Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 4203538, 14 pages
https://doi.org/10.1155/2019/4203538

Research Article

Air Target Threat Assessment Based on Improved Moth Flame Optimization-Gray Neural Network Model

Longfei Yue,[1] Rennong Yang,[1] Jialiang Zuo ,[1] Hao Luo,[2] and Qiuliang Li[2]

[1]*Air Traffic Control and Navigation College, Air Force Engineering University, Xi'an 710051, China*
[2]*Graduate College, Air Force Engineering University, Xi'an 710038, China*

Correspondence should be addressed to Jialiang Zuo; ylf1172969690@163.com

Received 10 June 2019; Revised 16 August 2019; Accepted 3 September 2019; Published 10 October 2019

Academic Editor: Oliver Schütze

Air target threat assessment is a key issue in air defense operations. Aiming at the shortcomings of traditional threat assessment methods, such as one-sided, subjective, and low-accuracy, a new method of air target threat assessment based on gray neural network model (GNNM) optimized by improved moth flame optimization (IMFO) algorithm is proposed. The model fully combines with excellent optimization performance of IMFO with powerful learning performance of GNNM. Finally, the model is trained and evaluated using the target threat database data. The simulation results show that compared with the GNNM model and the MFO-GNNM model, the proposed model has a mean square error of only 0.0012 when conducting threat assessment, which has higher accuracy and evaluates 25 groups of targets in 10 milliseconds, which meets real-time requirements. Therefore, the model can be effectively used for air target threat assessment.

1. Introduction

Air target threat assessment refers to comprehensively considering various factors affecting the target threat value, establishing a reasonable indicator system, and quantifying it, and then establishing a threat assessment model to evaluate the target threat value. As battlefield environment becomes more and more complex and incoming targets are characterized by high speed, high maneuverability, stealth, anti-jamming, and remote precision guidance, the pressure on air defense system is increasing. Therefore, establishing a reasonable threat assessment model and conducting a rapid and accurate threat assessment of incoming targets is a prerequisite for target allocation in air defense operations and an important support for efficient command and control.

At present, a lot of research studies have been carried out on air target threat assessment at home and abroad. Two methods are mainly used. One is the method of reasoning and the other is the method of machine learning.

The method of reasoning is to analyze the relationship between index values and threat values and combine expert experience and prior probability to infer the threat values of different targets. This method mainly includes the Bayesian network [1–3] and intuitionistic fuzzy logic [4–6]. The Bayesian network method can intuitively express the relationship between indicators and threats, but the determination of prior probability depends too much on expert experience. Therefore, subjectivity is too strong. An intuitionistic fuzzy logic method can better deal with qualitative indicators, but the rules of reasoning are complicated and the coupling relationship between the indicators is not considered enough.

The machine learning method is to establish a threat assessment index system, quantify each index, and then learn the nonlinear relationship between the index values and the threat values through machine learning. This method mainly includes neural network [7, 8] and support vector machine [9]. Neural network has strong nonlinear fitting ability, but it needs large sample training. Support vector machine has advantages for small sample prediction, but it has disadvantages of unreasonable index selection and low evaluation accuracy.

Therefore, how to choose a reasonable index system and scientific quantitative methods, while overcoming the strong

subjective deficiencies of traditional methods and improving the accuracy of assessment have always been the focus of research on threat assessment.

GNNM is a predictive model that combines the advantages of small sample prediction with gray system theory and the advantages of self-learning of neural network. Among them, gray system theory [10] is a systematic scientific theory proposed by Prof. Deng to predict the eigenvalues of uncertain system behaviors. Gray system modeling requires few samples, and the requirement for distribution law of data is not high. Neural network is a mathematical model mainly used to learn the nonlinear function relationship between input data and output data, with the advantages of parallel computing and self-learning, self-organization, and robustness. Combining gray theory with neural network to establish a gray neural network model can make full use of the advantages of both and improve the prediction accuracy. The performance of GNNM depends mainly on the choice of initial weights, so how to choose better initial weight parameters is particularly important.

Moth flame optimization (MFO) algorithm [11] is a new intelligent optimization algorithm [12–14] proposed by Prof. Mirjalili in 2015. It has the advantages of less adjustment parameters, faster convergence, and simple implementation. Hence, it is often used in engineering optimization problems [15, 16]. However, MFO also has some shortcomings in the process of optimization. For example, the selection of current optimal solution in each iteration leads to trap in local optima and premature convergence, and the global optimization performance is poor due to low population diversity. Therefore, after improving the optimization performance of MFO, the optimal initial weight parameter is obtained by IMFO and the learning performance of GNNM is improved.

Therefore, this paper combines the actual situation of air target threat assessment, selects a reasonable indicator system, establishes a threat assessment framework, and scientifically quantifies each indicator. Then, this paper improves the global optimization performance of the original MFO by introducing Tent chaos, Lévy flight, and Metropolis criterion. Then this paper combined IMFO with GNNM, a threat assessment model based on IMFO-GNNM is established. Finally, the proposed model is evaluated from accuracy and real-time performance. The results show that the proposed model has higher accuracy and better real-time performance in threat assessment. Therefore, this study has certain advantages and important practical significance.

2. Threat Assessment Problem Modeling

Air target threat assessment needs to analyze the air target and our asset to obtain the threat value of air target, thus providing a basis for our command and control. Due to the particularity of combat problem, on one hand, the accuracy of threat assessment should be considered, and on the other hand, the real-time performance of threat assessment should be considered. Therefore, it is crucial to choose a reasonable and accurate threat assessment indicator system and an efficient threat assessment method. Since air target threat assessment is not only related to the threat capability of target, but also related to the spatial location of target and the value of our asset. This paper considers threat level, threat capability, and threat extent to establish a target threat assessment framework, and it scientifically and reasonably quantifies each threat attribute. Assessment framework is shown in Figure 1.

2.1. Threat Level. Since the value of asset on the battlefield is different, the threat value of air target is related to the value of our asset. The higher the value of our asset, the higher the target threat level and the higher the threat value. The value of our asset is generally related to the political, economic, and military value of the asset. Therefore, refer to the importance of asset evaluation standard [17], and a threat level evaluation function is established, as shown in the following equation:

$$C_i = \frac{\ln\left(\prod_{j=1}^{s} C_{ij}\right) + 1}{\ln\left(10^s\right) + 1} = \frac{\sum_{j=1}^{s} \ln\left(C_{ij}\right) + 1}{\ln\left(10^s\right) + 1}. \quad (1)$$

In the formula, C_{ij} represents the value of the jth evaluation factor of our asset i and its value range $C_{ij} \in [1, 10]$, and s is the number of evaluation factors. The value of the asset is usually given by the superior command.

2.2. Threat Capability. The target threat capability includes the penetration ability of the target to our asset and the degree of damage caused by the successful penetration. Among them, the penetration ability is related to the target type and interference ability and the degree of damage is determined by the target damage ability.

2.2.1. Target Type. There are many types of air targets. Different types of targets are different in size, radar cross-section (RCS), and threat values. Air targets are mainly classified into aircraft and guided weapons according to their size. Aircraft mainly includes bombers (BA), fighter bombers (FB), and armed helicopters (AH). Guided weapons mainly include precision guided bombs (PGB), air-to-ground missiles (AGM), cruise missiles (CM), tactical ballistic missiles (TBM), and antiradiation missiles (ARM). Therefore, the domain expert group quantifies the RCS size of different types of targets, and the results are shown in Table 1.

2.2.2. Damage Ability. The target damage ability is determined by target damage probability and our survival probability. The damage probability is obtained through the intelligence data, and the survival probability is obtained by analyzing the factors such as the invulnerability and protection of our asset.

(a) When the air target is an aircraft type, the probability of damage is related to the type, quantity, and single-damage probability of the weapon carried by the aircraft. Therefore, its damage probability is the joint damage probability of different types of weapons, and its damage ability is quantified according to the following equation:

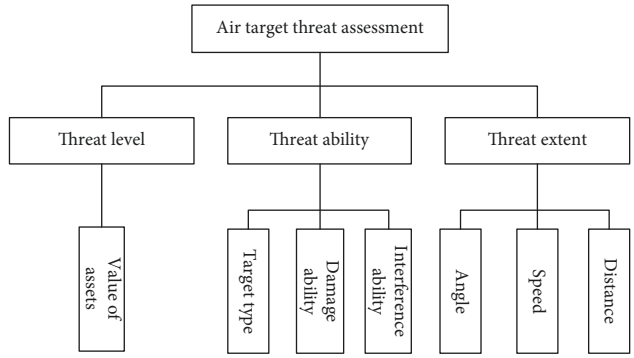

Figure 1: Target threat assessment framework.

