International Journal of Innovative Computing, Information and Control
Volume **15**, Number **5**, October **2019**

ICIC International ©2019 ISSN 1349-4198
pp. 1967–1987

AN IMPROVED GRASSHOPPER OPTIMIZATION ALGORITHM FOR TASK SCHEDULING PROBLEMS

Ran Zhao[1,2], Hong Ni[1,2], Hangwei Feng[1,2], Yaqin Song[1,2] and Xiaoyong Zhu[1,2,*]

[1]National Network New Media Engineering Research Center
Institute of Acoustics, Chinese Academy of Sciences
No. 21, North 4th Ring Road, Haidian District, Beijing 100190, P. R. China
{ zhaor; nih; fenghw; songyq }@dsp.ac.cn; *Corresponding author: zhuxy@dsp.ac.cn

[2]School of Electronic, Electrical and Communication Engineering
University of Chinese Academy of Sciences
No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, P. R. China

Received December 2018; revised April 2019

Abstract. *Grasshopper Optimization Algorithm (GOA) is a novel meta-heuristic algorithm for optimization problems. GOA is easy to implement but it cannot make full utilization of every iteration, and it is easy to fall into the local optimal. To improve the performance of GOA, an Improved Grasshopper Optimization Algorithm (IGOA) was proposed in this paper. Firstly, the nonlinear comfort zone parameter was used to promote the utilization of the iterations of the algorithm. Then the Lévy flight mechanism was applied to increasing the randomness and expanding the local search radius. At last the random jumping strategy was introduced to help the algorithm jump out of the local optimal. Several experiments involving 29 well-known benchmark functions were conducted and the performance of IGOA was compared with the basic GOA, Opposition-Based Learning GOA (OBLGOA), Whale Optimization Algorithm (WOA), Ant Lion Optimizer (ALO), Dragonfly Algorithm (DA) and Particle Swarm Optimization (PSO). IGOA was also applied to the task scheduling problem in the resource-constrained system. The results of the experiments demonstrated that the proposed IGOA outperformed GOA and other algorithms.*

Keywords: Grasshopper optimization algorithm, Nonlinear parameter, Lévy flight, Random jumping strategy, Task scheduling problem

1. Introduction.

An optimization problem is a process to search for the global optimal value and the corresponding best location of a mathematical function [1]. An optimization problem can be abstracted in many areas. While cloud computing and edge computing are developing rapidly, the problem of task scheduling becomes more and more prominence. The task scheduling problem can be abstracted as a kind of optimization problem in mathematics. It is essential for a scheduling system that the performance of the scheduling algorithm is outstanding. A superior algorithm can optimize the task execution results and reduce much extra makespan and budget.

Recently, many studies have been developed on task scheduling problems. As the study goes on, many meta-heuristic algorithms are used to handle complicated optimization problems. Meta-heuristic algorithms have the ability to find the global optimum with simple operation and less overhead. Grasshopper Optimization Algorithm (GOA) is a novel meta-heuristic algorithm proposed by S. Saremi et al. in 2017 [1]. GOA, which is inspired by the natural behavior of the grasshopper swarm, makes utilization of the swarm

DOI: 10.24507/ijicic.15.05.1967

intelligence to solve optimization problems. The two tendencies of the optimization search process, which are exploration and exploitation, coincide with the migration behaviors of the grasshopper swarm.

GOA has been widely used in many areas since it was proposed. S. Łukasik et al. used GOA to generate accurate data clusterings and compare the performance with the standard K-means algorithm [2]. N. Rajput et al. applied GOA to solving three types of economic dispatch problems in electrical power systems, and the results of the experiments showed that GOA is superior to other methods [3]. Z. Elmi and M. Ö. Efe employed GOA to search for a better path in robot navigation [4]. As far as our research results are concerned, there are not many improvements on the algorithm itself or the applications for GOA on handling the task scheduling problems in cloud computing or edge computing.

While there are many advantages with the GOA algorithm, the disadvantage of being easy to fall into the local optimum also prevents the search process from finding a better solution. To overcome the disadvantages and improve the accuracy of GOA when handling the optimization problems, an improved grasshopper optimization algorithm was proposed in this paper. And the proposed IGOA was applied on solving the task scheduling problems. The main academic contributions of this paper are as follows.

- Attach a nonlinear comfort zone parameter to the original GOA to enhance the search ability of the algorithm by making full utilization of every iteration.
- Introduce a local search mechanism based on Lévy flight to improve the performance of the algorithm by expanding the search radius of the search agents.
- Add a random jumping strategy to promote the capability of jumping out of the local optimum and to continue the influence of the newly obtained information of the jumping action.

