# Genetic Algorithm Based on Grid Maps for Solving Robot Path Planning Problem

Yin-Yin Bao, Yu Liu*, Jie-Sheng Wang, Jia-Xu Liu

**Abstract—A genetic algorithm (GA) based on the grid map is proposed to solve the robot path planning problem. It divides the grid map into two categories: obstacle and non-obstacle. It then generates a series of paths using GA that continuously improves their fitness to find feasible paths. To determine the efficiency of the proposed algorithm, various factors are adjusted, such as crossover probability, variation probability, and path smoothness weight in different maps. The objective was to find the optimal number of generations required to reach a relatively stable state while also ensuring an average proximity to the optimal path. Experimental results obtained through simulation on MATLAB software demonstrate that the algorithm achieves a high performance and robustness in finding feasible paths within a relatively short period. Moreover, it exhibits good scalability and adaptability when applied to different environments. Consequently, the algorithm offers an effective solution to the robot path planning problem and has substantial practical application value. It can serve as a valuable reference tool for robot path planning.**

***Index Terms—path planning, genetic algorithm, grid method***

## I. Introduction

The rapid development of robotics has led to the widespread use of robots in various aspects of daily life, industrial manufacturing, and healthcare. The primary objective of robot path planning is to find the optimal path within a given environment, enabling the robot to reach a specified target point within a set time frame [1]. This problem involves the robot's motion capabilities, environmental complexity, motion constraints, and sensor errors. Solving the robot path planning problem enhances the robot's autonomy and intelligence and enables it to adapt more flexibly to different environments and task requirements. Its practical applications span multiple fields, including industrial manufacturing, medical and healthcare, and military combat. In industrial manufacturing, addressing the robot path planning problem enhances production efficiency and quality while reducing costs and the need for human intervention. It improves surgical accuracy and safety in healthcare, mitigating surgical risks and patient suffering. Additionally, solving the robot path planning problem in military combat enhances combat efficiency and safety and minimizes casualties and losses among soldiers [2-3]. Currently, research on robot path planning primarily focuses on methods based on graph theory, search algorithms, and optimization algorithms. Among these, optimization algorithm-based methods demonstrate strengths in global search capabilities and self-adaptability, making them particularly effective in solving complex problems. The robot path planning problem is a complex optimization problem involving considerations such as the robot's motion capabilities, environmental complexity, motion constraints, and sensor errors [4-6]. Therefore, addressing the robot path planning problem has become a significant research focus.

Genetic algorithms have been widely applied to mobile robot path-planning problems in the past decade, leveraging the principles of evolution in nature. Genetic algorithms offer several advantages, such as global search capability, adaptivity, and parallelism, making them effective in solving these complex problems [7]. Consequently, the application of genetic algorithms in robot path planning has emerged as a prominent research area in recent years. In domestic research, several advancements have been made in applying genetic algorithms to robot path planning. For instance, Ref. [8] applied a genetic algorithm in joint space to plan the motion path of each robot joint while considering kinematic and dynamic constraints. This planning approach can readily apply the results to real-time robot control. Ming Zhou et al. proposed a genetic algorithm-based method for robot path planning in continuous space. They employed link graph modeling in the planning space to initially search for viable paths using advanced algorithms in graph theory. Subsequently, they used a genetic algorithm to optimize the path points, gradually obtaining an improved walking route. The chromosome encoding in this method prevents the generation of invalid paths, and basic genetic algorithms alone can complete the path planning. However, in complex environments with numerous obstacles, establishing link graphs can be challenging [9]. Chen et al. investigated a genetic algorithm solution for path-planning problems in complex environments. They designed efficient genetic path operators tailored to the characteristics of complex environments and proposed a novel calculation method to measure the fitness of individual paths. Through experiments, they demonstrated the algorithm's strong robustness [10]. Lei Yamin et al. adopted the grid method to represent the robot's working

Manuscript received May 29, 2023; revised September 5, 2023. This work was supported by the Basic Scientific Research Project of Institution of Higher Learning of Liaoning Province (Grant No. LJKZ0293), and Postgraduate Education Reform Project of Liaoning Province (Grant No. LNYJG2022137).

Yin-Yin Bao is a postgraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: bao17640667213@163.com).

Yu Liu is an associate professor of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (Corresponding author, phone: 86-0412-2538246; fax: 86-0412-2538244; lnasacl@126.com).

Jie-Sheng Wang is a professor of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: wang_jiesheng@126.com).

Jia-Xu Liu is a postgraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: 1849226542@qq.com).

environment model, encoding it with ordinal numbers. They used a combination of right-angle coordinates and ordinal numbers to generate the initial path population via genetic algorithms. The paths were then optimized to find the shortest route. Deletion and insertion operators were added to meet the obstacle avoidance requirements in path planning. Simulation results confirmed the efficacy and feasibility of genetic algorithms for obstacle avoidance and path planning [11].

Wang Jun et al. completed the robot's path planning in discrete space using a genetic algorithm, yielding satisfactory simulation results. However, it should be noted that this planning relies on a deterministic environment model, where the positions of obstacles in the workspace are known and fixed [12]. In addition to genetic algorithms, other optimization algorithms have contributed to solving robot path-planning problems. Chumming Ren et al. focused on path planning in robot navigation and proposed the GAA algorithm, which improves upon the ant colony algorithm by incorporating genetic algorithm concepts. The algorithm employs grid partitioning to describe the robot's environment. Simulation results demonstrate that this method effectively enhances path search, optimization, and smoothing [13]. Rosas et al. introduced the memEAPF method, which combines membrane computation with a genetic algorithm (a membrane heuristic evolutionary algorithm based on a layered membrane structure) and an artificial potential field approach to finding parameters for generating feasible and safe paths for mobile robot path planning [14]. Yang et al. utilized a deep Q-network (DQN) algorithm, a form of deep reinforcement learning, to address multi-robot path planning problems. The improved DQN algorithm combines Q-learning, empirical replay mechanisms, and productive body-based neural network techniques to achieve faster convergence and learn path-planning solutions more efficiently, enhancing multi-robot path-planning efficiency [15].

This paper proposes a genetic algorithm based on a grid map to address the robot path planning problem. The paper is organized as follows: Section II introduces the fundamental principles of the grid map-based genetic algorithm for solving robot path planning problems. Section III presents experimental simulations and a detailed analysis of the results. Finally, the conclusion of the paper is provided.

## II. Genetic Algorithm Based on Grid Map for Solving Robot Path Planning Problem

### *A. Genetic algorithm learning algorithm*

1) Principle of the algorithm

Algorithms are derived from populations consisting of a specific number of individuals and many chromosomes. In genetic algorithms, the reproductive process of a population begins with the presence of genetics, including genetic algorithms [16]. This population represents a collection of potential solutions wherein each individual carries a unique expression through its chromosomes. Chromosomes are the primary carriers of genetic information materials, consisting of multiple genes that determine the individual's external characteristics, such as hair color (black vs. yellow) or eyelid type (single vs. double). Binary coding is often employed to simplify the coding of genes, where a primitive population initially selects and gradually evolves the desired phenomenon through generations based on the principle of survival of the fittest. In each generation, individuals are selected based on the desired criteria. At the same time, genetic inheritance laws are applied to allow for free combination, crossover, and mutation, creating a new population with diverse sets of individuals. This iterative process enables the population to become better adapted to the environment. The optimal individual from the final generation can be an approximate optimal solution to the problem. For problems related to finding the extreme value of a function, the process can be mathematically described as follows:

$$\begin{cases} max\, f\,(x); \\ x \in R; R \in U \end{cases} \tag{1}$$

where, $x$ is the independent variable $\max f(x)$ is the functional equation， and $x \in R; R \in U$ is the condition. Where $U$ is the entire set of real numbers. $R$ is the subset of $U$. $R$ is the set of all solutions. $x$ is the solution of the functional equation, so $x$ belongs to the set.

*2)* Principle of the algorithm

The core concept of genetic algorithms lies in inheritance, where genetic information is transmitted from parents to offspring. Therefore, the crossover approach and the probability of variation play a crucial role in this process. The crossover approach is a method of transferring genes from parents to offspring, significantly contributing to the genetic algorithm's convergence. By combining genetic material from both parents, the crossover operator promotes the exchange of genetic information, producing diverse offspring. On the other hand, the variation operator enhances the genetic diversity of the offspring population. It introduces small random changes or mutations to the genetic material inherited from the parents. This combination of crossover and variation helps improve the offspring population's characteristics and ultimately meets the desired objectives..

### *B. Flow and formulation of the algorithm*

Genetic algorithms are typically employed to solve solution optimization and search problems. They draw inspiration from evolutionary phenomena and unique characteristics observed in various natural species. Through continuous study and development, genetic algorithms simulate and utilize biological concepts such as exchange, mutation, and natural selection. However, it is essential to note that improper selection of parameters such as population size, mutation probability, crossover probability, and path length ratio could result in convergence to a local optimum rather than the desired global optimum. The primary operational procedure of a genetic algorithm can be summarized as follows:

(1) Initialize the population: A population is randomly generated, and several individuals are randomly generated within the population. A finite number of iterations is also limited.

(2) Individual selection: Randomly select an individual in the population and observe its ability to survive and

reproduce.

(3) Selection operation: The selection operator is applied to the population to select a suitable target for crossover plied to the population. The crossover operator plays a central role in the genetic algorithm, and its probability is between 0 and 1.

(5) Variation operation: The variation operator is applied to the population. That is, the probability of a change in the genes on the chromosomes of the individuals in the population is between 0 and 1.

(6) Final judgment: judge whether the result satisfies the demand close to the optimal solution. Termination procedure.

The application of the genetic algorithm involves several crucial operations, including coding, fitness evaluation, and initial population selection. These factors significantly impact the outcomes of the experiments.

1) Individual coding

Coding is a genetic representation of a viable solution. Its coding is the unique numbering of an individual or a chromosome of a population. Thus, it is distinguished from other individuals or chromosomes.

Binary coding, the most crucial feature of this method, is the simplicity of programming. The principle is that the individuals in the population are converted into a 0-1 string and then operated on separately. For $n$-dimensional $f(x)$ , $x = (X_1, X_2, \cdots, X_n)$ and $x_i \in [u_i, v_i](i = 1,2, \cdots, n)$, each dimensional variable of length size 1, the total length of the encoding of $x$ is $L = \sum_{i=1}^{n} 11$, and the encoding space is

$$S^L = \{a_1, a_2, a_3, \cdots, a_k\} \tag{2}$$

Individual bit string structure on this space:

$$a_k = (a_{k1}^1, a_{k2}^1, a_{k3}^1, \cdots, a_{kl1}^2, a_{k1}^2, a_{k2}^2, a_{k3}^2, \cdots, a_{kl2}^2, a_{k1}^3, a_{k2}^3, a_{k2}^3, \cdots, a_{kl3}^3, \cdots, a_{k1}^n, a_{k2}^n, a_{k3}^n, \cdots, a_{kl1}^n), a_{kl}^i \in \{0,1\} \tag{3}$$

The binary bit string is $S_k$ ,The decoding function is $\Gamma^i\colon \{0,1\}^{1_i} \rightarrow [u_i, v_i]$ as follows.

$$X_i = \Gamma^i(a_k) = u_i + \frac{u_i - v_i}{2^{l_i - 1}} \left( \sum_{j=1}^{l_i} a_{kj}^i 2^{l_i - j} \right), i = 1,2, \cdots, n \tag{4}$$

Then the decoding function for the whole $S_k$ is

$$\Gamma = \Gamma^1 \times \Gamma^2 \times \Gamma^3 \times \cdots \Gamma^n \tag{5}$$

2) Initial population setting

(1) A primordial population is generated randomly. Where N represents the number of individuals in the population, define the set of chromosomes of the N individuals in the population as T. Then say: $S^N = \{\overline{X} = (X_1, X_2, X_3, \cdots, X_N), \quad X_N \in S(i \leq N)\}$ is the population space of T. A population is a collection of individuals or chromosomes through which each individual can represent an initial solution of an optimization function [17]. The reverse learning optimization approach is a standard initial population method [18]. In this approach, a population T(N) is randomly generated, and another new population OT(N) is generated based on the reverse learning strategy. The two populations are combined to obtain a new population P(N), where the N chromosomes of the initial population are and mutation.

