




A new hybrid algorithm for path planning of mobile robot

Ting-Wei Zhang¹ · Guang-Hui Xu¹  · Xi-Sheng Zhan² · Tao Han²

Accepted: 17 August 2021 / Published online: 24 August 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

The selection of algorithm is the most critical part in the mobile robot path planning. At present, the commonly used algorithms for path planning are genetic algorithm (GA), ant colony algorithm (ACA), and firefly algorithm (FA). Among them, FA is more typical. FA has the disadvantage of being easily trapped into a local optimal solution. In order to improve this disadvantage, this paper proposes a new hybrid algorithm which is based on GA and FA. The core idea of this new algorithm is that when the FA falls into the local optimal solution, the local optimal fireflies would be regarded as a group, and the group is subjected to the selection, crossover and mutation operations in the GA. Finally, the optimal firefly individual can be obtained from genetic operations. Theoretical and experimental results have verified that the new hybrid algorithm can improve the accuracy and performance of the FA. Applying the new hybrid algorithm to path planning can improve the robot's reaction ability and computing power in path planning.

Keywords Firefly algorithm · Genetic algorithm · Mobile robot · Path planning

1 Introduction

Path planning has always been a hot topic in the field of robot search and one of the key technologies for mobile robot technology research. The path planning of mobile robot is actually to select an optimal or suboptimal obstacle avoidance path that can be connected from the starting point to the ending point in the task area by referring to a certain parameter. The essence is to obtain the optimal or feasible solution under several constraints [1].

✉ Guang-Hui Xu
xgh@hbut.edu.cn

¹ Hubei Key Laboratory for High-Efficiency Utilization of Solar Energy and Operation Control of Energy Storage System, School of Electrical and Electronics Engineering, Hubei University of Technology, Wuhan 430068, China

² Mechatronics and Control Engineering, Hubei Normal University, Huangshi 435002, China

Path of the mobile robot can be divided into a path of a static known environment and a path of a dynamic unknown environment. The static path planning is named as the global path planning, while the dynamic path planning is named as the local path planning [2]. Local path planning is a kind of real-time dynamic path planning performed by the robot based on the local environment information collected by the sensors carried by the robot during the task execution. It is characterized by high flexibility and real-time performance [3].

However, as it relies on local environment characteristics, the path obtained may be only local optimal rather than global optimal [4]. For global path planning, it is first necessary to establish an abstract all-region environment map model, and then obtain the global optimal or a better path on the all-region map model by using the optimal searching algorithm, and finally guide the mobile robot to move safely toward the target point in a real situation. It mainly involves two parts: one is the understanding of environmental information and the construction of map model, and the second is all-region path search and robot guidance [5].

Based on above information, this paper studies the path planning problems based on the static global path planning. The selection of algorithm is the most critical part in the mobile robot path planning. At present, the commonly used algorithms for global path planning are GA [6], ACA [7], and PSO [8, 9]. With the development of mobile robot technology, path planning technology has achieved some research results, but in the path planning algorithm design, each algorithm has its advantages and disadvantages [10]. Therefore, more and more intelligent optimization algorithms are constantly emerging [11].

In 2008, Yang proposed a new swarm intelligence algorithm- FA. This algorithm which solves the optimal searching problem by simulating the mutual attraction behavior of fireflies due to the light intensity is a stochastic optimization algorithm based on the intelligence of biota [12].

FA is a genetic calculation method between GA and evolutionary programming [13]. Similar to other algorithms, it is also based on groups. However, instead of using the evolutionary operators, it treats each individual as a particle without a volume in the search space, moving freely in the search space. In the mobile search, the mobile distance is determined by the luminous intensity and light intensity absorption coefficient of its surrounding companions [14].

FA has the common characteristics of group intelligence algorithms: (1) they all have group-based optimization technology, that is, there are multiple search tracks, which have strong ease of implementation and parallelism; (2) they only need to use the value of the target itself, which is very versatile [15]. Because of the easy implementation and strong versatility of FA, more and more scholars combine it with various optimization problems.

FA has a short history of development. Although it has certain advantages over other algorithms in terms of optimization speed and accuracy, there are still some problems to be solved in the FA [16]. For example, the algorithm is easy to fall into the local optimal solution; the performance of the algorithm is too dependent on the selection of parameters; the algorithm is prone to premature convergence during the implementation process, etc [17, 18].

Motivated by the above work, this paper studies the optimization of FA to solve the problem that the FA is easy to fall into the local optimal solution. In the face of problem that the FA is easy to fall into the local optimal solution, some scholars have proposed different optimization strategies. Among them, dynamic adaptive parameter strategy is more typical [19]. The core idea of this strategy is to use dynamic formulas to update the adaptive parameters so that they finally meet the condition that all fireflies can converge to one point. However, a large number of simulation results show that the dynamic adaptive strategy has some errors in the implementation process, and the parameter update speed is slow [20].

This paper proposes a new genetic firefly algorithm (GFA), which is based on FA and GA. The core idea of this new algorithm is that when the FA falls into the local optimal solution, the local optimal fireflies would be regarded as a group, and the group is subjected to the selection, crossover and mutation operations in the GA. Finally, the optimal firefly individual can be obtained from genetic operations. Experimental results show that the new algorithm has a faster optimization speed than dynamic adaptive parameter strategy, and can solve the problem that FA is easy to fall into the local optimal solution. The main contributions of this paper are as follows:

- (1) We successfully combined the FA with the GA and propose a new hybrid algorithm based on this model.
- (2) The hybrid algorithm can not only solve the problem that the FA is easy to fall into the local optimal solution, but also improve the performance of the FA.
- (3) Hybrid algorithm can improve the computing power and reaction speed of mobile robot in path planning.
- (4) Compared with other algorithms, the hybrid algorithm has higher superiority and better research significance.

The specific content of this paper is as follows. The FA with its mathematical model and GA with its mathematical model are introduced in Sect. 2. The mathematical model of GFA and optimization experiment are also described in Sect. 3. Application in path planning and simulation results are reflected in Sect. 4. Finally, some concluding remarks and future works are drawn in Conclusions.

2 Mathematical models of algorithms

2.1 Mathematical model of FA

The search process of the FA relies on the attractiveness between individuals to produce movement, that is, the brighter fireflies have bigger attractiveness and always attract the darker fireflies to move to them. Each firefly in the group will have its own luminosity and correspond to a reference solution to the problem. Here, three kinds of remarks are required [13].

Remark 1 Fireflies are gender-neutral, any fireflies can be attracted by better (brighter) fireflies;

Remark 2 The attractiveness of fireflies is directly proportional to their luminosity. For any two fireflies, the darker firefly move toward brighter firefly;

Remark 3 If there is no fireflies brighter than the given fireflies, fireflies will move randomly;

According to the principle that darker fireflies are attracted to brighter fireflies, the position and luminosity of fireflies would be constantly updated, so as to find the brightest firefly and complete the optimization process [15].

For any two fireflies X_i and X_j ($i \neq j$), their attractiveness can be expressed as follows:

$$A_{ij}(r_{ij}) = A_0 e^{-\gamma r_{ij}^2} \tag{1}$$

where A_0 is the attractiveness when $r = 0$.

The parameter γ represents light absorption coefficient, which is always set as a constant. r_{ij} is the distance from X_i and X_j , which is defined by the following:

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{d=1}^D (x_{id} - x_{jd})^2} \tag{2}$$

where $d = 1, 2, 3, \dots, D$, and D is the number of maximum problem dimensions [21].

The luminosity of fireflies is related to the objective function, and its definition is as follows:

$$L_{ij}(r_{ij}) = L_0 e^{-\gamma r_{ij}^2} \tag{3}$$

$$L_i(r_i) = L_0 e^{-\gamma r_i^2} \tag{4}$$

where L_i represents the luminosity of the firefly X_i , and L_{ij} represents the relative luminosity between X_i and X_j . L_0 is the absolute luminosity at $r = 0$.

Similarly, for X_i and X_j , the darker firefly always moves to the brighter one, assuming that X_i is darker than X_j , then the process that X_i moves to X_j can be defined as follows:

$$x_{id}(t + 1) = x_{id}(t) + A(r_{ij}) \times (x_{jd}(t) - x_{id}(t)) + \alpha \varepsilon \tag{5}$$

where $\alpha \in [0, 1]$ represents the step factor. ε is a random value uniformly distributed in the range $[-0.5, 0.5]$, and t is the iteration number [21].