Table 1: Target type quantification.

Type	BA	FB	AH	PGB	AGM	CM	TBM	ARM
Threat	0.1	0.3	0.5	0.9	0.7	0.7	0.7	0.7

$$T_{d_{K_iT_j}} = \begin{cases} \dfrac{\left(1-\left(1-p^i_{\mathrm{PGB}}\right)^{N^{BA}_{\mathrm{PGB}}}\right)\left(1-\left(1-p^i_{\mathrm{AGM}}\right)^{N^{\mathrm{BA}}_{\mathrm{AGM}}}\right)\left(1-\left(1-p^i_{\mathrm{CM}}\right)^{N^{\mathrm{BA}}_{\mathrm{CM}}}\right)}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{BA}, \\ \dfrac{\left(1-\left(1-p^i_{\mathrm{PGB}}\right)^{N^{\mathrm{FB}}_{\mathrm{PGB}}}\right)\left(1-\left(1-p^i_{\mathrm{AGM}}\right)^{N^{\mathrm{FB}}_{\mathrm{AGM}}}\right)\left(1-\left(1-p^i_{\mathrm{ARM}}\right)^{N^{\mathrm{FB}}_{\mathrm{ARM}}}\right)}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{FB}, \\ \dfrac{\left(1-\left(1-p^i_{\mathrm{PGB}}\right)^{N^{\mathrm{AH}}_{\mathrm{PGB}}}\right)\left(1-\left(1-p^i_{\mathrm{AGM}}\right)^{N^{\mathrm{AH}}_{\mathrm{AGM}}}\right)}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{AH}. \end{cases} \tag{2}$$

In the formula, $d_{K_iT_j}$ represents the damage ability of the target j to the asset i, p^i_{survival} represents the survival probability of the asset i, p^i_{PGB} represents the probability of damage of the PGB to the asset i, and $N^{\mathrm{BA}}_{\mathrm{PGB}}$ represents the number of PGB mounted by the BA.

(b) When the air target is a guided weapon, its damage ability is quantified by

$$T_{d_{K_iT_j}} = \begin{cases} \dfrac{p^i_{\mathrm{PGB}}}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{PGB}, \\ \dfrac{p^i_{\mathrm{AGM}}}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{AGM}, \\ \dfrac{p^i_{\mathrm{TBM}}}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{TBM}, \\ \dfrac{p^i_{\mathrm{ARM}}}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{ARM}, \\ \dfrac{p^i_{\mathrm{PGB}}}{p^i_{\mathrm{survival}}}, & T_j \in \mathrm{PGB}. \end{cases} \tag{3}$$

2.2.3. Interference Ability. According to the electronic interference pod and interference means, the target interference capability is classified into five levels: super strong, strong, medium, weak, and no. According to Miller's quantitative theory [18], it is quantized to 0.9, 0.7, 0.5, 0.3, and 0.1.

2.3. Threat Extent. The target threat extent mainly measures the threat of the target to our asset from the spatial situation. As shown in Figure 2, T represents the incoming target, K represents our asset, D represents the target distance, and θ represents the airway angle, that is, the angle between the projection of the target velocity in the horizontal plane and the connection between the target and the asset.

It can be seen from Figure 2 that the closer the target distance, the higher the target speed and the higher the target threat value. In addition, considering killing boundary of the target, when the distance is fixed, the smaller the airway angle, the shorter the airway short cut, the more likely our asset is to fall into the target kill zone, and the higher the target threat value. Therefore, the target threat extent is measured by the three dimensions of the airway angle, the target speed, and the target distance.

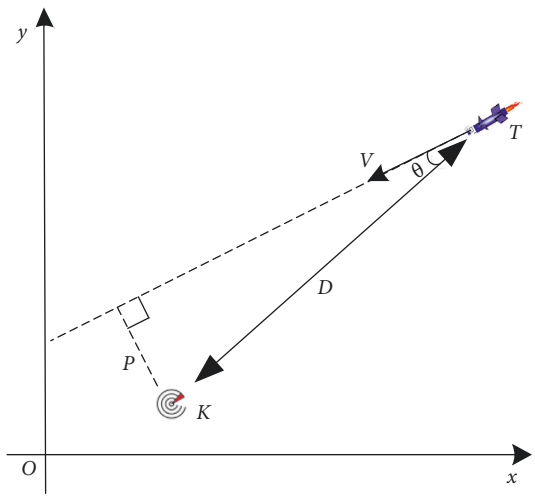

Figure 2: Schematic diagram of relationship between target and asset.

2.3.1. Airway Angle Threat. The airway angle usually varies in the range of [0°, 180°], but it is generally considered that when the airway angle is within [0°, 90°], the target is approaching to the asset. At this time, the larger the airway angle, the larger the airway short cut and the lesser the threat extent. Hence, the negative exponential function is used to quantify the airway angle. The formula is as follows:

$$T_\theta = \begin{cases} e^{-\theta_{K_iT_j}}, & \theta_{K_iT_j} \in [0^\circ, 90^\circ], \\ 0, & \text{else.} \end{cases} \tag{4}$$

2.3.2. Speed Threat. The higher the target speed, the shorter the response time and the higher the threat value of the target. Therefore, the speed threat is quantified by

$$T_v = 1 - e^{-0.005v}. \tag{5}$$

2.3.3. Distance Threat. The target distance is the distance between the target and the asset. The target distance directly affects the threat value of the target. The closer to our asset, the higher the threat and the farther the distance, the lower the threat value. Therefore, distance threat is quantified by

$$T_D = \frac{\max(D) - D_i}{\max(D) - \min(D)}. \tag{6}$$

3. Construction of IMFO-GNNM

3.1. GNNM. Suppose there is a sequence

$$x^{(0)} = \left(x^{(0)}(1), x^{(0)}(2), \ldots, x^{(0)}(n)\right). \tag{7}$$

We can get a new sequence by accumulating (AGO):

$$\begin{aligned} x^{(1)} &= \left(x^{(1)}(1), x^{(1)}(2), \ldots, x^{(1)}(n)\right) \\ &= \left(x^{(0)}(1), x^{(0)}(1) + x^{(0)}(2), \ldots, x^{(0)}(1) + x^{(0)}(2) \right. \\ &\quad \left. + \cdots + x^{(0)}(n)\right). \end{aligned} \tag{8}$$

It is easy to know $x^{(1)}$ is exponential growth law, so according to the idea of derivative and difference, we can construct differential equations to fit $x^{(1)}$.

For the sake of convenience, the original sequence $x^{(0)}$ is recorded as $x(t)$, the new sequence $x^{(1)}$ is recorded as $y(t)$, and the prediction result is $z(t)$; so, the differential equation expression of the gray neural network with n parameters is as follows:

$$\frac{dy_1}{dt} + ay_1 = b_1y_2 + b_2y_3 + \cdots + b_{n-1}y_n. \tag{9}$$

In the formula, the output variable is y_1, the input variable is $y_i\,(i = 2, \ldots, n)$, and a and b_i are parameters.

Solving the differential equation we can get

$$\begin{aligned} z(t) &= \left(y_1(0) - \frac{b_1}{a}y_2(t) - \frac{b_2}{a}y_3(t) - \cdots - \frac{b_{n-1}}{a}y_n(t)\right)e^{-at} \\ &\quad + \frac{b_1}{a}y_2(t) + \frac{b_2}{a}y_3(t) + \cdots + \frac{b_{n-1}}{a}y_n(t). \end{aligned} \tag{10}$$

If $d = ((b_1/a)y_2(t)) + ((b_2/a)y_3(t)) + \cdots + ((b_{n-1}/a)y_n(t))$, we will derive

$$\begin{aligned} z(t) &= \left((y_1(0) - d) \cdot \frac{e^{-at}}{1 + e^{-at}} + d \cdot \frac{1}{1 + e^{-at}}\right) \cdot \left(1 + e^{-at}\right) \\ &= \left((y_1(0) - d) \cdot \left(1 - \frac{1}{1 + e^{-at}}\right) + d \cdot \frac{1}{1 + e^{-at}}\right) \cdot \left(1 + e^{-at}\right) \\ &= \left((y_1(0) - d) - y_1(0) \cdot \frac{1}{1 + e^{-at}} + 2d \cdot \frac{1}{1 + e^{-at}}\right) \cdot \left(1 + e^{-at}\right). \end{aligned} \tag{11}$$

Then we map it to the neural network to get a gray neural network. Its structure is shown in Figure 3.

