Application of Improved Moth-Flame Optimization Algorithm for Robot Path Planning

XUEFENG DAI AND YANG WEI
School of Computer and Control Engineering, Qiqihar University, Qiqihar 161006, China
Corresponding author: Xuefeng Dai (daixuefeng203@163.com)

This work was supported in part by the General Project of Fundamental Scientific Research of Education Department of Heilongjiang Province, China, under Grant 135409421, and in part by the National Natural Science Foundation, China, under Grant 61672304.

ABSTRACT
Path planning is the focus and difficulty of research in the field of mobile robots, and it is the basis for further research and applications of robots. In order to obtain the global optimal path of the mobile robot, an improved moth-flame optimization (IMFO) algorithm is proposed in this paper. The IMFO features the following two improvements. Firstly, referring to the spotted hyena optimization (SHO) algorithm, the concept of historical best flame average is introduced to improve the moth-flame optimization (MFO) algorithm update law to increase the ability of the algorithm to jump out of the local optimum; Secondly, the quasi-opposition-based learning (QOBL) is used to perturb the location, increase the population diversity and improve the convergence rate of the algorithm. In order to evaluate the performance of the proposed algorithm, the IMFO algorithm is compared with three existing algorithms on three groups of different types of benchmark functions. The comparative results show that the IMFO algorithm is effective and has good performance in terms of jumping out of local optimum, balancing exploitation ability and exploration ability. Finally, the IMFO algorithm is applied to the path planning of the mobile robot, and computer simulations confirmed the algorithm’s effectiveness.

INDEX TERMS
Moth-flame optimization, quasi-opposition-based learning, spotted hyena optimization, path planning.

I. INTRODUCTION
Path planning technology is one of the core elements of research in the field of mobile robotics, and its purpose is to generate an optimal or near-optimal collision-free path from the starting point to the end point. Commonly used criteria for generating paths include the shortest path length, the smallest turn angle, and the best safety. Depending on whether the mobile robot knows a priori information about its resident environment, the path planning problem can be divided into two categories: global path planning and local path planning. In global path planning, the robot has a priori information about the environment. However, in local path planning, the mobile robot has completely or partially unknown information about the environment, except for knowing the starting point and the target point. The traditional path planning algorithms are the artificial potential field method, genetic searching approach, genetic algorithm, etc. Recently, an improved path initialization method is proposed to address the fact that the path initialization process can affect the performance of genetic algorithms, and simulation experiments show that the method can obtain high-quality paths in a shorter period of time [1]. The reference [2] addressed the shortcomings of the artificial potential field method in path planning, which is prone to fall into local optimum, and improved the repulsive function and combined it with a fuzzy inference strategy to better achieve multi-robot path planning. The reference [3] combined neural networks with hierarchical reinforcement learning, first using neural networks to perceive the environment and perform feature extraction, making them environmentally adaptive to represent action functions, and then using hierarchical reinforcement learning to map the current state of these actions, satisfying the needs of mobile robots. In reference [4], an incremental training approach for path planning was proposed to overcome disadvantages of low efficiency, poor
convergence, and weak generalization ability of reinforcement learning algorithms developed in a 3D path simulation environment, and better results were achieved. Although the traditional method has achieved better results, it is impractical to solve the mobile robot path planning problem by the traditional method alone. For example, the artificial potential field method has the problem of unreachable path, and the neural network algorithm and graphical searching approach have the problem of large amount of calculation. In recent years, swarm intelligence algorithm has been favored by scholars and achieved better results due to its better optimization ability. The reference [5] applied the quantum evolutionary algorithm (QEA) to robot path planning and compared it with the genetic algorithm, and the simulation experiments verified the effectiveness of QEA. The reference [6] studied the robot path planning problem under static obstacle space based on the dragonfly algorithm and achieved better results. Although the swarm intelligence algorithm has a good performance, it has problems such as slow convergence speed and easy to fall into local optimal solutions. The reference [23] applied the slime mould optimization algorithm (SMOA) to robot planning and the simulation experiments verified the effectiveness of SMOA. For this reason, some scholars have improved the swarm intelligence algorithm from different perspectives. In reference [7], an adaptive firefly algorithm was proposed based on the adaptive strategy, which improves the problem that the traditional firefly algorithm tends to fall into local optimum. The reference [8] proposed the improved artificial fish swarm algorithm (IAFSA), which improved the update formula based on the traditional ASFA and introduced the backward learning strategy to improve the quality of the initial solution and increase the convergence speed of the algorithm. The reference [9] improved the pheromone update strategy of the ant colony algorithm and combined it with the geometric path method to obtain an improved algorithm with faster convergence and better performance. The reference [10] combined the particle swarm algorithm with the gravitational search algorithm to balance the global and local search capabilities of particles. The reference [10] combined chaos based particle swarm algorithm with the ant colony optimization algorithm Comparative simulation experiments show that the chaos-based particle swarm optimization-ant colony optimization has a rapid search speed and can obtain solutions with similar qualities. These improved algorithms have faster convergence, higher accuracy and better performance compared to the standard algorithm for path planning applications. However, the research space is limited, so it is necessary to explore new algorithms and apply them to robot path planning. For example MFO algorithm [11] firefly algorithm [12], cuckoo search algorithm [13], bacterial foraging algorithm [14], gray wolf optimization algorithm [15]. Among the above algorithms, the MFO algorithm has received special attention due to its advantages such as simple search mechanism and few adjustment parameters, which was proposed by S. Mirjalili in 2015. Unfortunately, there are disadvantages such as slow convergence speed in the late iteration of the algorithm and easy to fall into local optimum. The reference [16] combined an improved differential evolution strategy to increase the diversity of the population and improve the performance of the algorithm. The reference [17] hybrid simulated annealing (SA) algorithm, relying on the mechanism of SA algorithm to accept non-optimal solutions, helps MFO algorithm to escape from local optimum. The reference [18] hybrid teaching and learning optimization algorithm improved the algorithm exploration and development capabilities. The reference [21] used opposition-based learning strategy to improve the MFO algorithm and verifies the effectiveness of the improved algorithm in benchmark function performance tests. The reference [22] improved the MFO algorithm update formulation and then uses the mutation operator to escape the local minimum. The performance of the improved algorithm is verified in structural optimization experiments. In order to better apply the MFO algorithm to mobile robot path planning, this paper proposed an improved moth-flame optimization algorithm. Firstly, the updated formulation of the MFO algorithm is improved by referring to the hunting mechanism and attack mechanism of the SHO algorithm [19]. Then the QOBL is introduced to increase the diversity of the population and help the algorithm jump out of the local optimum in the late iteration.