The remainder of this paper is organized as follows. A literature survey about classical and novel optimization algorithms is proposed in the second section. The basic theory of grasshopper optimization algorithm is introduced in Section 3. The proposed improved grasshopper optimization algorithm is described in detail in Section 4. Several experiments of 29 benchmark functions are implemented and the results are shown in Section 5. A model of task scheduling problems is proposed with experiments about 7 algorithms in Section 6. Some conclusion and future work are proposed in the last section.

2. Related Work.

Many meta-heuristic algorithms are introduced to optimization problems. GA is a classical meta-heuristic algorithm proposed by Goldberg in 1988, and it introduces the theory of natural selection into the process of optimization with several natural operators including mutation, crossover, and selection [5, 6, 7, 8]. While the performance of GA is pretty good, the operations of GA are too complicated to implement, and it is not suitable for some situations. Some meta-heuristic algorithms are inspired by the natural behavior of insects, fishes, birds, and other group creatures. Particle Swarm Optimization (PSO) is a classical meta-heuristic algorithm proposed by Kennedy in 1995. The principle of PSO is simple, and the performance is remarkable [9, 10, 11]. Ant Colony Optimization algorithm (ACO) is inspired by the natural foraging behavior of ants between the nest and the food sources. ACO makes utilization of chemical pheromone to communicate among the swarm of ants [12, 13].

Some novel meta-heuristic optimization algorithms are proposed recently. There are not many pieces of research on the improvement of those algorithms. The Ant Lion Optimizer (ALO) proposed in 2015 is inspired by the hunting behavior of antlions [14]. Whale Optimization Algorithm (WOA) was proposed in 2016. WOA simulates the natural hunting behavior of whales [15]. Dragonfly Algorithm (DA) proposed in 2016 is inspired by the static and dynamic behaviors of the dragonfly swarm in nature [16].

In 2017, S. Saremi et al. proposed a novel meta-heuristic optimization algorithm called Grasshopper Optimization Algorithm (GOA). GOA simulates the migration behavior of the grasshopper swarm by utilizing the influence of the interaction within the swarm and the wind influence outside the swarm to find the target food [1]. The GOA algorithm makes utilization of swarm intelligence, which fixes the search direction and finds the best or the approximate best location by sharing the experience among the grasshopper swarm. GOA also uses the evolutionary approach with several iterations to make the swarm intelligence efficient.

Some improved algorithms based on GOA have been developed. OBLGOA was proposed by A. A. Ewees et al. in 2018 [17]. The opposition-based learning strategy was introduced to generate an opposite solution as a candidate according to the current search position. The OBL strategy could improve the convergence rate of the algorithm, but the improvement was limited because of its lack of randomness. S. Arora and P. Anand proposed chaotic grasshopper optimization algorithm in 2018 [18]. The chaotic maps were applied to the algorithm to improving the performance of GOA. 10 chaotic maps were employed to evaluate the impact of the chaos theory. The results were not very particularly ideal because the chaotic factors were not suitable when handling many benchmark functions. This paper proposed a new algorithm based on GOA to solve optimization problems and task scheduling problems.

The proposed method involves Lévy flight. Lévy flight is a kind of random search walk proposed by Paul Lévy [16]. X. Yang and S. Deb formulated a meta-heuristic algorithm called Cuckoo Search (CS) based on Lévy flight [19]. L. Ying et al. applied Lévy to the Lévy flight trajectory-based Whale Optimization Algorithm (LWOA) [20] in 2017. Random walk can be generated by Lévy flight in both local and global scope, which can make the algorithm more creative.

Task scheduling problem is a problem to schedule several tasks to several nodes under constraints. Task scheduling problem can be an optimization problem. Many algorithms are applied to solving the task scheduling problem. Some algorithms based on Best Resource Selection (BRS) such as Max-Min, Min-Min, and Sufferage are traditional methods to solve task scheduling problems. Some meta-heuristic algorithms such as PSO and PSO-based improved algorithms are novel methods to handle task scheduling problems [21].

3. **Grasshopper Optimization Algorithm.**

Grasshopper optimization algorithm simulates the insect swarm behavior of grasshoppers [1]. The grasshopper swarm migrates over a long distance to find a new habitat with food. In this process, the interaction among grasshoppers influences each other inside the swarm. The power of the wind and the gravity outside the swarm influence the trajectory of the grasshoppers. The target of food is also an important influence factor.

With the influence of the three factors mentioned above, the migration process is divided into two stages which are exploration and exploitation. In the exploration stage, the grasshoppers are encouraged to move rapidly and abruptly to find more potential target areas. In the stage of exploitation, the grasshoppers tend to move locally to find better target areas. The two migration tendencies of exploration and exploitation to find a food source are achieved by grasshoppers naturally. This process can be abstracted as an optimization problem. The grasshopper swarm is abstracted as a swarm of search agents.

