Journal of Robotics and Control (JRC)
Volume 3, Issue 3, January 2022
ISSN: 2715-5072 DOI: 10.18196/jrc.31143

Mobile Robot Path Planning in a Trajectory with Multiple Obstacles Using Genetic Algorithms

Wahyu Rahmaniar[1] and Amalia Eka Rakhmania[1,2]
[1]Department of Electrical Engineering, National Central University, Taoyuan, Taiwan
[2]Department of Electrical Engineering, State Polytechnic of Malang, Malang, Indonesia
Email: wahyu.rahmaniar@gmail.com

***Abstract*— Path planning is an essential algorithm to help robots complete their task in the field quickly. However, some path planning algorithms are computationally expensive and cannot adapt to new environments with a distinctly different set of obstacles. This paper presents an optimal path planning based on a genetic algorithm (GA) that is proposed to be carried out in a dynamic environment with various obstacles. First, the points of the feasible path are found by performing a local search procedure. Then, the points are optimized to find the shortest path. When the optimal path is calculated, the position of the points on the path is smoothed to avoid obstacles in the environment. The simulation results show that the proposed algorithm successfully finds the optimal path in an environment with multiple obstacles. Compared to a traditional GA-based method, our proposed algorithm has a smoother route and a faster computation time due to path optimization. Therefore, this makes the proposed method advantageous in a dynamic environment.**

Keywords—genetic algorithm, mobile robot, optimal path, path planning, shortest path.

I. INTRODUCTION

The industrial revolution is claimed to be the era of automation, such as businesses, research facilities, and academic institutions around the world are researching robotics. Robots are increasingly prominent in our daily lives and have taken a significant part in humanitarian assistance [1], services industry [2], security[3], and drone photography [4], [5]. Robots are developed to make human life more comfortable, both for daily activities such as cleaning [6], cooking [7], assisting the elderly [8] and customer in a shopping center [9], even with risky activities such as monitoring hazardous environment [10] or for outdoor surveillance [11]. One of the crucial things that can help a robot complete its task quickly and accurately is path planning.

Path planning is an action to produce a path or movement series from the initial position to the desired state with a collision-free performance under certain restrictions and limitations [12]. The path planning algorithms aim to provide robot mobility to overcome specific obstacles without human intervention while achieving the shortest path with minimal energy consumption and the lowest running time [13]. The shortest path is a common constraint on path planning algorithms. While the robot follows the shortest path, it takes less time to do the work, and energy consumption will be kept to a minimum. Since the robot is equipped with limited energy, it is crucial to keep the energy as long as possible.

In general, the path planning algorithm is divided into two main implementations: probabilistic and artificial potential fields, *i.e*., genetic algorithms. Some algorithms for path planning are computationally expensive, especially when new environments and constraints are introduced. These algorithms cannot adapt to the new environment or recognize a different set of constraints. A method in [14] presents an outline of an autonomous mobile robot path planning algorithm that focuses on techniques to provide an ideal route for the robot to traverse the environment. Dijkstra algorithm is exploited in [15] to find the shortest path in a two-dimensional grid planar robot workspace. This method reduces time complexity since one adjacent grid node is only inspected once. But, this can only be implemented in a static environment with a single constraint. When the environment changes or more constraints are added to the system, the performance was not evaluated. Authors in [13] used information from sensors installed on the robot to avoid obstacles. A near-shortest path for mobile robot (NSPMR) algorithm was introduced based on local path planning with energy consumption and processing time limits. Though, the number of obstacles the robot can avoid is limited and static. If the obstacles increases and their position change, the robot needs to restart the sensing process, so the processing time will increase.

A primal-dual based heuristic for planning robot paths with different loading conditions is proposed in [16]. Since the robot has a wide variety of payloads, the time consumption and work handling of the system is heterogeneous. Nevertheless, this method can cause the expansion of the selected nodes and increase the computation time when several active subsets with the same constraint value are selected and go through the iteration process. Moreover, if the iteration result does not meet the constraints, the discarded components will be reselected to undergo the same procedure. This will certainly increase the processing time. Rapidly-exploring random tree (RRT) is implemented in [17] and [18] to modify the current path when an unknown obstacle blocks the path. Although the process shows high

flexibility in path planning, the higher number of dynamic obstacles resulted in higher computational cost.