Algorithm 1 represents the pseudo code of the FA, where T_{\max} is the maximum number of iterations and N is the maximum number of populations.

Algorithm 1 Iterative algorithm for FA

```

1: population initialization, put  $t = 0$ 
2: generate  $N$  fireflies(solutions)  $X_i, i = 1, 2, \dots, N$ 
3: compute the fitness value of each firefly based on equation
   (3)
4: loop
5:   while  $t < T$  do
6:     for  $i = 1$  to  $N$  do
7:       for  $j = 1$  to  $N$  do
8:         if  $L_i > L_j$  then
9:            $X_j$  moves to  $X_i$  based on equation (5)
10:          evaluate the luminosity and position of  $X_j$ 
11:        end if
12:      end for
13:    end for
14:     $t++$ 
15:  end loop
16: output result

```

2.2 Mathematical model of GA

GA uses a certain coding method to map the solution space to the coding space. Each code corresponds to a solution of the problem, which called an individual or chromosome, and then randomly determines the starting group of individuals, which can be called a population.

In subsequent iterations, individuals are first selected based on the size of fitness value, and then individuals are crossed and mutated according to various genetic operators. After these operations, a new population will be generated, which is more adaptable to the environment than the previous generation. This continues until the optimization criterion is met [22]. Finally, the decoded last-generation individual can be used as the optimal solution to the problem [23].

The three main operations in genetic operations are selection, crossover, and mutation. Among them, the selection operation (selection operator) is used to determine the crossover individuals, that is, which individual will be copied according to the individual's fitness value; the crossover operation (crossover operator) refers to replacing and reorganizing the partial structure of the two parents to generate new individuals.

The role of crossover operation is to generate new individuals, which is also a genetic operation that plays a central role in GA. Various crossover operators include two basic contents: (1) Pair individuals randomly and decide whether each pair needs crossover operation according to the preset crossover probability [24]. (2) Set the intersection of paired individuals, and exchange part of the structure of these paired individuals;

Mutation operation (mutation operator) refers to the inverse change of certain gene values of individuals in a group. There are two aims in the mutation operation:

(1) Make GA have local random search ability. (2) Maintain the diversity of groups [25]. The operation of the mutation operator is generally divided into two steps: (1) The loci are randomly determined within the code string range of all individuals in the population [26]. (2) The gene values of these loci are mutated with a preset mutation probability [27].

The basic process of GA can be summarized as follows:

- (1) A certain number of initial populations are randomly generated, and each individual is expressed as a gene code of a chromosome;
- (2) Calculate the fitness of each individual, and judge whether it meets the optimization criteria. If it meets, output the best individual and end the calculation, otherwise go to step 3;
- (3) Select regenerative individuals based on fitness. Individuals with high fitness have a high probability of being selected, while individuals with low fitness may be eliminated;
- (4) Perform crossover and mutation operations to generate new individuals;
- (5) Get a new generation of population and return to step 2.

The pseudo code of the GA is shown in Algorithm 2, where N is the maximum population and T is the maximum number of iterations.

Algorithm 2 Iterative algorithm for GA

```

1: parameters initialization, put  $t = 0$ 
2: generate random population of chromosomes  $H$ 
3: loop
4:   while  $t < T$  do
5:     select parents from  $H$  to create offspring
6:     apply crossover operation using crossover rate
7:     apply mutation operation using mutation rate
8:     evaluate new fitness values
9:     sort and rank fitness values
10:     $t++$ 
11:  end loop
12: get best solution

```

2.3 Defect verification

Although the FA has good performance on many optimization problems, it still has some shortcomings. For example, the optimization performance depends on the setting of control parameters, the convergence speed is slow, and it is easy to fall into the local optimal solution on complex problems.

This paper will focus on the problem that the FA is easy to fall into the local optimal solution. First, we will verify the existence of shortcomings of the algorithm. Selecting six binary nonlinear functions as the optimization objectives, the objective functions are shown in Table 1.

Table 1 Six binary nonlinear functions

| Function | Search range | Global optimum |
|--|----------------------|----------------|
| $f_1(x, y) = \frac{\sin x}{x} + \frac{\sin y}{y}$ | $x, y \in [-5, 5]$ | 2 |
| $f_2(x, y) = y \sin(2\pi x) + x \cos(2\pi y)$ | $x, y \in [-2, 2]$ | 4 |
| $f_3(x, y) = 0.5 + \frac{\sin^2(x^2 - y^2) + 0.5}{1 + 0.001(x^2 + y^2)}$ | $x, y \in [-10, 10]$ | 1 |
| $f_4(x, y) = \left\{ \sum_{i=1}^5 i \cos[(i+1)x + i] \right\} \times \left\{ \sum_{i=1}^5 i \cos[(i+1)y + i] \right\}$ | $x, y \in [-10, 10]$ | 190.001 |
| $f_5(x, y) = \frac{1}{[(x+1)^2 + (y-2)^2 + 1]} + \frac{4}{[x^2 + (y+1)^2 + 1]} + \frac{2}{[(x-4)^2 + (y+1)^2 + 1]}$ | $x, y \in [-5, 5]$ | 4 |
| $f_6(x, y) = 1.5e^{[-(x-3)^2 - (y-2)^2]} + e^{[-(x+2)^2 - (y-4)^2]} + 3e^{(-x^2 - y^2)} + 1.2e^{[-(x+4)^2 - (y+1)^2]} + 1.5e^{[-x^2 - (y+4)^2]} + 1.6e^{[-(x-4)^2 - (y+3)^2]}$ | $x, y \in [-5, 5]$ | 3 |

The reason for choosing these binary functions is that they are multimodal functions. As shown in Fig. 1, there are multiple local maxima in the domain of these functions. Once FA falls into the local optimum, some fireflies will gather near these local maxima.

FA is used to optimize these objective functions. Setting the number of fireflies to 20, the step factor α to 0.8, and the light absorption coefficient γ to 0.3. The performance of the algorithm depends on the choice of parameters, the inspiration for parameters selection comes from [19].

Figure 1 shows the results of FA to optimize objective functions $f_5(x, y)$ and $f_6(x, y)$. Among them, Figs. (a) and (b), respectively, represent the three-dimensional function graphs of $f_5(x, y)$ and $f_6(x, y)$.

From the results shown in Fig. 1, it can be found out that the binary nonlinear functions have some local extreme points in their definition domain, and the maximum extreme point of the function is near (0, 0). Besides the above analysis, some fireflies are distributed discretely around local extreme points, and most of the fireflies are not near the maximum extreme point. Thus, it can be confirmed that FA is easy to fall into the local optimal solution.

2.4 The model of AFA

Based on the core idea of FA (dark fireflies will be attracted by bright fireflies), it can be deduced that in an ideal situation, all fireflies should be close to the brightest firefly, and eventually converge to one point [28]. So for any two fireflies X_i and X_j :

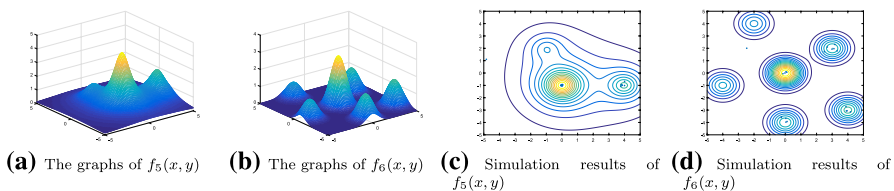


Fig. 1 Simulation results

$$\lim_{t \rightarrow \infty} X_i(t) = \lim_{t \rightarrow \infty} X_j(t), \forall i \neq j. \tag{6}$$

$$\lim_{t \rightarrow \infty} X_i(t + 1) = \lim_{t \rightarrow \infty} X_i(t). \tag{7}$$

where $i, j = 1, 2, 3, \dots, N$. Eq. (6) shows that all fireflies eventually converge to one point. Eq. (7) indicates that the position of the fireflies after convergence will no longer change.