The main contributions of this paper are summarized as follows:

1. The moth position update mechanism is improved by introducing the concept of historical optimal flame average by referring to the update formula of SHO algorithm. This mechanism improves the overall performance of the algorithm.

2. The QOBL strategy is used to update the moth positions during the iterative process to further improve the overall performance of the algorithm.

3. The IMFO algorithm is capable to achieve results (feasible paths) considering distance (minimum path length), safety (avoiding collisions) and smoothness.

This paper is organized as follows. Section II presents the details of the moth-flame optimization algorithm, the spotted hyena optimization algorithm, the quasi-opposition-based learning. Section III of the IMFO algorithm is discussed in detail. Section IV selects three different types of benchmark functions to evaluate IMFO algorithm performance. Section V solves the mobile robot path planning problem with IMFO algorithm. Section VI presents the conclusions of this study.

II. RELATED WORKS

A. MOTH-FLAME OPTIMIZATION ALGORITHM

The MFO algorithm is inspired by the special navigation mechanism of moths at night. In this mechanism, the moths fly at an angle to the moon. However, this flight mechanism is only effective when the moth is far away from the moon. When an artificial light source is present, the moth is attracted to the artificial light source and moves in a spiral around it.
Inspired by this natural phenomenon, the process of a moth flying in a spiral around a flame is abstracted as a process of finding an optimal solution. The moth is assumed to be a candidate solution to the solution problem, and the variable to be solved is the position of the moth in space. Thus, by changing its position vector, the moth can fly in one, two, three, or even higher dimensions. The population position matrix and the fitness function matrix of the moths are denoted by M and OM, respectively.

The flame matrix in the MFO algorithm is similar in structure to the moth population matrix in that both represent solutions to the problem. The difference between the two is that the flame is used to store the elite solutions of the moth population. That is, the MFO algorithm sorts the moth populations according to the fitness value size after each iteration, and then stores the sorted moth populations into the flame matrix. The flame position matrix and the flame fitness value matrix are denoted by F and OF, respectively. For swarm intelligence algorithms, the position update mechanism is a central part of the algorithm. In the MFO algorithm individual moths keep perturbing in a spiral flight around the flame as the number of iterations of the algorithm increases until the optimal solution is found. The mathematical description is divided into two parts: flame-tending behavior and flame-abandoning behavior.

1) FLAME-TENDING BEHAVIOR
In nature moths are flame-tending and moths move towards the closest flame according to a special navigation mechanism. This means that in the MFO algorithm, the moth updates its position only according to the flame that corresponds to it. The update mechanism is that the moth spirals around the corresponding flame and swims around the flame along the spiral flight path. The update formula is shown in (1):

\[ S(M_i, F_j) = D_{i,j} \cdot e^{b} \cdot \cos(2\pi t) + F_j \] (1)

where \( D_{i,j} \) indicates the distance of the \( i \)th moth for the \( j \)th flame, \( b \) is a constant for defining the shape of the logarithmic spiral, and \( t \) is a random number in \([-1, 1]\) which is used to determine the distance of the next position of the moth from the flame, when \( t=−1 \) the moth is closest to the flame, \( t=1 \) the moth is farthest from the flame. When there are enough moths and flames, the moths are able to search the vast majority of the solution space, thus ensuring the exploitation capability of the algorithm.