S. Saremi presented the mathematical model of the migration of the grasshopper swarm [1]. The simulating equation is shown as follows:

$$X_i = S_i + G_i + A_i \tag{1}$$

where X_i is the position of the i-th search agent, S_i represents the force of social interaction, G_i represents the influence factor of gravity force on the i-th search agent, and A_i represents the influence factor of the wind. S_i is defined as follows:

$$S_i = \sum_{j=1, j\neq i}^{N} s(d_{ij})\widehat{d_{ij}} \tag{2}$$

where d_{ij} defines the Euclidean distance between the i-th and the j-th search agent in space, and it is calculated as $d_{ij} = |x_j - x_i|$, $\widehat{d_{ij}}$ is the unit vector from the i-th to the j-th search agent, defined as $\widehat{d_{ij}} = \frac{x_j - x_i}{d_{ij}}$, and s is a function which shows the social relationship affection in the grasshopper swarm. The s function is defined as follows:

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{3}$$

where e is the Natural Logarithm, f represents the concentration of attraction and the parameter of l shows the attractive length scale.

When it is used to handle the mathematical optimization problem, some change factors should be added into Equation (1) to optimize the mathematical module. The parameters of G_i and A_i which represent the outside influence should be replaced by the parameter of food target. Thus, the equation is reformulated as follows:

$$x_i = c\left(\sum_{j=1, j\neq i}^{N} c\frac{u-l}{2}s(|x_j - x_i|)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T_d} \tag{4}$$

where u and l represent the upper and lower boundaries of the search space respectively, and $\widehat{T_d}$ is the food target position which represents the best fitness position which all the search grasshoppers can find in all-time in the mathematical module. Besides, the parameter c is the comfort zone parameter changing to balance the process of exploitation and exploration which is calculated as follows:

$$c = cmax - iter\frac{cmax - cmin}{MaxIteration} \tag{5}$$

where $cmax$ and $cmin$ are the maximum value and the minimum value of c respectively, $iter$ represents the current iteration, and $MaxIteration$ represents the maximum number of iterations.

Equation (4) should be iterated over as an evolution for the optimal solution. The evolution should be stopped when the termination condition is reached. Usually, it is when the preset maximum iteration number is reached. After the process of the evolution ends, the algorithm can get the approximate best fitness and its corresponding target position.

4. Improved Grasshopper Optimization Algorithm.

GOA has a simple theory foundation, and it is easy to implement. On the other hand, it has some disadvantages which prevent the algorithm from getting better solutions. The linearly decreasing comfort zone could not help the original GOA to make full utilization of every iteration. The original algorithm has little variability because of the lack of random factors. The algorithm is easy to fall into a local optimum. To handle the disadvantages, three improvements were introduced which were the nonlinear comfort zone parameter, the local search mechanism based on Lévy flight and the random jumping strategy. The details of the three improvements are explicitly described in this part.

4.1. **Nonlinear comfort zone parameter.** The original GOA varies the radius of the comfort zone, which can make the search agents converge towards the global optimum solution through iterations. GOA uses the comfort zone parameter to restrict the search space. In the exploration stage, the comfort zone parameter should be large enough to make the search agents get enough space to find the approximate optimum rapidly. In the exploitation stage, the restrictive factor should be small to search accurately towards the local optimum and avoid the over-speed movement of the search agents. However, the linearly decreasing factor could not make the search ability harmony with the exploration and the exploitation stage during the search iterations.

To match the two search stages and enhance the search ability of the algorithm, the sigmoid function was introduced to this work. The sigmoid function is a commonly used threshold function and a nonlinear adjustment factor. It is widely used in the field of information science. The formula of the sigmoid function is shown as follows:

$$f(x) = \frac{1}{1+e^{-x}} \tag{6}$$

A nonlinear comfort zone parameter based on a variant of the sigmoid function is proposed as follows:

$$m = \frac{-0.5}{1+e^{(-1.5(cx-5)+2\sin(cx))}} + u \tag{7}$$

where u is the adjustment parameter and its value should be in the interval $[0, 1]$. And cx is defined as follows:

$$cx = \frac{v(iter+50)}{MaxIteration} \tag{8}$$

where v is the accuracy adjustment factor and its value should be in the interval $[1, 10]$.

4.2. **Local search mechanism based on Lévy flight.** All the parameters of GOA are deterministic. The lack of randomness might lead to the lack of creativity during the search iterations, and every search agent could only search the determinate position. Introducing random factor to a deterministic system is a commonly used method to improve its performance.

Lévy flight is a random search walk proposed by Paul Lévy [16], and it is an efficient mathematical method to provide a random factor. Because Lévy flight is very complicated to implement, a simulating algorithm is used here as follows:

$$Levy(d) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{9}$$

where d is the dimension of the problem, r_1, r_2 are two random numbers in $[0, 1]$ and β is a constant number which is set to 1.5 according to S. Mirjalili in [16]. σ is calculated as follows:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times 2\left(\frac{\beta-1}{2}\right)}\right)^{\frac{1}{\beta}} \tag{10}$$

where $\Gamma(x) = (x-1)!$.