According to Eqs. (1), (2), (3), (5), (6) and (7), we can get:

$$\begin{aligned} \lim_{t \rightarrow \infty} X_i(t + 1) - X_i(t) &= 0 \\ \Rightarrow \lim_{t \rightarrow \infty} A_{ij}(r_{ij}) \times \lim_{t \rightarrow \infty} (X_i(t) - X_j(t)) \\ &+ \varepsilon \times \lim_{t \rightarrow \infty} \alpha = 0 \\ \Rightarrow A_0 \times \lim_{t \rightarrow \infty} e^{-\gamma r_{ij}^2} \times \lim_{t \rightarrow \infty} (X_i(t) - X_j(t)) \\ &+ \varepsilon \times \lim_{t \rightarrow \infty} \alpha = 0 \\ \Rightarrow 0 + \varepsilon \times \lim_{t \rightarrow \infty} \alpha &= 0 \\ \Rightarrow \lim_{t \rightarrow \infty} \alpha &= 0. \end{aligned} \tag{8}$$

From Eq. (8), we can find that when the fireflies finally converge to one point, the step factor α will also approach to zero. Based on this inference, many scholars regard whether the step factor α is 0 as a condition to determine whether the FA falls into the local optimal solution, and have proposed a variety of adaptive firefly algorithms (AFA) [29].

The main ideas of these algorithms are to change the dynamic value of the step factor α to make it finally approach to 0, so as to solve the problem that the FA is easy to fall into the local optimal solution [30].

Some scholars have proposed the following adaptive parameter dynamic strategies to update the parameter α , and the parameter update formulas are:

$$\alpha(t + 1) = (1 - \frac{t}{T_{\max}})\alpha(t). \tag{9}$$

$$\alpha(t + 1) = (\frac{1}{9000})^{\frac{1}{t}}\alpha(t). \tag{10}$$

It has been confirmed that the parameter update speed of AFA is slow [31]. Even in the face of some complicated optimization problems, AFA can not play a very good optimization effect [32].

3 A hybrid algorithm based on GA and FA

3.1 The model of GFA

In this section, this paper introduces a new hybrid algorithm – GFA . The core idea of the GFA is that when the FA falls into the local optimal solution, the local

optimal fireflies would be regarded as a group while the luminosity formula would be regarded as the fitness function. Then, the group will be genetically operated to obtain the brightest firefly [33].

The algorithm model of GFA is as follows: Suppose there are n fireflies trapped in local optimal solution, and each firefly has its own luminosity. Considering the luminosity expression formula as a fitness function, then each firefly has its own fitness value corresponding to its luminosity [34].

By regarding n fireflies trapped into the local optimal solution as a parent group, selection operation, cross operation and mutation operation are performed to the parent group to finally obtain the brightest firefly through continuous genetic iterations [35].

However, according to the algorithm model of FA, all fireflies trapped in the local optimal solution will approach the brightest firefly. So, there must be a firefly X_j , its fitness value (luminosity) I_j is greater than other fireflies [36]. This phenomenon will lead to the highest probability of X_j being selected in selection operation. Based on this theory, this paper introduces a selection operation method of roulette wheel [37].

Roulette wheel selection method, also known as ratio selection method, its basic idea is dividing the ratio on the roulette according to the individual fitness value. When rotating the wheel, the number of the pie chart pointed by the pointer changes continuously [38].

Finally, roulette wheel would be stationary, and the number which is pointed by the pointer will be the number selected by the roulette wheel [39]. The model of roulette wheel can be described as follows:

Assuming that there are n individuals in a group, and the fitness value of i is f_i , then the probability that i will be selected is as follows:

$$P_i = \frac{f_i}{\sum f_i}, i = 1, 2, 3, \dots, n. \quad (11)$$

The cumulative probability can be expressed as follows:

$$CP_i = \sum_{j=1}^i P_j. \quad (12)$$

It can be drawn from Eqs. (11) and (12) that when the fitness value of i is the largest, the corresponding selection probability is also the largest. However, this paper finds that if the fitness value of an individual is much larger than the fitness value of other individuals, its proportion in the disc pie chart will also be much larger than the proportion of the pie chart occupied by other individuals [40].

In this case, the roulette wheel selection method will converge prematurely or even fail. In order to improve the convergence of roulette wheel selection method and prevent its premature convergence, this paper proposes a new selection probability formula:

$$P_i = Z \times \frac{f_i}{\sum f_i}, i = 1, 2, 3, \dots, n, \quad (13)$$

where Z is the proportional function and its expression is as follows:

$$Z = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}} \times \frac{1}{n - 1}. \quad (14)$$

If the fitness value of an individual is much larger than the fitness value of other individuals. Through Eqs. (12) and (13), the proportion in the disc pie chart can be adjusted, and the roulette wheel selection method can be prevented from prematurely converging. The new roulette wheel selection method is applied to the GFA, and the pseudo code of the GFA is sorted out as shown in Algorithm 3.

Algorithm 3 Iterative algorithm for GFA

```

1: population initialization, put  $t = 0$ 
2: generate  $N$  fireflies(solutions)  $X_i, i = 1, 2, \dots, N$ 
3: compute the fitness value of each firefly based on equation
   (3)
4: loop
5:   while  $t < T$  do
6:     for  $i = 1$  to  $N$  do
7:       for  $j = 1$  to  $N$  do
8:         if  $L_i > L_j$  then
9:            $X_j$  moves to  $X_i$  based on equation (4)
10:          evaluate the luminosity and position of  $X_j$ 
11:          if  $\alpha_{X_j} \neq 0$  then
12:             $H = []$ 
13:            for  $\beta = 1$  to  $j$  do
14:               $H = [H, f(X_\beta)]$ 
15:            end for
16:          end if
17:        end if
18:      end for
19:    end for
20:     $t++$ 
21:  end loop
22: output  $H$ 
23: parameters initialization, put  $t = 0$ 
24: loop
25:   while  $t < T$  do
26:     select parents from  $H$  to create offspring
27:     luminosity as fitness values
28:     evaluate  $P_i$  of each firefly based on equation (12)
29:     evaluate  $CP_i$  of each firefly based on equation (11)
30:     simulate the operation of roulette and pick out offspring
31:     apply crossover operation using crossover rate
32:     apply mutation operation using mutation rate
33:     evaluate new fitness values
34:     sort and rank fitness values
35:      $t++$ 
36:   end loop
37: get best solution

```

The basic process of GFA can be summarized as follows:

Step 1: **Population initialization**—generate N fireflies(solutions) X_i , $i = 1, 2, \dots, N$, and compute the fitness value of each firefly based on Eqs. (3) and (4).

Step 2: **Firefly movement**—If the fitness value $f(X_i)$ is better than $f(X_j)$, X_i moves toward X_j based on Eq. (5) and compute the new fitness value of X_j .

Step 3: **Judgment condition**—If $\alpha_{X_j} \neq 0$, Put new $f(X_j)$ obtained in step 2 into Set H.

Step 4: **Genetic manipulation**—Regard luminosity value as fitness value, select parents from H to create offspring. Evaluate P_i and CP_i of each firefly based on Eqs. (12) and (13). Then, simulate the operation of roulette and pick out offspring.

Step 5: **Other operations**—Perform crossover and mutation operations to generate new individual.

Step 6: **Termination condition**—Get a new generation of population and return to step 2.

Figure 2 shows the Schematic flowchart of GFA.

In the previous section, we used some binary nonlinear functions to prove that the FA is easy to fall into a local optimal solution. In this section, we will focus on using the proposed algorithm to solve this problem. Still choose $f_5(x, y)$ and $f_6(x, y)$ shown in Table 1 as the objective functions. Setting the number of fireflies to 20, the step factor α to 0.8, and the light absorption coefficient γ to 0.3. Finally, simulation results are shown in Fig. 3.

As shown in Fig. 3, GFA can make the fireflies finally approach the maximum extreme point of the function. Thus, proposed algorithms can solve the problem that the FA is easy to fall into the local optimal solution.

In addition, this paper sorts out the simulation results of two algorithms when optimizing the objective functions, as shown in Fig. 4. Table 2 summarizes the computational results (Function value and Number of iteration) of these two algorithms.

As shown in Table 2, for these 6 binary nonlinear functions, both algorithms have found maximum values of the functions. However, it is worth noting that the number of iterations required by the two algorithms for optimization is different.

Specifically, for most functions, GFA requires fewer iterations than AFA. For example, for f_5 and f_6 , GFA only needs 3 or 4 times to calculate the optimal value of the function. In a sense, it can be confirmed that GFA has faster calculation speed than AFA, and can solve the problem that the FA is easy to fall into the local optimal solution faster.