Since the advent of artificial intelligent methods, path planning algorithms have been developed by introducing a genetic algorithm (GA) for node selection or possible path detection. A GA-based method was presented to find the shortest path for robots in a static environment and a fixed number of obstacles [19], [20]. These methods ensure the convergence of the algorithm, and the optimal path chosen is the shortest. However, there is no discussion of the time consumption of the algorithm. Time consumption appears to be high since the algorithms will calculate all possible paths between the starting and destination points with a randomly selected initial point. The selection of the adjacent points should be made based on the fixed conditions of the workspace map. If there is no prior knowledge about the position of obstacles, these methods cannot perform optimally. GA is integrated in [21] to find the Bezier curve control points to define a smoother and more consistent path. The purpose of this method is to reduce energy loss in the robot. But, as the algorithm emphasizes path quality, the processing time increases. The computation time will be much higher when the environment has more obstacles.

A method in [22] improves a new mutation operator in dynamic environments to find the optimal path faster. However, due to the mutations which are not randomized, the complexity of the algorithm is high. The algorithm will check all free nodes close to the mutation node and compare them with the fitness value of the complete path instead of selecting the nodes one by one. GA enhancement is proposed in [23] using multi-domain inversion to increase the number of offsprings. This method was claimed to be able to overcome the slow convergence and preterm population in traditional. As the number of offspring increases, a fitness condition is used to eliminate unwanted offsprings during the calculation. Although this method has superior performance compared to conventional methods, the complexity of this algorithm is increasing and sacrificing computational time. A method in [24] combined GA with the travelling salesman problem (TSP). This method claims that the simulated algorithm is feasible and could be implemented in real-world experiments. Since the GA and TSP have high computational costs, the authors need to consider the cost function regarding the obstacle position and numbers.

This paper proposes an improvement of the GA-based path planning algorithm that can be implemented in a dynamic environment with changeable obstacles. Since there is no prior information about the adjacent points between the starting and destination nodes, GA is used to find the feasible path by performing a local search procedure. The points found are then optimized to get an optimal path that focused on the shortest and smoother routes while avoiding obstacles in the trajectory. Centralized computation is implemented so that there will be no additional burden to the systems in the robot. Thus, the resulting path can be efficient.

The remainder of the paper is organized as follows. Section II and III introduce the path planning objectives and an overview system of the proposed method. Section IV presents the results and discussion. Finally, conclusions are drawn in Section V.

II. PATH PLANNING OBJECTIVES

A. Genetic Algorithm

In this paper, the simulation of the autonomous mobile robot is designed for viewing the workspace, locating the robot position in the environment, recognizing obstacles, and improving the path route. Therefore, path planning is a crucial step for the navigation of the robot since it is challenging to ensure convergence towards a dynamic environment.

GA works simultaneously with individual populations and explores several new areas in the solution space in parallel [25]. Environmental information is known beforehand and before the navigation process [26]. Thus, it can reduce the probability of the search space trapped within the local minimum to reach the target [27], [28]. The genetic algorithm process is described as follows:

1) Population

GA is initiated by randomly generating an initial population representing the possible paths (chromosomes) from the start point to the target point [29]. However, all of the possible shortest paths are initially considered as a population. If the entire population is found to be blocked during path planning by a robot collision, the configuration space is increased locally to ensure that alternative paths can be found. Therefore, the configuration space is only extended as needed to reduce the computational costs of finding a path.

2) Fitness function

The population of paths is evaluated using the fitness function during each reproduction cycle [30]. The fitness function used in this method considers the values for the path length, smoothness or angle of the path, and the number of feasible steps as follows

$$F(\text{path}) = \sum F(\text{length}),\ F(\text{feasible step}),\ F(\text{smoothness}) \quad (1)$$

The smoothness of the path is considered because a smoother route directly impacts the time cost of the robot to travel from the starting point to the target point. If the path has rough angles, the robot has to slow down for each turn which increases the total travel time. The feasible paths include additional points generated for avoiding the obstacles in the path. Therefore, the fitness function in this paper can produce a path with a minimum length, minimum time, and minimum speed to reach the target.