(4) Crossover operation: The crossover operator is ap

made up of the best-adapted individuals [11]. A random method was used for binary coding. Number the individuals as (0-1). Also in the case of $X = (X_1, X_2, \cdots, X_i, \cdots, X_n)$, generate as follows:

$$X_i = \begin{cases} 1 & r > 0.5 \\ 0 & r \leq 0.5 \end{cases} \tag{6}$$

3) Adaptation value function

The adaptation value represents an individual's ability within a population to adapt and survive in a given environment and its potential to produce offspring. The fitness function in a genetic algorithm is commonly called the evaluation function. It serves as an indicator of the individuals' merit within the population. It is important to note that the optimization objective function typically involves both positive and negative aspects, and therefore, it is necessary to ensure a proper mapping relationship between them. So that the fitness should be non-negative, the selection function $T\colon g \rightarrow f$ is transformed so that for the optimal solution $x*$ satisfies.

$$maxf(x^*) = opt(x^*). x * \in [u, v] \tag{7}$$

If the solution space a point $f(x)$, can be converted to correspond to the degree of adaptation $F(x)$ to get to find the best, if the objective sought for the maximum problem, then the function needs to be constructed as

$$F(x) = \begin{cases} f(x) + C_{min} & if \quad f(x) + C_{min} \\ 0 & if \quad f(x) + C_{min} \end{cases} \tag{8}$$

where, $C_{min}$ is generally taken as a small value. $f(x)$ is the objective function.

(2) If the range is pre-selected with uncertainty for the threshold value, the relative pres-election range can be obtained from the maximum value of the objective function:

$$F(x) = \frac{1}{1+c-f(x)}, c \geq 0, c \geq f(x) \tag{9}$$

4) Crossover operations

Crossover manipulation is a crucial method for preserving populations in genetic algorithms. It combines desirable genes through cross-fertilization, resulting in new individuals with novel combinations. This strategy ensures the retention of desired genes. The crossover strategy is as follows.:

(1) One commonly used crossover strategy is the single-point crossover. This method requires a single exchange point. By selecting a gene from each chromosome and exchanging their positions, new individuals are created, incorporating the exchanged genes.

$$\begin{cases} s_1 = a_{11}, a_{12}, \cdots, a_{1l_1}, a_{1l_2}, \cdots, a_{1L} \\ s_2 = a_{21}, a_{22}, \cdots, a_{2l_1}, a_{2l_2}, \cdots, a_{2L} \end{cases} \tag{10}$$

If the selected intersection is $x \in \{1,2, ..., L - 1\}$,set $x$=1, then the next generation of individuals obtained after the intersection is

$$\begin{cases} {s_1}^1 = a_{11}, a_{12}, \cdots, a_{1l_1}, a_{1l_2}, \cdots, a_{1L} \\ {s_2}^1 = a_{21}, a_{22}, \cdots, a_{2l_1}, a_{2l_2}, \cdots, a_{2L} \end{cases} \tag{11}$$

(2) Multi-point crossover. The purpose is to increase the carrying information of individual chromosomes. The operation is a random multi-point crossover of selected individuals. Selection of location points:

$$\begin{cases} x_1, x_2, \cdots, x_k \in \{1,2,\cdots,L-1\}, x_k \le x_{k+1}, \\ k = 1,2,\cdots,k-1 \end{cases} \quad (12)$$

Dividing the L gene loci into $K+1$ gene loci set:

$$\begin{cases} Q_k = \{l_k, l_k+1, \ldots, l_{k+1}\}, k = 1,2,\ldots,K+1; \\ l_1 = 1; l_{k+2} = L+1 \end{cases} \quad (13)$$

The arithmetic form is:

$$O(p_c, k) = \begin{cases} a_{1l}{}^1 = a_{2i}, a_{2l}{}^1 = a_{1i}, i \in Q_k, & evennumber \\ a_{1l}{}^1 = a_{1i}, a_{2l}{}^1 = a_{2i}, & Other \end{cases} \quad (14)$$

Generation of new individuals:

$$\begin{cases} s_1{}^1 = a_{11}, a_{12}, \cdots, a_{1l_1}, a_{1l_2}, \cdots, a_{1L} \\ s_2{}^1 = a_{21}, a_{22}, \cdots, a_{2l_1}, a_{2l_2}, \cdots, a_{2L} \end{cases} \quad (15)$$

5) Mutation operations

The mutation operation substitutes the value of a part of the individual gene with the value of other alleles to combine into a new individual, which is also reversed for 0-1 codes [19-20]. The method is as follows:

By-position variation. The coder first codes the individuals. It is known that each gene has a small probability of mutation. The actual number is coded for individual $X = (x_1, x_2, x_3, \ldots, x_i, \ldots, x_n)$, and the gene will change in the restricted region. In by-site mutation, individuals are selected to increase or shorten changes in one of their genes $x_i (1 \le j \le n)$:

$$x_j{}^1 = x_k + \mu \quad (16)$$

Chromosome 1 after mutation. mutation step 2 is set according to the problem.

6) Selecting operations

The selection operation is an effective procedure aimed at preserving and safeguarding the genetic information encoded in the chromosomes of a population. This operation is designed to prevent the loss of individual genes and typically involves evaluating the fitness of individuals, thereby enhancing the overall convergence performance. One type of selection operation is adaptation value proportional selection, which is based on the relative fitness levels of individuals within the population. Through effective means, individuals are selected in proportion to their level of adaptation. For example, in a population of 2 individuals, their fitness level determines the probability of selecting a particular individual.

$$p_s(a_i) = \frac{f(a_i)}{\sum_{j=1}^{n} f(a_i)}, i = 1,2,\ldots,n \quad (17)$$

where, $\sum_{j=1}^{n} f(a_i)$ is the sum of all individual fitness $f(a_i)$ in population n. Dispersion in the offspring population is determined by (2-18). The expected number of individuals in the population to survive:

$$P(a_i) = n * p_s(a_i), i = 1,2,\ldots,n \quad (18)$$

The combination of these points allows for the construction of a flow chart shown in Fig. 1.

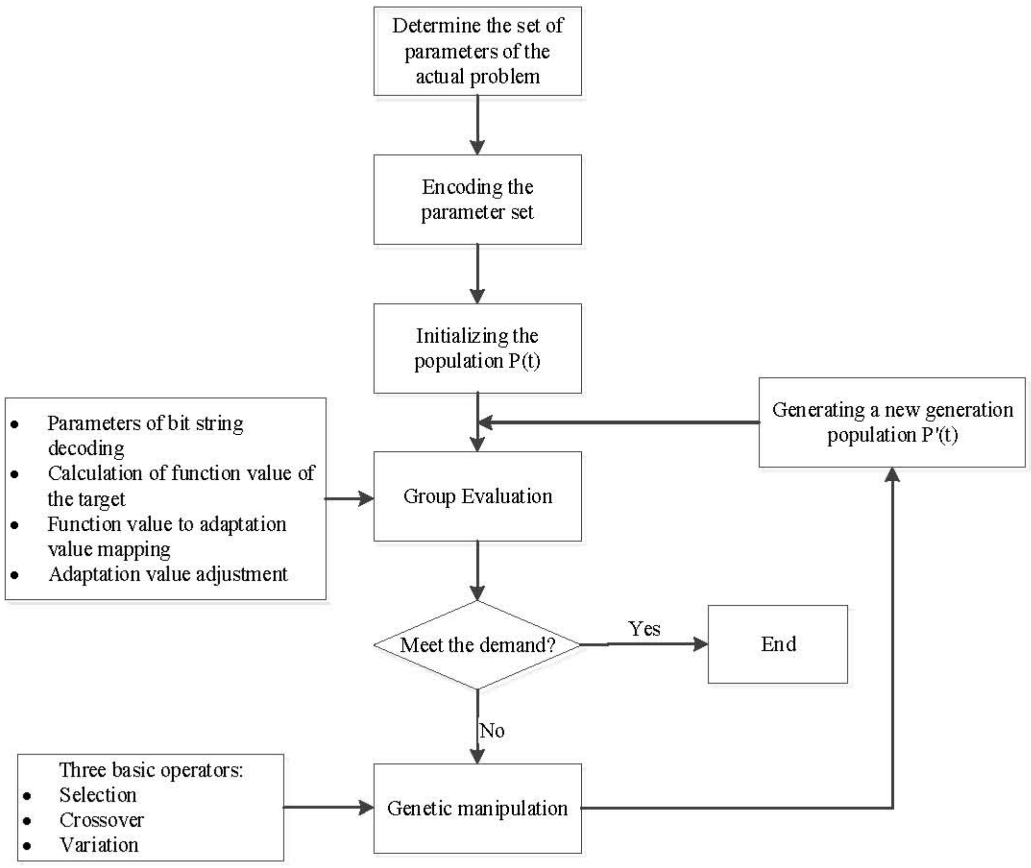

Determine the set of parameters of the actual problem
Encoding the parameter set
Initializing the population P(t)
Generating a new generation population P'(t)
Parameters of bit string decoding
Calculation of function value of the target
Function value to adaptation value mapping
Adaptation value adjustment
Group Evaluation
Meet the demand?
Yes
End
No
Three basic operators:
Selection
Crossover
Variation
Genetic manipulation

Fig. 1 Algorithm flow chart.

### C. *Other designs*

Map creation In this paper, the grid method establishes the robot's operating space. The robot needs to build a map before performing path planning. The smaller the number of grids in the map, the larger the area of small grids. When constructing the environment model, it is essential to consider various aspects to ensure accurate representation and address potential collision issues. The map's complexity level increases as the grids become more extensive and the grid area smaller, allowing for more precise information representation. However, this also leads to increased complexity in path searching. Therefore, the following points are taken into account during the building of the environment model:

(a) The height of buildings is not considered, and the robot's walking space is treated as a two-dimensional plane.

(b) The size and location of obstacles are defined, and no movable obstacles are present.

(c) The robot is treated as a point during planning.

A grid map space is created, where a two-dimensional rectangle represents the robot's operating space. If the obstacle area is smaller than the area of a single square grid, it can be considered a small square grid. If the obstacle area is more extensive, multiple square grids can represent it. Map 1 illustrates a two-dimensional map, where the white area represents the robot's motion space, and the black area represents the obstacle space.

We establish a two-dimensional coordinate system to construct the map, setting the origin at the lower left corner (1, 1). Each grid position can be precisely expressed using coordinates. We number the grids, starting from 0, beginning from the bottom left corner. A conversion formula is used to relate the numbering and the coordinates.

$$X = int\,(N/G_{SIZE}) + 1 \tag{19}$$

$$Y = (N\%G_{SIZE}) + 1 \tag{20}$$

where, $G_{SIZE}$ is the number of grids and int is an integer.

1) Code framework

The flow chart illustrating the general structure of the algorithm is displayed in Fig. 2. Well-designed code often follows a concise, step-by-step process, and the genetic algorithm is no exception. It is predominantly utilized for solving optimization problems as an intelligent and efficient optimization algorithm. The main steps of the genetic algorithm include population initialization, fitness function calculation, selection, crossover, and mutation. For instance, let us consider an example where the starting position is set to 0, the target position is 399, and the coordinates range from (1, 1) to (20, 20). The initial population requires the generation of multiple random feasible paths. Generating feasible paths involves two primary steps. Firstly, the robot generates an interrupted path by instructing it to record the spaces and trajectory it traverses while adhering to the constraint that at least one grid in each row and column must be free space. This results in an interrupted path. The next step involves connecting these interrupted paths into a continuous path.