Of course, it is not enough to rely on the computational results shown in Table 2 to prove the competitiveness of the proposed algorithm. The performance of GFA is reflected experimentally.

3.2 Simulation analysis

In this section, we will conduct more experiments to verify the competitiveness of the proposed algorithm, we select another nine famous test functions to test the

performance of GFA. The expressions of the famous test functions are shown in Table 3, the global optimum values of these functions are all 0. At the same time, we introduce some other algorithms for comparison. The selected algorithms are listed below:

- Standard FA (**FA**)—[13].
- Genetic Algorithm (**GA**)—[22].
- Sparrow Algorithm (**SSA**)—[41].
- Adaptive Parameters FA (**AFA**)—[19].
- Particle Swarm Algorithm (**PSO**)—[42].
- Whale Optimize Algorithm (**WOA**)—[43].

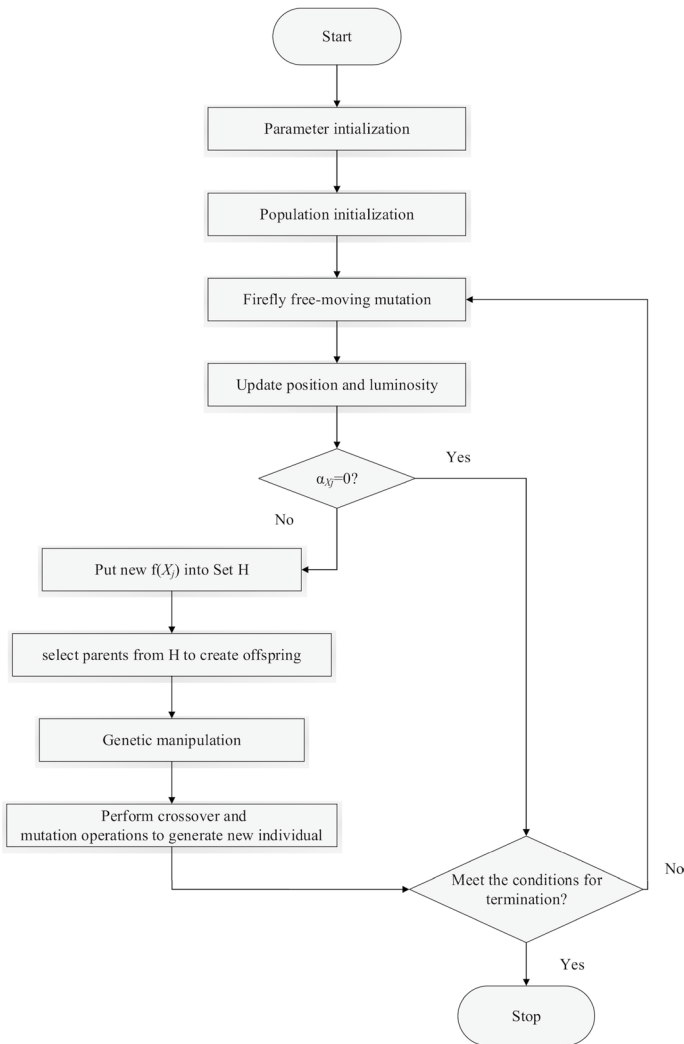


Fig. 2 The schematic flowchart of GFA

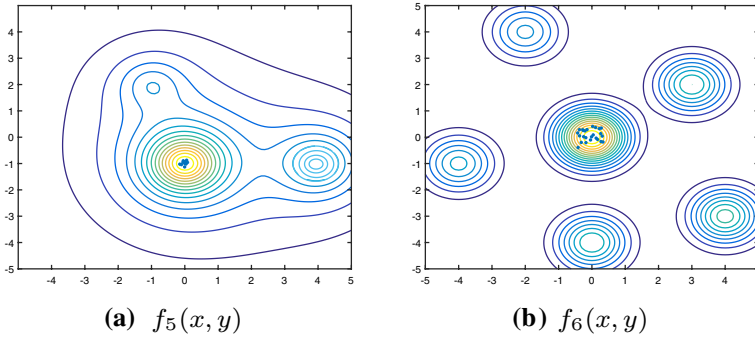


Fig. 3 Nonlinear optimization results by using GFA, AFA

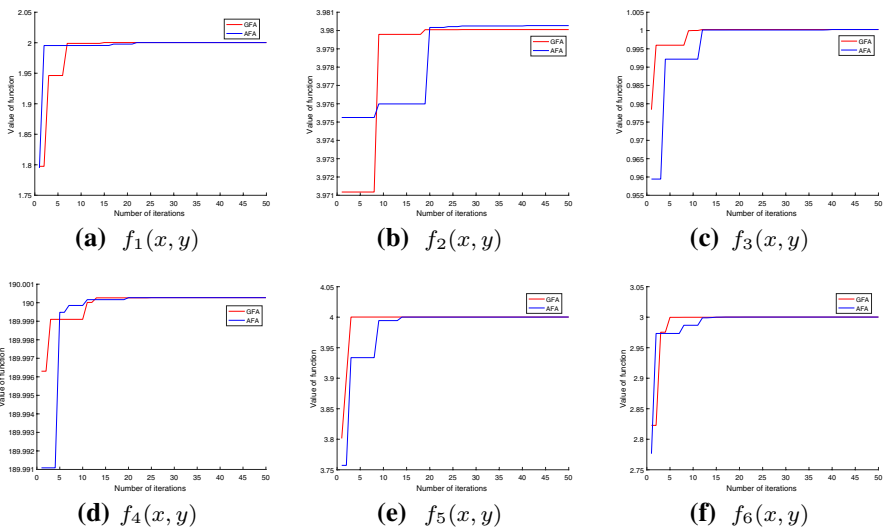


Fig. 4 The simulation results of GFA and AFA

Set the problem dimension D to 50, the maximum population N to 100, the maximum iteration number T_{\max} to 1000, the light absorption coefficient γ to 0.8, and the step factor α to 0.6. In the experiment, above selected algorithms were all run 100 times. The inspiration for parameters selection comes from [19].

It is worth noting that a fixed initial population should be used for all experiments to ensure the fairness of the experiment. In addition, for any algorithm and its variants, we use the same initial parameters.

Finally, simulation results were obtained, as shown in Fig. 5, where the abscissa represents the number of iterations, and the ordinate represents the best function value.

From the results shown in Fig. 5, it can be found out that for these nine test functions, the function values optimized by these algorithms are different, where

Table 2 Computational results achieved by GFA and AFA

| Function | GFA | | AFA | |
|-------------|----------|------------|----------|------------|
| | Value | Iterations | Value | Iterations |
| $f_1(x, y)$ | 2 | 7 | 2 | 22 |
| $f_2(x, y)$ | 3.98 | 18 | 3.982 | 20 |
| $f_3(x, y)$ | 1 | 7 | 1 | 11 |
| $f_4(x, y)$ | 190.0004 | 13 | 190.0004 | 21 |
| $f_5(x, y)$ | 4 | 3 | 4 | 14 |
| $f_6(x, y)$ | 3 | 4 | 3 | 12 |

the results obtained by GFA and GA are often better than other algorithms. In particular, for most test functions, the optimization effect of the FA is not good.

In order to identify the best algorithm, the Friedman test [44] is conducted on the data shown in Table 4. The results are listed in Table 5.

According to Table 5, we can found out that GFA, WOA, and GA have lower relevance rank than AFA, FA, PSO, although the gap between these three algorithms is not large. To be specific, their relevance ranks are 1.37, 2.08, 2.29.

Lower rank represents stronger relevance, obviously, GFA achieves the best rank. Thus, GFA performs best in optimization for these nine benchmark functions. As compared with other algorithms, its performance is better so that better solutions can be produced of the test functions.

Although the proposed algorithm performs best in optimization for these 9 benchmark functions, we still find many drawbacks in the experimental process for proposed algorithm. First, the proposed algorithm depends on the choice of parameters. Then, although proposed algorithm can solve the problem that FA is easy to fall into the local optimal solution, it has the problem of slow convergence. Finally, the proposed algorithm is a hybrid algorithm, on the basis of the original algorithm, it only maintains the diversity of the population, but does not improve the diversity.

4 Application in mobile robot path planning

4.1 Simulation environment

Abstractly expressing the surrounding environment of a mobile robot with a set of data, establishing a two-dimensional or three-dimensional environmental model, and obtaining environmental data that the mobile robot can understand and analyze is the basic premise of robot path planning [45].

