "Dynamics of Heel Strike in Bipedal Systems with Circular Feet", *Proceedings of EUCOMES08,* 2009. pp. 455-462.

[3]. 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. 2012.

[4]. R. Kala, A. Shukla, and R. Tiwari, "Robotic Path Planning in Static Environment Using Hierarchical Multi-Neuron Heuristic Search and Probability Based Fitness", *Neurocomputing*, Vol. 74, No. 14-15, pp. 2314-2335, 2011.

[5]. S. Hassan and J. Yoon, "Haptic Assisted Aircraft Optimal Assembly Path Planning Scheme Based on Swarming and Artificial Potential Field Approach", *Adv. Eng. Softw.,* Vol. 69, pp. 18-25. 2014.

[6]. R. Volpe and P. Khosla, "Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments", *IEEE Trans Syst Man Cybern*, 20(6), pp. 1423-1436, 1990.

[7]. W. Wu, Z. Q. Sen, J. B. Mbede, and H. Xinhan. “Research on Path Planning for Mobile Robot Among Dynamic Obstacles”, *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, Vancouver, BC, Canada, Vol 2, 2001, pp. 763-767.

[8]. Y. Shan, “Study on Submarine Path Planning Based on Modified Ant Colony Optimization Algorithm”, *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, Changchun, 2018, pp. 288-292.

[9]. Y. Huang, Y. Gu and Z. Zheng, “Research on the Path Planning of Hair-Insertion Robot Arm Based on Ant Colony Optimization”, *2018 37th Chinese Control Conference (CCC)*, Wuhan, 2018, pp. 5191-5195.

[10]. Y. Long, Y. Su, H. Zhang and M. Li, “Application of Improved Genetic Algorithm to Unmanned Surface Vehicle Path Planning”, *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, Enshi, 2018, pp. 209-212.

[11]. 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, 182-193, 2005.

[12]. M. Chen and A. Zalzala, "Safety Considerations in the Optimization of Paths for Mobile Robot Using Genetic Algorithms", *IEEE Int. Conf. Genet. Algorithms Eng. Syst. Innov. Appl*., pp. 299-306, 1995.

[13]. R. Soltani, H. Tawfik, J. Y. Goulermas, and T. Fernando, "Path Planning in Construction Sites: Performance Evaluation of the Dijkstra A* and GA Search Algorithms", *Adv. Eng. Informatics*, Vol. 16, pp. 291-303, 2002.

[14]. K.K. Kwaśniewski, and Z. Gosiewski. “Genetic Algorithm for Mobile Robot Route Planning with Obstacle Avoidance”. *Acta Mechanica et Automatica*, 12.2, pp. 151-159, June 2018.

[15]. S.B. Mane and S. Vhanale. “Genetic Algorithm Approach for Obstacle Avoidance and Path Optimization of Mobile Robot", *Computing, Communication and Signal Processing,* Springer, Singapore, 2019, pp. 649-659.

[16]. W. Rahmaniar and A.E. Rakhmania. “Mobile Robot Path Planning in a Trajectory with Multiple Obstacles Using Genetic Algorithms”, *Journal of Robotics and Control (JRC)*, 3.1, pp. 1-7, 2022.

[17]. A.T. Ismail, A. Sheta, and M. Al-Weshah, "A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment", *J. Comput. Sci.*, Vol. 4, pp. 341-344, 2008.

[18]. W. Jenkins, "A Decimal-coded Evolutionary Algorithm for Constrained Optimization", *Comput. Struct.,* Vol. 80, pp. 471-480, 2002.

[19]. J. H. Li, M. J. Lee, S. H. Park, and J. G. Kim, "Real Time Path Planning for A Class of Torpedo-type AUVs in Unknown Environment", *IEEE/OES Auton. Underw. Veh. AUV2012,* 2012, pp. 0-5.

[20]. Q. Li, W. Zhang, Y. Yin, Z. Wang, and G. Liu, "An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots, *Sixth Int. Conf. Intell. Syst. Des. Appl*., Vol. 2, 2006, pp. 637-642.

[21]. D. Whitley, "A Genetic Algorithm Tutorial", *Statistics and Computing,* 4*,* Springer-Link, pp. 65-85, 1994.

[22]. Z. Y. Z. Yao and L. M. L. Ma, "A Static Environment-based Path Planning Method by Using Genetic Algorithm", *Comput. Control Ind. Eng. (CCIE), 2010 Int. Conf.*, Vol.2, 2010, pp. 405-407.

[23]. H. Qu, K. Xing, and T. Alexander, "An Improved Genetic Algorithm with Co-evolutionary Strategy for Global Path planning of Multiple Mobile Robots", *Neurocomputing*, Vol. 120, pp. 509-517, 2013.

Oyas Wahyunggoro was born in Yogyakarta, Indonesia. He received his undergraduate degree (Ir.) in electrical engineering from Universitas Gadjah Mada (UGM), Indonesia, Feb 1993, a master's degree (MT) in electrical engineering from Universitas Gadjah Mada (UGM), Yogyakarta, May 2001, and a Ph.D. degree in automation and control system from Universiti Teknologi PETRONAS, Malaysia, Oct 2011. Currently, he is an Associate Professor in the Department of Electrical and Information Engineering, Engineering Faculty, Universitas Gadjah Mada, Indonesia. His research focuses are BMS, applying intelligent systems on automation and control systems, and biomedical signal processing. His expertise group is Signal-System-Control.

Hendri Himawan Triharminto received a B. Eng. in Electrical Engineering from Naval Science Institute and Technology. He completed master's and doctoral degrees in Electrical Engineering and Information Technology from Gadjah Mada University in 2017. His current research is robotic, especially control systems, path planning, image processing, and Simultaneous Localization and Mapping.

Adha Imam Cahyadi received his B.Eng. from the Department of Electrical Engineering, Faculty of Engineering, Universitas Gadjah Mada, Indonesia, in 2002, a master's degree in Control Engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand, in 2005, Thailand, and Doctor of Engineering from Tokai University, Japan in 2008. He currently serves as chairman for the bachelor program in Electrical Engineering at the Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Indonesia. His research interests are mechanical control systems, telemanipulation systems, as well as Unmanned Aerial Vehicles and Battery Management Systems.