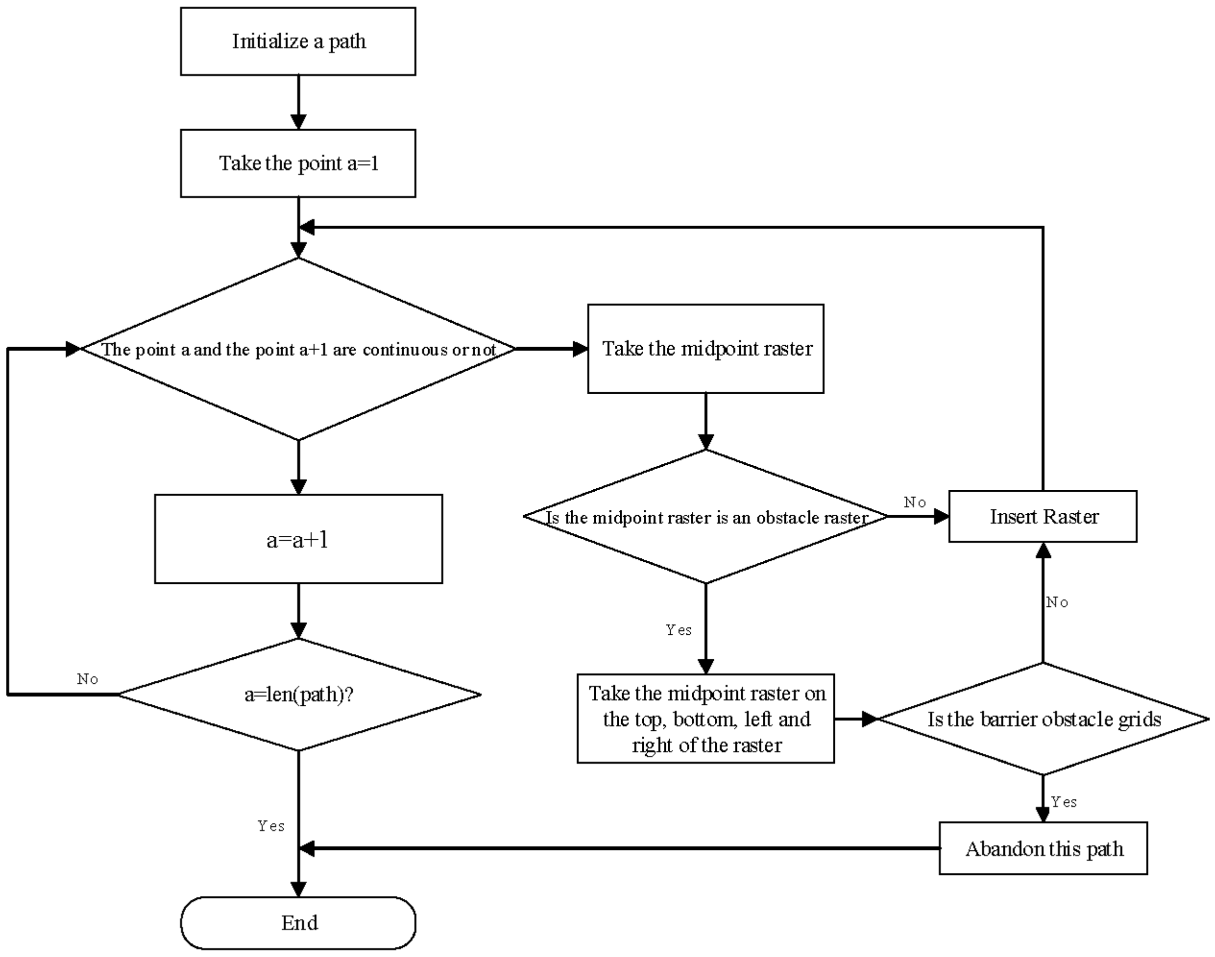

Initialize a path
Take the point a=1
The point a and the point a+1 are continuous or not
Take the midpoint raster
a=a+1
Is the midpoint raster is an obstacle raster
No
Insert Raster
No
Yes
No
a=len(path)?
Take the midpoint raster on the top, bottom, left and right of the raster
Is the barrier obstacle grids
Yes
Yes
Abandon this path
End

Fig. 2 Algorithm flow chart.

To determine the continuity of each taken out grid, one can employ the following method of judgment.

$$D = max\{abs(x^{a+1} - x^a), abs(y^{a+1} - y^a)\} \quad (21)$$

If $D$=1, The grid connects if not disconnected, otherwise the opposite. If there is discontinuity, we take the midpoint of the two points. The calculation is as follows.

$$X_{\text{new}} = \text{int}\left(\frac{x_{a+1}+x_a}{2}\right) \quad (22)$$

$$Y_{\text{new}} = \text{int}\left(\frac{y_{a+1}+y_a}{2}\right) \quad (23)$$

If the midpoint grid is not an obstacle, you must determine whether someone has already selected this grid. If the selected grid is unoccupied and not part of the current path, you should add it. However, you must select a new adjacent grid if it is an obstacle. You should delete the path if no available unoccupied grids are adjacent to the obstacle. When selecting a new grid that meets the necessary conditions, insert it between two disconnected grids in the path. The process should then continue by checking if the new grid is contiguous with the existing path. You should take the midpoint grid and continue the evaluation if it is not contiguous. This cycle should continue until someone fills the entire path with contiguous grids.

## III. ALGORITHM SIMULATION AND RESULT ANALYSIS

This paper focuses on modifying various parameters, including crossover probability, variation probability, and path smoothness weight, to determine the number of generations required to reach a relatively stable state while approaching the optimal path on average. The experimental simulation system used for this study includes the Windows 10 flagship operating system, Intel i7-8 processor, 8GB RAM, and MATLAB 2017b software.

### *A. Map construction*

The maps were created using MATLAB 2017b. Four maps, represented in Fig. 3, were constructed with 400 grids ranging from coordinates (1, 1) to (20, 20). The black grids represent obstacles, while the white grids indicate free space.

### *B. Simulation experiment and result analysis*

(1) Parameters were set as follows: path length weighting (a) = 1, path smoothness weighting (b) = 7, population number = 200, crossover probability = 0.8, variation probability = 0.2, and iteration number = 50. The figure below shows the simulation results in Fig. 4-7 and the table in Table I.

(2) Parameters were set as follows: path length weighting (a) = 1, path smoothness weighting (b) = 9, population number = 200, crossover probability = 0.8, variation probability = 0.2, and iteration number = 50. The figures and results in Fig. 8-11 and Table II show the simulation results.

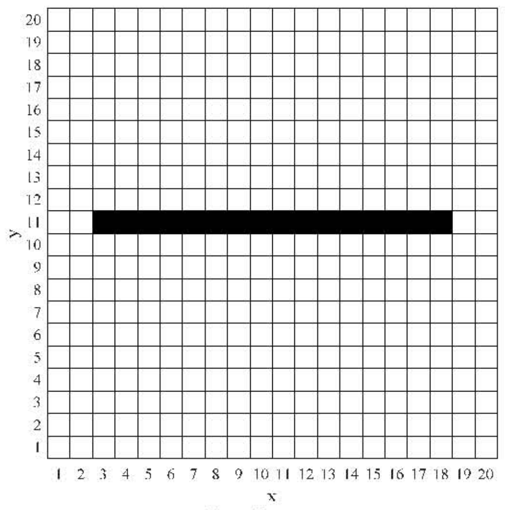
(3) Parameters were set as follows: iteration number = 100, population number = 200 (constant), path length weighting (a) = 1, crossover probability = 1, path smoothness weighting (b) = 7, and variation probability = 0. Fig. 12-15 and Table III present the simulation results.

(4) Parameters were set as follows: path smoothness weighting (b) = 7, offspring generation number = 50, path length weighting (a) = 1, population number = 200, crossover probability = 0, and variation probability = 1. The simulation results are shown in Fig. 16-19 and Table IV.

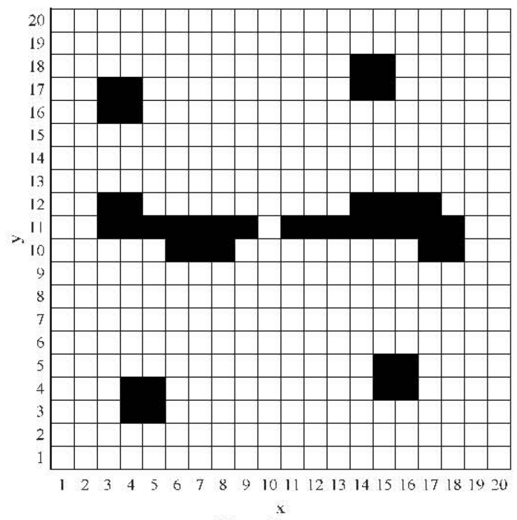
Based on our observations of Map 1, Map 2, Map 3 and Map 4, we can draw some conclusions. First, when dealing with more complex maps (e.g., Maps 3 and 4), the initial path length and the time required for descent are also longer. The number of iterations required to reach a steady state is also significantly higher than for relatively simple maps. After further adjusting the different parameters of the maps, we find that the number of iterations required for the optimal and average paths to reach relative stability increases as the path smoothness weights change and the path length changes. In addition, when the mutation operation is re-shifted, and only the crossover probability is retained, the curvature of the paths can be observed from Fig. 8-11. Moreover, the mutation operation plays a vital role in path planning. We are observing the curvature of the paths in Fig. 8-11, which shows that the curvature increases significantly when only the crossover probabilities are present. This suggests that mutation operations are necessary to find the shortest paths, and the optimal solution cannot be achieved by relying only on crossover probabilities.

A comparison of Tables I and III shows that in Fig. 4-7, the final stabilized distances are between 25 and 35, and the number of iterations required is between 3 and 5. However, in Fig. 12-15, the final stabilization distances are between 60 and 100. Specifically, 60 to 63 in Fig. 12, 70 in Fig. 13, 100 in Fig. 14, and 80 in Fig. 15. As the complexity of the maps increases, the initial path lengths, the descent times, and the number of iterations required to reach the stabilized state during processing also increase. Adjusting the path smoothness weights affects the stability of the optimal and average paths, while removing the mutation operation to retain only the crossover probability increases path curvature and failure to reach the shortest path. The final stabilization distance and the number of iterations required will also vary from map to map.

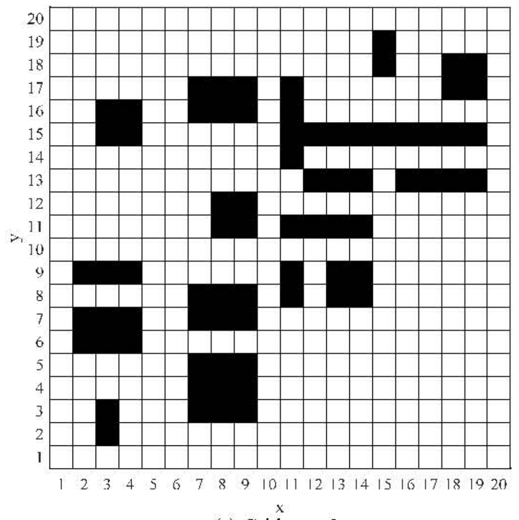
The efficiency and accuracy of path planning can be optimized by combining the crossover and variance probabilities and adjusting the operating parameters appropriately. Experimental results show that on relatively simple maps, the optimal path can be obtained in about five generations, while in complex cases, 15 to 25 generations may be required. These conclusions are important guiding significance for optimal path planning and prove that genetic algorithms have specific efficiency and stability in this field. Through this experiment, we investigated the combination of crossover and variance probability and explored the effect of parameter tuning on path planning. The experimental results highlight the importance of considering the crossover probability and the variance probability to achieve the objectives. The experimental results show that by removing the crossover probability and retaining only the variance probability, a smooth distance state can be achieved within a few generations, which aligns with the desired goal. Compared to retaining only the crossover probability, we can infer that the variance probability contributes significantly to the effectiveness of path planning. The effect of parameter tuning on path planning has also been investigated. When dealing with relatively simple maps, proper tuning of the running parameters usually requires about five generations to obtain optimal paths. However, 15 to 25 generations may be required in complex cases to obtain the optimal path. Therefore, on relatively simple maps, combining the crossover and variance probabilities and adequately tuning the operating parameters requires about five generations to obtain the optimal path. For complex cases, more iterations may be required.