3) Encoding of chromosomes

The chromosome or path represents a candidate solution to find an optimal path planning [31]. A chromosome consists of a start point, a target point, and points that are generated along the path for the mobile robot to pass. These points are referred to as chromosomal genes. In our method, the points are represented as pixel locations in the two-dimensional environment by (x_n, y_n).

4) Selection method

The selection procedure stores the best genes on chromosomes for transfer to new generations. First, the objective function values of all chromosomes are found. Then, fitness values are assigned to the chromosomes according to the value of the objective function. The rank-based fitness is given to prevent some chromosomes from becoming dominant in the population. In the last step, the two chromosomes with the highest probability are selected as parents to mate via crossover to produce new chromosomes.

5) Crossover operator

The crossover operator takes two parents and divides their features in half at the center position. Half of each parent is then used to form a new individual. So that two offsprings are produced. The crossover of two infeasible chromosomes can generate a new infeasible path. To solve this problem, every chromosome must be checked. If it intersects with an obstacle while generating the initial population, the genes intersecting on the chromosomes are randomly changed to a feasible one.

6) Mutation operator

All candidate chromosomes in the population undergo random mutations after the crossover operation. This procedure is applied with uniform probability to all genes of all individuals in the population. The mutation operator takes a random point from the path and replaces it with a random path to its neighboring point. Therefore, mutations can increase population diversity and avoid premature convergence.

B. Environment

In this paper, four main points (*i.e.*, *A*, *B*, *C*, and *D*) are simulated to be selected as a start point (*S*) and a target point (*T*) in a two-dimensional environment. Obstacles (blue square) can be randomly generated and their location information is known, as shown in Fig. 1. The path planning is designed to find the optimal path from *S* to *T* within ten randomly generated points (small blue circle) that do not collide with the obstacles. Then, a robot (yellow circle) is converted into a single point in the field of computational geometry [32]. The points on the path are indicated by

$$p = \left[S, p_0, p_1, p_2, \mathrm{L}\ , p_n, p_{n+1}, T\right] \tag{2}$$

where p_n is the n-th gene of chromosomes or points in the path. The coordinate of p_n is denoted as (x_n, y_n).

In our method, length and smoothness are used as path optimization which is related to energy loss in the path. If the distance of the robot from the obstacles during running is less than the sensor safe distance, this can be a dangerous situation that can damage the robot. Therefore, a considerable distance between the robot and the obstacle is required for a safer route. The objectives designed to improve the optimal route of the robot as follows:

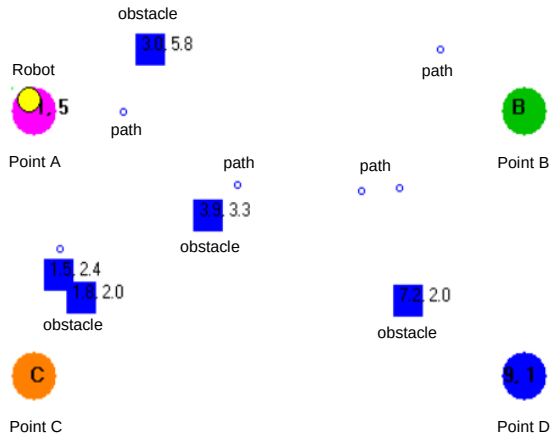


Fig. 1. Environmental simulation.

1) Path length

The path length can be calculated as the sum of the distances between each point in a path as follows

$$\lambda_p = \sum_{n=0}^{L} d\left(p_n, p_{n+1}\right) \tag{3}$$

where L is the length of the chromosome, d is the distance of every two points connected which can be calculated using the Euclidean distance as follows

$$d\left(p_n, p_{n+1}\right) = \sqrt{\left(x_n - x_{n+1}\right)^2 + \left(y_n - y_{n+1}\right)^2} \tag{4}$$

where $p_{n+1} = \left(x_n, y_n\right)$ and $p_{n+1} = \left(x_{n+1}, y_{n+1}\right)$, p_{n+1} consists of the position of the next horizontal and vertical point.