Network output is

$$y_1 = w_{31}c_1 + w_{32}c_2 + \cdots + w_{3n}c_n - \theta_{y1}. \tag{12}$$

The weight of the input layer is $w_{11} = a$, the weight of the input layer and the hidden layer is $w_{21} = -x_1(0)$, $w_{2i} = (2b_{i-1}/a)$, $i = 2, \ldots, n$, the weight of the hidden layer and the output layer is $w_{3j} = a + e^{-at}$, $j = 1, \ldots, n$, the output of the input layer is $h = 1/(1 + e^{-at})$, the output of the hidden layer is $c_1 = hw_{21}$, $c_i = y_i(t)hw_{2i}$, $i = 2, \ldots, n$, and the deviation of the output layer is $\theta_{y1} = (1 + e^{-at})(d - y_1(0))$. Note that when the input variable is $n - 1$, the number of hidden layer nodes is n.

3.2. IMFO

3.2.1. MFO. The MFO was inspired by the lateral positioning of moths during night flight [11]. The moths often

keep the same angle with the moon when flying. Because they are far away from the moon, the moth and the moon are connected in parallel at different times. So, the moths can fly in a straight line. However, in reality, the moth is very close to the flame, and the moth still maintains the angle with the flame, so that the moth flies along the equiangular spiral to the flame. Hence, there is a "moth to flame," as shown in Figure 4. Prof. Mirjalili was inspired by this isometric spiral function to invent the MFO algorithm.

In the original MFO algorithm, **M** is the moth matrix, that is, the agent that searches for the solution, $\mathbf{Z}_{\mathrm{M}}$ is the moth fitness value matrix, **F** is the flame position matrix, that is, the current optimal solution, and $\mathbf{Z}_{\mathrm{F}}$ is the flame fitness value matrix, as shown in the formula (13) and (14). At the initial moment, the flame matrix and the moth matrix have the same dimensions:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,d} \end{bmatrix},$$
$$\mathbf{Z}_{\mathrm{M}} = \begin{bmatrix} z_{m1} \\ z_{m2} \\ \vdots \\ z_{mn} \end{bmatrix}, \tag{13}$$

$$\mathbf{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,d} \end{bmatrix},$$
$$\mathbf{Z}_{\mathrm{F}} = \begin{bmatrix} z_{f1} \\ z_{f2} \\ \vdots \\ z_{fn} \end{bmatrix}, \tag{14}$$

where n is the number of moths and d is the number of dimensions.

The flame positions are sorted according to the fitness values from small to large, and the moths fly around the sorted flames, along the isometric spiral of equation (15), and update their position by changing the parameter t:

$$M_i = D_i e^{bt} \cos(2\pi t) + F_j, \tag{15}$$

where M_i is the new position of the ith moth, D_i, formulated as $D_i = |M_i - F_j|$, is the distance between the ith moth and the jth flame, b is the isometric spiral parameter, which determines the shape of the isometric spiral, and t is a random number between [−1, 1], controlling the distance between the moth and the flame, as shown in Figure 5. The smaller the t is, the closer the moth is to the flame. By changing t, the moth can reach the position around the flame, thus enhancing the local optimization performance of the algorithm.

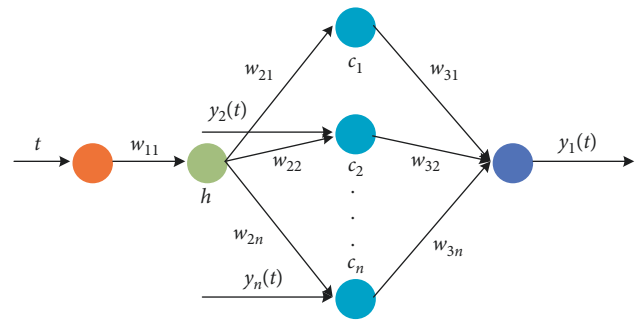

Figure 3: Network structure of GNNM.

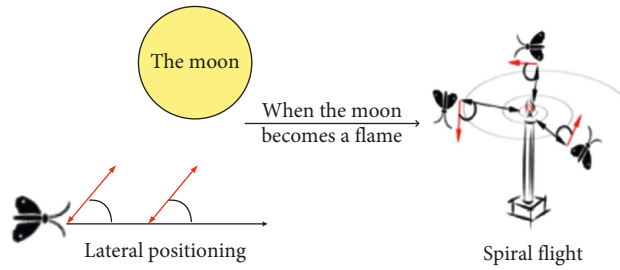

Figure 4: Lateral positioning flight and moth to flame.

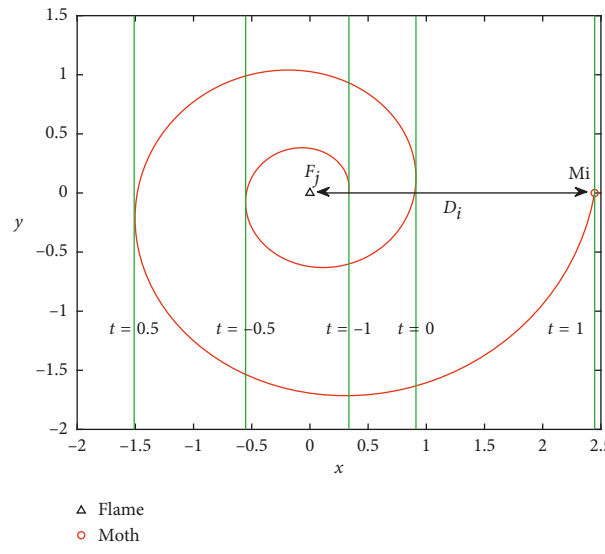

Figure 5: The position of moth changes with time.

In order to improve the search efficiency of the algorithm, MFO adopts the flame adaptive reduction mechanism, which makes the later convergence faster, without continuing to optimize around the inferior solution. The number of flames is adaptively reduced by

$$N_{\mathrm{F}} = \mathrm{round}\left(N_{\max} - I \times \frac{N_{\max} - 1}{I_{\max}}\right), \tag{16}$$

where N_{F} is the current number of flames, $N_{\max}$ is the maximum number of flames, I is the current number of iterations, and $I_{\max}$ is the maximum number of iterations. The number of flames varies with the number of iterations as shown in Figure 6.

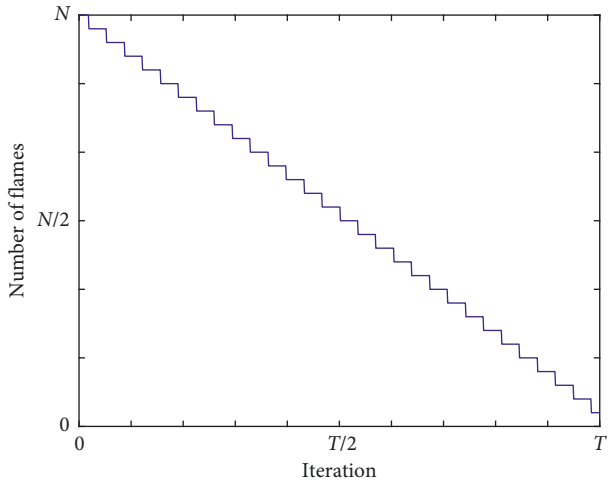

Figure 6: The number of flames decreases with the number of iterations.

When the number of flames is less than the number of moths, the moths fly around the worst-fitted flame.

3.2.2. Tent Chaos. The chaotic sequence [19–22] has ergodicity and randomness. The chaotic sequence can make the initial solution distribution more uniform, which is beneficial to find the global optimization solution. At present, two chaotic sequences are mainly used, namely logistic mapping and tent mapping. Shan et al. [23] proved that tent mapping is better ergodic than logistic mapping. Therefore, the tent mapping is used to optimize the initial population in this paper. Tent mapping and logistic mapping distribution are shown in Figure 7. Figure 7(a) is a tent map result of mapping 400 times from 0.01 to 0.5 according to the tent mapping formula, and Figure 7(b) is a logistic map result of mapping 200 times from 3 to 4 according to the logistic mapping formula.