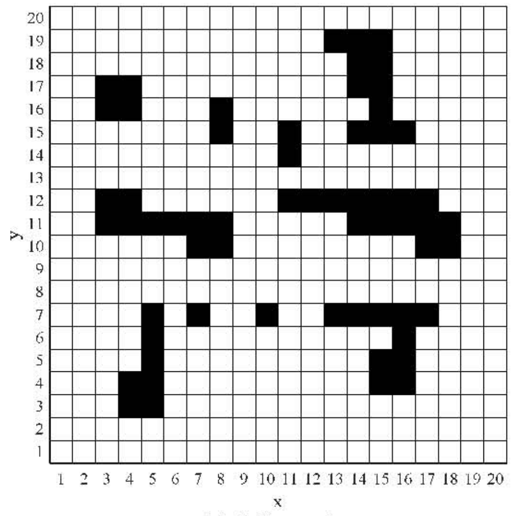
(a) Grid map 1



(b) Grid map 2

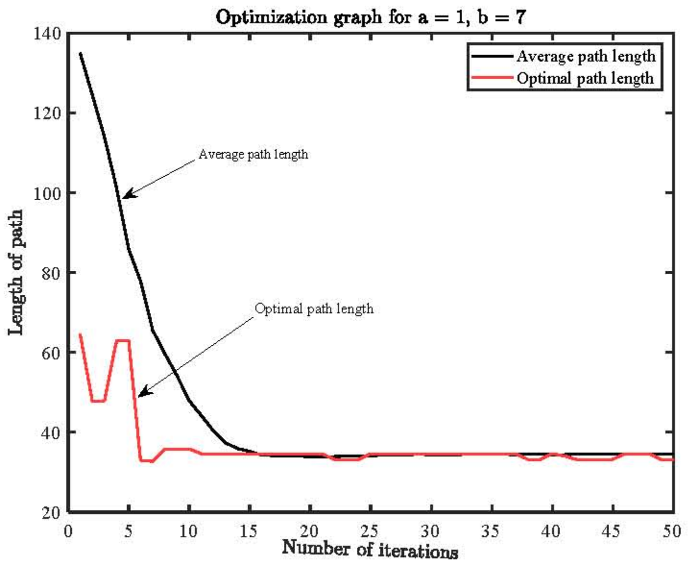


(c) Grid map 3



(d) Grid map 4

Fig. 3 Grid maps.



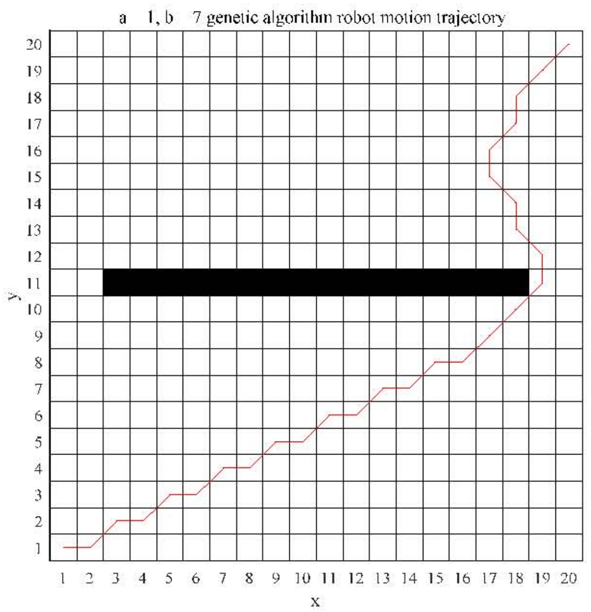
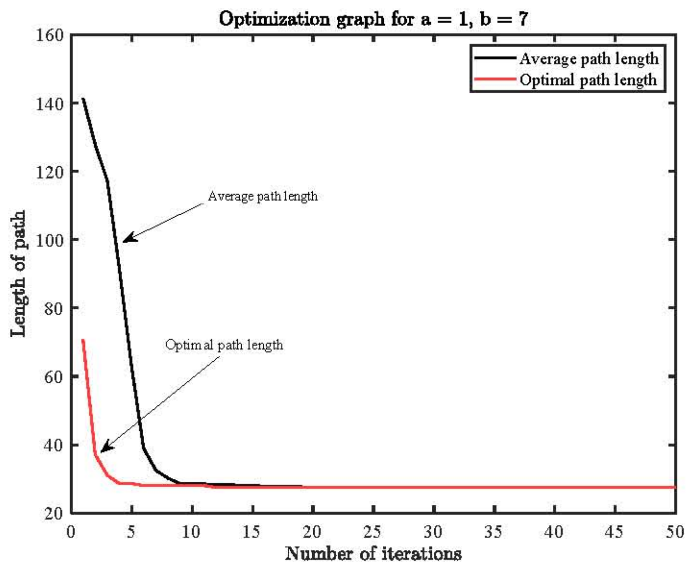


Fig. 4 Path length and route 1.



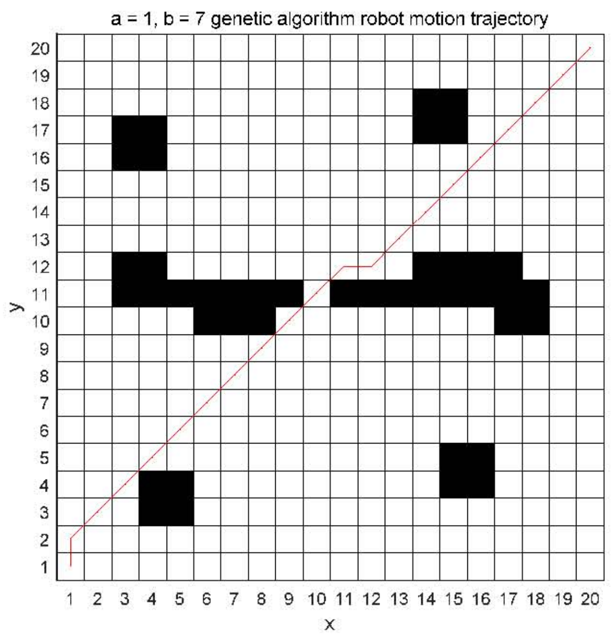
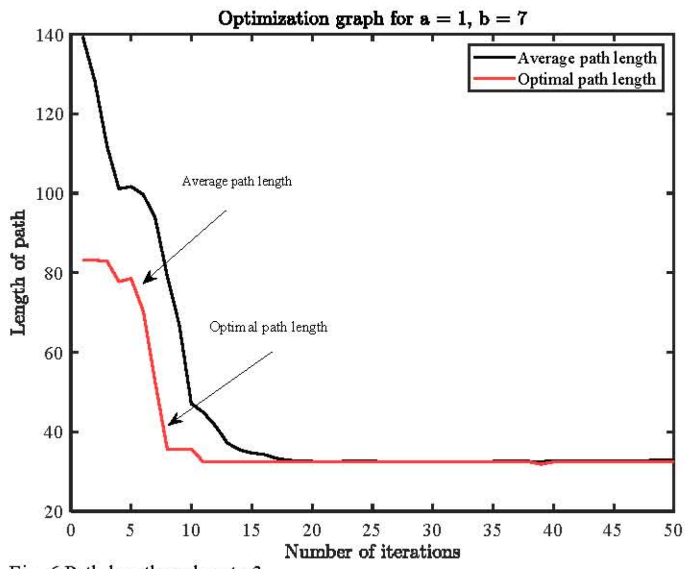


Fig. 5 Path length and route 2.



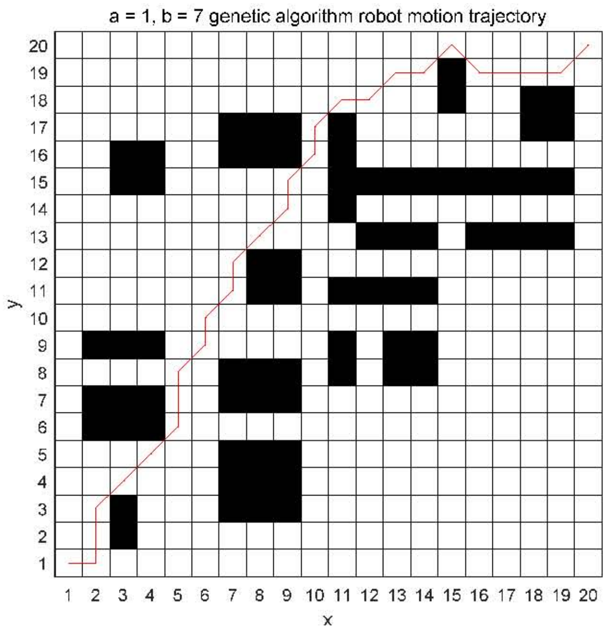
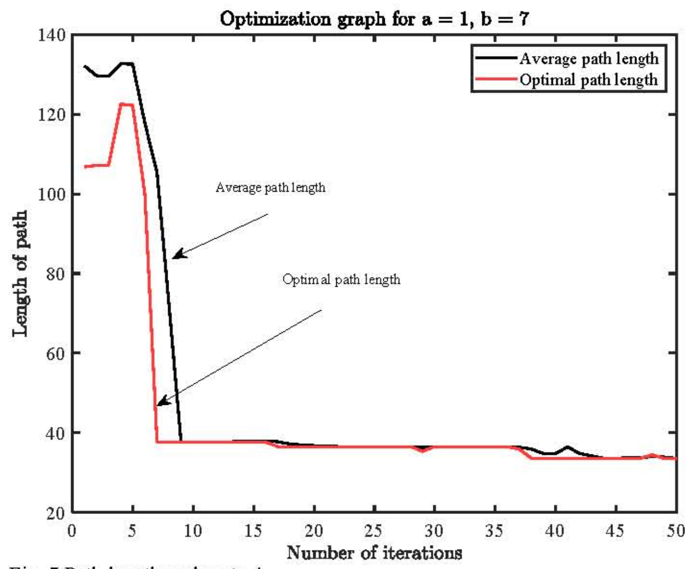


Fig. 6 Path length and route 3.



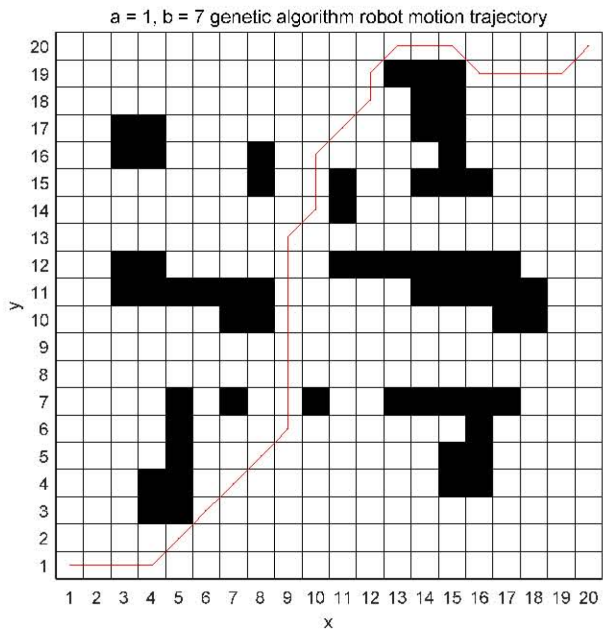


Fig. 7 Path length and route 4.