2) Path smoothness

The path smoothness can be assumed as the degree of path sleekness. A smoother route can reduce the energy loss of the robot when running along the path. First, the average turning angle of the path is calculated by [33]

$$\theta\left(p_n, p_{n+1}, p_{n+2}\right) = \pi - \cos^{-1}\left(\frac{\left(x_{n+1} - x_n\right)\left(x_{n+2} - x_{n+1}\right) + \left(y_{n+1} - y_n\right)\left(y_{n+2} - y_{n+1}\right)}{d\left(p_n, p_{n+1}\right) \times d\left(p_n, p_{n+2}\right)}\right) \tag{5}$$

Then, the total average turning angle in λ_p can be determined by

$$\omega\left(p\right) = \frac{1}{L}\sum_{n=0}^{L}\left|\theta\left(p_i, p_{i+1}, p_{i+2}\right)\right| \tag{6}$$

III. PROPOSED METHOD

The proposed method process is summarized in Algorithm 1. First, variables are initialized and information on the environment is stored. The S_n and T_n are selected by user, and several points p_n are generated along the path. The obstacles O_i are randomly generated and their positions are known. The obstacles positions are stored as p_n. O_i is added as population *POP*. In *POP*, the objectives of each individual are evaluated.

If the stopping conditions have not been reached, a new population *POP'* is generated. Individuals are selected from *POP* by a selection operator. Individuals in *POP* are paired and a crossover operator is performed. Every individual in *POP'* is evaluated and the infeasible solutions examined. If there is a possibility that an infeasible path can occur, then the mutation operator is carried out. Individuals in *POP* are classified according to the set of feasible and infeasible solutions. This process is repeated until a solution is found that satisfies the optimal path planning and a fixed number of generations is reached.

Algorithm 1: Path planning pseudocode

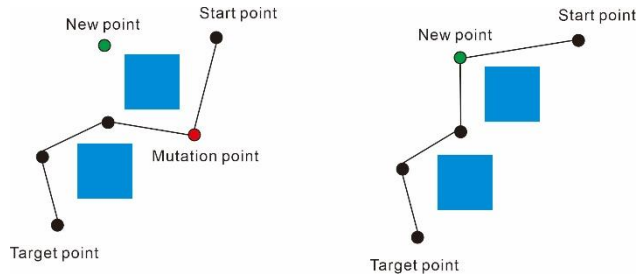
1. Initializing and setting the environment, population-scale N_{pop}, the maximum iteration of GA N, number of obstacles N_{obs}, the position of every obstacle O_i, the crossover probability α_c, the mutation probability α_m, and the whole population set p_L.
2. Determine start position S_n, target position T_n, and path point p_n.
3. Calculate visible space of S_n as V_n.
4. **While** $i < N_{pop}$
   Set O_i to store in p_n .
   Add p_n to O_i.
   $S_j = S_n$ and $V_{Si} = V_n$.
   While $T_n \neq V_{Sn}$
   Select a p_n from V_{Si}, $p_i = p_n$.
   Add T_n into O_i.
   Add O_i into population *POP*.
   $i = i + 1$
5. Determine the feasible path and start the genetic operation.
6. **While** $j < N$
   Calculate chromosome fitness in the population.
   Perform a selection operation.
   Perform a crossover operation.
   If the crossover point can cause infeasible solution
   Then modify the chromosome by a mutation operation.
   New generated individual stored in child population *POP'*.
7. Output the optimal solution.
8. The robot starts moving to travel along the path.

A. Path Planning Improvements

In the selection operator, the best individuals are selected and stored in *POP*. Two individual in *POP* are selected and their features are swap by a crossover operation. The location of the points and individual in *POP* are evaluated to improve the optimal solution as follows:

1) Shortest path

A single point variation is used to determine the shortest path when passing by the obstacles. A point in the path is randomly selected and replaced with a point randomly generated around an obstacle. The path that is produced after the mutation can be better or worse than the previous path. Therefore, mutations are continued until the shortest path is optimal. Fig. 2 shows the point mutation to find the shortest path.



(a) (b)

Fig. 2. Mutation of points on the path to obtaining the shortest path. (a) A randomly selected point (red circle) is replaced by a randomly generated point (green circle). (b) New path results.