In this paper, the grid method is used. The principle is to regard the surrounding environment as a two-dimensional (three-dimensional) plane and divide the plane into grids with two-value (three-value) information of equal area.

In each grid, the amount of information about the surrounding environment is stored. The grid map designed here is a 10×10 ($10 \times 10 \times 10$) terrain matrix,

Table 3 Test functions used in the experiments

| Function name | Function | Search range | Global optimum |
|--------------------|---|--------------|----------------|
| Sphere | $f_1(x) = \sum_{i=1}^D x_i^2$ | [-100, 100] | 0 |
| Schwefel 1.2 | $f_2(x) = \sum_{i=1}^D (\sum_{j=1}^i x_j)^2$ | [-100, 100] | 0 |
| Schwefel 2.21 | $f_3(x) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i$ | [-10, 10] | 0 |
| Schwefel 2.22 | $f_4(x) = \max\{ x_i , 1 \leq i \leq D\}$ | [-50, 50] | 0 |
| Schwefel 2.26 | $f_5(x) = \sum_{i=1}^D -x_i \sin(\sqrt{ x_i })$ | [-500, 500] | 0 |
| Step | $f_6(x) = \sum_{i=1}^D [x_i + 0.5]$ | [-100, 100] | 0 |
| Quartic with Noise | $f_7(x) = \sum_{i=1}^D i \times x_i^4 + \text{random}[0, 1)$ | [-1.3, 1.3] | 0 |
| Rastrigin | $f_8(x) = \sum_{i=1}^D [x_i^2 - 10 \cos 2\pi x_i + 10]$ | [-5, 5] | 0 |
| Griewank | $f_9(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$ | [-600, 600] | 0 |

black areas indicate obstacles, and white areas indicate no obstacles. Path, here select the evaluation criteria of the shortest distance path planning, that is, the shortest path planning problem [46].

In addition, when using the grid method to build an environmental model, in order to convert the environmental information into data that can be recognized by the mobile robot, the serial number method is generally used to mark the environmental map information, that is, the grids are sequentially accumulated from number 0 to the last grid.

For a grid map with coordinates, each grid point has a unique coordinate. Assuming that the grid size is x rows and y columns, the position corresponding to the i -th grid is shown in Eq. (15).

$$\begin{cases} x_i = a \times [\text{mod}(i, y) - a/2] \\ y_i = a \times [x + a/2 - \text{ceil}(i/x)] \end{cases} \tag{15}$$

where a is the side length of each small square pixel.

In this paper, each grid has a side length of 1 cm and its serial number is from 0 to 99(999). $\text{Ceil}(n)$ is the smallest integer greater than or equal to the value n , and $\text{mod}(i, y)$ is the remainder of i divided by y [47].

Here, some definitions are required.

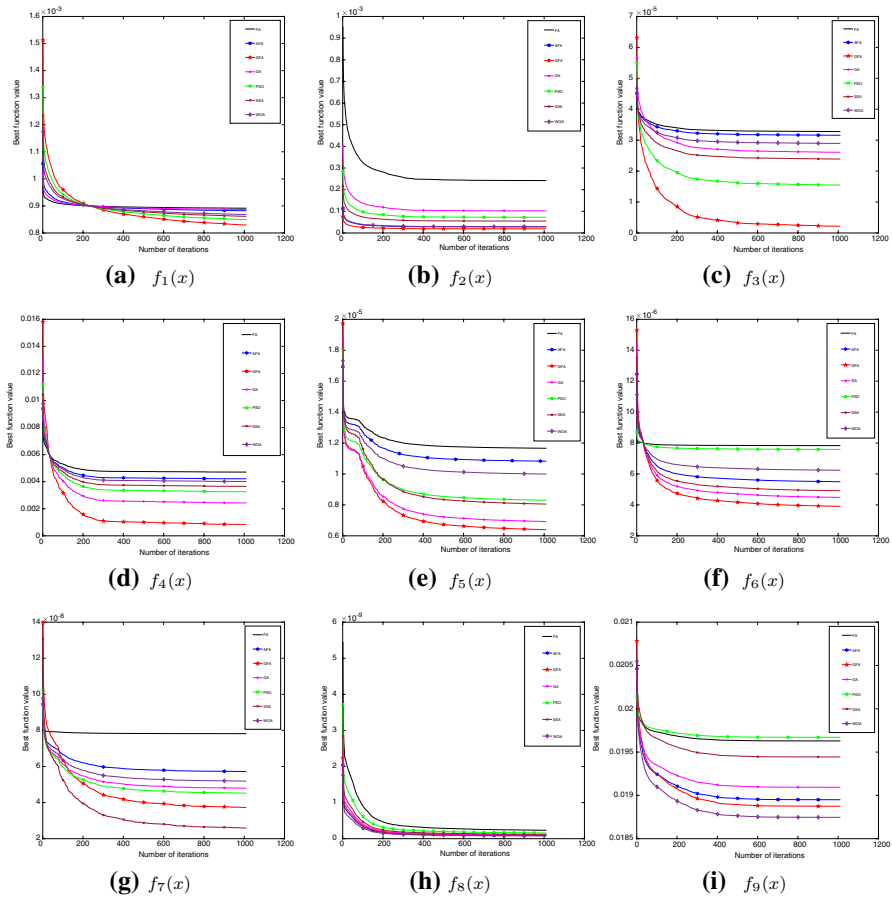


Fig. 5 Simulation results of 9 test functions for 7 algorithms under 50 dimensions

Definition 1 Two-dimensional path planning evaluation function $L_{2D}(n)$:

$$L_{2D}(n) = \sqrt{(x_n - x_{end})^2 + (y_n - y_{end})^2}. \tag{16}$$

where (x_n, y_n) is the coordinate of the center point of the n grid, (x_{end}, y_{end}) is the coordinate of the end point of the planned path. $L(n)$ represents the total distance of the planned path.

Definition 2 Three-dimensional path planning evaluation function $L_{3D}(n)$:

$$L_{3D}(n) = \sqrt{(x_n - x_e)^2 + (y_n - y_e)^2 + (z_n - z_e)^2}. \tag{17}$$

Table 4 Computational results achieved by different seven algorithms

| | | <i>GFA</i> | <i>GA</i> | <i>AFA</i> | <i>FA</i> | <i>PSO</i> | <i>WOA</i> | <i>SSA</i> |
|----------|------|----------------|-----------|------------|-----------|------------|----------------|----------------|
| $f_1(x)$ | Mean | 2.0E-07 | 3.1E-05 | 1.4E-03 | 3.2E-02 | 5.2E-05 | 7.9E-05 | 9.1E-04 |
| | Best | 0.0E+00 | 3.4E-18 | 2.4E-14 | 7.8E-09 | 4.8E-22 | 4.1E-20 | 5.2E-18 |
| $f_2(x)$ | Mean | 1.6E-09 | 2.3E-04 | 2.9E-07 | 5.2E-01 | 3.1E-05 | 4.2E-08 | 8.3E-06 |
| | Best | 0.0E+00 | 3.7E-15 | 7.8E-28 | 1.2E-11 | 1.7E-17 | 4.8E-29 | 6.4E-25 |
| $f_3(x)$ | Mean | 8.6E-07 | 4.8E-03 | 2.2E-02 | 8.1E+00 | 7.3E-05 | 6.9E-03 | 7.9E-04 |
| | Best | 0.0E+00 | 9.8E-12 | 8.4E-07 | 7.1E-04 | 9.5E-20 | 5.4E-11 | 3.9E-16 |
| $f_4(x)$ | Mean | 3.6E-03 | 5.2E-03 | 1.7E-01 | 8.7E-01 | 1.5E-02 | 8.4E-02 | 2.9E-02 |
| | Best | 4.1E-08 | 5.8E-07 | 4.1E-06 | 6.8E-05 | 7.7E-07 | 1.6E-06 | 9.3E-07 |
| $f_5(x)$ | Mean | 6.9E-08 | 1.6E-07 | 1.8E-03 | 2.6E-01 | 6.9E-07 | 3.7E-04 | 5.3E-07 |
| | Best | 5.8E-18 | 9.9E-15 | 4.1E-10 | 3.9E-04 | 4.1E-14 | 8.8E-12 | 3.5E-14 |
| $f_6(x)$ | Mean | 4.7E-06 | 8.8E-06 | 4.8E-05 | 1.9E-02 | 9.3E-03 | 8.2E-05 | 3.1E-05 |
| | Best | 2.1E-11 | 5.9E-10 | 9.3E-10 | 3.9E-03 | 4.1E-04 | 1.2E-09 | 7.8E-10 |
| $f_7(x)$ | Mean | 6.1E-08 | 4.2E-07 | 1.3E-05 | 5.6E-02 | 1.9E-07 | 5.9E-07 | 3.2E-08 |
| | Best | 5.0E-14 | 2.1E-13 | 4.6E-11 | 5.9E-04 | 7.9E-14 | 4.9E-12 | 4.1E-16 |
| $f_8(x)$ | Mean | 2.0E-13 | 7.0E-13 | 3.2E-14 | 3.5E-09 | 3.4E-12 | 2.4E-14 | 4.6E-13 |
| | Best | 5.3E-24 | 7.8E-24 | 3.8E-24 | 6.7E-15 | 2.7E-23 | 0.0E+00 | 7.1E-24 |
| $f_9(x)$ | Mean | 8.2E-02 | 4.7E-01 | 1.2E-01 | 8.5E-01 | 9.1E-01 | 6.8E-02 | 7.1E-01 |
| | Best | 2.9E-04 | 6.2E-04 | 3.4E-04 | 9.2E-03 | 9.8E-03 | 1.8E-04 | 6.7E-03 |