TABLE I. PATH LENGTH UNDER DIFFERENT PARAMETERS

| | Map 1 | | Map 2 | | Map 3 | | Map 4 | |
|---|---|---|---|---|---|---|---|---|
| | Optimal | Average | Optimal | Average | Optimal | Average | Optimal | Average |
| 1st | 64.63 | 142.40 | 70.73 | 141.43 | 83.11 | 139.36 | 106.77 | 132.12 |
| 2nd | 47.90 | 133.06 | 37.07 | 127.79 | 83.11 | 128.19 | 107.11 | 129.67 |
| 3rd | 47.90 | 125.10 | 30.97 | 117.13 | 82.87 | 111.79 | 107.11 | 129.60 |
| 4th | 62.87 | 113.27 | 28.63 | 91.85 | 77.70 | 101.06 | 122.53 | 132.68 |
| 5th | 62.87 | 98.79 | 28.63 | 63.22 | 78.53 | 101.62 | 122.18 | 132.54 |
| 6th | 33.56 | 80.03 | 28.04 | 38.95 | 70.28 | 99.66 | 99.94 | 117.74 |
| 7th | 33.56 | 60.92 | 28.04 | 32.53 | 52.63 | 93.80 | 37.66 | 105.28 |
| 8th | 33.56 | 45.97 | 28.04 | 30.30 | 35.56 | 79.12 | 37.66 | 71.47 |
| 9th | 32.97 | 39.66 | 28.04 | 28.59 | 35.56 | 66.68 | 37.66 | 37.66 |
| 10th | 32.97 | 35.80 | 28.04 | 28.57 | 35.56 | 47.02 | 37.66 | 37.66 |
| 11th-15th | 32.97 | 34.90 | 28.04 | 28.50 | 32.38 | 44.88 | 37.66 | 37.66 |
| 16th-20th | 32.97 | 35.07 | 27.46 | 28.40 | 32.38 | 41.42 | 37.66 | 37.66 |
| 21th-30th | 32.97 | 33.50 | 27.46 | 28.29 | 32.38 | 37.10 | 37.66 | 37.66 |
| 31th-40th | 32.97 | 33.45 | 27.46 | 28.13 | 32.38 | 35.45 | 37.66 | 37.99 |
| 41th-50th | 32.97 | 33.47 | 27.46 | 28.04 | 32.38 | 34.65 | 37.66 | 37.99 |

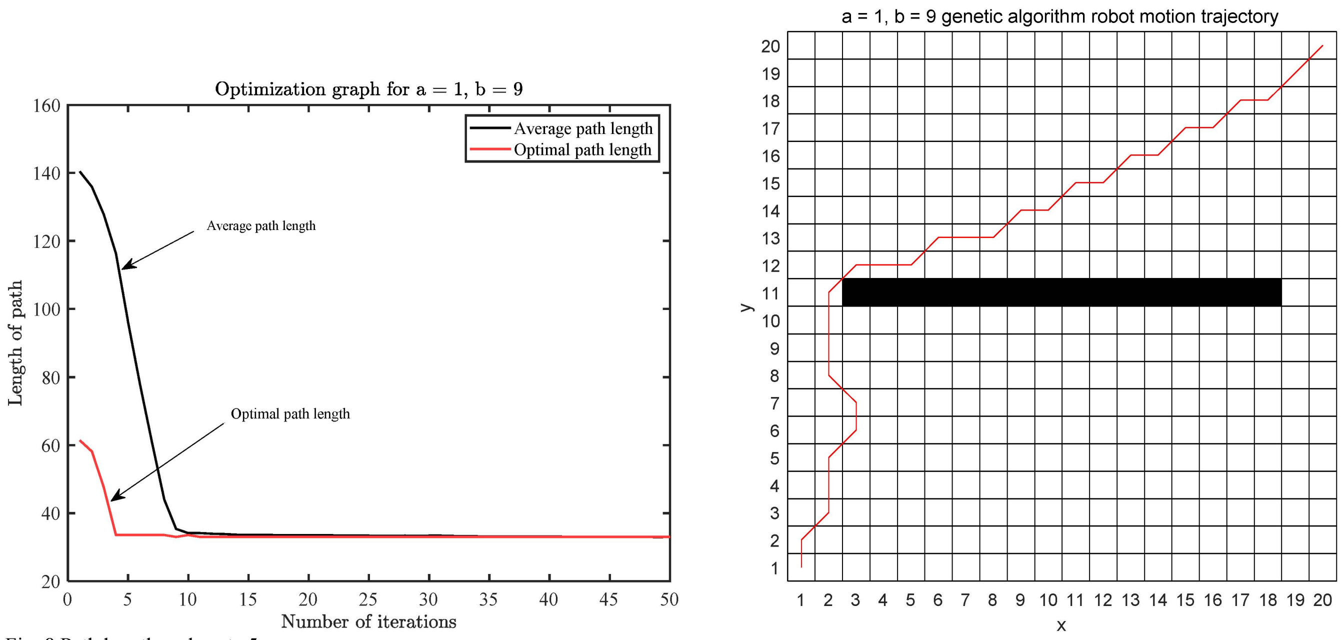


Fig. 8 Path length and route 5.

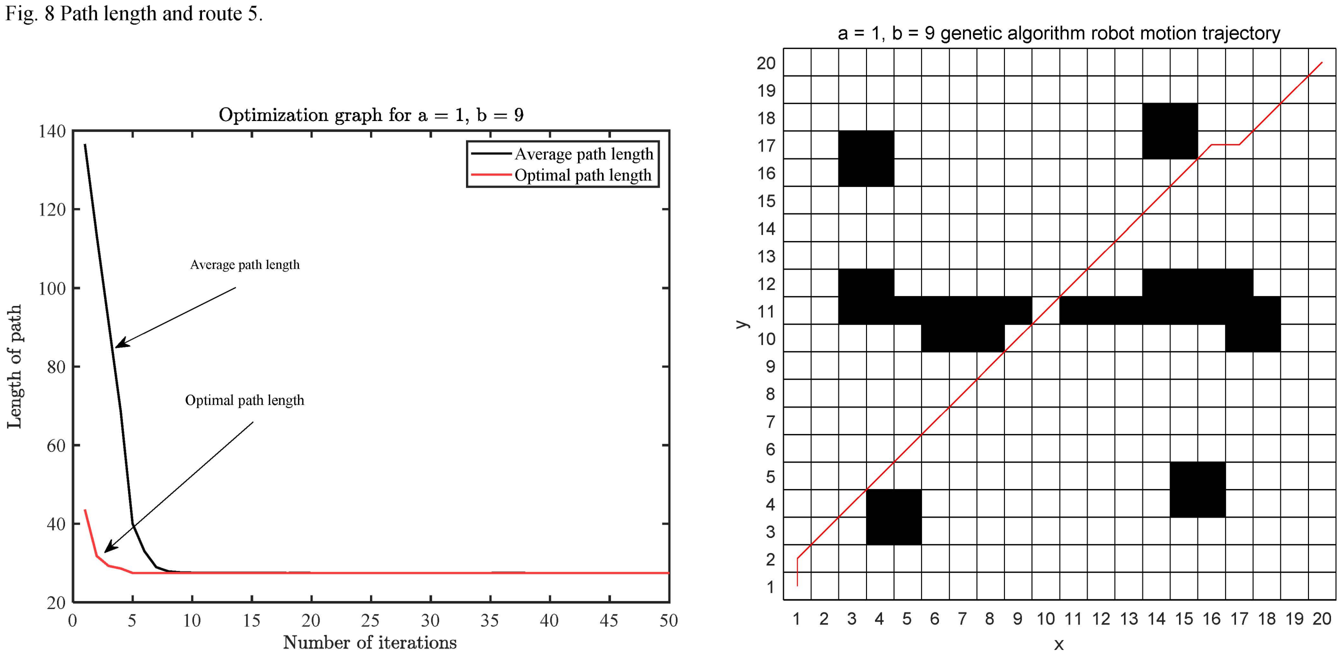


Fig. 9 Path length and route 6.

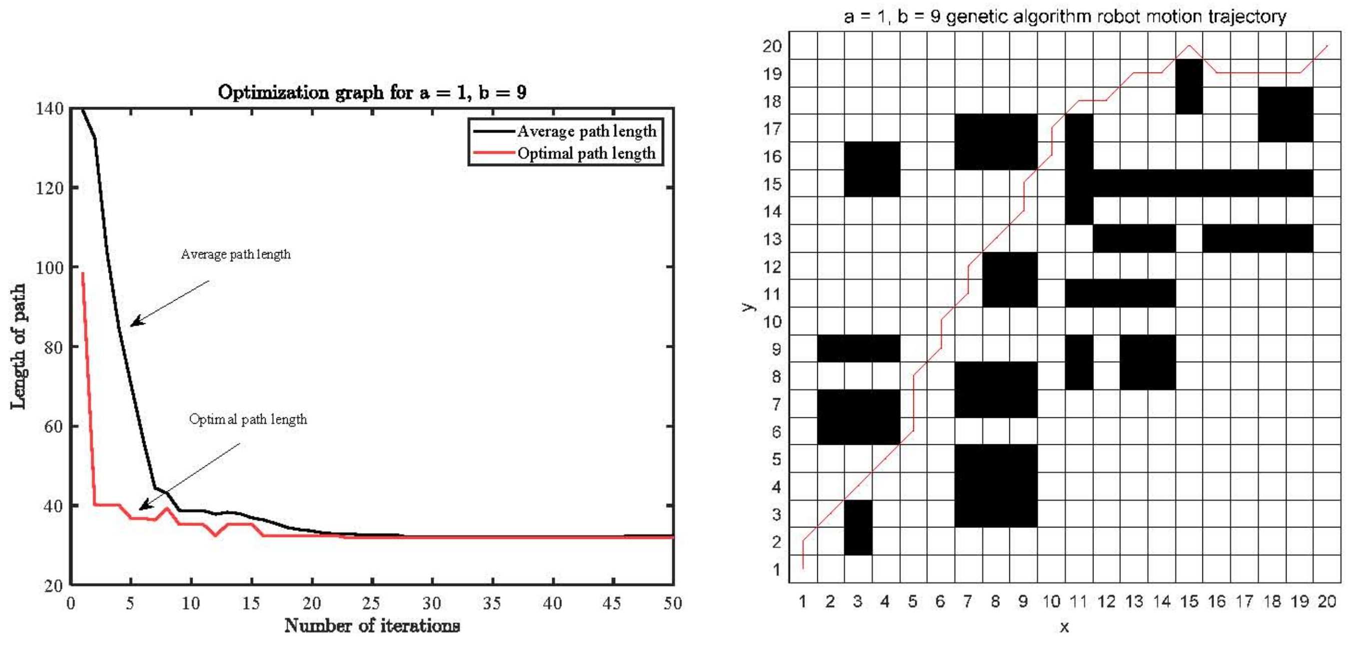


Fig. 10 Path length and route 7.