Lévy flight could give vision to all the search agents when they are moving towards the optimal position. The search agents could see the small areas around them. To expand the search radius of the search agents and enhance the ability to find the optima, a local search mechanism based on Lévy flight is proposed. When a location updating process is finished, the position of every search agent should be adjusted through Lévy flight with a certain probability. The adjustment formula is defined as follows:

$$X_i = X_i + 10c \times s_{threshold} \times Levy(dim) \times X_i \tag{11}$$

where $s_{threshold}$ is the threshold parameter which controls the direction and the probability of the variation. $s_{threshold}$ is calculated as follows:

$$s_{threshold} = sign(x_{trans} - 1) + sign(x_{trans} + 1) \tag{12}$$

where $sign(x)$ is the sign function and x_{trans} is a random number in $[-3, 3]$.

4.3. **Random jumping strategy.** The basic theory of GOA is elementary. The algorithm only focuses on the process of convergence to the global optimum and ignores the mechanism about jumping out of the local optimum. Hence the search process of GOA was easy to be trapped in local optimum, and the search could not go further. The advantage that GOA was simple to implement could not contribute to getting rid of the local optimum, which could be a disadvantage of GOA instead.

To promote the ability to jump out of the local optimum, a random jumping strategy is introduced. When a search agent finds an optimal position, the new position can replace the old target position. When it does not, the random jumping equation starts to work. It is described as follows:

$$X_i^{new} = ((0.5 - rand) \times 2 + 1)X_i \tag{13}$$

where X_i is the position of the i-th search agent, and X_i^{new} is the new position after random jumping. If X_i^{new} has better fitness, it will replace X_i. Thus, action of jumping out occurs successfully.

The evolution formula of the original GOA only takes the best position obtained by the current iteration as the search direction, and it ignores some other useful information. To continue the influence of the newly obtained information of the jumping action, the evolution formula of the location is transformed as follows:

$$X_i^{iter+1} = m \times c \times S_i + (1 - p)\widehat{T_d} + p \times X_i^{iter} \tag{14}$$

where p is the coefficient parameter to control the impact of the position of the search agent. p is initialized as 0 at the first iteration. If the search agent does not jump or it fails to jump out of the local optimal location, p is still set to 0 to make sure that the evolution can be affected only by S_i and T_d. When the search agent jumps out successfully, p is set to a variable linearly decreasing to 0 in three iterations to continue the influence of the behavior of jumping. After some trial, the decreasing step of p is set to 0.3478 which is not discussed in this paper. Thus, p is calculated in this work as follows:

$$p = \begin{cases} p - 0.3478 & p > 0 \\ 0 & p \leq 0 \\ 3 \times 0.3478 & \text{when jumping out successfully} \end{cases} \tag{15}$$

4.4. **Procedure of IGOA.** This paper proposed an Improved Grasshopper Optimization Algorithm (IGOA). The procedure of the IGOA is divided into four stages which were the initialization stage, the evolution stage, the fitness updating stage and the jumping stage.

In the initialization stage, the parameters are set, and the original positions of all the search agents are initialized randomly. The best target position and the corresponding fitness are also calculated in this part. The search loop starts to work in the evolution stage. Every search agent moves towards the target position by Equation (14). The nonlinear comfort zone parameter m is set by Equation (7). After that, every search agent conducts Lévy flight in a particular probability by Equation (11) and a new position is generated. In the updating stage, the fitness of the new position is calculated. If the new fitness is better than the global fitness, the new position can replace the old global target. If the new fitness is not more optimal than the global target, it comes to the jumping stage. In this stage, the search agent tries to jump out of the local optimum by Equation (13) and a new fitness is calculated. If the new fitness is better than personal fitness, the new position can replace the old personal position. Parameter p is updated by Equation (15) as well. So far, one iteration in the loop is finished. After the maximum number of iterations is reached, the loop ends, and the fitness and the target position are presented as the final result. The pseudo code of the IGOA is shown as Algorithm 1. The figure framework of the procedure of IGOA is shown as Figure 1.

Algorithm 1 Improved Grasshopper Optimization Algorithm

```
1: initialize the parameters
2: initialize the swarm position with random matrix
3: calculate the original target fitness and mark the target position
4: while ($iter < MaxIteration$ and $target\ fitness > destination\ fitness$) do
5:    set the nonlinear comfort zone parameter $m$ by Equation (7)
6:    update $x_i$ by Equation (14)
7:    $x_i$ conducts Lévy flight by Equation (11)
8:    calculate the fitness
9:    if current fitness is better than the target fitness then
10:      update the target fitness and the target position
11:   else
12:      $x_i$ jumps out by Equation (13)
13:      calculate the fitness
14:      if current fitness is better than the personal fitness then
15:         update the personal position
16:      end if
17:      set parameters $p$ by Equation (15)
18:   end if
19: end while
20: Return target fitness and target position
```

