Jixin Liu
Yanbin Cai
Yue Cao ✉

https://doi.org/10.21278/TOF.483056623
ISSN 1333-1124
eISSN 1849-1391

A ROBOT PATH-PLANNING METHOD BASED ON AN IMPROVED GENETIC ALGORITHM

Summary

When solving path-planning problems, a traditional genetic algorithm has some drawbacks such as being prone to falling into premature convergence, a relatively slow convergence rate, and generating multiple invalid paths during crossover and mutation operations. It also depends heavily on the initial population and empirical core parameters. In this paper, a robot path-planning method based on an improved genetic algorithm is proposed. The crossover and variation probabilities of the genetic algorithm are given by an adaptive function during the population iterations, and deletion and optimisation operators are proposed to improve the performance of the algorithm. The reference population is introduced for those inferior individuals eliminated by the main population, and the high-quality gene fragments among them are extracted and added to the main population to speed up the search procedure and to avoid missing the optimal solution. The simulation results show that the adaptive function speeds up the convergence of the algorithm and ensures the searching ability. The addition of the deletion and optimisation operators shortens the length of the optimal path. The reference population significantly accelerates the convergence speed of the algorithm and ensures the stability of the population throughout the process.

Key words: *path planning; genetic algorithm; optimisation operator; reference population*

1. Introduction

In recent years, as research on robotics has intensified, the role of robots in modern industrial technology has become increasingly important [1], and the requirements for robot interaction with the outside world have increased [2]. Path planning is one of the most basic and important requirements for robots. It is about how to find a path from the starting point to the target point in the space where the robot is working by using sensors and computers to obtain information about the external environment and to avoid any obstacles effectively. The traditional path-planning algorithm is Dijkstra's algorithm from 1959 [3]. In 1966, graph theory was introduced by Doran and Michie to the problem of path planning [4]. In 1968, the famous Grid Method was used to solve the robot path planning problem. In response to the low efficiency of Dijkstra's algorithm, Hart and Nilsson designed an A* algorithm [5] to help solve the problem. However, in a complex environment, because of the large amount of memory

space required, the algorithm takes longer to solve the optimal value, and the performance deteriorates. Therefore, numerous scholars have made improvements based on traditional algorithms. Song et al. [6] used image processing methods to build raster maps and added three smoothers to increase the continuity when the A* algorithm generates collision-free paths, and this method is superior in both turns and paths. In Dijkstra's algorithm, the presence of some nodes in the deterministic environment reduces the algorithm's speed. Fusic et al. [7] consider the nearest nodes to find the optimal path, which reduces the time needed. Hliwa et al. [8] combine the Firefly Algorithm [9] with Tabu Search, introducing the TS algorithm in short segments to optimise the path and improve the algorithm's robustness by avoiding local optima. Although improved traditional algorithms have greatly enhanced the efficiency and performance of mobile robot path planning, they still show shortcomings in adaptability, high computational requirements, and low optimisation efficiency in complex environments.

Since the 1990s, intelligent algorithms have also been successfully introduced into robot path-planning problems [10], resulting in significant accomplishments. Cao et al. [11] improved the Ant Colony Optimisation (ACO) algorithm by increasing the amount of pheromone on short paths in each cycle and dynamically adjusting its evaporation rate to enhance the probability of transitions. The results showed that this algorithm improved global optimisation capability and significantly increased search speed. Kamalova et al. [12] proposed a new multi-objective Grey Wolf Optimisation (GWO) algorithm to address the problem of mobile robot route planning. Brand et al. [13] employed the Firefly Algorithm to solve the path-planning problem for mobile robots in dynamic environments. The results indicated relatively low path lengths and computational costs. Additionally, Saraswati et al. [14] combined the Cuckoo Search and Bat Algorithm. The Cuckoo Search algorithm was used to search for local optimal solutions, and the obtained optimal solution was then used as an input for the Bat Algorithm to obtain a global optimal solution.

Genetic algorithms employ a population-based search strategy and possess strong global search capability, parallel processing ability, and strong adaptability [15-17]. Therefore, they are widely used in path-planning problems. However, traditional genetic algorithms have certain dependencies on the generation of the initial population, are prone to getting stuck in local optima, and can generate a large number of invalid paths during crossover and mutation operations, thereby increasing computational complexity. Therefore, to overcome the shortcomings of traditional genetic algorithms in path planning, algorithmic improvements and optimisations need to be made according to specific problems. Luan et al. [18] provided a dynamic mutation rate and switchable global-local search method based on the mutation operator of traditional genetic algorithms. Through these improvements, the premature convergence of genetic algorithms and the long computation time of fitness calculations in meme algorithms were reduced. By utilising segmented cubic Bezier curves with continuous curvature, smooth paths were provided for differential wheeled robots. Mohamed et al. [19] also combined genetic algorithms with Bezier curves to improve the smoothness of paths by using genetic algorithms to search for the control points of Bezier curves. López-González et al. [20] utilised parallel genetic algorithms to achieve artificial distributed skills and shared the best method that converges to the desired distance while avoiding collisions, ultimately reaching consensus on the solution. Lee et al. [21] proposed combining genetic algorithms with directional guidance factors to achieve obstacle avoidance, reduce unnecessary computations, and accelerate convergence speed. Han et al. [22] introduced a one-time transformation crossover algorithm to generate optimal offspring and applied the minimisation of dual-path constraints to obtain the shortest path. Lee [23] suggested combining the initialisation population with directed acyclic graphs to generate initial paths and continuously optimise them, allowing the algorithm to obtain the optimal path in a short period. However, these improvement methods are prone to generating many invalid paths during crossover and mutation operations, relying on the initial population, and requiring core parameters to be given

based on experience. To solve this problem, this paper proposes an initial population generation method based on midpoint interpolation. In addition, the design of the fitness calculation function for the algorithm model has always been a key point in genetic algorithms. This study designs a new fitness function that achieves a reduction in computational complexity while satisfying the condition of finding the shortest path. To prevent the algorithm from being limited by the initial population and to reduce the decrease in the algorithm's global search capability, a reference population is introduced to enhance the efficiency of global search. Additionally, deletion operators are included to remove duplicate paths in individuals, and optimisation operators are added to improve path length and smoothness.