2) Position update

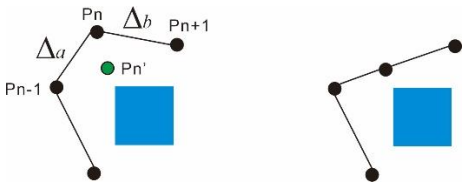
Several three continuous points (p_{n-1}, p_n, p_{n+1}) around obstacles are selected to examine for their positions in the path, as shown in Fig. 3. The position of p_n is updated by

$$p_n' = p_n + \Delta \tag{7}$$

where Δ is the displacement of the p_n in the solution space, can be calculated by

$$\Delta = \Delta_a \alpha + \Delta_b \beta \tag{8}$$

where α and β are the velocity constraints in the updated position which are set to ±1% of the vector distance between the upper and lower boundaries of the solution space, $\Delta_a = p_{n-1} - p_n$, and $\Delta_b = p_{n+1} - p_n$.



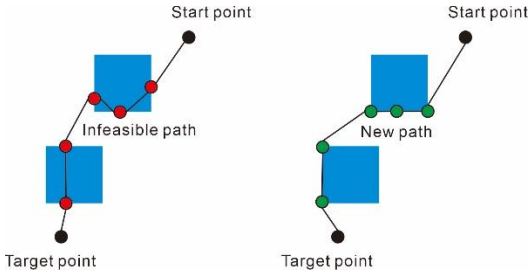
(a) (b)

Fig. 3. Update point positions. (a) Three continuous points are selected and the center point is moved to a new position (green circle). (b) New path results.

B. Obstacle Avoidance Approach

In the case of infeasible path, the solution operator is performed to change the position of several points to create a feasible path. The infeasible path usually occurs when two or more points intersect an obstacle. Every two vertices are connected to form a new set of edges between the obstacle and the two selected points (p_n, p_{n+1}), as shown in Fig. 4. The edge that intersects with the obstacles will be deleted. Djikstra algorithm is applied to achieve the optimal shortest path based on graph vertices and edges. The path p can be

adjusted by adding a new point q. Thus, the new path is generated as $p' = [S, p_0, p_1, p_2, \text{L}\ , p_{n-1}, q, \text{L}\ \ p_n, p_{n+1}, T]$.



(a) (b)

Fig. 4. Obstacle avoidance. (a) The points in the path are intersecting the obstacle (red circle). (b) New path results with a new point position (green circle).

IV. RESULTS AND DISCUSSION

We compared the results with the traditional GA method to evaluate the performance of the proposed method. The robot is run from a start point to a target point with 5, 8, and 10 obstacles in the environment. The numbers (*x*,*y*) indicate the center position of the obstacles. Fig. 5, 6, and 7 show the results of path planning using the traditional GA method and the proposed method.

In Fig. 5 (a), point 4 has a non-optimal location which makes the path longer, where it should have been between points 3 and 5 could have a shorter path. Between point 5 and the target point some improper points cause the path to intersect with the obstacle. Our proposed method can solve this problem, as shown in Fig. 5(b), the points 4 and 5 are updated so that the path can be shorter and avoid the obstacles. The same situation occurs in Fig. 6(a) and 7(a), the path generated by the traditional GA method cannot solve the problem of the infeasible path which makes the path not optimal and collides with obstacles. Fig. 6(b) and 7(b) show that our proposed method fixes these common problems of path planning and produces the optimal path while avoiding obstacles. A solution is found when the best fitness value meets the average fitness value, as shown in Fig. 5(c), 6(c), and 7(c).

Table I shows that the proposed method has shorter and smoother path compared to the traditional GA method. Table II summarizes the performance results of our proposed method. The more obstacles generated in the path would allow the algorithm to produce more individual generations and mutation operators. More mutations occur because the path has more obstacles, which means it is more difficult to find the optimal path solution.

TABLE I. PERFORMANCE COMPARISON

Method	Average Results		
	Time (s)	Path Length (pixels)	Smoothness
Traditional GA	10.68	565	7.8
Proposed Method	8.57	521	6.5