5. **Experimental Results on Benchmarks.**

5.1. **Experimental setup.** To evaluate the performance of the proposed IGOA, a series of experiments were conducted. In this work, IGOA was compared with 6 meta-heuristic algorithms including the original Grasshopper Optimization Algorithm (GOA), the Opposition-Based Learning GOA (OBLGOA), three recently proposed algorithms, Whale Optimization Algorithm (WOA), Dragonfly Algorithm (DA) and Ant Lion Optimizer (ALO), and a classical heuristic algorithm, Particle Swarm Optimization (PSO). The parameters of the other 6 algorithms compared with IGOA were set as [1, 9, 14, 15, 16, 17] described.

In this part, 29 well-known benchmark functions were used to test the search ability of the proposed IGOA. The benchmarks are divided into 3 types which can evaluate the different capabilities of the algorithms. Benchmarks F_1-F_7 listed in Table 1 are unimodal functions with only one optimal location which can estimate the ability of exploitation. Benchmarks F_8-F_{23} listed in Table 2 and Table 3 are multimodal functions with several

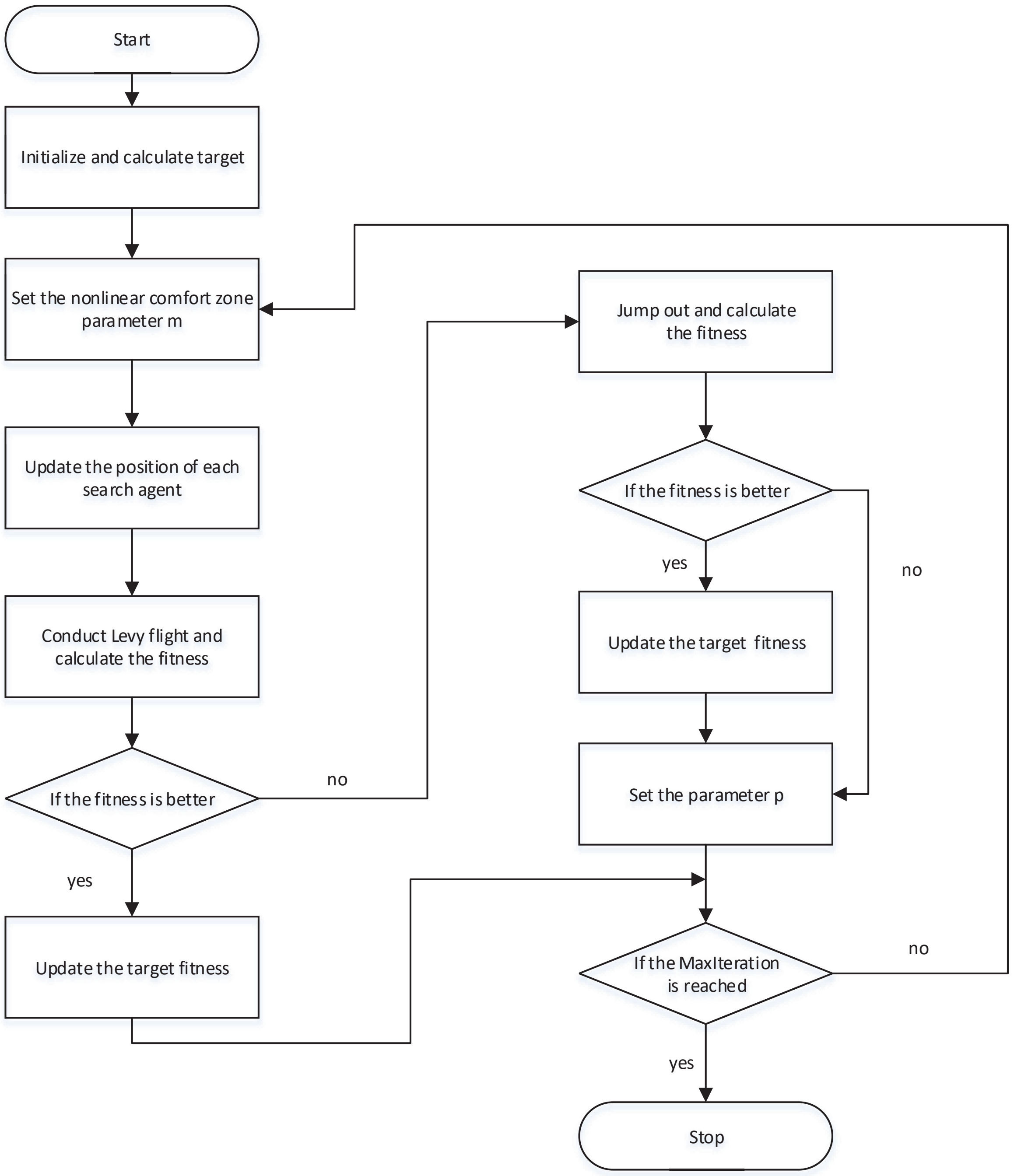
Start
Initialize and calculate target
Set the nonlinear comfort zone parameter m
Update the position of each search agent
Conduct Levy flight and calculate the fitness
If the fitness is better
yes
no
Update the target fitness
Jump out and calculate the fitness
If the fitness is better
yes
no
Update the target fitness
Set the parameter p
If the MaxIteration is reached
no
yes
Stop