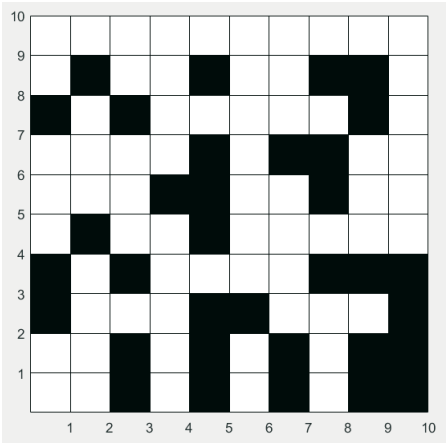
2. Environmental modelling

The grid-based method constructs the environmental space as a two-dimensional plane, considering only the length and width dimensions of obstacles to simplify the environment model. The two-dimensional map is divided into small matrix blocks of equal size using the grid-based approach. Each matrix block is assigned a feature value, representing the state of that block. Thus, the grid map can be represented in the form of a numerical matrix. If a matrix value in the grid is "1", this indicates that the corresponding area is an obstacle region, while a value of "0" signifies that the area is an obstacle-free space.

Assuming that the two-dimensional plane created by the raster method is 10 × 10, the numerical matrix is given in Fig. 1(a), and the created grid map is shown in Fig. 1(b).

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

(a) Environmental Map Number Matrix



(b) Environmental Map1

Fig. 1 Environmental Map representation

The white areas in the diagram represent obstacle-free spaces, while the shaded squares indicate obstacles. When the robot is moving, it needs to avoid the black-shaded areas and choose the path represented by the white squares. The robot's position in the environment can be described using coordinates (x, y), which represent the position in the x-th row and y-th column. The path taken by the robot from the starting point S to the end point E can be represented as a set of path points (x, y).

3. Path-planning model based on an improved genetic algorithm

3.1 Individual coding

In this algorithm, a decimal encoding method is used, which represents the complete path of a mobile robot moving in a grid map from the starting point to the destination. The encoding set consists of grid indices, where each element represents a grid block that the robot traverses along the complete path.

First, let us assume that each matrix block in the grid map has a side length of 1. The index of the bottom-left grid block is defined as 1. Then, the grid blocks are sorted in a left-to-right, bottom-to-top order. The midpoint of the matrix block with index 1 is defined as (0.5,0.5). Therefore, the matrix block at coordinates (0.5,1.5) is represented by the index 2. The coordinate relationship between grid coordinates (x_i,y_i) and grid index N is given by the equation as follows:

$$\begin{cases} N_i = (x_i - 1) \times G_{size} + (y_i - 1) \\ x_i = \text{int}(N - 1 / G_{size}) + 1 \\ y_i = \text{mod}(N - 1 / G_{size}) + 1 \end{cases} \tag{1}$$

where N_i is the raster serial number, and G_{size} denotes the dimensionality of the map G, and mod is the remainder of the result of the equation, and int denotes rounding the result of the equation.

3.2 Initialising the population based on midpoint interpolation

In path-planning problems, traditional genetic algorithms often generate the initial population by restricting the values of the first and last components of each individual. These values represent the coordinates of the starting and end points, respectively. The other intermediate points are randomly generated between the starting and end points. This random method of generating the initial population is convenient and efficient as it generates the entire population at once. However, it may also yield a significant number of unfeasible solutions. Due to the random nature of point generation, there is a high possibility of generating line segments between consecutive points that intersect with shaded areas on the map. This leads to a large number of invalid calculations and reduces the efficiency of the algorithm.

To address this issue, this algorithm utilises an interpolation operator for the initialisation of the population. The specific steps of this operation are as follows:

Step 1: Select one grid cell per row.

Based on the assumption that the robot can only move one grid cell at a time on the map, each robot needs to explore step by step from the starting point towards the destination, eventually creating a complete path. Based on this assumption, we can infer that on the grid map, except for the rows containing the starting point and the destination, there must be at least one white grid cell that the robot passes through in each row. Therefore, for any individual, one can select any white grid cell index in each row as its path. Refer to Fig. 2 for an illustration:

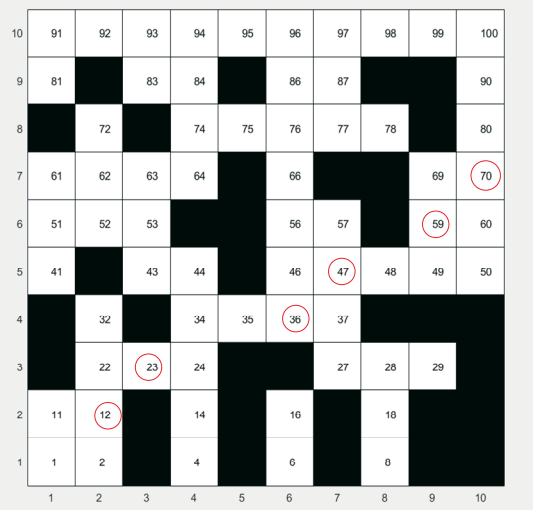


Fig. 2 Path diagram before initial individual interpolation

In Fig. 2, the individual with the starting point at the lower left corner numbered 1 is the starting point, and the grid cell numbered 80 is the destination. In between, a random grid cell was chosen as a path point in each row. The resulting initial path is (1, 12, 23, 36, 47, 59, 70, 80).

Step 2: Check whether adjacent path points are consecutive.