2) FLAME-ABANDONING BEHAVIOR
The number of flames in the MFO algorithm decreases as the number of iterations increases in the process of finding the optimal solution, enabling the moth to fully search the neighborhood space of the more optimal solution and ensuring the exploitation capability of the algorithm. The update formula is shown in (2):

\[ F_N = \text{round}(K - l \ast \frac{K - 1}{L}) \] (2)

where \( l \) is the current number of the iteration, \( K \) is the maximum number of flames, and \( L \) indicates the maximum number of iterations.

B. SPOTTED HYENA OPTIMIZATION ALGORITHM
The SHO algorithm is derived from the hunting and foraging mechanism of the African spotted hyena population, which mainly includes the process of searching, encircling, hunting, and attacking prey.

1) ENCIRCLING PREY
Spotted hyenas first rely on vision to determine the location of prey, then sort the population according to the proximity of prey, and set the spotted hyena closest to the prey as the current optimal position. The rest of the individuals in the population are updated with reference to the optimal position, thus surrounding the prey. The specific formulas refer to (1)-(5) in the literature [19].

2) HUNTING
Spotted hyenas rely on a trusted population network and the ability to locate prey for hunting, and the mechanism is specifically updated with (3)-(6).

\[ D_h = B \cdot P_h - P(k) \] (3)
\[ P(k) = P_h - E \cdot D_h \] (4)
\[ C_h = P_k + P_k + \cdots + P_{k+N} \] (5)

where \( P_h \) is the position of first best spotted hyena, \( D_h \) the distance between the first best spotted hyena and other spotted hyenas, \( P_k \) the position of other spotted hyenas, \( C_h \) a group of \( N \) number of optimal solutions and \( N \) the number of spotted hyenas which is calculated as follows:

\[ N = \text{Count}_{nos}(P_h, P_{h+1}, \ldots, (P_h + M)) \] (6)

where \( M \) is a random variable in \([0.5, 1]\), \( nos \) the number of solutions.

3) ATTACKING PREY
After completing the encirclement of the prey, the spotted hyena will wait for an opportunity to attack the prey. When the convergence factor \( E \) is less than 1 and greater than -1, the spotted hyena will launch an attack on the prey. In the SHO algorithm, the mathematical formulation of the attack behavior is to find the average of the current optimal solution cluster as shown in (7).

\[ P_{i+1} = \frac{C_h}{N} \] (7)

4) SEARCH FOR PREY
Spotted hyenas search for prey based on the location of spotted hyenas in the best search group \( C_h \) to search for prey, however, when the convergence factor \( E \) is greater than 1 and less than -1, the spotted hyenas move away from each other to search and attack prey again, thus performing a global search.
C. QUASI-OPPOSITION-BASED LEARNING

The main idea of Opposition-based learning is to improve the optimization performance of the swarm intelligence algorithm by simultaneously evaluating the current solution and its inverse solution and retaining them on a merit basis. The reverse point is defined as follows:

Definition 1 (Opposite Point in the n-Dimensional Space) [20]:

Let \( x = (x_1, x_2, \ldots, x_n) \) be a point in n-dimensional space and \( x_i \in (a_i, b_i) \), then its opposite point \( \bar{x} \) can be calculated by using (8).

\[
\bar{x}_i = a_i + b_i - x_i \tag{8}
\]

Definition 2 (Quasi-Opposite Point in the n-Dimensional Space) [20]:

The quasi-opposite number is the number generated between the center of the search space and the opposite point. Mathematically, it is expressed in (9).

\[
x^q_i = \text{rand} \left( \frac{a_i + b_i}{2}, \bar{x}_i \right) \tag{9}
\]

III. PROPOSED IMFO ALGORITHM

In order to increase the global searching ability and enhance the robustness of the MFO algorithm, an improved moth flame algorithm called IMFO algorithm is proposed in this paper.

A. MOTIVATION OF IMPROMING MFO ALGORITHM

The MFO algorithm is a swarm intelligence optimization algorithm, which has a simple structure that is easy to learn and use, and has fewer parameters to adjust for a less demanding operating environment. However, the algorithm suffered from the following shortages.

1) SLOW CONVERGENCE SPEED

The spiral flight search of MFO algorithm has good local search ability and coverage search level for the target area, and it can quickly converge to the optimal solution in the early iterations, but after a certain number of iterations, the spiral flight search will only do some small updates to the current population, which makes the convergence of the algorithm slow in the late iterations.

2) LOCAL OPTIMAL

When the MFO algorithm finds the local optimal solution, other moth individuals will quickly gather near the local optimal solution, and because the MFO algorithm itself does not have a mechanism to jump out of the local optimal, it is difficult for the moths to escape after flying to the local optimal region, thus leading the algorithm to fall into the local optimal region and the phenomenon of premature convergence. In addition, the adaptive reduction mechanism of the number of flames in the MFO algorithm itself, although the algorithm is guaranteed to converge to a certain result in the iterative process, the mechanism also reduces the diversity of the algorithm population, which exacerbates the occurrence of premature convergence phenomenon to a certain extent.