FIGURE 1. The figure framework of the procedure of IGOA

local optimums which can evaluate the exploration capability [1]. Benchmarks F_{24}-F_{29} listed in Table 4 are composite functions [22] which combine some basic test functions under a framework and are more complicated. They can evaluate the performance of getting out of local optimum. In Tables 1-4, Dim represents the dimension of the benchmark functions, Range is the search boundary of the optimization problems, and $f_{\min}$ is the optimal fitness of the functions.

A series of experiments related to the 29 benchmark functions described above were conducted with Matlab code. For F_1-F_{23}, each algorithm was evolved for 500 iterations with 30 search agents to make a complete search process. For F_{24}-F_{29}, a search process

TABLE 1. Unimodal functions

Function	Dim	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0

contained 100 iterations. Each search process would be repeated 30 times for every algorithm to eliminate contingency and some statistical data, such as average (avg), standard deviation (std), best fitness of 30 (best) and worst fitness of 30 (worst), was calculated to compare the performance of the algorithms. Besides, the Wilcoxon rank-sum test was conducted, and the p-value was calculated to demonstrate the statistical significance of the results.

5.2. **Evaluation of exploitation capability.** F_1-F_7 are unimodal functions with only one global optimum, which can test the exploitation capability of the algorithms. If an algorithm has a superior ability of exploitation, it can search more accurately and find a solution closer to the global optimum.

The results on F_1-F_7 are shown in Table 5. It could be seen that IGOA could get the best average results in F_5-F_7, and in F_1-F_4 IGOA behaved only worse than WOA. As for std and worst value, IGOA could also perform better than the other algorithms in F_3-F_7, which indicates that the proposed algorithm can reduce the probability of getting terrible solutions and promote the stability of the algorithm. Compared with GOA and OBLGOA, the proposed IGOA can significantly improve the exploitation capability of the original algorithm.

5.3. **Evaluation of exploration capability.** F_8-F_{23} are multimodal functions with several local optimums, which can test the exploration capability of the algorithms. If an algorithm cannot do well in exploration, the search will most likely fall into local optimum when dealing with the multimodal functions and even the best ability of exploitation cannot help. A wrong direction can probably lead to a wrong result.

The results on F_8-F_{23} are presented in Table 6 and Table 7. It can be found that the proposed IGOA can get the best average fitness in 11 of 16 benchmark tests and in F_{10}, F_{20} and F_{23} IGOA can get the second best result. The results about standard deviation show that the stability of IGOA might not be as excellent as it shows in average fitness, but it still is the best of all the 7 algorithms in most of the tests. The results on best

TABLE 2. Multimodal functions-1

Function	Dim	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500, 500]$	$-418 \times Dim$
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i-1}^{n} u(x_i, 10, 100, 4) + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_x, 5, 100, 4)$	30	$[-50, 50]$	0

fitness and worst fitness can show that the proposed IGOA has the most reliable ability to find the best solution in almost all the benchmark tests and get the minimum probability to find the worst solution. In comparison with the original GOA and OBLGOA, the proposed IGOA can naturally enhance the performance of exploration of the algorithm a lot.

5.4. **Evaluation of capability of getting rid of local optimum.** F_{24}-F_{29} are the composite test functions, which combine some basic benchmark functions under a particular framework to construct new controllable test functions. The composite benchmark

Table 3. Multimodal functions-2

Function	Dim	Range	$f_{\min}$
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65, 65]$	0
$F_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_i(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 5]$	0.3979
$F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	$[-2, 2]$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	$[1, 3]$	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 1]$	-3.32
$F_{21}(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.4028
$F_{23}(x) = -\sum_{i=1}^{1} 0\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.5363

functions are more complicated and more challenging than multimodal test functions, and they are more convincing when evaluating the search capability of an algorithm.

According to the results on F_{24}-F_{29} listed in Table 8, it can be seen that the proposed IGOA can be very competitive compared with the other algorithms. In F_{24} and F_{26} IGOA can get the best fitness and in F_{25}, F_{27} and F_{29} IGOA can get the second-best results. IGOA cannot perform best nor worst in the aspect of the standard deviation on all the benchmark tests. IGOA can perform best in 4 tests and 3 tests respectively on best value and worst value. Although IGOA can perform generally in terms of standard deviation, it can outperform other algorithms in the potency of finding a better solution and controlling risk. Compared with the original GOA and the OBLGOA, IGOA performs far better. According to the results on composite function tests, it can be demonstrated that IGOA has the ability to handle such complex and challenging problems.