Fig. 2 shows that in the selected path (1, 12, 23, 36, 47, 59, 70, 80), some adjacent path points cannot be directly connected. The line would pass through a shaded area. Therefore, it is necessary to check the continuity of each adjacent pair of path points in the selected path:

$$\gamma = \begin{cases} 0 & T \le 1 \\ 1 & T > 1 \end{cases} \tag{2}$$

$$T = \max\{\text{abs}(x_{i+1} - x_i), \text{abs}(y_{i+1} - y_i)\} \tag{3}$$

where $\gamma = 0$ indicates that the two path points are continuous and can be connected directly, and $\gamma = 1$ denotes a discontinuity between two path points and requires interpolation. abs denotes an absolute value operation, and x_i and x_{i+1} denote the line coordinates of the i-th path point and the line coordinates of the i+1-th path point, respectively. y_i and y_{i+1} denote the column coordinates of the i-th path point and the column coordinates of the i+1-th path point, respectively.

Step 3: Midpoint interpolation

When the discontinuity between two path points is judged as disjoint after Step 2, the path between the two disjoint points is made continuous by using the midpoint interpolation method, as in equation (4):

$$\begin{cases} x_{new} = \text{int}\left(\dfrac{x_{i+1} + x_i}{2}\right) \\ y_{new} = \text{int}\left(\dfrac{y_{i+1} + y_i}{2}\right) \end{cases} \tag{4}$$

where x_{new} and y_{new} are the coordinate value of the new coordinate point inserted between the non-adjacent points.

Step 4: Generate the entire initial population

Steps 1 to 3 are completed to generate the first individual in the initial population. Next, Steps 1 to 3 are repeated until the population reaches the desired size.

3.3 Fitness function

In this algorithm, the fitness function is defined as:

$$f_i = \frac{1}{\sum_{j=2}^{l_i} \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}} + C \tag{5}$$

C is a large positive number, which is used to prevent the small fitness of an individual in the population from leading to errors in the calculation and to avoid the fitness of a poor individual in the population differing too much from that of a high-quality individual. This is influenced by the optimal value in the subsequent selection process, which makes its probability of being selected too low, thus reducing the diversity of the algorithm.

3.4 Adaptive crossover and mutation operators

1. Adaptive crossover and mutation probability functions

In traditional genetic algorithms, the probabilities for crossover and mutation are typically set by the user based on their empirical experience. This approach can have negative effects on the efficiency and convergence of the algorithm. Therefore, it is beneficial to have an adaptive probability function for each individual in the population to control their crossover and mutation.

The adaptive crossover probability function proposed in this paper is as follows:

$$P_c = \begin{cases} P_{c_{\min}} + \sin c\left(\dfrac{f' - f_{avg}}{f_{\max} - f_{avg}}\right) \times \dfrac{f_{avg}}{f_{\max}} \times \left(P_{c_{\max}} - P_{c_{\min}}\right) & f' \geq f_{avg} \\ P_{c_{\max}} & f' < f_{avg} \end{cases} \tag{6}$$

where P_c is the crossover probability of paired individuals, and $P_{c_{\min}}$ and $P_{c_{\max}}$ are the lower and upper bounds of the given crossover probability range, respectively. f' denotes the largest fitness value of the two individuals to be crossed. $f_{\min}$ is the mean of the sum of the fitness of all individuals, and $f_{\max}$ is the fitness value of the best individual in the population.

The proposed adaptive mutation probability function is as follows:

$$P_m = \begin{cases} P_{m\min} + \sin c\left(\dfrac{f' - f_{avg}}{f_{\max} - f_{avg}}\right) \times \dfrac{f_{avg}}{f_{\max}} \times \left(P_{m\max} - P_{m\min}\right) & f' \geq f_{avg} \\ P_{m\max} & f' < f_{avg} \end{cases} \tag{7}$$

$$\sin c(x) = \frac{\sin(\pi \cdot x)}{\pi \cdot x} \tag{8}$$

where P_m is the probability of the mutation of an individual, and $P_{m_{\min}}$ and $P_{m_{\max}}$ are the lower and upper bounds of the given range of mutation probabilities, respectively. f' denotes the fitness value of an individual, and f_{avg} is the mean of the sum of the fitness of all individuals in the population, and $f_{\max}$ is the fitness value of the best individual in the population.

From the practical problem, we know that the range of f' and f_{avg} is $[0, f_{\max}]$. When f' takes the value of f_{avg}, then:

$$\sin c\left(\frac{f' - f_{avg}}{f_{\max} - f_{avg}}\right) = \sin c(0) = 1 \tag{9}$$

When f' takes the value of f_{avg}, then:

$$\sin c\left(\frac{f' - f_{avg}}{f_{\max} - f_{avg}}\right) = \sin c(1) = 0 \tag{10}$$

From equations (9) and (10), we can conclude that the value range of $\sin c\left(\dfrac{f' - f_{avg}}{f_{\max} - f_{avg}}\right)$ is [0,1]. When the new species and the mutated population collide with obstacles, f' is less than the average of the population. To promote individual evolution, the crossover and mutation probabilities are set to their maximum values, aiming to make individuals converge towards fitter individuals and improve the average fitness of the population.

When f' is greater than the average fitness of the population, it is important to consider the overall average fitness of the population. The higher the ratio of the average fitness of all individuals in the population to the fitness of the best individual, the higher the level of evolution in the population. Therefore, individuals in the population should be assigned higher crossover and mutation probabilities to explore a larger search space. Additionally, the crossover and mutation probabilities of individuals in the population should be weighted based on their individual fitness f'. A higher f' value indicates a higher-quality individual or chromosome that should be preserved. Therefore, relatively lower probabilities should be assigned until f reaches the fitness of the best individual in the population, minimising the mutation and crossover probabilities. Conversely, if f' is smaller, higher probabilities should be assigned. This approach implements an adaptive genetic algorithm that considers the overall average fitness, the optimal level, and the individual level within the population.

2. Crossover operation

In this algorithm, the crossover operation searches for the common gene points between paired chromosomes as crossover points. If there are multiple common gene points, one of them is randomly chosen as the crossover point. Then, the structure of the chromosome is exchanged after the crossover point.

3. Mutation operation

The mutation operation in this algorithm utilises the midpoint interpolation method used during the population initialisation process. When an individual needs to undergo mutation, two points on the chromosome are randomly selected as mutation points. The path between these two points is removed, and a continuous path between the mutation points is regenerated using the midpoint interpolation method. This method overcomes the drawback of traditional mutation operators randomly mutating to an unfeasible solution at a specific point, thereby increasing the computational efficiency of the algorithm and reducing the computational overhead.