The Tent mapping formula is as follows:

$$x_{t+1} = \begin{cases} 2x_t, & 0 \le x_t \le 0.5, \\ 2(1 - x_t), & 0.5 < x_t \le 1. \end{cases} \tag{17}$$

3.2.3. Lévy Flights. Since the original MFO algorithm selects the current optimality for each iteration and is prone to trap in local optima, Lévy flight is introduced. Lévy flight, different from the Brownian motion, is a random motion proposed by French mathematician Paul Pierre Lévy [24–26], which is characterized by a large number of small step movements mixed with a small amount of large step movement, as shown in Figure 8. Figure 8 is the result of flying 3000 times according to the Lévy formula. It can also be seen from this picture that Lévy flight has large steps and small steps.

The step size of each step movement and direction obeys Lévy distribution [27], as shown in the following equation:

$$L \sim \frac{\mu \times u}{|v|^{1/\beta}},$$
$$\mu = \left(\frac{(\Gamma(1+\beta) \times \sin((\pi \times \beta)/2)) \times rand}{\Gamma((1+\beta)/2) \times \beta \times 2^{(\beta-1)/2}} \right)^{1/\beta}. \tag{18}$$

Among them, $\Gamma(\lambda)$ is a standard gamma function, u and v obey the normal distribution, and β is a constant.

The current optimal solution updates as follows:

$$x_i^{t+1} = x_i^t + \alpha L, \tag{19}$$

where α is a scale factor that controls the learning step size for each step. In this way, after moving a small step, it will also move a large step size, which balances exploration and exploitation of the search space and helps to jump out of the local optimal solution.

3.2.4. Metropolis Criterion. The Metropolis criterion is the core of the simulated annealing algorithm. It accepts the current inferior solution with a certain probability, so that the algorithm has the ability to jump out of the local optimum and converge to the global optimal solution.

The Metropolis criterion uses the following process (take the minimization problem as an example):

(1) Adding the Lévy flight to the current optimal solution S_1 to generate a new solution S_2.

(2) Weight the new solution and the old solution according to equation (20) to get the final new solution. The new solution obtained in this way is beneficial to retain the advantages of the old solution and reduce the influence of the disturbance error:

$$S_2 = \text{alpha} \times S_2 + (1 - \text{alpha}) \times S_1. \tag{20}$$

(3) Calculate the increment, $df = f(S_2) - f(S_1)$, $f(S_1)$ is the cost function of S_1.

(4) According to the Metropolis criterion, the formula is expressed as follows:

$$P = \begin{cases} 1, & df < 0, \\ e^{-(df/T)}, & df \ge 0. \end{cases} \tag{21}$$

If $df < 0$, we accept the new solution; otherwise, we accept the new solution with the probability of $e^{-(df/T)}$.

3.2.5. IMFO. The IMFO algorithm is given in Algorithm 1.

3.3. IMFO-GNNM. Because the neural network learns the functional relationship between input and output by adjusting the weight connection, the learning effect depends on its weight parameters. In order to enhance the learning effect of GNNM, IMFO is used to optimize its weight parameters, and then GNNM with optimized weights is used to

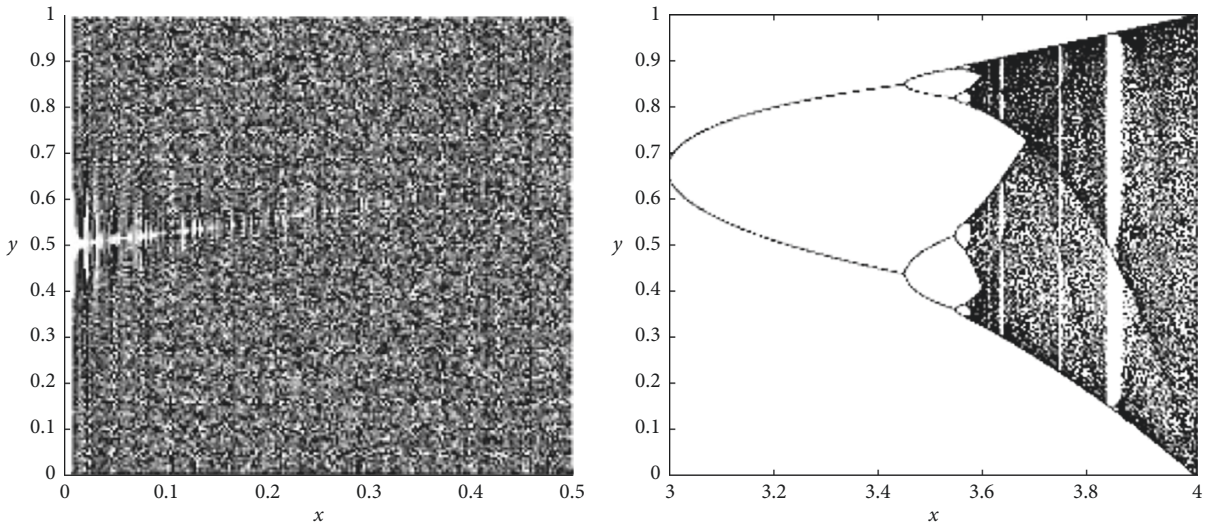

Figure 7: Tent map and logistic map. (a) Tent map. (b) Logistic map.

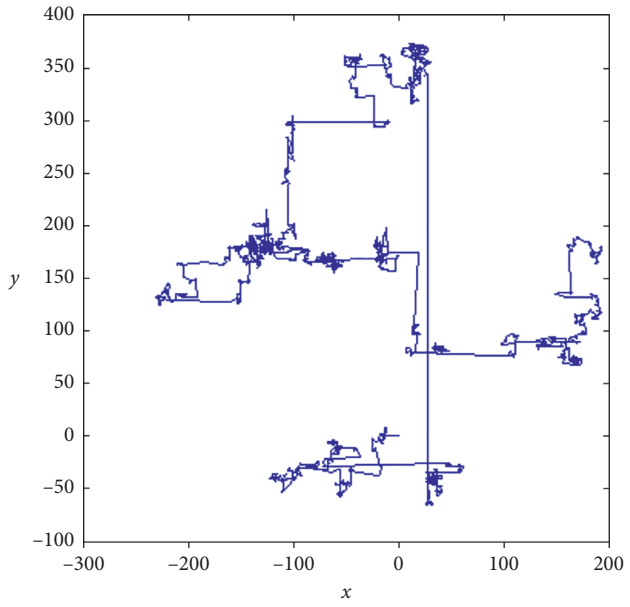

Figure 8: Lévy flights path.

predict. Therefore, the IMFO-GNNM algorithm is obtained, and the algorithm flow is shown in Figure 9.

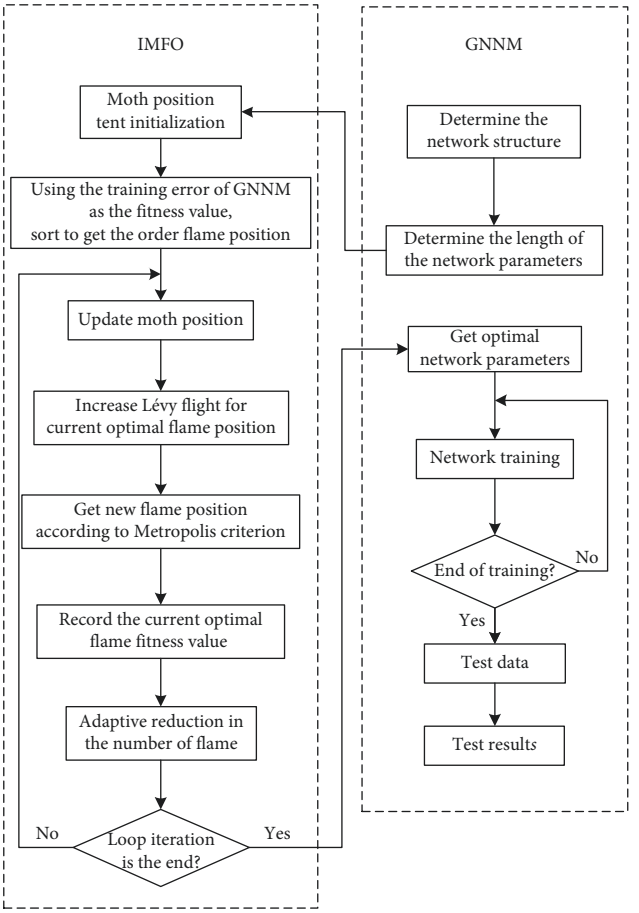

Figure 9: Flow chart for the proposed algorithm.