5.5. **Significance of the results.**

The comparison based on average value and standard deviation for 30 independent operations did not compare the difference between

TABLE 4. Composite functions

Function	Dim	Range	$f_{\min}$
$F_{24}(CF1)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	30	$[-5, 5]$	0
$F_{25}(CF2)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function, $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	30	$[-5, 5]$	0
$F_{26}(CF3)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function, $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1, 1, 1, \ldots, 1]$	30	$[-5, 5]$	0
$F_{27}(CF4)$ f_1, f_2 = Ackley's Function, f_3, f_4 = Rastrigin's Function, f_5, f_6 = Weierstrass Function, f_7, f_8 = Griewank's Function, f_9, f_{10} = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 5/32, \ldots, 5/32]$	30	$[-5, 5]$	0
$F_{28}(CF5)$ f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley's Function, f_9, f_{10} = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, \ldots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}]$ $= [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	30	$[-5, 5]$	0
$F_{29}(CF6)$ f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley's Function, f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}]$ $= [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100,$ $0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	30	$[-5, 5]$	0

each operation. It was still possible that the results of the experiment contained certain contingency. To dispel this contingency and demonstrate the significance of the results of the experiment, the Wilcoxon rank-sum test was introduced in this work. Wilcoxon rank-sum test is a nonparametric test of the null hypothesis, and it is used to determine whether two independent datasets were from the same distributed population. In this work, p-values about the statistical data between IGOA and each of the other algorithms on F_1-F_{29} were calculated. When p-value is less than 0.05, it can be considered that the difference between the two samples is significant.

From Table 9 it can be seen that the p-values between IGOA and another compared algorithm in most of the benchmark tests are less than 0.05. Some p-values are more than 0.05 in some of the test functions which have a higher probability of achieving the same optimal solution by different algorithms, such as F_{14}, F_{19}, F_{20}, F_{21}, F_{22}, and F_{23}. By analyzing the results of the composite functions, p-values are more than 0.05 between

Table 5. Results of unimodal functions

Function	Type	IGOA	GOA	WOA	DA	ALO	PSO	OBLGOA
F_1	avg	3.3538E-15	0.8386	**1.0082E-71**	0.0012	17.1234	6.6310E-06	2.77E-05
	std	1.9129E-15	0.8473	**5.3671E-71**	0.0008	17.8648	2.1612E-05	1.57E-05
	best	8.0120E-16	0.0683	**5.0175E-87**	0.00010	2.4120	1.5608E-07	7.75E-06
	worst	9.1885E-15	4.4591	**2.9415E-70**	0.0030	78.3488	0.0001	6.68E-05
F_2	avg	2.4358E-08	10.2444	**1.5782E-51**	47.0419	4.9289	0.0953	0.0136
	std	1.0173E-08	22.2516	**4.8668E-51**	43.4815	3.7734	3.0196E-01	0.0377
	best	9.0029E-09	0.0290	**2.5368E-57**	3.8197	1.4484	0.0003	0.0011
	worst	5.7194E-08	79.1046	**2.1838E-50**	120.4026	21.3472	1.6419	0.1505
F_3	avg	0.0121	1789.3452	43942.9825	4632.0793	1154.2060	227.7776	**0.0061**
	std	0.0211	1030.4488	1.6119E+04	2008.3302	1332.5907	81.0377	**0.0021**
	best	**8.6626E-12**	450.4535	17269.6957	1883.2010	249.1874	127.3043	0.0020
	worst	0.0983	4603.9086	85296.2126	10156.9819	5729.0798	425.9704	**0.0103**
F_4	avg	0.0257	9.7756	56.3543	16.9378	31.4847	3.2311	**0.0165**
	std	0.0182	3.5013	25.3903	4.3129	8.2314	1.1774	**0.0081**
	best	**9.7116E-07**	3.0335	3.2199	6.5808	17.7108	1.4918	0.0010
	worst	0.0694	19.5647	89.1869	57.2014	48.8229	5.5785	**0.0297**
F_5	avg	**26.4488**	965.6578	28.1591	348.5174	1615.4578	61.9257	28.3790
	std	**0.3046**	1572.3600	0.4797	553.5976	2814.6516	64.9991	0.3087
	best	25.8503	25.6988	27.2726	28.4537	143.9488	**1.7275**	27.6749
	worst	**27.0365**	7522.9967	28.7708	2223.6927	14636.9499	268.0065	28.7678
F_6	avg	**1.4451E-06**	0.8997	0.3866	0.0023	20.5677	2.2973E-05	1.2542
	std	**4.8437E-07**	2.0343	0.2498	0.0055	29.2793	5.0099E-05	0.4237
	best	**5.2803E-07**	0.0203	0.0856	0.0001	3.0268	1.5390E-07	0.6616
	worst	**2.5210E-06**	11.0963	1.0626	0.0306	148.1366	0.0002	2.1687
F_7	avg	**0.0010**	0.0234	0.0032	0.2504	0.1588	0.0274	0.0014
	std	**0.0014**	0.0106	0.0032	0.0768	0.0934	0.0113	0.0007
	best	**1.0746E-05**	0.0090	5.1138E-05	0.1003	0.0467	0.0115	0.0006
	worst	0.0072	0.0602	0.0118	0.4105	0.3750	0.0538	**0.0033**