3.5 Deleting operator

During the iterative search process of the algorithm, the path followed by the robot is likely to contain detours, where it goes in a circle and returns to a specific point before moving forward. In the individual encoding, this is reflected as having repeated path points. By using a deletion operator, the paths between the repeated path points, as well as one of the repeated path points, can be removed. This eliminates unnecessary paths, not only reducing computational complexity but also increasing the fitness of the individual. This is because the removal of redundant paths results in a shorter total path length.

3.6 Optimisation operator

The optimisation operator is applied to each path, where every three points are considered as a group. Let us assume that the i-th path point in path D is denoted as $D(i)$. The optimisation operator makes a judgement on $D(i\text{-}1)$ and $D(i+1)$, and if the connecting line of $D(i\text{-}1)$ and $D(i+1)$ does not pass through the obstacle, then $D(i)$ is deleted and only $D(i+1)$ and $D(i+1)$ are left. If the connecting line passes through the obstacle, continue to check the next point $D(i+1)$ as the centre point until all path points in all paths are checked.

According to the grid map, if a line passes through an obstacle, the distance from the centre point of the obstacle to the line must be less than $\sqrt{2}$. Therefore, the formula for determining whether the line between $D(i\text{-}1)$ and $D(i+1)$ passes through the obstacle is:

$$dl = \frac{|(y_2 - y_1)x_k + (x_1 - x_2)y_k + (x_2y_1 - x_1y_2)|}{\sqrt{(y_2 - y_1)^2 + (x_1 - x_2)^2}} \tag{11}$$

where y_2 and y_1 denote the row coordinates of $D(i+1)$ and $D(i-1)$, respectively, and x_2 and x_1 denote the column coordinates of $D(i+1)$ and $D(i-1)$, respectively. x_k and y_k are the column coordinates and row coordinates of the obstacle. When dl is less than $\sqrt{2}$, it means that the line of $D(i-1)$ and $D(i+1)$ crosses the obstacle, otherwise the line does not pass through the obstacle, and $D(i)$ can be deleted.

3.7 Reference populations

During the process of the genetic algorithm, all operations are based on the initial population generated before the algorithm starts. Selection, crossover, and mutation all act upon the initial population. It can be said that the quality of the initial population largely determines the efficiency of the entire algorithm. If the initial population contains more high-quality information that is closer to the optimal solution, the algorithm can quickly approach the optimal solution and converge at a fast rate. However, this is only a rare occurrence, as most of the time the initial population generated is far from the optimal solution and requires multiple iterations to make progress. Additionally, the initial population to be generated is near a local optimum, causing the algorithm to linger around the local optimum for a considerable amount of time. It relies on crossover and mutation operators escaping premature convergence and approaching the next optimal solution. Therefore, generating the initial population with a higher probability can significantly improve the efficiency of the algorithm.

In the proposed algorithm, this article introduces the concept of a reference population. When the initial population A is generated, a second initial population is simultaneously generated as a reference population, referred to as population B. Both populations evolve and iterate together, and all genetic operators act both on population A and population B. Every t generation, population B undergoes crossover with population A. The fittest n individuals in population B are transferred to population A, replacing the n individuals with the lowest fitness values in population A. At the same time, population A replaces the n individuals with the lowest fitness values in population B. Fig. 3 illustrates the specific process of interchanging individuals between the two populations.

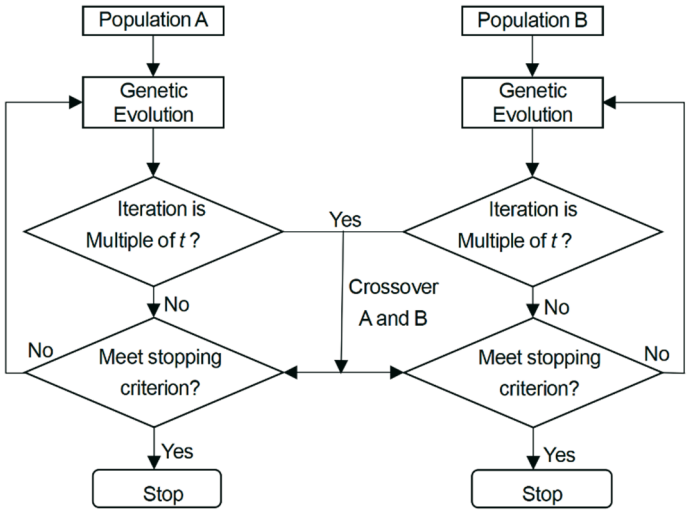


Fig. 3 Flow chart of crossover between reference and main populations

4. Simulation experiments and results analysis

In this paper, the crossover and mutation probabilities are designed as an adaptive function according to the population fitness based on the traditional genetic algorithm for the path-planning problem, and the deletion operator, optimisation operator and reference population are added to the iterative process of the algorithm.

To verify the performance and practicality of the proposed improved genetic algorithm for path planning problems, simulation experiments are conducted on Map 1 for each step of improvement in the improved genetic algorithm, and a comparison and analysis are made with the previous step of improvement, respectively. Fig. 4 shows that the bottom-left corner is the starting point, and the top-right corner is the end point.

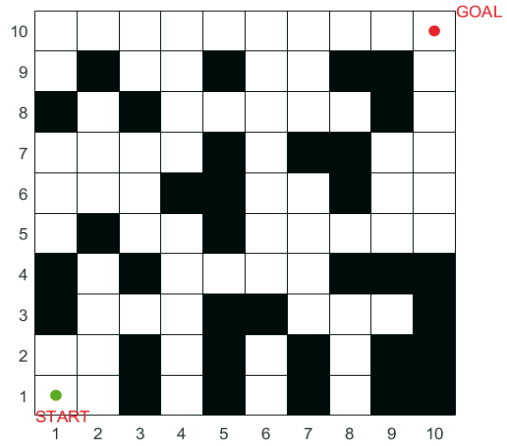


Fig. 4 Map 1 for Simulation and Analysis

4.1 Adaptive genetic algorithm simulation analysis

The population size of the traditional genetic algorithm is set at 300, the number of iterations at 150, the crossover probability at 0.8, and the mutation probability at 0.05. The population size of the genetic algorithm containing the adaptive operators is set at 300, the number of iterations at 150, the crossover probability at [0.5, 0.8], and the mutation probability at [0.01, 0.1]. Figure 6 shows the simulation results of the algorithms on Map 1.