Bold values represent the best value for comparison between different algorithms in the experimental results

Table 5 Friedman test results

| Algorithm | Relevance rank |
|------------|----------------|
| <i>AFA</i> | 3.47 |
| <i>GFA</i> | 1.37 |
| <i>FA</i> | 4.18 |
| <i>GA</i> | 2.29 |
| <i>PSO</i> | 3.18 |
| <i>WOA</i> | 2.08 |
| <i>SSA</i> | 2.73 |

Bold values represent the best value for comparison between different algorithms in the experimental results

Definition 3 Robot movement method:

Octree search strategy: the robot can move freely between adjacent grids in eight nearby directions during the search process. It is assumed that k is the next grid node to be selected, which can be defined as:

$$k = \min_{n=1}^8(L(n)). \tag{18}$$

Definition 4 All environmental boundaries are represented by obstacles.

4.2 Simulation results and analysis

In this section, this paper applies the GFA to the path planning of mobile robot, and analyzes simulation results. The performance of GFA for path planning is also reflected experimentally, AFA, FA, and GFA are tested in the same grid environment for many times.

The specific parameters of GFA, FA, and AFA used in path planning are shown in Table 6. The inspiration for parameters selection comes from [46].

In the experiment, the above-mentioned algorithms were all operated for 200 times. We have selected nine different grid environments to evaluate the algorithm for path planning, most of which are more complicated.

Table 7 shows the constraint conditions of each grid environment in 9 different grid environments, where *Start* represents the starting point number in 2D (3D) environments, *End* represents the ending point number in 2D (3D) environments, and L_{\min} represents the shortest path length in different grid environments.

The simulation results are shown in Fig. 6, where the line marked with some circles represents GFA.

From the simulation results, it can be found out that three algorithms have successfully found a path from the starting point to the ending point, and achieved completing avoidance of all obstacles. To compare the advantages and disadvantages of three algorithms more effectively, the minimum path length and the number of iterations in the same grid environment are recorded. As shown in Fig. 7.

Figure 7 shows the simulation results of the three algorithms in different dimensional grid environments. The blue waveform represents GFA, the orange waveform represents AFA and the red waveform represents FA. Similarly, the abscissa represents the number of iterations while the ordinate represents the minimum path length.

When algorithm finds the optimal path, the waveform will start to stabilize. In order to identify the best algorithm, the Friedman test [44] is conducted on the data shown in Tables 8 and 9. The results are listed in Table 10.

It is worth noting that in Tables 8 and 9, $L(cm)$ represents the shortest path length calculated by the algorithm, $T(n)$ represents the number of iterations required for the algorithm to calculate the shortest path length, and $t(s)$ represents the time that it takes for the algorithm to calculate the result.

From the data resulting in Table 10, some conclusions can be drawn. As far as the algorithm alone is concerned, both L and T achieve better relevance ranks than t with this conclusion is valid in both 2D and 3D environments. To be specific, the relevance ranks for L are 1.00, 2.92, 2.89, and 1.78, which means that L achieves the best rank.

Thus, for these nine different grid environments, L is an important parameter for evaluating path planning, while T is ranked behind it.

Table 6 Simulation parameters

| Parameter | <i>GFA</i> | <i>FA</i> | <i>AFA</i> |
|---|------------|-----------|------------|
| | Value | Value | Value |
| Coefficient of disturbance (ϵ) | 0.2 | 0.2 | 0.2 |
| Firefly induction radius (R) | 20 | 20 | 20 |
| Number of iterations (T) | 100 | 100 | 100 |
| Number of firefly (N) | 200 | 200 | 200 |
| Step factor (α) | 0.8 | 0.8 | 0.8 |
| Light absorption coefficient(γ) | 0.3 | 0.3 | 0.3 |
| Crossover probability (P_c) | 0.6 | – | – |
| Mutation probability (P_m) | 0.2 | – | – |
| Induction radius expansion (R) | 20 | 20 | 20 |

Table 7 Environmental parameters

| Environment | <i>Start</i> | <i>End</i> | $L_{\min}(2D)$ | $L_{\min}(3D)$ |
|-------------|--------------|------------|----------------|----------------|
| | Value | Value | Value | Value |
| <i>E1</i> | 4(4) | 86(854) | 8.2 | 25.9 |
| <i>E2</i> | 11(11) | 87(864) | 9.0 | 27.2 |
| <i>E3</i> | 0(0) | 88(874) | 14.2 | 26.6 |
| <i>E4</i> | 0(0) | 88(874) | 14.8 | 25.0 |
| <i>E5</i> | 0(0) | 88(874) | 11.3 | 25.3 |
| <i>E6</i> | 0(0) | 99(984) | 18.8 | 26.6 |
| <i>E7</i> | 5(5) | 85(844) | 6.9 | 26.0 |
| <i>E8</i> | 5(5) | 85(844) | 8.6 | 24.8 |
| <i>E9</i> | 5(5) | 85(844) | 9.0 | 25.0 |

In the case of the comparison between the three algorithms, the rank for L calculated by GFA is lower than the ranks calculated by the other two algorithms in different dimensions.

In addition, the Friedman test [44] results of three algorithms (overall) for these different 9 grid environments are also recorded in Table 10. In detail, their relevance ranks are 1.09, 2.92, 1.95 under 2D environment and 1.10, 2.94, 1.97 under 3D environment, which means that GFA achieves the best rank.

Thus, GFA performs best in path planning for these 9 grid environments. As compared with other algorithms, its performance is far better so that better path solutions can be produced under different grid environments.

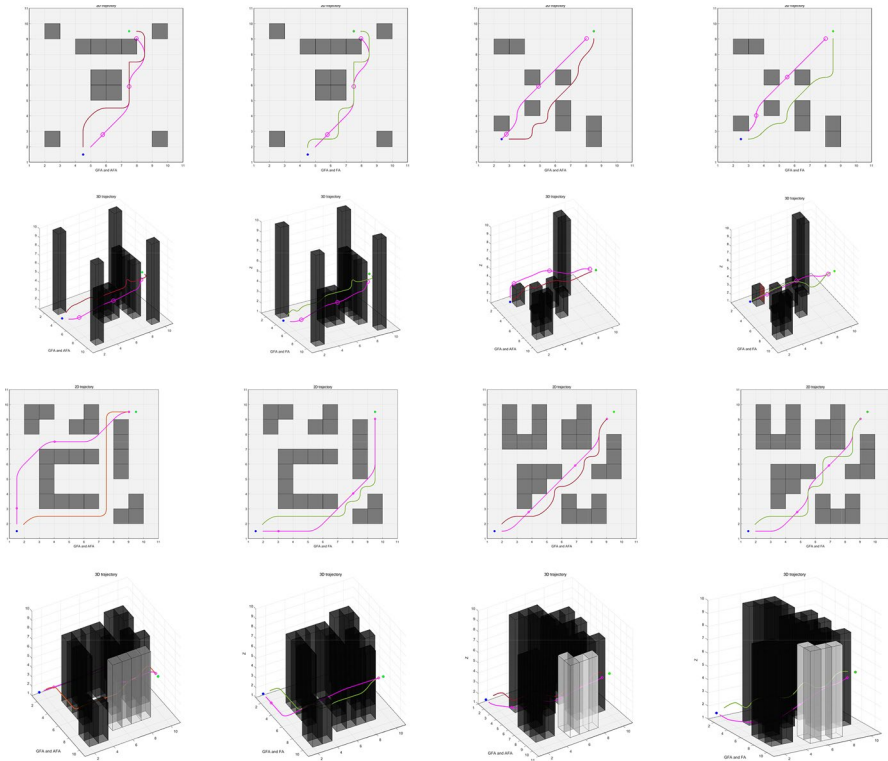
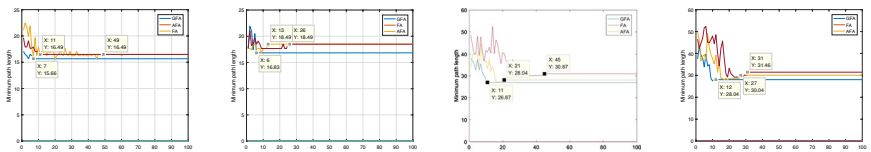