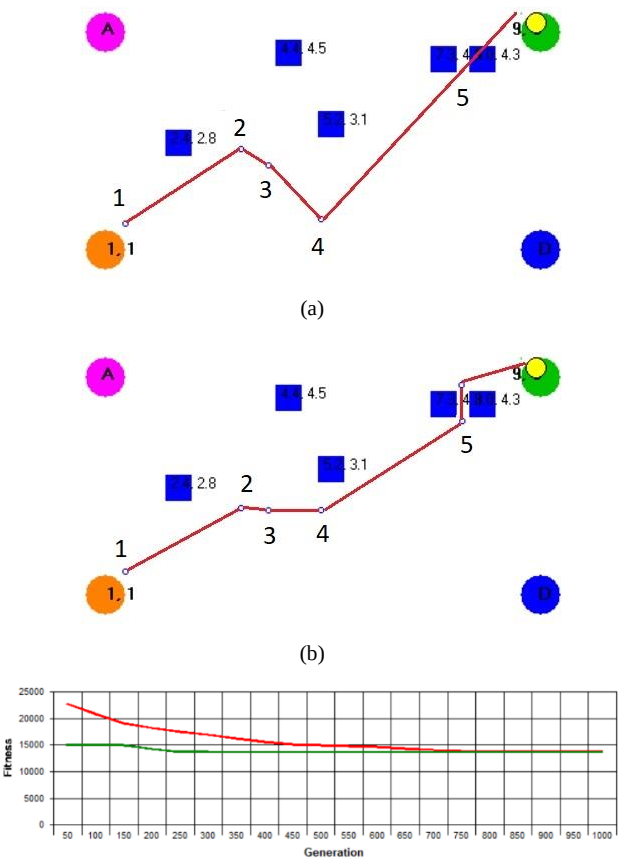


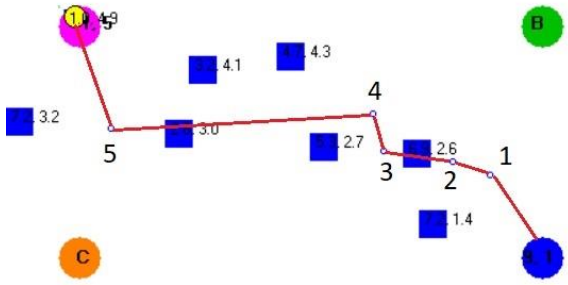
Fig. 5. Path planning results of 5 obstacles. (a) Traditional GA. (b) Proposed method. (c) Average fitness value (green) and the best fitness value (red).

TABLE II. PERFORMANCE RESULTS

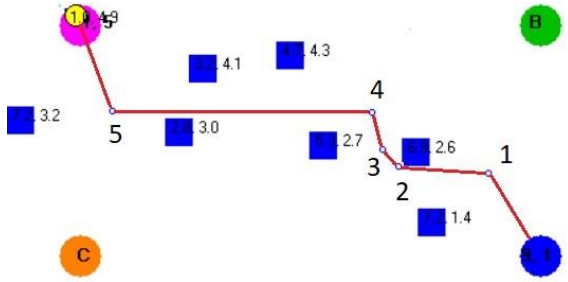
Total Number of Obstacles	Generations	Mutations	Average Fitness Value
5	1002	1955	13646
8	1008	2056	14283
10	1012	2086	16382

In Fig. 6(a), the traditional GA method cannot find a path that can avoid the obstacle between points 2 and 3, the same situation happens between points 4 and 5. Our proposed method updates the position for point 2 so that the path between points 2 and 3 do not hit obstacles, which also fix the position of point 5 so that the path is more optimal, as shown in Fig. 6(b).

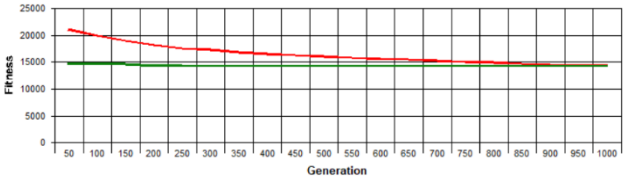
In Fig. 7(a), there are 5 points that are not optimal to create a path between a start point to a target point. Our proposed method makes the path more optimal with a few additional points to avoid obstacles around point 1 and a shorter path between points 2 and 4, as shown in Fig. 7(b).