B. IMFO ALGORITHM

To address the above two shortcomings, this paper conducts improvements in two ways. The concept of historical best flame average is introduced to improve the update formula of MFO algorithm, and the improved update formula enhances the population diversity as well as the ability to escape from local optimum. Equation (9) is again applied to the updated moth population to execute the proposed backward learning strategy, which further enhances the population diversity and the ability to escape from the local optimum. The concept of historical best flame mean is inspired by the SHO algorithm, and the hunting mechanism in the SHO algorithm is similar to the construction of flame matrix in the MFO algorithm, both of which construct clusters of optimal solutions, although the MFO algorithm does not effectively use the information of the flame matrix. Therefore, the updated equations after the modification are (10) and (11).

\[
S(M_i, F_j) = \text{Average}_y \cdot e^{bt} \cdot \cos(2\pi t) + F_j \tag{10}
\]

\[
\text{Average}_y = \frac{\sum_{y=1}^{f} |F_y - M_i|}{f} \tag{11}
\]

where \( F_y \) denotes the flame that is better than the flame \( F_j \) corresponding to moth \( i \), \( f \) is the number of flames that

Algorithm 1 IMFO Algorithm

1: Initialize the position of moths;
2: Initialize the iteration count \( l=1 \);
3: While \( l < L+1 \);
4: Update flame number \( F_M \) by (2);
5: \( OM = \text{FitnessFunction}(M) \)
6: if \( l == 1 \)
7: \( F = \text{sort}(M) ; \text{OF} = \text{sort}(OM) ; \)
8: else
9: \( F = \text{sort}(M(l-1), M(l)) ; \text{OF} = \text{sort}(OM(l-1), OM(l)) ; \)
10: end if
11: for \( i = 1 : n \)
12: for \( j = 1 : d \)
13: Update \( t \);
14: Update the position of each moth using formula (10);
15: Update the Quasi-Opposite position of each moth using formula (10);
16: end for
17: end for
18: \( \text{OFM} = \text{FitnessFunction}(FM) \)
19: if \( \text{OFM} < OM \)
20: \( OM = \text{OFM} ; F = FM ; \)
21: end if
22: \( l = l+1 \);
23: end while
are better than the flame corresponding to moth $i$. $Average_y$ denotes the average of the flames that are better than the flame corresponding to moth $i$, which is the historical best flame average.

### IV. EXPERIMENTAL COMPARISONS ON CLASSICAL BENCHMARK SET

In the study of optimization, it is common to test the performance of an algorithm using benchmark functions. In order to verify the effectiveness of the IMFO algorithm, three sets of benchmark test functions with different characteristics were selected from the literature [18] to test the performance of the proposed IMFO algorithm from different perspectives and compare it with the MFO algorithm, PSO algorithm, GWO algorithm, SHO algorithm and the OMFO algorithm. As shown in Tables 1 to 3, the three sets of benchmark test functions are: unimodal benchmark functions $F_1$-$F_7$, multimodal benchmark functions $F_8$-$F_{13}$, and fixed-dimension multimodal benchmark functions $F_{14}$-$F_{23}$.

#### TABLE 1. Unimodal benchmark functions.

| Function | Dim | Range   | $f_{min}$ |
|----------|-----|---------|-----------|
| $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 10  | [-100,100] | 0         |
| $F_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$ | 10  | [-10,10] | 0       |
| $F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ | 10  | [-100,100] | 0         |
| $F_4(x) = \max \{|x_i|, 1 \leq i \leq d\}$ | 10  | [-100,100] | 0         |
| $F_{i}(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | 10  | [-30,30] | 0         |
| $F_6(x) = \sum_{i=1}^{n} (|x_i + 0.5|)^2$ | 10  | [-30,30] | 0         |
| $F_7(x) = \sum_{i=1}^{n} i \cdot x_i^4 + \text{random}[0,1]$ | 10  | [-1.28,1.28] | 0        |

#### TABLE 2. Multimodal benchmark functions.

| Function | Dim | Range   | $f_{min}$ |
|----------|-----|---------|-----------|
| $F_8(x) = \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$ | 10  | [-500,500] | -418.98×10 |
| $F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10]$ | 10  | [-5.12,5.12] | 0       |
| $F_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) + 20 + e\right)$ | 10  | [-32,32] | 0       |
| $F_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | 10  | [-600,600] | 0       |
| $F_{12}(x) = \frac{\pi}{d} [10 \sin(\pi y_1) + \sum_{i=1}^{d} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1}) + 1 + (y_n - 1)^2]] + \sum_{i=1}^{n} u(x_i,10,100,4)$ | 10  | [-50,50] | 0       |
| $y_i = 1 + \frac{x_i + 1}{4}$ | 0   | [-50,50] | 0       |
| $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$ | 0   | [-50,50] | 0       |
| $F_{13}(x) = 0.1 \cdot [\sin^2(3\pi x_1) + \sum_{i=1}^{d} (x_i - 1)^2 [1 + \sin^2(1 + 3\pi x_i)]] + (x_d - 1)^2 [1 + \sin^2(2\pi x_a)] + \sum_{i=1}^{d} u(x_i,5,100,4)$ | 10  | [-50,50] | 0       |
TABLE 3. Fixed-dimension multimodal benchmark functions.