4. Results and Discussion

4.1. Verification Tests by Benchmark Functions. This paper selects six classical benchmark functions in CEC2010 to test the performance of the algorithm, as shown in Table 2. Among them, the Sphere function is a smooth monotonic function with only one global minimum, which is used to test the convergence speed of the algorithm. The Rosenbrock function is a nonconvex unimodal function with multiple local extremums, which is mainly used to test the convergence speed and optimization accuracy of the algorithm; Schwefel 2.26, Rastrigin, Ackley, and Griewank functions are complex multimodal functions with multiple local minima. They are mainly used to test the global optimization performance of the algorithm and ability to

```
(1) Initialize solution population using tent chaos map
(2) iteration = 1
(3) while (iteration ≤ Max_iteration)
(4)    OM = FitnessFunction(M)
(5)    if iteration = = 1
(6)       F = sort(M)
(7)       OF = sort(OM)
(8)    else
(9)       F = sort(M_{t−1}, M_t)
(10)      OF = sort(M_{t-1}, M_t)
(11)   end if
(12)   for i = 1 : n
(13)      for j = 1 : d
(14)         update t
(15)         calculate D with respect to the corresponding flame
(16)         update M(i, j) using equation (15) with respect to the corresponding flame
(17)      end for
(18)   end for
(19)   update the position of the current optimal agent using Lévy-flight
(20)   F_lévy = Lévy(F)
(21)   OF_lévy = FitnessFunction(F_lévy)
(22)   using the Metropolis criterion for OF and OF_lévy
(23)   update the position best flame obtained so far
(24)   update flame number using equation (16)
(25)   iteration = iteration + 1
(26) end while
```

Algorithm 1: Improvement moth flame optimization algorithm.

jump out of local extremums. These functions can more comprehensively examine the optimization performance of the algorithm.

In order to evaluate the effectiveness of the IMFO algorithm, this paper uses the above benchmark function to test and compare it with the MFO algorithm. The simulation was carried out with MATLAB R2014a. The parameters were set as follows: the number of moths is 30, the maximum number of iterations is 1000, the helix parameter is 1, the Lévy flight parameter is $\beta = 1.5$, and the weighting coefficient of the new solution and the old solution is 0.5. In order to reduce the error, the fitness mean obtained by continuously running 30 times is taken as the test result.

The fitness convergence curves of the 10-dimensional benchmark function test are shown in Figure 10–15. The fitness mean and standard deviation of the 10-dimensional and 50-dimensional benchmark function tests are shown in Table 3.

It can be seen from Figure 10–15 that for the unimodal function, the IMFO algorithm converges faster than the MFO algorithm, and the optimization precision is higher. For the multimodal function, the MFO algorithm basically traps in local optima and the convergence speed is slow. However, the IMFO algorithm can jump out of the local optimal trap and search for the optimal solution all the time. It has good global optimization performance and faster convergence speed.

It can be seen from Table 3 that the mean and standard deviation of the optimization of the IMFO algorithm are smaller than the mean and standard deviation of the MFO algorithm, which proves that the IMFO algorithm has better performance and good robustness. Especially for the Schwefel function and the Rastrigin function, the MFO algorithm converges prematurely and stagnates in advance to local optima, while the IMFO algorithm almost reaches the global minimum.

In summary, because the IMFO algorithm introduces Tent chaotic sequence initialization, Lévy flight, and Metropolis criterion, the initial population distribution is more uniform. At the same time, it has better ability to jump out of local optima in function optimization, and the global optimization performance is better.

4.2. Threat Assessment Tests

4.2.1. Test Settings. According to the threat assessment framework, the input dimension of the neural network is determined to be 7, that is, threat value, target type, damage ability, interference ability, airway angle, target speed, and target distance, and the output dimension is 1, that is, threat value. Therefore, the gray neural network structure is 1-1-8-1.

The IMFO algorithm parameter setting is the same as above, dimension is 7 and maximum running times of GNNM is 200. The experimental data randomly selected 100 groups from the target threat database, of which 75 groups are training sets and 25 groups are test sets. Some data are shown in Table 4.

Table 2: Benchmark functions.

Function	Equation and range	Range	Optimum	Property
Sphere	$f(X) = \sum_{i=1}^{D} x_i^2$	[−100, 100]	0	Unimodal
Rosenbrock	$f(X) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[−30, 30]	0	Unimodal
Schwefel 2.26	$f(X) = -\sum_{i=1}^{D} (x_i \sin \sqrt{\lvert x_i \rvert})$	[−500, 500]	−418.98	Multimodal
Rastrigin	$f(X) = \sum_{i=1}^{D} (100(x_i^2 - 10\cos(2\pi x_i) + 10))$	[−5.12, 5.12]	0	Multimodal
Ackley	$f(X) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{D} x_i^2}) - \exp((1/n)\sum_{i=1}^{D}\cos(2\pi x_i) + 20 + e)$	[−30, 30]	0	Multimodal
Griewank	$f(X) = (1/1000)\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(x_i/\sqrt{i}) + 1$	[−600, 600]	0	Multimodal

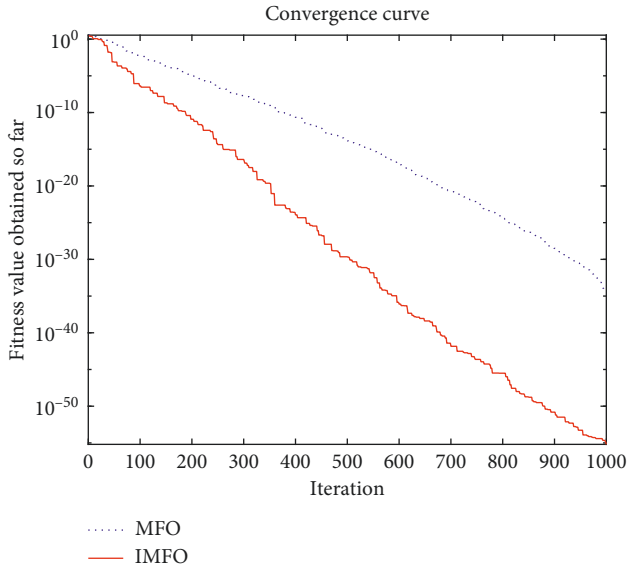

Figure 10: Sphere (10 dimensions).

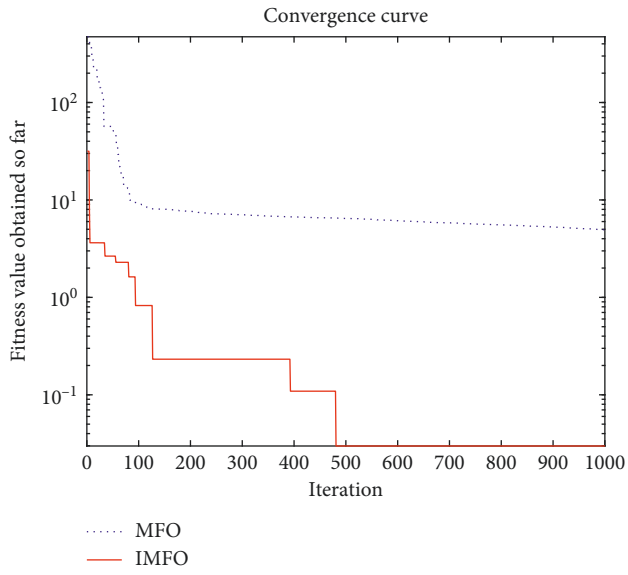

Figure 11: Rosenbrock (10 dimensions).