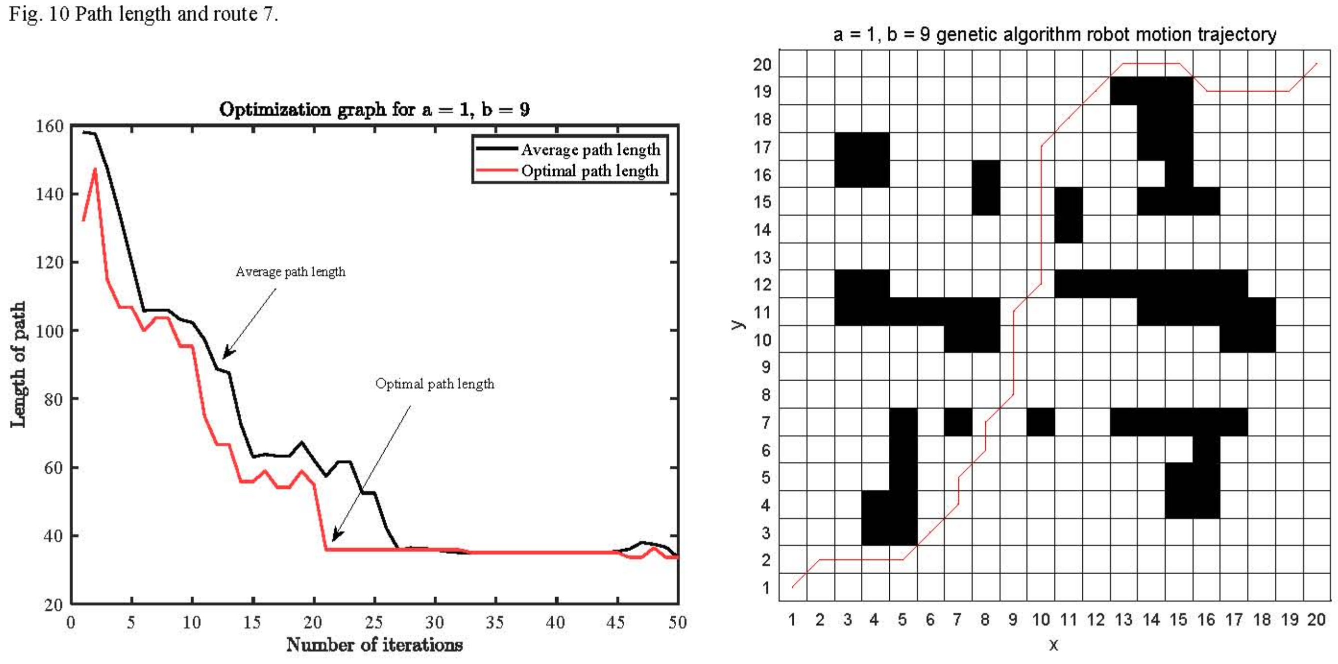
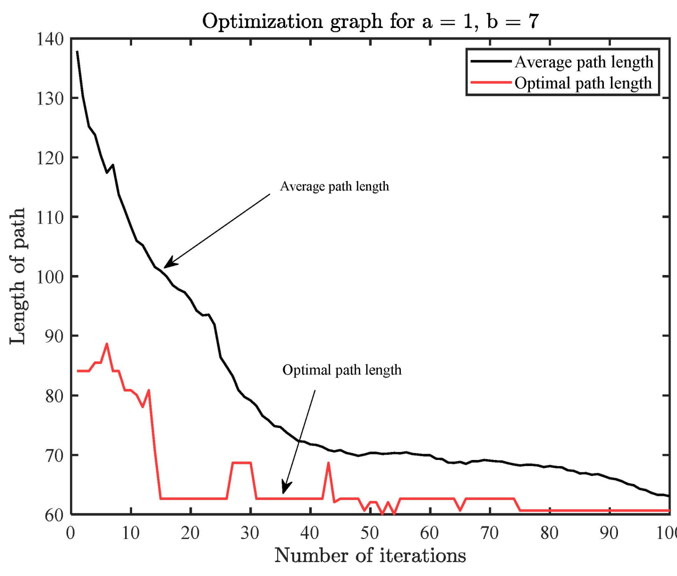


Fig. 11 Path length and route 8.

TABLE II. PATH LENGTH UNDER DIFFERENT PARAMETERS

| | Map 1 | | Map 2 | | Map 3 | | Map 4 | |
|---|---|---|---|---|---|---|---|---|
| | Optimal | Average | Optimal | Average | Optimal | Average | Optimal | Average |
| 1st | 61.46 | 140.39 | 43.56 | 136.51 | 98.53 | 139.44 | 132.08 | 157.96 |
| 2nd | 58.14 | 135.85 | 31.80 | 113.41 | 40.14 | 132.38 | 147.25 | 157.46 |
| 3rd | 47.56 | 127.77 | 29.21 | 91.34 | 40.14 | 103.69 | 114.77 | 147.22 |
| 4th | 33.56 | 116.30 | 28.63 | 68.57 | 40.14 | 84.67 | 106.77 | 134.10 |
| 5th | 33.56 | 96.16 | 27.46 | 40.03 | 36.73 | 70.42 | 106.77 | 119.89 |
| 6th | 33.56 | 78.03 | 27.46 | 32.97 | 36.73 | 56.64 | 99.94 | 105.77 |
| 7th | 33.56 | 60.67 | 27.46 | 28.96 | 36.38 | 44.43 | 103.60 | 105.98 |
| 8th | 33.56 | 44.00 | 27.46 | 27.89 | 39.21 | 43.05 | 103.60 | 105.98 |
| 9th | 32.97 | 35.36 | 27.46 | 27.63 | 35.21 | 38.74 | 95.36 | 103.12 |
| 10th | 33.56 | 34.14 | 27.46 | 27.51 | 35.21 | 38.68 | 95.36 | 102.33 |
| 11th-15th | 34.63 | 36.71 | 27.46 | 27.51 | 35.21 | 38.65 | 66.63 | 87.55 |
| 16th-20th | 34.87 | 35.33 | 27.46 | 27.55 | 32.38 | 35.44 | 55.80 | 61.48 |
| 21th-30th | 32.97 | 34.47 | 27.46 | 27.53 | 32.38 | 34.43 | 34.97 | 52.36 |
| 31th-40th | 32.97 | 34.19 | 27.46 | 27.54 | 31.80 | 33.93 | 33.56 | 34.60 |
| 41th-50th | 32.97 | 33.97 | 27.46 | 27.56 | 31.80 | 33.27 | 33.56 | 34.16 |



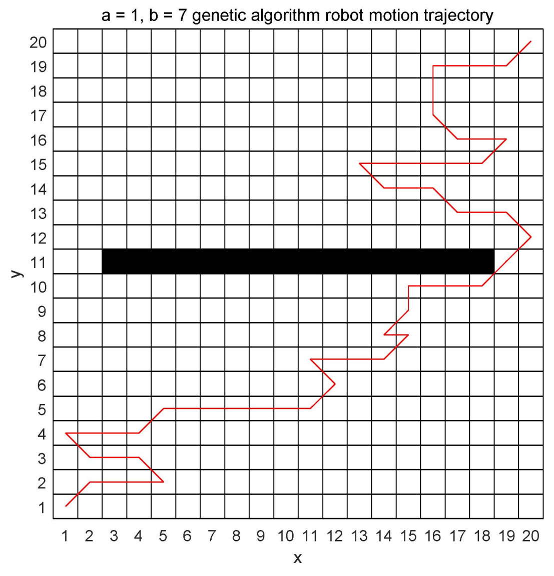
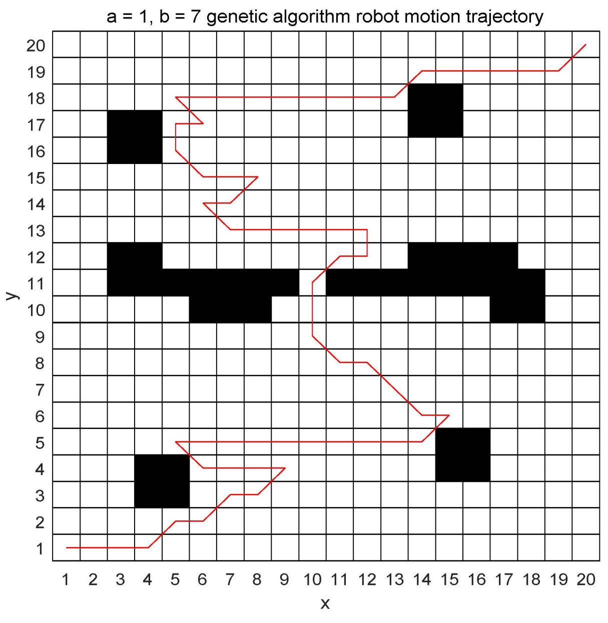


Fig. 12 Path length and route 9.



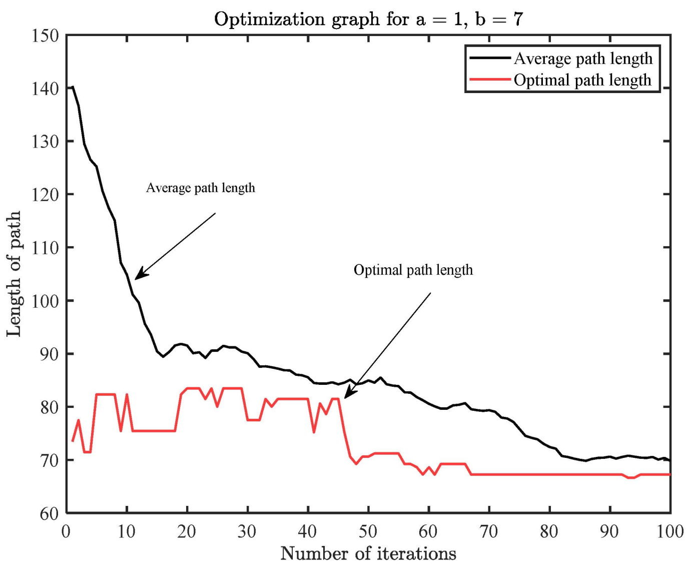
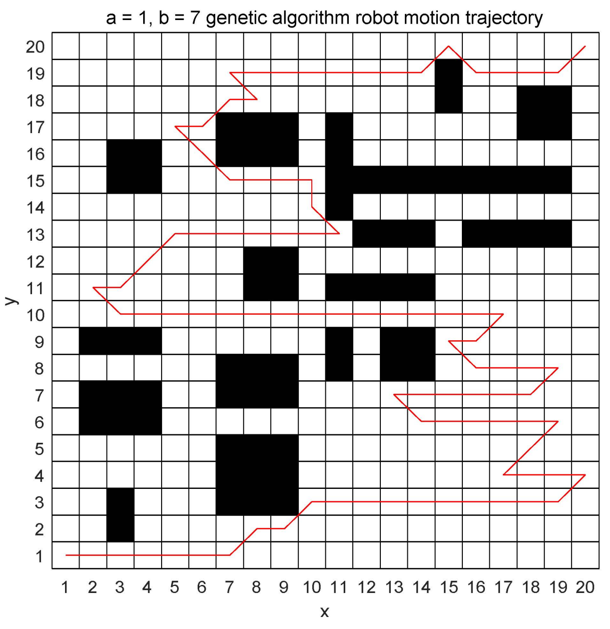


Fig. 13 Path length and route 10.



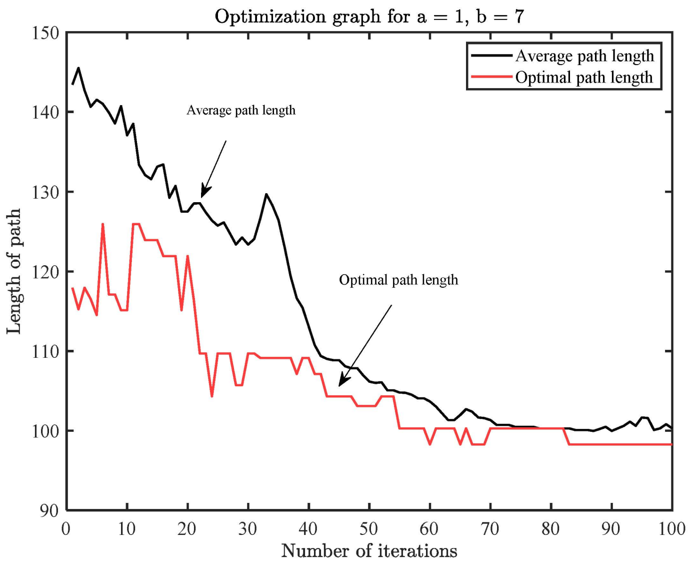


Fig. 14 Path length and route 11.