| Function | Dim | Range     | $f_{\text{min}}$ |
|----------|-----|-----------|------------------|
| $F_{14}(x)$ | 2   | [-65,65]  | 1                |
| $F_{15}(x)$ | 2   | [-5,5]    | 0.0003           |
| $F_{16}(x)$ | 2   | [-5,5]    | -1.0316          |
| $F_{17}(x)$ | 2   | [-5,5]    | 0.398            |
| $F_{18}(x)$ | 2   | [-2.2]    | 3                |
| $F_{19}(x)$ | 3   | [1,3]     | -3.86            |
| $F_{20}(x)$ | 6   | [0,1]     | -3.32            |
| $F_{21}(x)$ | 4   | [0,10]    | -10.1532         |
| $F_{22}(x)$ | 4   | [0,10]    | -10.4028         |
| $F_{23}(x)$ | 4   | [0,10]    | -10.5363         |

TABLE 4. Statistic of the algorithm in 30 runs on each unimodal benchmark functions.

| Functions | IMFO | MFO   | PSO   | GWO   | SHO   | OMFO   |
|-----------|------|-------|-------|-------|-------|--------|
| F1        | Mean | 0.000E+00 | 3.276E-13 | 1.549E-23 | 1.706E-64 | 0.000E+00 | 0.000E+00 |
| std       | 0.000E+00 | 7.454E-13 | 3.748E-23 | 6.229E-64 | 6.229E-64 | 0.000E+00 | 0.000E+00 |
| F2        | Mean | 0.000E+00 | 6.667E-01 | 7.314E-13 | 4.703E-37 | 0.000E+00 | 0.000E+00 |
| std       | 0.000E+00 | 2.537E+00 | 2.012E-12 | 6.303E-37 | 6.303E-37 | 0.000E+00 | 0.000E+00 |
| F3        | Mean | 0.000E+00 | 4.952E-02 | 1.138E-07 | 1.626E-28 | 0.000E+00 | 0.000E+00 |
| std       | 0.000E+00 | 8.497E-02 | 2.041E-07 | 5.343E-28 | 5.343E-28 | 0.000E+00 | 0.000E+00 |
| F4        | Mean | 0.000E+00 | 6.105E-01 | 1.328E-06 | 1.840E-20 | 0.000E+00 | 0.000E+00 |
| std       | 0.000E+00 | 8.119E-01 | 1.142E-06 | 2.754E-20 | 2.754E-20 | 0.000E+00 | 0.000E+00 |
| F5        | Mean | 0.000E+00 | 3.366E+03 | 4.959E+00 | 6.408E+00 | 6.408E+00 | 6.408E+00 |
| std       | 0.000E+00 | 1.639E+04 | 1.783E+00 | 6.838E+00 | 9.617E-02 | 9.617E-02 | 9.617E-02 |
| F6        | Mean | 0.000E+00 | 1.616E-13 | 5.480E-24 | 3.099E-06 | 1.030E-01 | 4.390E-02 |
| std       | 0.000E+00 | 3.025E-13 | 1.103E-23 | 1.049E-06 | 3.926E-01 | 7.050E-03 | 7.050E-03 |
| F7        | Mean | 4.128E-05 | 7.936E-03 | 6.125E-03 | 5.077E-04 | 6.331E-05 | 6.511E-05 |
| std       | 4.814E-05 | 8.658E-03 | 3.751E-03 | 3.856E-04 | 6.403E-05 | 6.023E-05 | 6.023E-05 |

optimal solution and no local optimal solution, which is suitable for verifying the development capability and convergence speed of the algorithm. The multimodal benchmark function has multiple local optimal solutions, and the number of local optimal solutions grows exponentially with the number of dimensions, which is suitable for verifying the exploration ability of the algorithm. Heuristic algorithms are stochastic optimization techniques, and therefore they have to be run more than 10 times for generating meaningful statistical results. In order to better verify the performance of the IMFO algorithm, the number of populations is set to 40, the maximum number of iterations is set to 500, and each algorithm is run 30 times independently on each test function in this paper. The performance of the four algorithms is compared according to the mean and standard deviation of the optimal solution of the benchmark function, and the statistical results are shown in Table 4 - Table 5. A comparison of the convergence curves of the four functions is shown in Fig. 1.
In this paper, Mean and Std represent the mean fitness value and standard deviation, respectively. Experimental results are listed in Tables 4, 5, and 6. The best results are denoted in bold characters.

According to the results of Table 4, IMFO algorithm is able to provide very competitive results. From this table and the convergence curves of the unimodal benchmark function in Fig. 1, it can be seen that the IMFO algorithm outperforms other algorithms in terms of overall performance. Therefore, the proposed algorithm has high performance to find the global minimum of unimodal benchmark functions.