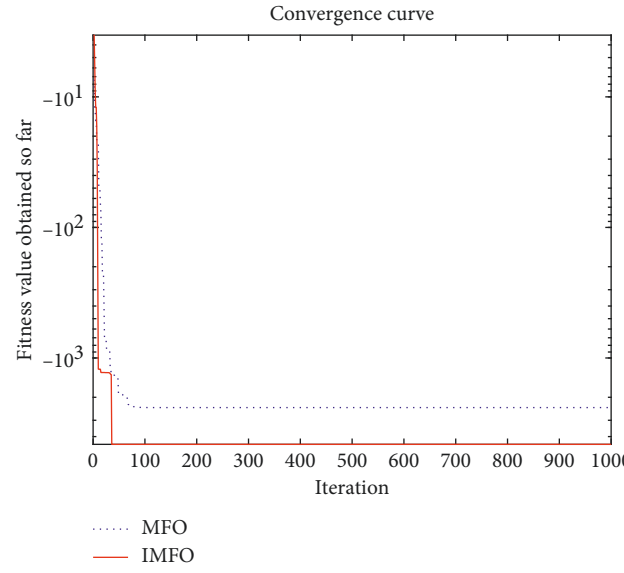

Figure 12: Schwefel (10 dimensions).

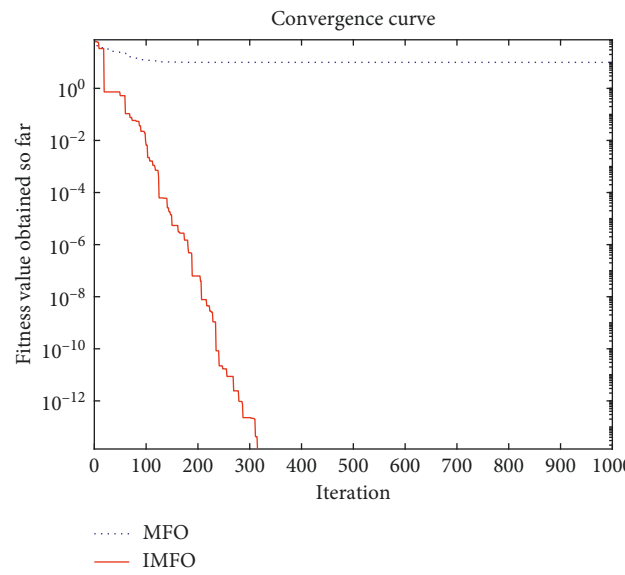

Figure 13: Rastrigin (10 dimensions).

Since the range of indicators is not uniform after quantification, the quantized data is normalized, and the data of each index is uniformly compressed into the interval [0, 1], which is beneficial to speed up the training of the model and improve the learning outcomes. The formula is as follows:

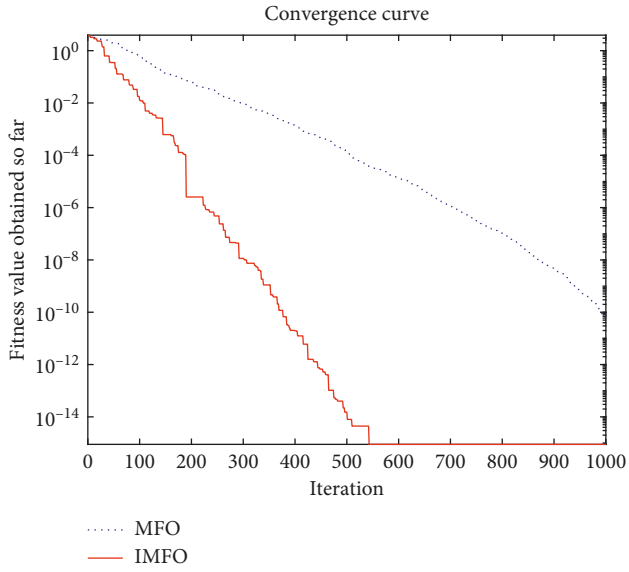

Figure 14: Ackley (10 dimensions).

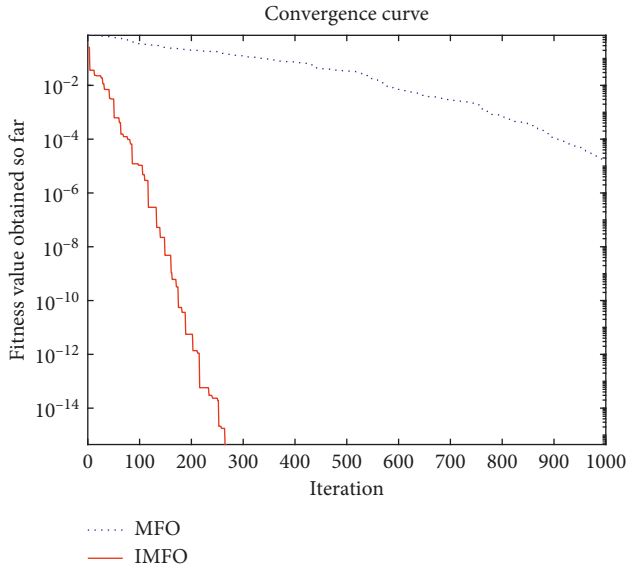

Figure 15: Griewank (10 dimensions).

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{22}$$

where x represents the actual value of a variable, $x_{\max}$ represents the maximum value, $x_{\min}$ represents the minimum value, and x' represents the normalized value.

In order to reduce the error, the experiment takes the average of 30 running results and compares it with the simulation results of the GNNM model, the original MFO-optimized GNNM model, and the IMFO-optimized GNNM model.

4.2.2. Accuracy Analysis. Firstly, the parameters of GNNM are optimized by the training set to obtain the optimal initialization parameters. The error evolution curve is shown in Figure 16.

Table 3: Optimization results comparison of benchmark functions.

Function	Dimension	Algorithm	Mean	Standard deviation
Sphere	10	MFO	8.738×10^{-35}	8.043×10^{-35}
		IMFO	$\mathbf{6.544 \times 10^{-55}}$	$\mathbf{1.936 \times 10^{-50}}$
	50	MFO	9.965×10^{-3}	4.507×10^{-2}
		IMFO	$\mathbf{3.346 \times 10^{-50}}$	$\mathbf{1.672 \times 10^{-48}}$
Rosenbrock	10	MFO	7.971	1.463
		IMFO	$\mathbf{8.954 \times 10^{-1}}$	$\mathbf{3.041 \times 10^{-1}}$
	50	MFO	4.873×10^{1}	1.676
		IMFO	$\mathbf{9.580 \times 10^{-1}}$	**1.637**
Schwefel 2.26	10	MFO	-0.999×10^{3}	1.649×10^{3}
		IMFO	$\mathbf{-4.188 \times 10^{3}}$	**1.403**
	50	MFO	-1.392×10^{4}	7.090×10^{2}
		IMFO	$\mathbf{-2.094 \times 10^{4}}$	$\mathbf{5.981 \times 10^{-2}}$
Rastrigin	10	MFO	5.597	6.085
		IMFO	$\mathbf{4.967 \times 10^{-16}}$	$\mathbf{3.455 \times 10^{-15}}$
	50	MFO	7.355×10^{1}	9.458
		IMFO	$\mathbf{3.678 \times 10^{-14}}$	$\mathbf{5.279 \times 10^{-11}}$
Ackley	10	MFO	4.209×10^{-15}	2.621×10^{-16}
		IMFO	$\mathbf{8.419 \times 10^{-16}}$	$\mathbf{3.847 \times 10^{-17}}$
	50	MFO	7.118×10^{-2}	2.937×10^{-1}
		IMFO	$\mathbf{8.882 \times 10^{-16}}$	$\mathbf{5.580 \times 10^{-18}}$
Griewank	10	MFO	1.314×10^{-4}	8.359×10^{-4}
		IMFO	$\mathbf{3.177 \times 10^{-17}}$	$\mathbf{3.564 \times 10^{-16}}$
	50	MFO	1.129×10^{-4}	9.423×10^{-4}
		IMFO	$\mathbf{1.761 \times 10^{-17}}$	$\mathbf{2.965 \times 10^{-15}}$

As can be seen from Figure 16, in the weight optimization process, IMFO has a faster optimization speed, a smaller mean square error, and better weight parameters.

After the three models are trained in the training set, the test set is input to test its accuracy. The GNNM model uses random initialization weights, the MFO-GNNM model uses MFO-optimized weight parameters, and the IMFO-GNNM model uses IMFO-optimized weight parameters. The test results are shown in Figure 17.