Fig. 6 Path trajectory in different environments



(a) 2D under environment 3 (b) 2D under environment 4 (c) 3D under environment 3 (d) 3D under environment 4

Fig. 7 Path results in different environments

5 Conclusion

As mentioned above, GFA can solve the problem that the FA is easy to fall into the local optimal solution faster, so that fireflies trapped into the local optimal solution converge to one point at a faster rate.

As verified by the theoretical and experimental results, GFA outperforms other algorithms on most of the test functions, it usually takes less iterations to find the optimal value of the function. Moreover, as compared with other algorithms, GFA

Table 8 Computational results achieved by three algorithms in different 2D environments

| | | <i>GFA(2D)</i> | | | <i>FA(2D)</i> | | | <i>AFA(2D)</i> | | |
|-----------|------|----------------|-------------|-------------|---------------|-------------|-------------|----------------|-------------|-------------|
| | | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> |
| <i>E1</i> | Mean | 10.56 | 8 | 3.014 | 13.87 | 24 | 3.036 | 12.79 | 15 | 3.023 |
| | Best | 8.35 | 5 | 3.002 | 9.57 | 19 | 3.016 | 8.89 | 12 | 3.009 |
| <i>E2</i> | Mean | 10.16 | 12 | 3.086 | 16.13 | 23 | 3.173 | 14.31 | 19 | 3.115 |
| | Best | 9.52 | 8 | 3.035 | 10.47 | 16 | 3.146 | 9.65 | 13 | 3.082 |
| <i>E3</i> | Mean | 17.14 | 11 | 3.055 | 19.33 | 57 | 3.133 | 18.76 | 16 | 3.094 |
| | Best | 15.66 | 7 | 3.021 | 16.49 | 49 | 3.106 | 16.49 | 11 | 3.051 |
| <i>E4</i> | Mean | 19.52 | 9 | 3.023 | 22.57 | 30 | 3.092 | 20.31 | 21 | 3.086 |
| | Best | 16.83 | 6 | 3.009 | 18.79 | 26 | 3.077 | 18.49 | 14 | 3.037 |
| <i>E5</i> | Mean | 13.85 | 16 | 3.481 | 21.34 | 41 | 3.732 | 19.08 | 27 | 3.428 |
| | Best | 11.42 | 14 | 3.343 | 17.75 | 36 | 3.711 | 16.52 | 23 | 3.300 |
| <i>E6</i> | Mean | 23.19 | 17 | 3.092 | 23.67 | 33 | 3.137 | 23.72 | 31 | 3.081 |
| | Best | 19.15 | 12 | 3.052 | 21.70 | 30 | 3.086 | 21.66 | 26 | 3.068 |
| <i>E7</i> | Mean | 9.32 | 10 | 3.208 | 10.23 | 25 | 3.214 | 10.14 | 18 | 3.213 |
| | Best | 7.97 | 8 | 3.162 | 8.66 | 19 | 3.193 | 7.05 | 12 | 3.175 |
| <i>E8</i> | Mean | 12.24 | 7 | 3.081 | 13.08 | 18 | 3.080 | 12.37 | 14 | 3.031 |
| | Best | 11.78 | 4 | 3.013 | 12.06 | 14 | 3.059 | 8.83 | 11 | 3.017 |
| <i>E9</i> | Mean | 12.88 | 9 | 3.554 | 18.50 | 24 | 3.918 | 14.60 | 12 | 3.652 |
| | Best | 9.46 | 6 | 3.541 | 15.11 | 18 | 3.893 | 13.39 | 10 | 3.606 |

Table 9 Computational results achieved by 3 algorithms in different 3D environments

| | | <i>GFA(3D)</i> | | | <i>FA(3D)</i> | | | <i>AFA(3D)</i> | | |
|-----------|------|----------------|-------------|-------------|---------------|-------------|-------------|----------------|-------------|-------------|
| | | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> | <i>L(cm)</i> | <i>T(n)</i> | <i>t(s)</i> |
| <i>E1</i> | Mean | 27.33 | 14 | 4.253 | 37.80 | 47 | 4.804 | 32.34 | 31 | 4.211 |
| | Best | 26.58 | 9 | 4.162 | 35.11 | 38 | 4.662 | 29.10 | 27 | 4.177 |
| <i>E2</i> | Mean | 27.93 | 18 | 3.579 | 31.82 | 33 | 4.280 | 32.86 | 26 | 3.700 |
| | Best | 27.52 | 15 | 3.271 | 30.69 | 26 | 4.156 | 29.19 | 18 | 3.516 |
| <i>E3</i> | Mean | 28.02 | 15 | 5.082 | 32.95 | 58 | 5.201 | 30.33 | 33 | 5.131 |
| | Best | 26.87 | 11 | 5.078 | 30.87 | 45 | 5.184 | 28.04 | 21 | 5.099 |
| <i>E4</i> | Mean | 28.44 | 13 | 5.018 | 33.27 | 32 | 5.105 | 30.53 | 34 | 5.069 |
| | Best | 28.04 | 12 | 5.012 | 31.46 | 31 | 5.070 | 30.08 | 25 | 5.027 |
| <i>E5</i> | Mean | 26.06 | 17 | 3.188 | 30.14 | 25 | 3.411 | 29.52 | 23 | 3.293 |
| | Best | 25.77 | 8 | 3.142 | 28.82 | 19 | 3.310 | 28.79 | 19 | 3.179 |
| <i>E6</i> | Mean | 30.46 | 28 | 7.433 | 38.90 | 51 | 7.726 | 36.03 | 28 | 7.201 |
| | Best | 29.72 | 22 | 6.612 | 35.24 | 48 | 6.922 | 35.66 | 20 | 6.839 |
| <i>E7</i> | Mean | 29.29 | 16 | 4.517 | 30.18 | 44 | 6.301 | 29.34 | 29 | 4.713 |
| | Best | 28.12 | 12 | 4.329 | 28.56 | 35 | 5.226 | 26.83 | 24 | 4.479 |
| <i>E8</i> | Mean | 27.20 | 10 | 3.631 | 31.25 | 34 | 4.037 | 30.48 | 19 | 3.994 |
| | Best | 25.13 | 6 | 3.580 | 30.50 | 29 | 4.006 | 30.26 | 18 | 3.849 |
| <i>E9</i> | Mean | 26.44 | 26 | 4.334 | 30.26 | 49 | 4.430 | 27.46 | 38 | 4.302 |
| | Best | 26.18 | 15 | 4.318 | 29.99 | 42 | 4.381 | 25.66 | 27 | 4.285 |

Table 10 Friedman test results of path planning

| | | $L(cm)$ | $T(n)$ | $t(s)$ |
|------------|----|-------------|-------------|--------|
| Algorithm | | Rank | Rank | Rank |
| <i>GFA</i> | 2D | 1.01 | 1.03 | 1.28 |
| | 3D | 1.02 | 1.08 | 1.22 |
| <i>FA</i> | 2D | 2.92 | 3.00 | 3.00 |
| | 3D | 2.89 | 2.92 | 3.00 |
| <i>AFA</i> | 2D | 1.78 | 2.00 | 2.08 |
| | 3D | 1.78 | 2.00 | 2.11 |
| | | $2D(rank)$ | $3D(rank)$ | |
| <i>GFA</i> | | 1.09 | 1.10 | |
| <i>FA</i> | | 2.92 | 2.94 | |
| <i>AFA</i> | | 1.95 | 1.97 | |

shows better algorithm performance, which is more suitable for solving practical problems.