From Table 5, Table 6 and the convergence curves of multimodal benchmark functions and fixed-dimension multimodal benchmark functions in Fig. 1, it can be seen that the performance of the algorithm proposed in this paper is better than other algorithms in most cases. These show that IMFO algorithm has a strong sense of stability and robust comparing from other algorithms. Overall, the results in Tables 4-6 and Fig. 1 show that the proposed method is effective in not only optimizing unimodal and multimodal functions but also optimizing fixed-dimension multimodal functions.

V. PATH PLANNING OF MOBILE ROBOT BASED ON IMFO ALGORITHM

A. FITNESS VALUE FUNCTION

In the mobile robot path planning problem, a feasible path represents a solution of the problem. In order to ensure the validity of the path, the coordinate points in the path need to

| Functions | IMFO | MFO | PSO | GWO | SHO | OMFO |
|-----------|------|-----|-----|-----|-----|------|
| **F8**    | Mean | -4.190E+03 | -3.227E+03 | -1.990E+03 | -2.736E+03 | -1.456E+03 | -2.532E+03 |
|           | Std  | 2.775E-12  | 3.389E+02  | 2.251E+02  | 4.133E+02  | 2.478E+02  | 3.281E+02  |

| **F9**    | Mean | 2.000E+00  | 2.187E+01  | 4.350E+00  | 4.132E-01  | 0.000E+00  | 0.000E+00  |
|           | Std  | 0.000E+00  | 1.076E+01  | 1.777E+00  | 1.109E+00  | 0.000E+00  | 0.000E+00  |

| **F10**   | Mean | 8.882E-16  | 7.029E-01  | 2.325E-12  | 6.691E-15  | 8.882E-16  | 8.882E-16  |
|           | Std  | 0.000E+00  | 3.638E+01  | 3.107E-12  | 1.741E-15  | 0.000E+00  | 0.000E+00  |

| **F11**   | Mean | 4.712E-32  | 4.092E-01  | 4.092E-01  | 2.318E-02  | 0.000E+00  | 0.000E+00  |
|           | Std  | 1.670E-47  | 4.215E-01  | 1.173E-24  | 1.243E-02  | 7.186E-02  | 1.819E-03  |

| **F12**   | Mean | 1.350E-32  | 4.029E-03  | 2.139E-24  | 3.914E-06  | 8.822E-01  | 3.926E-02  |
|           | Std  | 5.567E-48  | 5.385E-03  | 4.524E-24  | 1.715E-06  | 1.751E-05  | 9.884E-03  |

In the mobile robot path planning problem, a feasible path represents a solution of the problem. In order to ensure the validity of the path, the coordinate points in the path need to
FIGURE 1. The convergence curves for part of the benchmark functions.
satisfy the conditions of no collision with obstacles, the shortest distance between coordinate points, and smoothness of the path. Therefore, in this paper, the path length, smoothness, and safety are used as three components to measure the path quality.

1) SHORTEST PATH LENGTH
In the field of mobile robots, shortest means the shortest total path length of the robot from the start Position to the goal Position. Therefore, the shortest path from the starting point to the end point through n path points is found by (11).

\[ f_1 = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \]  

(12)

where \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\) denote the coordinates of the current point and the next adjacent point, respectively.

2) PATH SMOOTHNESS
As one of the important criteria to measure the quality of the path, adding path smoothing will increase the fitness value of the path as the number of turns increases, thus reducing the probability of a more tortuous path being selected, and the path smoothing formula is shown in (13).

\[ f_2 = \frac{n_1 \pi}{4} + \frac{n_2 \pi}{2} \]  

(13)

where \(n_1\) indicates the number of turning points in the path with 45 degrees; and \(n_2\) indicates the number of turning points in the path with 90 degrees.

3) SAFETY
The effect is to improve the safety of the path. When a path point falls into an obstacle, it indicates that the path is unsafe, so the path is discarded and a new path is initialized.

In summary, the formula of the fitness function used in this paper is shown in (14).

\[ f = f_1 + f_2 \]  

(14)

B. PATH PLANNING STEPS FOR MOBILE ROBOTS BASED ON IMFO ALGORITHM
Step1: Initialize the algorithm parameters, randomly generate \(N\) feasible paths \(X = (x_1, x_2, \ldots, x_i), i = (1, 2, \ldots, N)\), Maximum iteration number \(T\);

Step2: Calculate the fitness value \(F\) of individual \(i\) according to the fitness function equation (14) and rank them in ascending order;

Step3: Update the number of flames, update the position of individual moths according to Equation (10), and calculate the quasi-opposite point \(OX\) of individual moths according to (9);

Step4: Calculate the fitness value \(OF\) of the quasi-opposite path according to (14), if the fitness value \(OF < F\) of the quasi-opposite path, \(X = OX\);

Step5: Determine whether the termination condition is satisfied, if so, terminate the program and output the optimal solution, otherwise jump to Step2 to continue the execution.