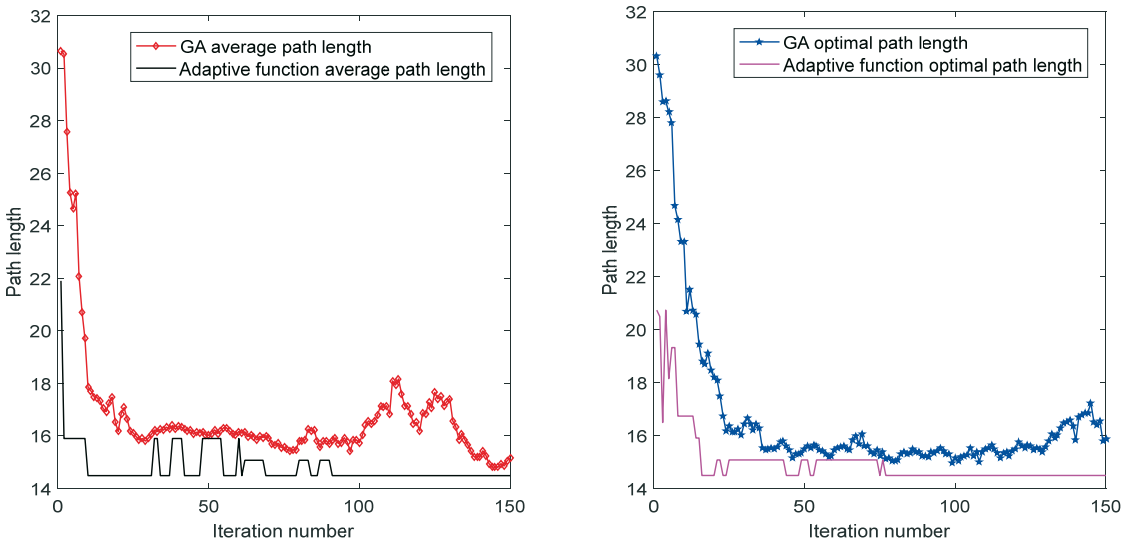


Fig. 5 Plot of path variation between adaptive GA and traditional GA under specified parameters

From Fig. 5, it can be inferred that the application of the proposed adaptive probability model in the genetic algorithm leads to a faster convergence speed. Additionally, in the adaptive genetic algorithm, the average fitness level of the population is closer to the optimal solution. This observation suggests that utilising the adaptive probability model improves the performance of the genetic algorithm. By dynamically adjusting the crossover and mutation probabilities based on the fitness of individuals within the population, the algorithm can adapt more effectively to the problem at hand, resulting in quicker convergence and a population that is closer to the optimal solution.

4.2 Simulation analysis of improved GA with deletion and optimisation operators

The deletion and optimisation operators are incorporated into the genetic algorithm based on the adaptive crossover and mutation simulated in Section 3.1, and the simulation results are compared with the adaptive algorithm in Fig 6 and Fig 7.

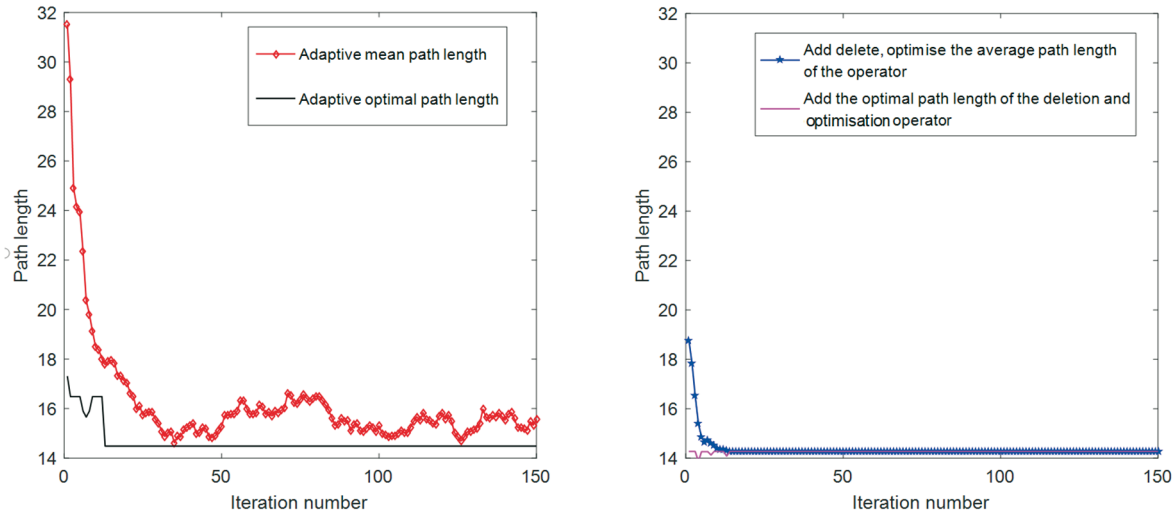


Fig. 6 Comparison of path changes after two operators with the adaptive algorithm (Map 1)