(a)

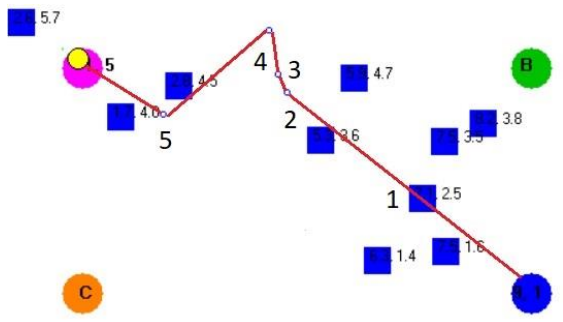


(b)

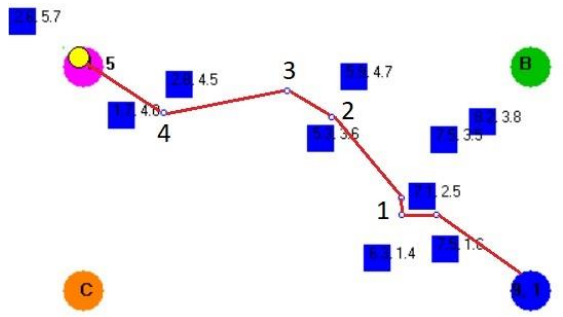


(c)

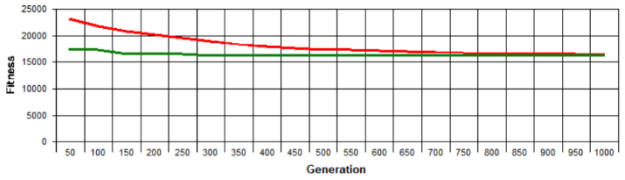
Fig. 6. Path planning results of 8 obstacles. (a) Traditional GA. (b) Proposed method. (c) Average fitness value (green) and the best fitness value (red).



(a)



(b)



(c)

Fig. 7. Path planning results of 10 obstacles. (a) Traditional GA. (b) Proposed method. (c) Average fitness value (green) and the best fitness value (red).

V. CONCLUSIONS

This paper has presented path planning improvements for a mobile robot using genetic algorithms in an environment with multiple obstacles. Two objectives to be optimized are applied to find the optimal path. The GA operators are proposed to accelerate the evolution of individual populations in the path. This method selects points according to the fitness value of the total path rather than the direction of movement through the mutated points. Compared with the traditional GA method, the proposed method shows that the chromosomes generated by path planning have better characteristics and a faster computation time in the solution space. The average fitness values and the number of generations generated on this improved GA are more suitable for optimal path planning than the traditional method. Thus, the resulting path is shorter and smoother and can be used as a travel path for a mobile robot in a real application. For future works, the characteristics of the sensors in the robot can be considered for better path planning implementation.

REFERENCES

[1] H. M. Gross, A. Scheidig, S. Müller, B. Schütz, C. Fricke, and S. Meyer, "Living with a mobile companion robot in your own apartment - Final implementation and results of a 20-weeks field study with 20 seniors," in Proc. of IEEE Int. Conf. Robot. Autom., 2019, pp. 2253–2259, 2019.

[2] C. J. Lai and C. P. Tsai, "Design of introducing service robot into catering services," in Proc. of ACM Int. Conf. Proceeding Ser., 2018, pp. 62–66.

[3] S. Meghana, T. V. Nikhil, R. Murali, S. Sanjana, R. Vidhya, and K. J. Mohammed, "Design and implementation of surveillance robot for outdoor security," in Proc. of IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. Proc., 2017, pp. 1679–1682.

[4] M. Takagi, "Japanese society: where humans and robots coexist," Int. J. Soc. Sci. Humanit., vol. 10, no. 1, pp. 13–16, 2020.

[5] W. Rahmaniar and A. E. Rakhmania, "Online digital image stabilization for an unmanned aerial vehicle (UAV)," J. Robot. Control, vol. 2, no. 4, pp. 234–239, 2021.

[6] S. M. B. P. Samarakoon, M. A. V. J. Muthugala, A. Vu Le, and M. R. Elara, "HTetro-infi: A Reconfigurable floor cleaning robot with infinite morphologies," IEEE Access, vol. 8, pp. 69816–69828, 2020.

[7] K. Junge, J. Hughes, T. G. Thuruthel, and F. Iida, "Improving robotic cooking using batch bayesian optimization," IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 760–765, 2020.

[8] G. Wilson et al., "Robot-enabled support of daily activities in smart home environments," Cogn. Syst. Res., vol. 54, pp. 258–272, 2019.

[9] M. Niemelä, P. Heikkilä, and H. Lammi, "A social service robot in a shopping mall," in Proc. of ACM/IEEE International Conference on Human-Robot Interaction, 2017, pp. 227–228.