C. EXPERIMENTAL RESULTS AND DISCUSSION
In order to verify the feasibility and effectiveness of the IMFO algorithm for mobile robot path planning, this paper compares it with the PSO algorithm, the MFO algorithm, the DA algorithm and the GWO algorithm in three environments. The maximum number of iterations for the four algorithms is set to 300 and the population size is set to 30. The starting point and end point are randomly generated in each environment, and each algorithm is run 100 times independently in each group of environments.

The simulations were conducted in Matlab 2019b environment on a personal computer which has an Intel Core i7 9750h 2.6GHz processor and 16 GB RAM.

The starting and ending points of the first environment are (43,49) and (35,2), the starting and ending points of the second are (45,12) and (24,47), and the starting and ending points of the third are (48,5) and (2,45). The results of the three simulation environments are shown in Table 7 to Table 9 and Fig.3 to Fig.7.

Figs. 3 to 7 and Tables 8 to 9 demonstrate that the IMFO algorithm in this paper can be applied to the mobile robot path planning problem as well as the PSO algorithm, DA algorithm and GWO algorithm. It can be seen from Table 7 to
Table 9 that in terms of the number of iterations of convergence, although in some cases the IMFO algorithm has more iterations of convergence than the MFO algorithm, the DA algorithm and the GWO algorithm. However, the convergence accuracy is higher than that of the MFO algorithm, the DA algorithm and the GWO algorithm. According to Figure 6 it can be seen that in scenario 1, the IMFO algorithm improves 1.42%, 2.08%, 3.94% and 1.61% over the MFO algorithm, the PSO algorithm, the DA algorithm and the GWO algorithm, respectively. In scenario 2, the IMFO algorithm improves 0.93%, 0.81%, 2.42% and 1.12% over the MFO algorithm, the PSO algorithm, the DA algorithm and the GWO algorithm, respectively. In scenario 3, the IMFO algorithm improves 0.56%, 0.68%, 1.83% and 1.02% over the MFO algorithm, PSO algorithm, the DA algorithm and the GWO algorithm, respectively. It can be seen from Fig. 7 that the average path length generated by IMFO algorithm is smaller than the MFO algorithm, the PSO algorithm, the DA algorithm and the GWO algorithm, respectively.
algorithm and the GWO algorithm in terms of path standard deviation, indicating that the IMFO algorithm is relatively stable in the path planning problem. To check whether the algorithms are statistically different or not, the t-test is performed. The level 0.05 of significance is considered. The results indicate that IMFO algorithm outperforms in all test environments because the p-values are far smaller than the level 0.05 of significance.

VI. CONCLUSION

In this paper, an IMFO algorithm has been proposed to address the shortcomings of MFO algorithm such as slow late convergence and easy to fall into local optimum. First, the update formula of MFO algorithm was improved by introducing the concept of historical optimal flame mean with reference to SHO algorithm, so that MFO algorithm can better utilize the information in the flame matrix. Then, the hybrid backward learning strategy proposed increases the population diversity. The performance test of 23 benchmark functions verifies that the IMFO algorithm has improved in stability and convergence accuracy compared with the MFO algorithm. Finally, the IMFO algorithm is applied to path planning, and the effectiveness of the IMFO algorithm is verified by simulation experiments. To the best of our knowledge, there are few papers that apply the MFO algorithm to mobile robot path planning. However, through the above digital experiments, it is obvious that there are still a few of imperfections in this paper. For example, how to apply the IMFO algorithm to dynamic path planning, and how to further improve the accuracy and stability of the IMFO algorithm. In the next step, we will further refine the proposed IMFO algorithm and compare the refined algorithm with more swarm intelligence algorithms.

REFERENCES

[1] J. Lee and D.-W. Kim, “An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph,” Inf. Sci., vol. 332, no. 1, pp. 1–18, Mar. 2016.
[2] T. Zhao, H. Li, and S. Dian, “Multi-robot path planning based on improved artificial potential field and fuzzy inference system,” J. Intell. Fuzzy Syst., vol. 39, no. 5, pp. 7621–7637, Nov. 2020.
[3] J. Yu, Y. Su, and Y. Liao, “The path planning of mobile robot by neural networks and hierarchical reinforcement learning,” Frontiers Neuroinformatics, vol. 14, p. 63, Oct. 2020, doi: 10.3389/fnbot.2020.00063.
[4] J. Gao, W. Ye, J. Guo, and Z. Li, “Deep reinforcement learning for indoor mobile robot path planning,” Sensors, vol. 20, no. 19, p. 5493, Sep. 2020, doi: 10.3390/s20195493.
[5] L. Gao, R. Liu, F. Wang, W. Wu, B. Bai, S. Yang, and L. Yao, “An advanced quantum optimization algorithm for robot path planning,” J. Circuits, Syst. Comput., vol. 29, no. 8, Jun. 2020, Art. no. 2050122, doi: 10.1142/S0218126620501224.
[6] S. Muthukumaran and R. Sivaramakrishnan, “Optimal path planning for an autonomous mobile robot using dragonfly algorithm,” Int. J. Simul. Model., vol. 18, no. 3, pp. 397–407, Sep. 2019.
[7] G. Xu, T.-W. Zhang, Q. Lai, J. Pan, B. Fu, and X. Zhao, “A new path planning method of mobile robot based on adaptive dynamic firefly algorithm,” Mod. Phys. Lett. B, vol. 34, no. 29, Oct. 2020, Art. no. 2050322, doi: 10.1142/S0217984920503224.
[8] Y. Zhang, G. Guan, and X. Pu, “The robot path planning based on improved artificial fish swarm algorithm,” Math. Problems Eng., vol. 2016, pp. 1–11, Aug. 2016, doi: 10.1155/2016/3297585.
[9] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, “An improved ant colony algorithm for robot path planning,” Soft Comput., vol. 21, no. 19, pp. 5829–5839, Oct. 2017.
[10] P. K. Das, H. S. Behera, and B. K. Panigrahi, “A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning,” Swarm Evol. Comput., vol. 28, pp. 14–28, Jun. 2016.
References