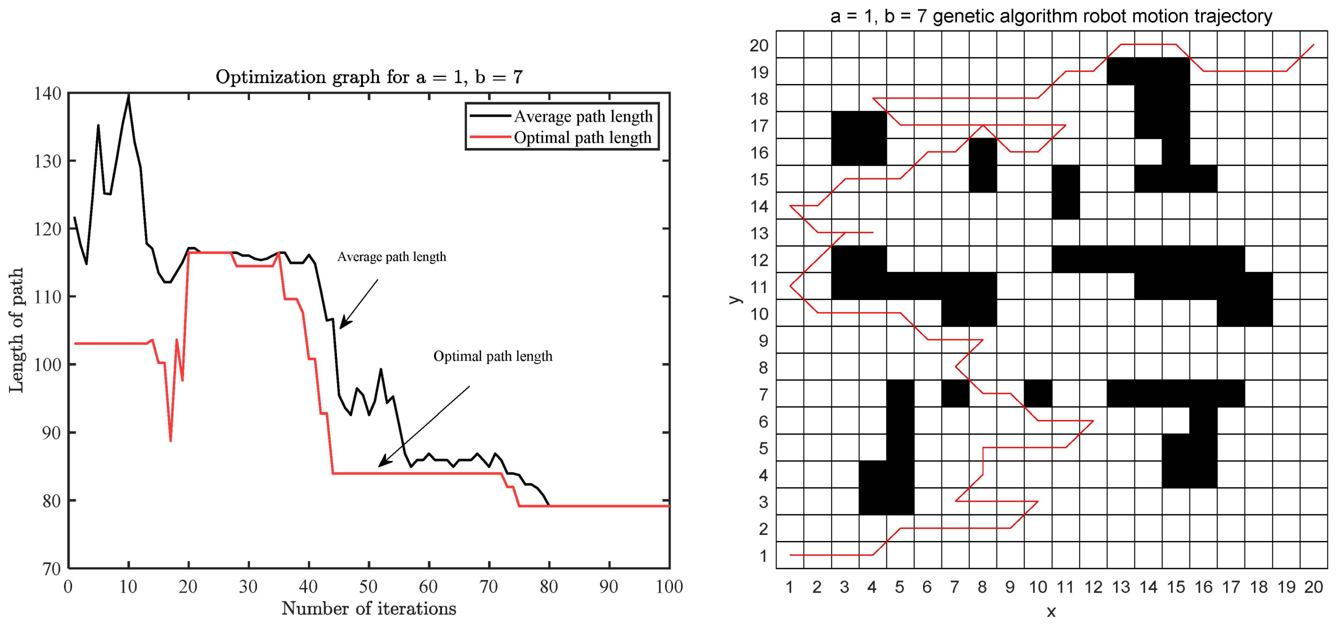


Fig. 15 Path length and route 12.

TABLE III. PATH LENGTH UNDER DIFFERENT PARAMETERS

| | Map 1 | | Map 2 | | Map 3 | | Map 4 | |
|---|---|---|---|---|---|---|---|---|
| | Optimal | Average | Optimal | Average | Optimal | Average | Optimal | Average |
| 1st | 84.04 | 137.86 | 73.46 | 140.34 | 117.94 | 143.41 | 103.01 | 121.68 |
| 2nd | 84.04 | 130.27 | 77.46 | 136.62 | 115.25 | 145.50 | 103.01 | 117.63 |
| 3rd | 84.04 | 125.14 | 71.46 | 129.40 | 117.94 | 142.69 | 103.01 | 114.78 |
| 4th | 85.46 | 123.75 | 71.46 | 126.44 | 116.53 | 140.65 | 103.01 | 124.57 |
| 5th | 85.46 | 120.33 | 82.28 | 125.19 | 114.53 | 141.50 | 103.01 | 135.16 |
| 6th | 88.63 | 117.40 | 82.28 | 120.55 | 125.94 | 141.00 | 103.01 | 125.15 |
| 7th | 84.04 | 118.72 | 82.28 | 117.41 | 117.11 | 139.91 | 103.01 | 125.05 |
| 8th | 84.04 | 113.75 | 82.28 | 115.00 | 117.11 | 138.53 | 103.01 | 130.06 |
| 9th | 80.87 | 111.09 | 75.46 | 107.07 | 115.11 | 140.70 | 103.01 | 135.36 |
| 10th | 80.87 | 108.39 | 82.28 | 104.86 | 115.11 | 137.06 | 103.01 | 139.29 |
| 11th-20th | 76.63 | 102.47 | 75.46 | 101.15 | 125.94 | 130.75 | 88.77 | 115.00 |
| 21th-30th | 74.63 | 84.97 | 75.46 | 81.85 | 109.70 | 127.53 | 116.43 | 117.09 |
| 31th-40th | 70.63 | 77.37 | 75.46 | 81.20 | 104.28 | 123.06 | 109.60 | 95.44 |
| 41th-50th | 62.63 | 75.16 | 67.21 | 80.54 | 100.28 | 107.84 | 92.77 | 91.07 |
| 51th-60th | 62.63 | 74.96 | 69.21 | 80.07 | 98.28 | 104.07 | 81.94 | 86.88 |
| 61th-70th | 62.63 | 65.10 | 69.21 | 79.68 | 100.28 | 101.34 | 81.94 | 80.72 |
| 71th-100th | 62.63 | 65.10 | 69.21 | 70.56 | 98.28 | 100.71 | 79.11 | 79.11 |

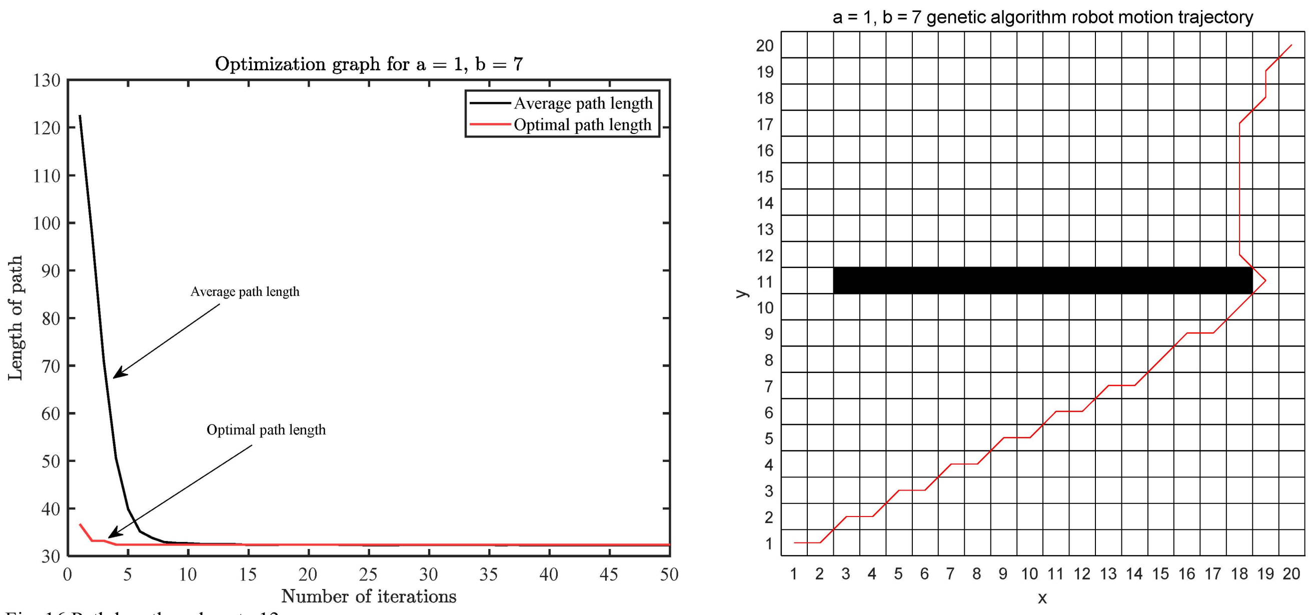


Fig. 16 Path length and route 13.

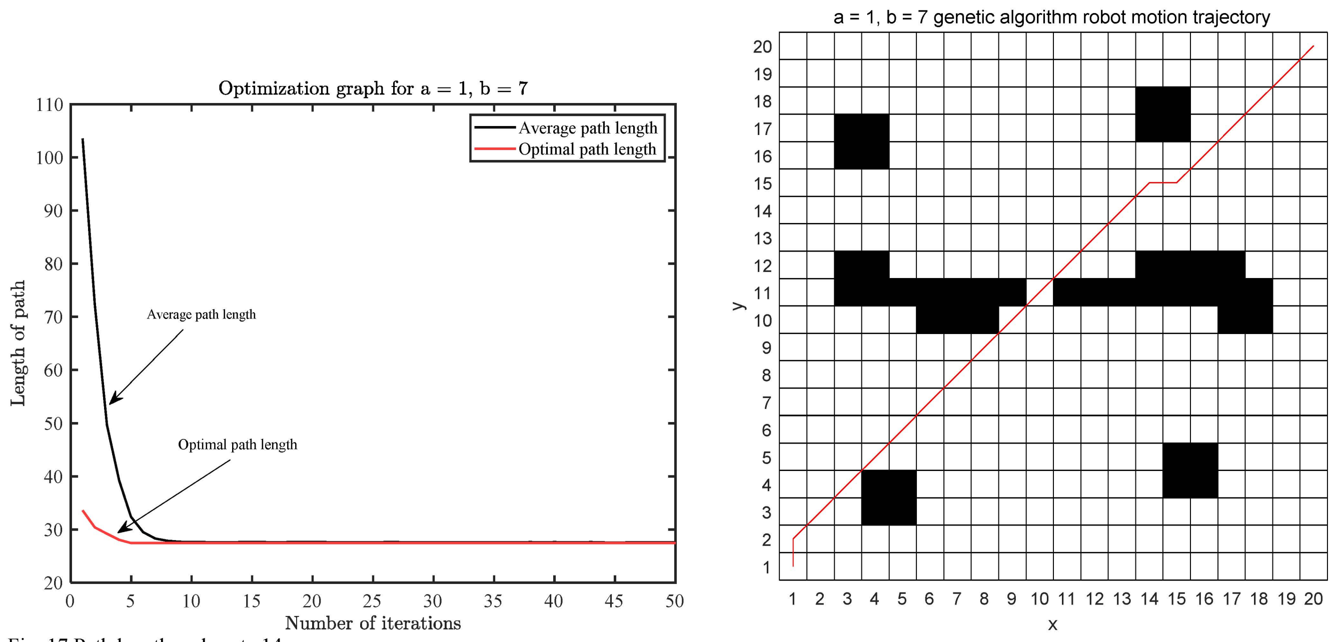


Fig. 17 Path length and route 14.

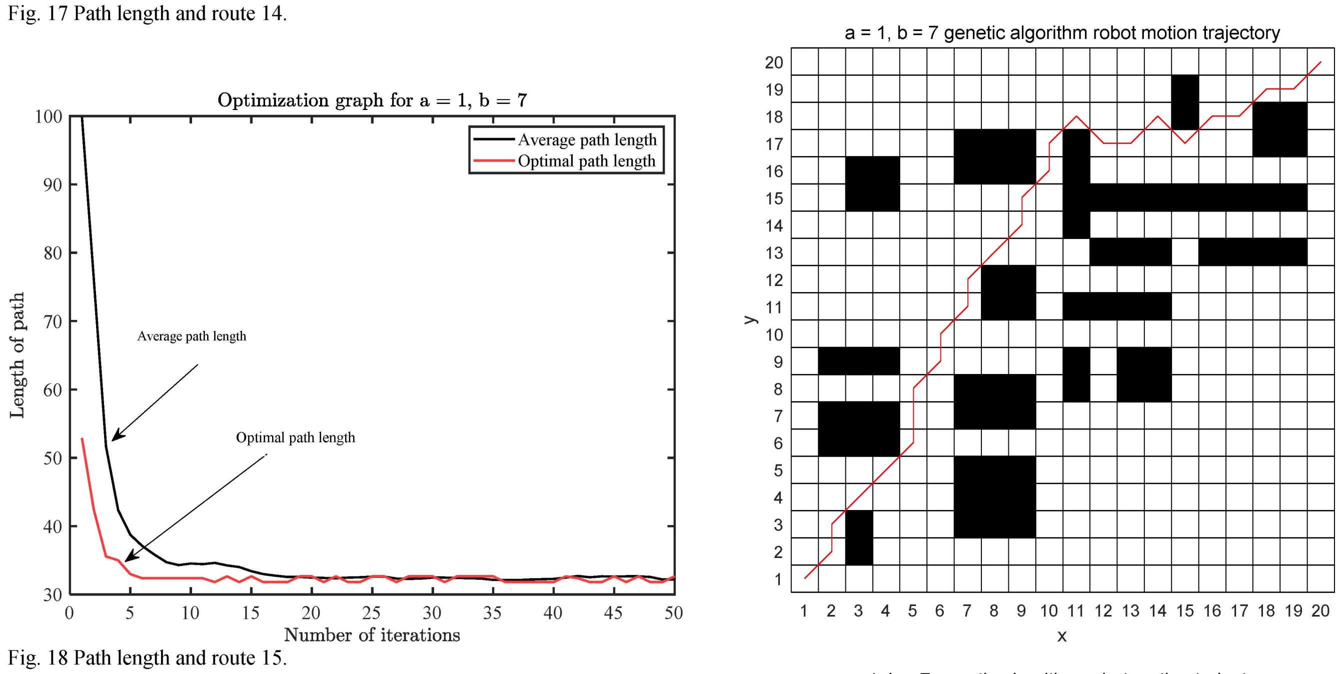


Fig. 18 Path length and route 15.