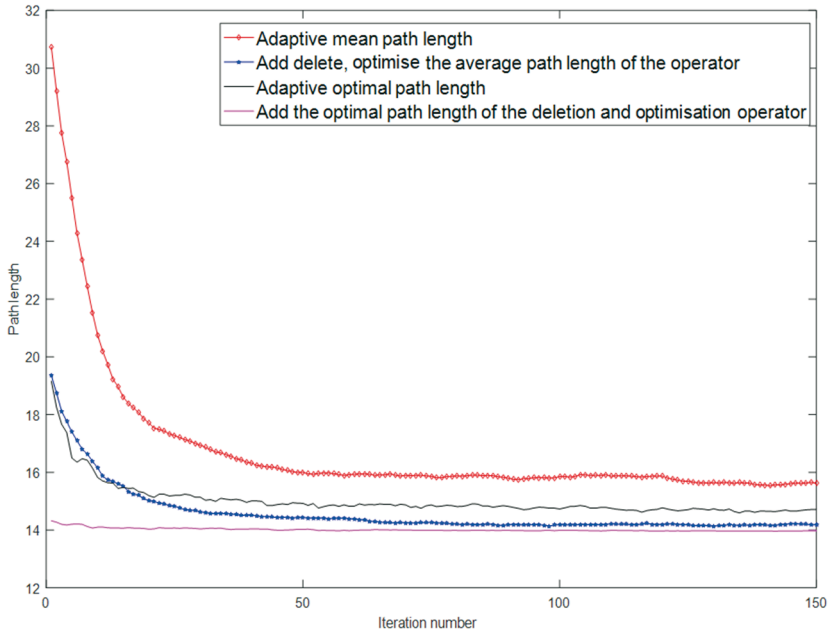


Fig. 7 Plot of the mean values of the two algorithms run one hundred times (Map 1)

Among them, Fig. 6 shows the curve of the change in the population path length for the adaptive genetic algorithm and the genetic algorithm with the addition of deletion and optimisation operators, which are both based on Map 1 and vary with the number of iterations. In Fig. 6, it can be observed that after adding deletion and optimisation operators to the adaptive genetic algorithm, the path length of the obtained optimal solution becomes shorter. The stand-alone adaptive genetic algorithm yields an optimal path length of 14.3 in Map 1, while the addition of deletion and optimisation operators reduces the optimal path length to 13.8, a decrease of 4%. Fig. 7 represents the average performance of both algorithms after running 100 times separately. The deletion and optimisation operators reduce the optimal path length by 6% and lead to faster convergence.

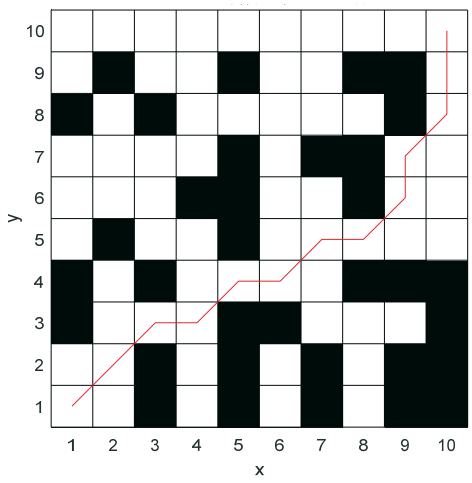


Fig. 8 Adaptive genetic algorithm path diagram

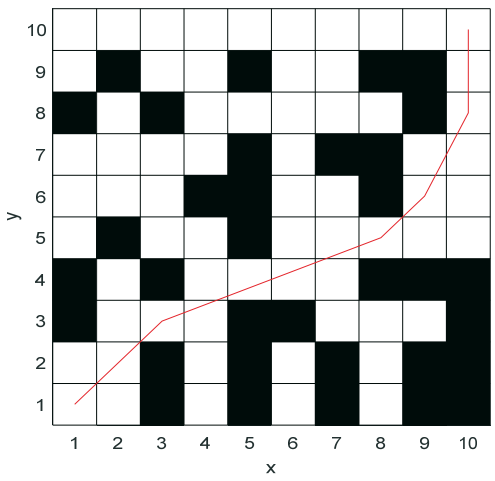


Fig. 9 Path diagram of genetic algorithm with deletion and optimisation operator

Fig. 8 and Fig. 9 show that the optimal path curve obtained by the regular adaptive genetic algorithm is not smooth enough. It contains multiple corners along the path. However, after adding the two operators, the smoothness of the path is significantly improved. This characteristic is more evident in areas of the map with fewer obstacles.

4.3 Comparison of improved GA based path planning with traditional GA

In this section, a comparative simulation experiment is carried out in the path-planning problem using a randomly generated 20*20 2D raster map (as shown in Fig 10:(a)) using an improved Genetic Algorithm(IGA, Genetic algorithm(GA), Particle Swarm Algorithm, Artificial Fish Swarm Algorithm(AFSA) and Grey Wolf Optimisation(GWO).

The population size for the traditional genetic algorithm is set at 300, the number of iterations at 150, the crossover probability at 0.8, and the mutation probability at 0.05. For the proposed improved genetic algorithm, the population size is set at 300, the number of iterations at 150, the crossover probability is set between 0.5 and 0.8, and the mutation probability between 0.01 and 0.1. Additionally, let t=3, and the inter-population crossover operation is stopped when t>100.