The expected value in the figure represents the actual threat value. It can be seen from the figure that when the unoptimized GNNM predicts the test set, the predicted value of the individual points deviates greatly from the expected value and the accuracy is not high enough. The prediction effect of the MFO-GNNM model has improved. The IMFO-GNNM model with optimized initial weights has the best weight value, and the deviation between the predicted value and the expected value is the smallest, and the prediction effect is the best.

The relative error and mean square error of the three models on the test set are compared below, as shown in Figure 18 and Table 5.

It can be seen from Figure 18 and Table 5 that the relative error of the individual points predicted by the GNNM model reaches about 0.8, the relative error is larger, and the mean square error is 0.013. The relative error of the MFO-GNNM model decreases, and the mean square error is reduced to 0.004; the IMFO-GNNM model has a small overall prediction error and a mean square error of only 0.0012. This indicates that MFO algorithm optimization weight can improve the learning effect of the model. The IMFO

Table 4: Ten sets of target threat database data.

Serial number	Threat level	Target type	Damage ability	Interference ability	Airway angle (°)	Target speed (m/s)	Target distance (km)	Threat value
1	0.63	BA	0.922	Strong	9	400	100	0.5707
2	0.34	BA	0.796	Medium	17	450	200	0.5333
3	0.35	FB	0.523	Super strong	5	600	100	0.6895
4	0.23	FB	0.674	Medium	18	550	240	0.5587
5	0.21	AH	0.646	Weak	12	110	150	0.7142
6	0.15	PGB	0.525	No	13	300	5	0.3015
7	0.25	AGM	0.884	No	11	740	40	0.5256
8	0.17	CM	0.746	No	9	690	15	0.5344
9	0.23	TBM	0.875	No	13	880	50	0.6188
10	0.13	ARM	0.627	No	16	750	8	0.6426

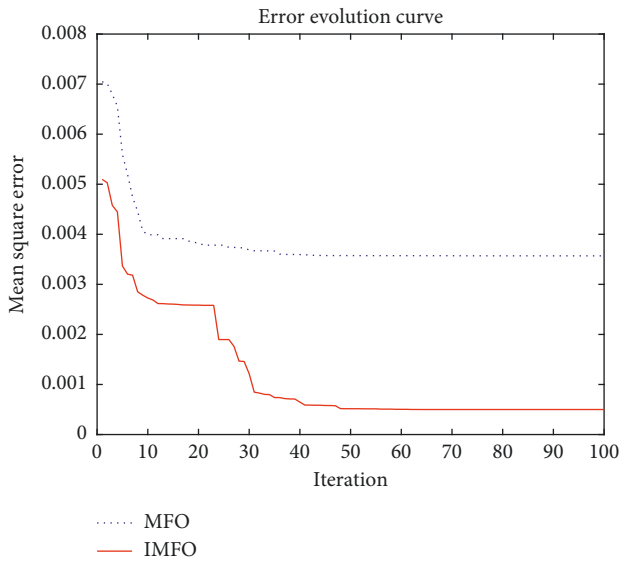

Figure 16: Error evolution curve.

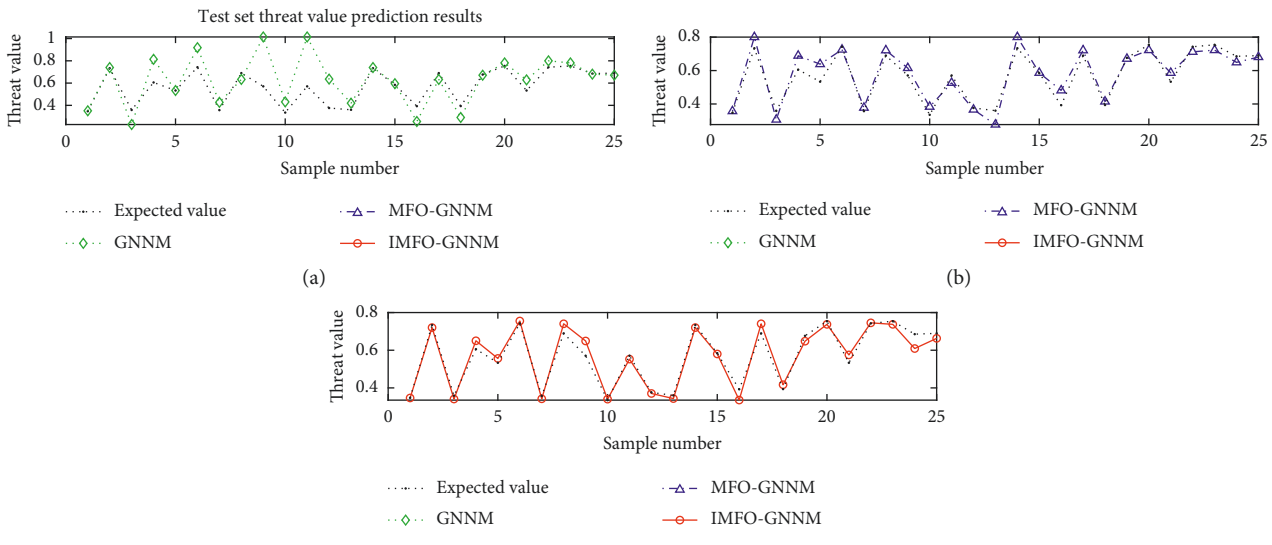

Figure 17: Assessment results of test set.

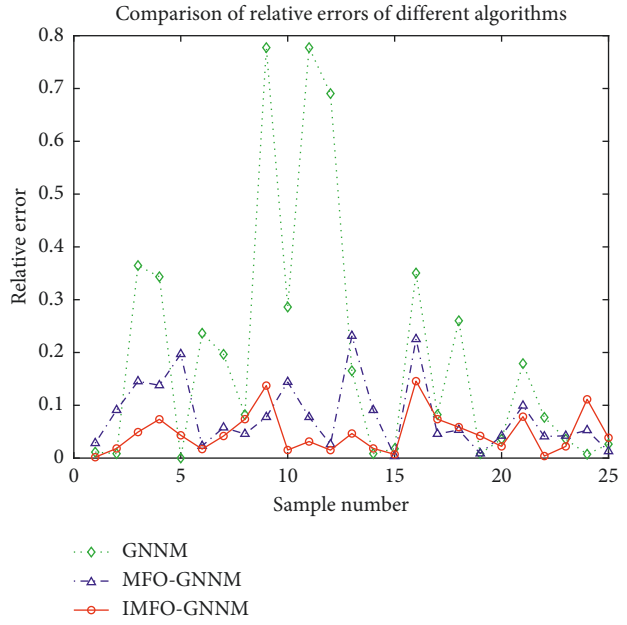

Figure 18: Relative error of test set.

Table 5: Mean square error of test set.

Models	GNNM	MFO-GNNM	IMFO-GNNM
MSE	0.013	0.004	0.0012

algorithm can obtain better initial weight parameters, so the prediction relative error is the smallest and the accuracy is the highest, which verifies the effectiveness of the proposed method.

4.2.3. Real-Time Performance Analysis. The training set is used to evaluate the running time of the three algorithms. The results are shown in Figure 19.

As can be seen from the figure, the running time of the three models: GNNM < IMFO-GNNM < MFO-GNNM. This is because the GNNM model directly inputs the training set for training after random initialization, while both the MFO-GNNM model and the IMFO-GNNM model need to be trained in the training set to obtain the optimized weight parameters, so the model runs for a long time. At the same time, the IMFO model has been improved and the search speed is faster, so the IMFO-GNNM model runs slightly faster than the MFO-GNNM model.

Considering that the neural network algorithm adopts offline training and online prediction, the evaluation time of 5, 10, 15, 20, and 25 sets of data is compared with the three algorithms. The results are shown in Figure 20.

With reference to Figure 20, the evaluation time of the three models is approximative. This is because the three models use the same network structure, but with different weights. Moreover, it takes less than 10 ms to evaluate 25 sets of data, which meets the real-time requirements.