[10] W. Rahmaniar and A. Wicaksono, "Design and implementation of a mobile robot for carbon monoxide monitoring," J. Robot. Control, vol. 2, no. 1, 2021.

[11] W. Rahmaniar, W. Wang, and H. Chen, "Real-time detection and recognition of multiple moving objects for aerial surveillance," Electronics, vol. 8, no. 12, pp. 1373–1390, 2019.

[12] Y. Chen, J. Liang, Y. Wang, Q. Pan, J. Tan, and J. Mao, "Autonomous mobile robot path planning in unknown dynamic environments using neural dynamics," Soft Comput., vol. 24, no. 18, pp. 13979–13995,

2020.
[13] H. T. Nguyen, H. X. Le, and V. Nam, "Path planning and obstacle avoidance approaches for mobile robot," Int. J. Comput. Sci. Issues, vol. 13, no. 4, pp. 1–10, 2016.
[14] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in Proc. of 4th Student Conf. Res. Dev. "Towards Enhancing Res. Excell. Reg., 2006, pp. 183–188, 2006.
[15] L. Sun, X. Liu, and M. Leng, "An effective algorithm of shortest path planning in a static environment," Int. Fed. Inf. Process., vol. 207, pp. 257–262, 2006.
[16] J. Bae and W. Chung, "Efficient path planning for multiple transportation robots under various loading conditions," Int. J. Adv. Robot. Syst., vol. 16, no. 2, pp. 1–9, 2019.
[17] D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using RRT," in Proc. of IEEE Int. Conf. Syst. Man, Cybern, 2017, pp. 1429–1434, 2017.
[18] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," Sensors, vol. 18, no. 2, 2018.
[19] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," in Proc. of Int. Conf. Electr. Electron. Comput. Eng., 2004, pp. 185–189, 2004.
[20] A. Ghorbani, S. Shiry, and A. Nodehi, "Using genetic algorithm for a mobile robot path planning," in Proc. of Int. Conf. Futur. Comput. Commun., 2009, pp. 164–166.
[21] J. Ma, Y. Liu, S. Zang, and L. Wang, "Robot path planning based on genetic algorithm fused with continuous bezier optimization," Comput. Intell. Neurosci., vol. 2020, pp. 1–10, 2020.
[22] 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.
[23] J. Xin, J. Zhong, F. Yang, Y. Cui, and J. Sheng, "An improved genetic algorithm for path-planning of unmanned surface vehicle," Sensors, vol. 19, no. 11, pp. 1–23, 2019.
[24] A. V. Le et al., "Complete path planning for a tetris-inspired self-reconfigurable robot by the genetic algorithm of the traveling salesman problem," Electron., vol. 7, no. 12, pp. 1–21, 2018.
[25] C. Messom, "Genetic algorithms for auto-tuning mobile robot motion control," Research Lett. in the Inf. and Math. Sci., vol. 3, pp. 129-134, 2002.
[26] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," in Procedia Computer Science, 2018, vol. 127, pp. 180–189.
[27] F. A. Afsar, M. Arif, and M. Hussain, "Genetic algorithm based path planning and optimization for autonomous mobile robots with morphological preprocessing," in Proc. of IEEE International Multitopic Conference, 2006, pp. 182–187.
[28] Y. Xue and J. Q. Sun, "Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm," Appl. Sci., vol. 8, no. 9, 2018.
[29] T. Tometzki and S. Engell, "Systematic initialization techniques for hybrid evolutionary algorithms for solving two-stage stochastic mixed-integer programs," IEEE Trans. Evol. Comput., vol. 15, no. 2, pp. 196–214, Apr. 2011.
[30] K. Rakesh and K. Mahesh, "Exploring genetic algorithm for shortest path optimization in data networks." Global J. Comp. Sci. Tech., vol. 10, no. 11, pp. 8-12, 2010.
[31] M. Muthiah and A. Saad, "Multi robot path planning and path coordination using genetic algorithms," in Proc. of SouthEast Conf. ACMSE, 2017, pp. 112–119.
[32] S. Das and S. Sarvottamananda, "Computing the minkowski sum of convex polytopes in $\Re^d$." arXiv:1811.05812, 2018.
[33] Y. Xue, "Mobile robot path planning with a non-dominated sorting genetic algorithm," Appl. Sci., vol. 8, no. 11, 2018.