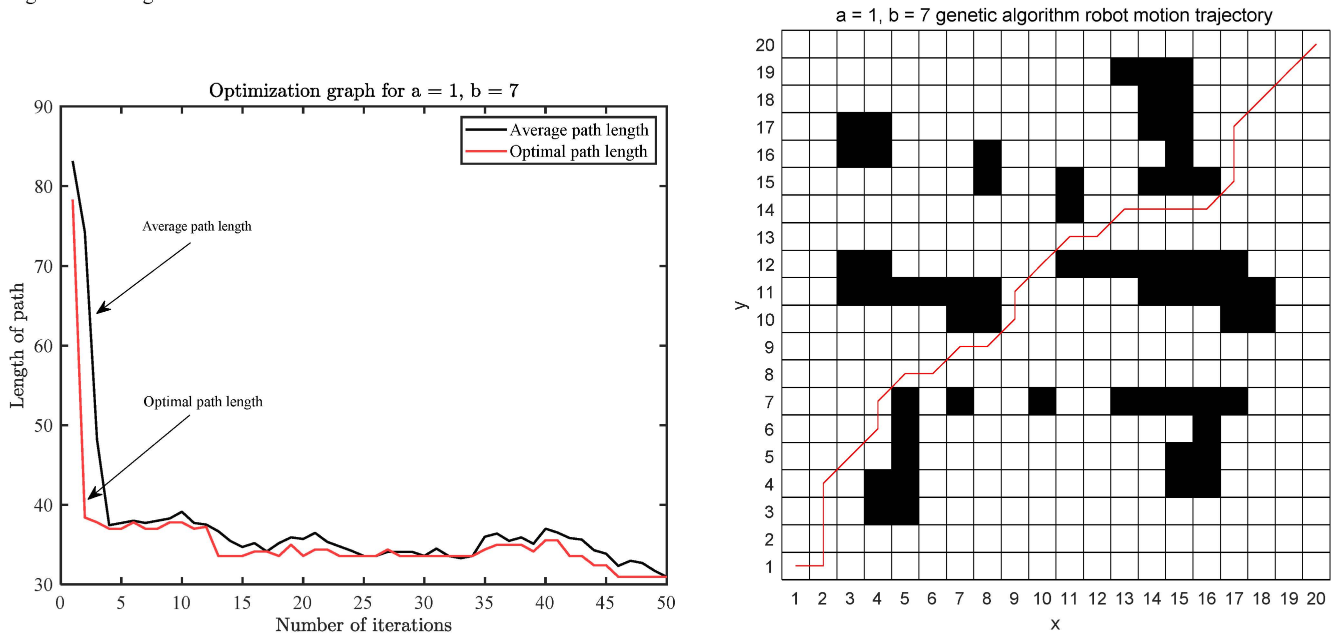


Fig. 19 Path length and route 16.

TABLE IV. PATH LENGTH UNDER DIFFERENT PARAMETERS

| | Map 1 | | Map 2 | | Map 3 | | Map 4 | |
|---|---|---|---|---|---|---|---|---|
| | Optimal | Average | Optimal | Average | Optimal | Average | Optimal | Average |
| 1st | 36.73 | 122.54 | 33.56 | 103.51 | 52.87 | 99.84 | 78.28 | 83.11 |
| 2nd | 33.21 | 97.90 | 30.38 | 72.32 | 42.38 | 76.06 | 38.38 | 74.25 |
| 3rd | 33.21 | 70.94 | 29.21 | 49.69 | 35.56 | 51.66 | 37.80 | 48.16 |
| 4th | 32.38 | 50.57 | 28.04 | 39.26 | 34.97 | 42.35 | 36.97 | 37.44 |
| 5th | 32.38 | 39.87 | 27.46 | 32.38 | 32.97 | 38.73 | 36.97 | 37.72 |
| 6th | 32.38 | 35.17 | 27.46 | 29.48 | 32.38 | 37.12 | 37.80 | 37.99 |
| 7th | 32.38 | 33.83 | 27.46 | 28.28 | 32.38 | 35.85 | 36.97 | 37.72 |
| 8th | 32.38 | 32.96 | 27.46 | 27.84 | 32.38 | 34.69 | 36.97 | 37.99 |
| 9th | 32.38 | 32.74 | 27.46 | 27.67 | 32.38 | 34.30 | 37.80 | 38.27 |
| 10th | 32.38 | 32.63 | 27.46 | 27.65 | 32.38 | 34.50 | 37.80 | 39.13 |
| 11th-15th | 32.38 | 32.56 | 27.46 | 27.60 | 32.38 | 34.43 | 33.56 | 35.72 |
| 16th-20th | 32.38 | 32.50 | 27.46 | 27.57 | 31.80 | 34.60 | 34.38 | 36.47 |
| 21th-30th | 32.38 | 32.50 | 27.46 | 27.55 | 32.63 | 34.23 | 33.56 | 35.33 |
| 31th-40th | 32.38 | 32.48 | 27.46 | 27.61 | 31.80 | 34.02 | 33.56 | 34.78 |
| 41th-50th | 32.38 | 32.35 | 27.46 | 27.63 | 32.63 | 33.38 | 30.38 | 30.86 |

In summary, the results of this study are of great instructive significance for optimizing the efficiency and accuracy of path planning. Genetic algorithms have shown some efficiency and stability in this field. However, it is better to consider using methods other than genetic algorithms to solve the problem in unusually complex cases.

## IV. CONCLUSION

This paper uses a genetic algorithm to solve the robot path planning problem under grid maps. The algorithm aims to find a feasible path with a high success rate and robustness quickly by continuously optimizing the path adaptation through genetic operation. The strong adaptability and scalability of the algorithm in various environments are verified through experiments. In this paper, the genetic algorithm based on a grid graph effectively solves the robot path planning problem, and the algorithm performs well in the experiments, efficiently searching the optimal path in a short time while maintaining a certain degree of robustness and success rate. The algorithm is adaptable and scalable and applies to various environments to solve the robot path planning problem. According to the experimental results, the algorithm performs well regarding running time, search efficiency, and success rate. Therefore, the algorithm has theoretical and practical value and is worth a valuable reference.

## REFERENCES

[1] J. Lian, W. Yu, K. Xiao, and W. Liu, "Cubic Spline Interpolation-based Robot Path Planning Using a Chaotic Adaptive Particle Swarm Optimization Algorithm," *Math. Probl. Eng.*, vol. 2020, pp. 1-20, 2020.

[2] S. Li, X. Wang, L. Hu, and Y. Liu, "Mobile Robot Path Planning Based on Q-learning Algorithm," *2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, Beijing, China, pp. 160-165, 2019.

[3] D. Zhang, X. You, S. Liu, and H. Pan, "Dynamic Multi-Role Adaptive Collaborative Ant Colony Optimization for Robot Path Planning," *IEEE Access*, vol. 8, pp. 129958-129974, 2020.

[4] S. G. Gu, "Research on Robot Path Planning and Tracking Algorithms," *J. Jiangxi Vocat. Tech. Coll. Electr.*, vol. 32, no. 3, pp. 13-14+17, 2019.

[5] T. Dai, and Y. Dong, "Application of Hybrid Intelligent Algorithm in Robot Path Plannin," *J. Zhaotong Univ.*, vol. 43, no. 05, pp. 51-54+59, 2021.

[6] Z. Xie, Q. Zhang, Z. Jiang, and H. Liu, "Robot Learning from Demonstration for Path Planning: A review," *Sci. China Technol. Sci.*, vol. 63, no. 8, pp. 1325-1334, 2020.

[7] X. Liang, D. Kou, and L. Wen, "An Improved Chicken Swarm Optimization Algorithm and Its Application in Robot Path Planning," *IEEE Access*, vol. 8, pp. 49543-49550, 2020.

[8] X. Zhang, Q. Sun, S. Cai, and Z. Liu, "Research on Robot Trajectory Planning Control Strategy," *Micromotors*, vol. 52, no. 11, pp. 76-81, 2019.

[9] F. Wang, J. Ren, and W. Liu, "Box-type Substation Fault Prediction Strategy Based on GA-BP Network," *J. Henan Polytech. Univ. (Nat. Sci.)*, vol. 38, no. 5, pp. 93-98, 2019.

[10] G. Chen and L. Shen, "Genetic Path Planning Algorithm for Complex Environment Path Planning," *Robot*, vol. 23, no. 1, pp. 40-44, 2001.

[11] Y. Lei and S. Wang, "Simulation Study of Robot Path Planning Based on Genetic Algorithm," *J. Changchun Univ.*, vol. 27, no. 4, pp. 1-3+32, 2017.

[12] J. Wang, P. Peng, and C. Xu, "Improved Genetic Algorithm Based on Path Planning for Robot," *Tech. Autom. Appl.*, vol. 29, no. 07, pp. 12-15, 2010.

[13] H. Shuyun, T. Shoufeng, S. Bin, T. Minming, and J. Mingyu, "Robot Path Planning Based on Improved Ant Colony Optimization," *2018 International Conference on Robots & Intelligent Systems (ICRIS)*, Changsha, China, pp. 25-28, 2018.

[14] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile Robot Path Planning Using Membrane Evolutionary Artificial Potential Field," *Appl. Soft Comput.*, vol. 77, pp. 236-251, 2019.

[15] Y. Yang, L. Juntao, and P. Lingling, " Multi-robot Path Planning Based on a Deep Reinforcement Learning DQN Algorithm," *CAAI Trans. Intell. Technol.*, vol. 5, no. 3, pp. 177-183, 2020.

[16] A. K. Rath, D. R. Parhi, H. C. Das, P. B. Kumar, and M. K. Mahto, "Design of a Hybrid Controller Using Genetic Algorithm and Neural Network for Path Planning of a Humanoid Robot," *Int. J. Intell. Unman. Syst.*, vol. 9, no. 3, pp. 169-177, 2021.

[17] 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.

[18] J. Min, "A Fireworks Explosive Immune Genetic Algorithm Based on Elite Opposition-based Learning," *J. Hefei Univ. Technol. (Nat. Sci.)*, vol. 43, no. 4, pp. 433-437, 2020.

[19] H. Zhang, Q. Zhao, Z. Cheng, L. Liu, and Y. Su, "Dynamic Path Optimization with Real-time Information for Emergency Evacuation," *Math. Probl. Eng.*, vol. 2021, pp. 1-9, 2021.

[20] C. Shao, G. Hu, and S. Li, "Gauss Mutation Grasshopper Optimization Algorithm for Path Planning of Welding Robot," *Mach. Design Manuf.*, vol. 386, no. 4, pp. 276-280, 2023.