Applying the GFA to the path planning of mobile robots can improve the computing power and reaction speed of mobile robots, which also can make mobile robots find a route to avoid obstacles more quickly in different dimensional environment.

The final experiments also show that when GFA optimizes the path planning, it can be shown to yield better solutions. As opposed to FA and AFA, GFA tends to show better competitiveness which means that it has good research value and significance.

There are still many points for improvement in the research work of this paper, such as how to apply the GFA in other fields, how to apply the GFA in a dynamic grid environment, and how to improve the performance of the GFA from other aspects.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61603127.

References

1. Patle BK, Babu GL, Pandey A (2019) A review: on path planning strategies for navigation of mobile robot. *Def Technol* 15(2):582–606
2. Pandey A, Parhi DR (2017) Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm. *Def Technol* 13(1):47–58
3. Daniel LC, Constantin V (2014) A 2D chaotic path planning for mobile robots accomplishing boundary surveillance missions in adversarial conditions. *Commun Nonlinear Sci* 10:3617–3627
4. Fan XJ, Jiang MY, Pei ZL (2018) Research on path planning of mobile robot based on genetic algorithm in dynamic environment. *Basic Clin Pharmacol* 124(13):54
5. Tang L, Liu GJ, Yang M (2020) Joint design and torque feedback experiment of rehabilitation robot. *Adv Mech Eng*. <https://doi.org/10.1177/1687814020924498.2020.924498>
6. Dewang HS, Mohanty PK, Kundu S (2017) A robust path planning for mobile robot using smart particle swarm optimization. *Proc Comput Sci* 133(6):290–297

7. Liu CG, Yan XH, Liu CY (2010) Dynamic path planning for mobile robot based on improved genetic algorithm. *Chin J Electron* 2:245–248
8. Wang PD, Feng ZH, Huang X (2018) An improved ant colony algorithm for mobile robot path planning. *Robot* 6:554–560
9. Anatolii VM, Vladimir VM, Leonid MS (2019) Adaptive genetic algorithms used to analyze behavior of complex system. *Commun Nonlinear Sci* 3(8):174–186
10. Maria GM, Tenreiro M, Azevedo P (2009) Trajectory planning of redundant manipulators using genetic algorithms. *Commun Nonlinear Sci* 7:2858–2869
11. Michael AL (2014) Metaheuristics in nature-inspired algorithms. *ACM* 14:1419–1422
12. Yang XH (2013) Multiobjective firefly algorithm for continuous optimization. *Eng Comput Germany* 2(4):175–184
13. Fister I, Fister II, Yang XH (2013) A comprehensive review of firefly algorithms. *Swarm Evol Comput* 13:34–46
14. Almasi ON, Rouhani M (2016) A new fuzzy membership assignment and model selection approach based on dynamic class centers for fuzzy SVM family using the firefly algorithm. *Turk J Electr Eng Co* 24(3):1797-U5073
15. Ali N, Othman MA, Husain MN (2014) A review of firefly algorithm. *ARPN J Eng Appl Sci* 10(3):1732–1736
16. Pal NS, Sharma S (2013) Robot path planning using swarm intelligence: a survey. *Appl Int J Comput*
17. Gong P, Wang W, Li F (2018) Sparsity-aware transmit beamspace design for FDA-MIMO radar. *Signal Process* 18(5):99–103
18. Abdelaziz A, Mekhamer S, Badr M (2015) The firefly metaheuristic algorithms: developments and applications. *Int Elect Eng J* 7(13):1945–1952
19. Wang H, Zhou XY, Sun H (2017) Firefly algorithm with adaptive control parameters. *Soft Comput* 17:5091–5102
20. Has T, Meybodi MR, Shahramirad M (2017) A new fuzzy firefly algorithm with adaptive parameters. *Int J Comput Intell Appl* 3(2):34–39
21. Tilahun SL, Ngnotchouye JM, Hamadneh NN (2019) Continuous versions of firefly algorithm: a review. *Artif Intell Rev* 3:445–462
22. Beasley D, Bull DR, Martin RR (1993) An overview of genetic algorithms. *Univ Comput* 4:170–181
23. Lee CK (2018) A review of applications of genetic algorithms in operations management. *Eng Appl Art Intell* 5(2):1–12
24. Annu L, Kunal G, Kriti C (2019) Genetic algorithm: a literature review. *IEEE Cloud Paral Comput*
25. Li F, Li QX, Li YF (2019) Imaging with 3-D aperture synthesis radiometers. *IEEE Trans Geosci Remote* 57(3):2395–2406
26. Han T, Guan ZH, Xiao B (2019) Distributed output consensus of heterogeneous multi-agent systems via an output regulation approach. *Neurocomputing* 131–137
27. Archana AV (2012) A survey on image contrast enhancement using genetic algorithm. *Int J Sci Res Publi* 2:1–3
28. Yu SH, Yang SL, Su SB (2013) Self-adaptive step firefly algorithm. *J Appl Math*
29. Yang XS (2009) Firefly algorithms for multimodal optimization. *Proc Stoch Algo Found Appl* 3:169–178
30. Baykasoglu A, Ozsoydan FB (2015) Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Appl Soft Comput* 36(11):152–164
31. Cheung NJ, Ding XM, Shen HB (2014) Adaptive firefly algorithm: parameter analysis and its application. *PLoS ONE* 11:e112634
32. Jing W, Gui YL (2019) A novel firefly algorithm with self-adaptive step strategy. *Int J Innov Comput Appl* 1:18–26
33. Keshav K, Vikram A (2015) A hybrid data clustering using firefly algorithm based on improved genetic algorithm. *Pro Comput Sci* 58(7):249–256
34. Xu GH, Qi F, Lai Q (2020) Fixed time synchronization control for bilateral teleoperation mobile manipulator with nonholonomic constraint and time delay. *IEEE Trans Circuits Syst II Express Briefs* 67(12):3452–2456
35. Xu GH, Zhang TW, Lai Q (2021) A new firefly algorithm with mean condition partial attraction. *Appl Int*. <https://doi.org/10.1007/s10489-021-02642-6>
36. Lai Q, Chen CY, Zhao XW (2019) Constructing chaotic system with multiple coexisting attractors. *IEEE Access* 7:24051–24056

37. Olympia R (2017) Genetic algorithm and firefly algorithm hybrid schemes for cultivation processes modelling. *Trans Comput Colle Intell* 23(6):196–211
38. Thammano A, Teek W (2015) A modified genetic algorithm with fuzzy roulette wheel selection for job-shop scheduling problems. *Int J General Syst* 4(2):499–518
39. Zhan XS, Chen LL, Wu J (2019) Optimal modified performance of MIMO networked control systems with multi-parameter constraints. *ISA Trans* 84:111–117
40. Greinecker M, Podczek K (2015) Purification and roulette wheels. *Econ Theor* 2:255–272
41. Yuan JH, Zhao ZW, Liu YP (2021) DMPPT control of photovoltaic microgrid based on improved sparrow search algorithm. *IEEE Access* 9:16623–16629
42. Sedighzadeh D, Masehian E, Sedighzadeh M (2020) A new generalized particle swarm optimization algorithm. *Math Comput Sim* 179(46):194–212
43. Alsghaier H, Akour M (2020) Software fault prediction using whale algorithm with genetics algorithm. *Soft Pra Exper* 31(2):155–167
44. Garcia S, Fernandez A, Luengo J (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Inf Sci* 180(20):2044–2064
45. Yang BW, Ding ZM, Yuan L (2020) A novel urban emergency path planning method based on vector grid map. *IEEE Access* 8:154338–154353
46. Ho JJ, Kim DH (2020) Local path planning of a mobile robot using a novel grid-based potential method. *Int J Intell Syst* 20(1):26–34
47. Xiao SC, Tan XJ, Wang JP (2021) A simulated annealing algorithm and grid map-based UAV coverage path planning method for 3D reconstruction. *Electronics* 10(7):55–67

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.