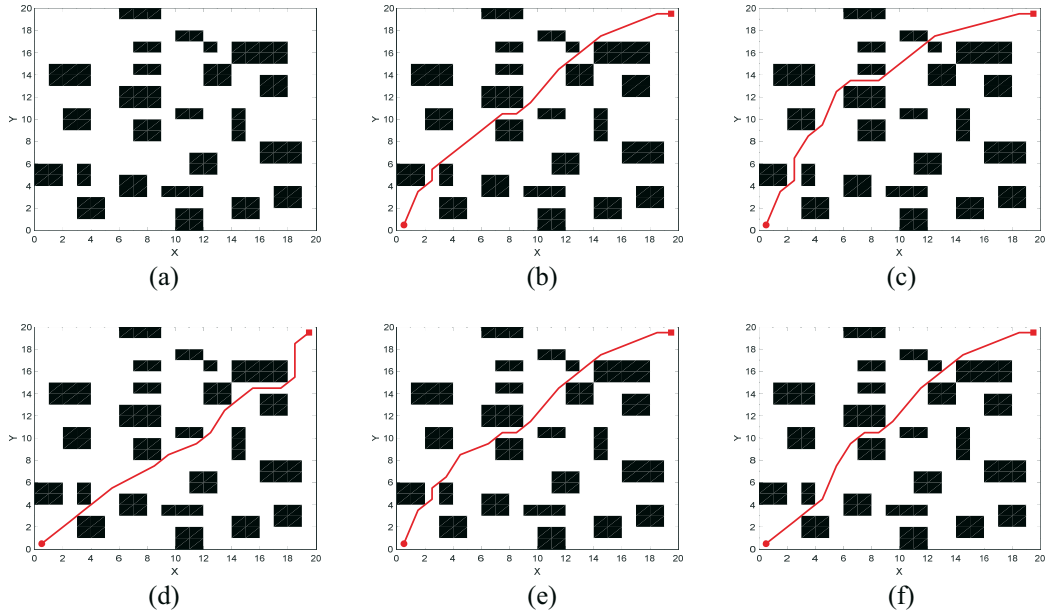


Fig. 10 (a) Random map (20×20 2D), (b) IGA, (c) GA, (d) PSO, (e) AFSA, (f) GWO

Table 1 Results of classical algorithms

planner	path length(m)	time(s)	success rate(%)
IGA	28.0122	4.203	99.8
GA	31.1782	6.952	93.5
PSO	31.8821	6.125	92.4
AFSA	30.3541	6.224	90.6
GWO	30.9512	5.549	94.5

Figure 10 shows the trajectory of the five algorithms running on the map Fig 11 (a), and Table 1 shows a comparison of the average path length, running time, and success rate of the five algorithms after running 100 times. It can be seen that the performance of the improved genetic algorithm is significantly better than the other algorithms. Compared with the other algorithms, IGA reduces the average path length by about 10%-14%, the running time by 23%-38%, and the success rate by 8%-10%.

To further verify and improve the performance of the genetic algorithm, Random maps Nathan Sturtevant/HOG2 (15% obstacles) of Nathan Sturtevant's Moving AI Lab public data set were selected, and 2D grids of 40×40 were selected as the experimental map (Fig 11:(a)).

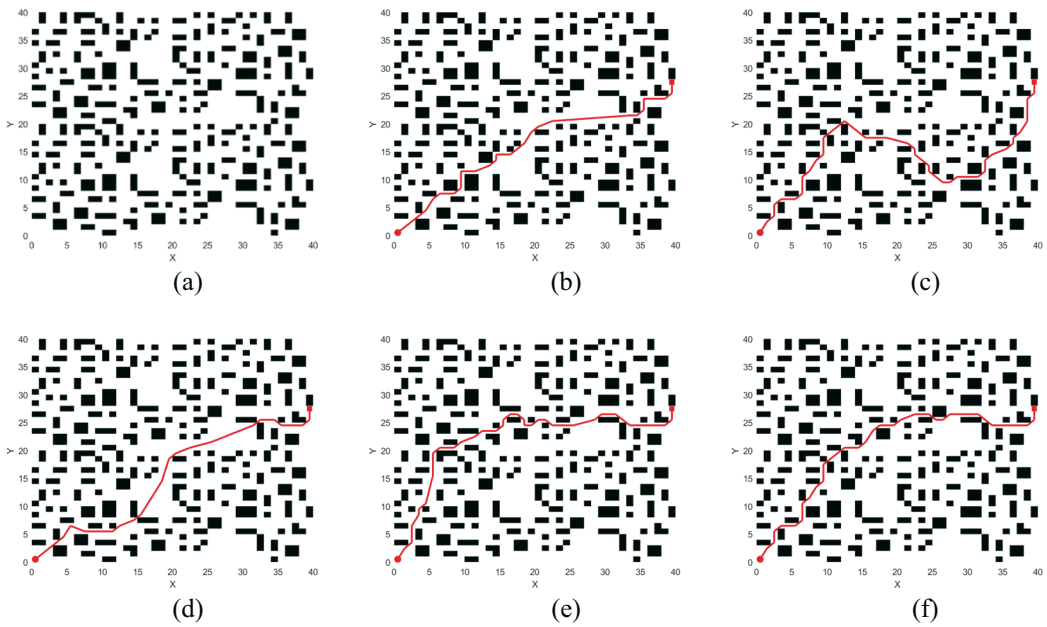


Fig. 11 (a) Random map (40×40 2D grids), (b) IGA, (c) GA, (d) PSO, (e) AFSA, (f) GWO

Table 2 Results of classical algorithms

planner	path length (m)	time (s)	success rate (%)
IGA	54.2362	10.624	99.7
GA	68.8209	16.257	89.8
PSO	59.6826	14.354	91.2
AFSA	61.2684	15.349	90.1
GWO	60.8209	15.756	92.4

Figure 11 shows the trajectory of the five algorithms running on the public dataset excerpt map Fig 11 (a), and Table 2 shows a comparison of the average path length, running time, and

success rate after 100 runs of the five algorithms. Compared with the other algorithms, IGA reduces the average path length by about 10%-21%, the running time by 28%-38%, and the success rate by 8%-11%.

5. Conclusion

The present study employs a grid-based method to establish an environment model for the robot's workspace. By satisfying the constraint of avoiding all obstacles, the path-planning problem is investigated using a genetic algorithm. Through the use of the proposed improved genetic algorithm in this study, an optimal path for the robot's movement is obtained.

The proposed model incorporates an adaptive crossover and mutation probability model. It also introduces the deletion and optimisation of genetic operators, significantly improving the generation of the initial population. Furthermore, the concept of reference population is introduced to avoid unnecessary computations and premature convergence, which are commonly associated with traditional genetic algorithms. The simulation results demonstrate that the performance of this algorithm is significantly superior to that of the traditional genetic algorithm. The existence of optimisation operators promotes smoother paths. The reference population greatly accelerates the convergence speed of the algorithm, while enhancing its stability. However, this algorithm still has room for improvement, particularly regarding the calculation aspect of the proposed reference population. The parameter t currently requires careful analysis based on the specific problem at hand and cannot adapt autonomously to changes in the population.