[11] S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowl.-Based Syst., vol. 89, pp. 228–249, Nov. 2015.

[12] S. L. Tilahun and J. M. T. Ngnotchouye, “Firefly algorithm for discrete optimization problems: A survey,” KSCE J. Civil Eng., vol. 21, no. 2, pp. 535–545, Feb. 2017.

[13] V. Bhargava, S. E. K. Fateen, and A. Bonilla-Petriciolet, “Cuckoo search: A new nature-inspired optimization method for phase equilibrium calculations,” Fluid Phase Equilibria, vol. 337, pp. 191–200, Jan. 2013.

[14] M. Wan, L. Li, J. Xiao, C. Wang, and Y. Yang, “Data clustering using bacterial foraging optimization,” J. Intell. Inf. Syst., vol. 38, no. 2, pp. 321–341, Apr. 2012.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014.

[16] S. Sapre and S. Mini, “A differential moth flame optimization algorithm for mobile sink trajectory,” Peer-Peer Netw. Appl., vol. 14, no. 1, pp. 44–57, Jan. 2021.

[17] G. I. Sayed and A. E. Hassanien, “A hybrid SA-MFO algorithm for function optimization and engineering design problems,” Complex Intell. Syst., vol. 4, no. 3, pp. 95–212, Oct. 2018.

[18] K. N. Reddy and P. Bojja, “A new hybrid optimization method combining moth–flame optimization and teaching–learning-based optimization algorithms for visual tracking,” Soft Comput., vol. 24, no. 24, pp. 18321–18347, Dec. 2020.

[19] G. Dhiman and V. Kumara, “Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications,” Adv. Eng. Softw., vol. 114, pp. 48–70, Dec. 2017.

[20] D. Guha, P. Roy, and S. Banerjee, “Quasi-oppositional backtracking search algorithm to solve load frequency control problem of interconnected power system,” Iranian J. Sci. Technol., Trans. Electr. Eng., vol. 44, no. 2, pp. 781–804, Jun. 2020.

[21] W. Apinantanon and S. Khamron, “OMFO: A new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems,” in Proc. ICIT, Bangkok, Thailand, 2017, pp. 22–31.

[22] S. Gholizadeh, H. Davoudi, and F. Fattahi, “Design of steel frames by an enhanced moth–flame optimization algorithm,” Steel Compos. Struct., vol. 24, no. 1, pp. 129–140, May 2017.

[23] A. Divya and P. S. Bharti, “Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots,” Appl. Soft. Comput., vol. 107, Aug. 2021, Art. no. 107372, doi: 10.1016/j.asoc.2021.107372.

[24] P. Chen, Q. Li, C. Zhang, J. Cui, and H. Zhou, “Hybrid chaos-based particle swarm optimization-ant colony optimization algorithm with asynchronous pheromone updating strategy for path planning of landfill inspection robots,” Int. J. Adv. Robot. Syst., vol. 16, no. 4, pp. 1–11, Jul. 2019, doi: 10.1177/1729881419859083.

Xuefeng Dai received the B.S. degree in electrical automation from Liaoning Technical University, Liaoning, China, in 1985, and the M.S. degree in automatic control theory and applications and the Ph.D. degree in control theory and control engineering from Harbin Engineering University, Heilongjiang, China, in 1992 and 2001, respectively. From 1992 to 1997, he was a Lecturer with the Department of Automation, Qiqihar University, Heilongjiang. From 2001 to 2004, he was a Postdoctoral Research Associate with the Institute of Robotics, Harbin Institute of Technology, Heilongjiang. His research interests include supervisory control of discrete event systems, neural networks, and coordinated control of multi-robot systems.

Yang Wei was born in Beijing, China, in 1997. He received the B.S. degree in automation from Qiqihar University, Qiqihar, China, in 2019, where he is currently pursuing the M.S. degree in control theory and control engineering. His research interest includes multiple-robot path planning.