In summary, the IMFO-GNNM model sacrifices the training time of the algorithm while improving its accuracy,

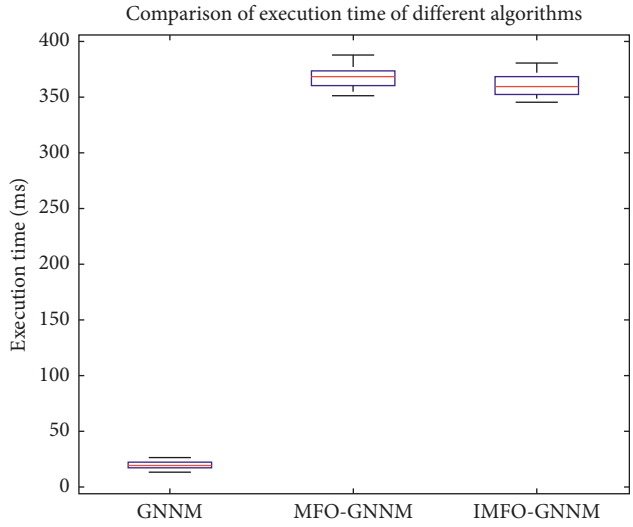

Figure 19: Elapsed time of the algorithms.

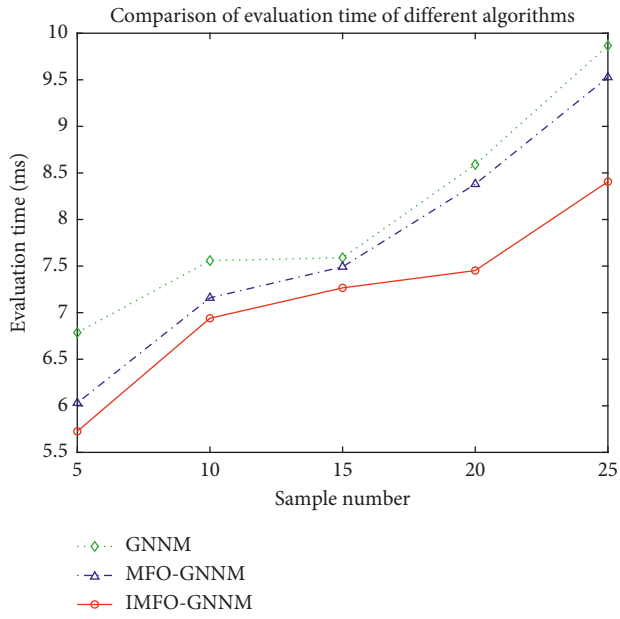

Figure 20: Assessment time of the algorithms.

but it does not affect the real-time assessment. Therefore, the IMFO-GNNM model can improve the accuracy of air target threat assessment and has better real-time performance.

5. Conclusions

(1) By analyzing the actual situation of air defense operations and considering threat level, threat ability, and threat extent, a reasonable air target threat assessment framework is established, and each indicator is scientifically quantified.

(2) The IMFO algorithm is proposed. By introducing Tent chaos, Lévy flight, and Metropolis criterion, the global optimization performance of the algorithm is improved.

(3) Combining powerful learning ability of GNNM and excellent optimization performance of IMFO, a new air target threat assessment model based on IMFO-GNNM is established, which has high accuracy and good real-time performance for threat assessment of incoming targets.

We will continue to study and propose better threat assessment models in the future. At the same time, we will study how to incorporate the proposed threat assessment model into the air defense system.

Data Availability

The data used to support the findings of this paper are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of the paper.

Authors' Contributions

Longfei Yue planned the work, completed the simulation experiment, and drafted the main part of the paper. Rennong Yang and Jialiang Zuo performed error analysis. Hao Luo and Qiuliang Li contributed to setup type.

Acknowledgments

The work described in this paper is partially supported by the National Natural Science Foundation of China under Grant no. 61503409.

Supplementary Materials

See the threat_data.xlsx file in the Supplementary Material for data. See the tent.m file and logistic.m file in the Supplementary Material for Figure 7. See the levy.m file in the Supplementary Material for Figure 8. See the main.m file in the Supplementary Material for Figures 10–15. (*Supplementary Materials*)

References

[1] S. Kumar and B. K. Tripathi, "Modelling of threat evaluation for dynamic targets using Bayesian network approach," *Procedia Technology*, vol. 24, pp. 1268–1275, 2016.

[2] Y. Wang, Y. Sun, J. Y. Li, and S.-T. Xia, "Air defense threat assessment based on dynamic Bayesian network," in *Proceedings of the 2012 International Conference on Systems and Informatics (ICSAI2012)*, pp. 721–724, Yantai, China, May 2012.

[3] Y. F. Liu, S. D. Chen, Z. Y. Zhao, and A. Zhang, "Threat assessment of manned/unmanned combat aerial vehicle formation air-to-ground attack based on FBNs," *Systems Engineering and Electronics*, vol. 34, no. 8, pp. 1636–1639, 2012.

[4] Y. Xu, Y. Wang, and X. Miu, "Multi-attribute decision making method for air target threat evaluation based on intuitionistic fuzzy sets," *Journal of Systems Engineering and Electronics*, vol. 23, no. 6, pp. 891–897, 2012.

[5] Y. Wang and X. Miao, "Intuitionistic fuzzy perceiving methods for situation and threat assessment," in *Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 578–582, Chongqing, China, May 2012.

[6] D. F. Chen, Y. Feng, and Y. X. Liu, "Threat assessment for air defense operations based on intuitionistic fuzzy logic," *Procedia Engineering*, vol. 29, pp. 3302–3306, 2012.

[7] H. Lee, B. J. Choi, C. O. Kim, J. S. Kim, and J. E. Kim, "Threat evaluation of enemy air fighters via neural network-based Markov chain modeling," *Knowledge-Based Systems*, vol. 116, pp. 49–57, 2017.

[8] H. Yang, C. Han, and C. Tu, "Air targets threat assessment based on BP-BN," *Journal of Communications*, vol. 13, no. 1, pp. 21–26, 2018.

[9] H. P. Cai, J. X. Liu, and Y. W. Chen, "Threat assessment of targets based on support vector machine," *Journal of China Ordnance*, vol. 2, no. 3, pp. 173–177, 2006.

[10] J. L. Deng, *The Primary Methods of Gray System Theory*, Huazhong University of Science and Technology Press, Wuhan, China, 2005, in Chinese.

[11] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[12] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[13] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, no. 3, pp. 46–61, 2014.

[14] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[15] P. Jangir, A. Kumar, I. N. Trivedi, R. Bhesdadiya, M. H. Pandya, and N. Jangir, "Moth-flame optimization algorithm for solving real challenging constrained engineering optimization problems," in *Proceedings of the IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pp. 1–5, Bhopal, India, March 2016.

[16] S. Khalilpourazari and S. Khalilpourazary, "An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems," *Soft Computing*, vol. 23, no. 5, pp. 1–24, 2019.

[17] J. P. Yang, J. Wang, and W. T. Liang, "Research on method of the threaten queuing based on an-missile," *Journal of China Academy of Electronics & Information Technology*, vol. 7, no. 4, pp. 432–436, 2012, in Chinese.

[18] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.

[19] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[20] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[21] X. Liao, J. Zhou, S. Ouyang, R. Zhang, and Y. Zhang, "An adaptive chaotic artificial bee colony algorithm for short-term hydrothermal generation scheduling," *International Journal of Electrical Power & Energy Systems*, vol. 53, no. 12, pp. 34–42, 2013.

[22] C. Xu, H. Duan, and F. Liu, "Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning," *Aerospace Science and Technology*, vol. 14, no. 8, pp. 535–541, 2010.

[23] L. Shan, H. Qiang, J. Li, and Z. Q. Wang, "Chaotic optimization algorithm based on tent map," *Control and Decision*, vol. 20, no. 2, pp. 179–182, 2005, in Chinese.

[24] A. V. Chechkin, R. Metzler, J. Klafter, and V. Y. Gonchar, "Introduction to the theory of lévy flights," *Anomalous Transport: Foundations and Applications*, vol. 135, no. 2, pp. 129–162, 2008.

[25] A. F. Kamaruzaman, A. M. Zain, S. M. Yusuf, and A. Udin, "Levy flight algorithm for optimization problems—a literature review," *Applied Mechanics and Materials*, vol. 421, pp. 496–501, 2013.

[26] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the 2009 IEEE World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 210–214, Coimbatore, India, December 2009.

[27] Z. Li, Y. Zhou, S. Zhang, and J. Song, "Lévy-flight moth-flame algorithm for function optimization and engineering design problems," *Mathematical Problems in Engineering*, vol. 2016, Article ID 1423930, 22 pages, 2016.