Acknowledgment

This work was supported by the Talent Introduction Project for Guangdong University of Petrochemical Technology under Grant 2020rc32. It was also supported by the Maoming City Science and Technology Plan Project under Grant 2021002, Guangdong Province Science and Technology Innovation Strategic Special Funding under Grant 2023S003042, and the National Key Research and Development Programme of China under Grant 2023YFB4704000.

REFERENCES

[1] Zhou S, Liu J, Wang Z, et al. Research on Design Optimization and Simulation of Regenerative Braking Control Strategy for Pure Electric Vehicle Based on EMB Systems. Transactions of FAMENA, 2023, 47(4): 33-49. https://doi.org/10.21278/TOF.474045522

[2] Vijayan S, Parameshwaran Pillai T. Application of a Machine Learning Algorithm in a Multi Stage Production System. Transactions of FAMENA, 2022, 46(1): 91-102. https://doi.org/10.21278/TOF.461033121

[3] Dijkstra E W. A note on two problems in connexion with graphs. Numerische mathematik, 1959, 1(1): 269-271. https://doi.org/10.1007/BF01386390

[4] Tanglay Onur, Dadario Nicholas B, Chong Elizabeth H. N. Tang Si Jie, Young Isabella M. Sughrue Michael E. Graph Theory Measures and Their Application to Neurosurgical Eloquence. Cancers, 2023, 15(2):556. https://doi.org/10.3390/cancers15020556

[5] Hart Peter, Nilsson Nils, Raphael Bertram. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2):100-107. https://doi.org/10.1109/TSSC.1968.300136

[6] Rui Song, Yuanchang Liu, Richard Bucknall. Smoothed A* algorithm for practical unmanned surface vehicle path planning. Applied Ocean Research, 2019, 83:9-20. https://doi.org/10.1016/j.apor.2018.12.001

[7] Fusic S. J., Ramkumar P., Hariharan K. Path planning of robot using modified Dijkstra Algorithm. National Power Engineering Conference (NPEC), Madurai, India, 2018, pp. 1-5.

[8] Hliwa H, Daoud M, Abdulrahman N, et al. Optimal Path Planning of Mobile Robot Using HybridTabu Search-Firefly Algorithm. International Journal of Computer Science Trends and Technology, 2018, 6(6): 7-15.

[9] Devanathan C, SureshBabu A. Multi objective optimization of process parameters by firefly algorithm during the friction stir welding of metal matrix composites. Transactions of FAMENA, 2021, 45(1): 117-128. https://doi.org/10.21278/TOF.451018520

[10] Zhou J, Gao J, Wang K, et al. Design optimization of a disc brake based on a multi-objective optimization algorithm and analytic hierarchy process method. Transactions of FAMENA, 2018, 42(4): 25-42. https://doi.org/10.21278/TOF.42403

[11] Cao J.. Robot Global Path Planning Based on an Improved Ant Colony Algorithm. Journal of Computer & Communications, 2016, 04(2):11-19. https://doi.org/10.4236/jcc.2016.42002

[12] Kamalova, Navruzov, Qian, et al. Multi-Robot Exploration Based on Multi-Objective Grey Wolf Optimizer. Applied Sciences, 2019, 9(14):2931. https://doi.org/10.3390/app9142931

[13] Brand M., Yu X. H.. Autonomous robot path optimization using firefly algorithm. 2013 International Conference on Machine Learning and Cybernetics. IEEE, 2013, 3: 1028-1032. https://doi.org/10.1109/ICMLC.2013.6890747

[14] Mbl Saraswathi, Gunji Bala Murali, B. B. V L. Deepak. Optimal Path Planning of Mobile Robot Using Hybrid Cuckoo Search-Bat Algorithm. Procedia Computer Science, 2018, 133. https://doi.org/10.1016/j.procs.2018.07.064

[15] Hart E, Ross P. A Systematic Investigation of GA Performance on Job Shop Scheduling Problems. Real-World Applications of Evolutionary Computing, 2000, Vol.1803:280-289. https://doi.org/10.1007/3-540-45561-2_27

[16] Madasamy B, Balasubramaniam P. Enhanced load balanced clustering technique for VANET using location aware genetic algorithm. Promet-Traffic&Transportation, 2022, 34(1): 39-52. https://doi.org/10.7307/ptt.v34i1.3785

[17] Ding F, Liu S, Li X. Pareto optimality of centralized procurement based on genetic algorithm. Tehnički vjesnik, 2022, 29(6): 2058-2066. https://doi.org/10.17559/TV-20220723180901

[18] PG Luan. NT Thinh. Hybrid genetic algorithm based smooth global-path planning fora mobile robot. Mechanics Based Design of Structures and Machines. 2021, 32(08): 1-17.

[19] Mohamed E, Alaa T, Aboul Ella H. Bezier Curve Based Path Planning in a Dynamic Field Using Modified Genetic Algorithm. Journal of Computational Science. 2017, 26(03): 1-41

[20] A. López-González, J . A. Meda Campaña, E. G. Hernández Martínez et al. Multi robot distance based formation using Parallel Genetic Algorithm. Applied Soft Computing Journal. 2019, 16(05): 1-25.

[21] Lee H Y, Shin H, Chae J. Path Planning for Mobile Agents Using a Genetic Algorithm with a Direction Guided Factor. Electronics, 2018, 7(10):212-232. https://doi.org/10.3390/electronics7100212

[22] Han Z, Wang D, F, et al. Multi-AGV path planning with double-path constraints by using an improved genetic algorithm. PLOS Clinical Trials, 2017, 12(7): e018174. https://doi.org/10.1371/journal.pone.0181747

[23] Lee J, Kim D W. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. Information Sciences: An International Journal, 2016, Vol. 332:1-18. https://doi.org/10.1016/j.ins.2015.11.004

Submitted: 15.07.2023

Accepted: 26.02.2024

Jixin Liu
Yanbin Cai
Yue Cao*
School of Automation, Guangdong University of Petrochemical Technology
Maoming 525000, China
*Corresponding author
caoyue2000@gdupt.edu.cn