Table 6. Results of multimodal functions-1

Function	Type	IGOA	GOA	WOA	DA	ALO	PSO	OBLGOA
F_8	avg	−7594.1662	−7728.4324	**−9969.6875**	−6189.11	−7320.0609	−6688.3058	−7901.9216
	std	767.0277	**593.4825**	1919.0327	1833.8338	886.1388	684.8647	591.8701
	best	−9009.3177	−8903.0523	−12564.8051	**−12568.5831**	−9628.8472	−7869.7845	−9598.7278
	worst	−5993.9317	−6468.5651	−5709.2023	−5417.6748	−6101.1009	−5224.2865	**−6823.4703**
F_9	avg	**0.0000**	9.4853	0.4119	8.8883	7.4670	4.5109	1.7919
	std	**0.0000**	5.4604	1.6505	4.5100	4.1855	2.8231	2.1937
	best	**0.0000**	1.9899	0.0000	0.9950	0.9959	0.9950	4.42E-09
	worst	**0.0000**	27.8586	8.1471	16.9143	16.9159	10.9445	6.9648
F_{10}	avg	1.30E-08	3.0892	**4.56E-15**	5.0040	9.9207	1.3594	0.0010
	std	3.13E-09	0.8305	**2.18E-15**	3.1845	3.9783	0.8583	0.0002
	best	8.58E-09	1.5021	**8.88E-16**	1.1582	3.3950	0.0002	0.0005
	worst	1.97E-08	4.5855	**7.99E-15**	12.3302	16.3216	2.8857	0.0015
F_{11}	avg	**5.50E-15**	0.6966	0.0069	0.0610	1.1803	0.0258	0.0002
	std	**5.71E-15**	0.1924	0.0379	0.0268	0.1822	0.0354	8.83E-05
	best	**6.66E-16**	0.3226	0.0000	0.0068	1.041	9.58E-07	7.36E-05
	worst	**2.52E-14**	1.0411	0.2076	0.1151	1.9360	0.1590	0.0004
F_{12}	avg	**8.93E-08**	5.6658	0.0242	14.9630	25.6688	0.5808	0.0347
	std	**3.28E-08**	2.3767	0.0162	7.2414	14.1457	0.8898	0.0211
	best	**4.62E-08**	1.8994	0.004	6.9778	6.3337	1.89E-07	0.0051
	worst	**1.85E-07**	9.7664	0.0832	35.5319	55.4036	3.4350	0.1028
F_{13}	avg	**0.0138**	8.7624	0.6039	24.9054	261.6986	0.2117	0.4087
	std	**0.0298**	9.0372	0.2536	15.4205	1186.8265	0.5112	0.2177
	best	**4.85E-07**	0.3005	0.1459	0.3854	1.3788	9.86E-07	0.1211
	worst	**0.0989**	35.2838	1.2995	56.9980	6534.7729	2.0239	1.1019

Table 7. Results of multimodal functions-2

Function	Type	IGOA	GOA	WOA	DA	ALO	PSO	OBLGOA
F_{14}	avg	3.5566	**0.9980**	2.7622	2.2142	1.7242	4.5076	3.0103
	std	2.9933	**6.4341E-16**	3.3587	2.1646	1.2701	3.0100	1.5672
	best	**0.9980**	0.9980	0.9980	0.9980	0.9980	0.9980	0.9980
	worst	10.7632	**0.9980**	10.7632	10.7632	5.9288	12.6705	5.9288
F_{15}	avg	**0.0003**	0.0071	0.0007	0.0032	0.0029	0.0038	0.0063
	std	**3.24E-05**	0.0089	0.0005	0.0062	0.0059	0.0076	0.0087
	best	**0.0003**	0.0007	0.0003	0.0006	0.0003	0.0003	0.0003
	worst	**0.0004**	0.0204	0.0022	0.0206	0.0204	0.0204	0.0210
F_{16}	avg	**−1.0316**	−1.0316	−1.03162	−1.0316	−1.0316	−1.0316	−1.0316
	std	**4.44E-16**	8.1630E-13	1.6137E-09	1.0917E-13	5.1620E-07	6.5195E-16	4.63E-06
	best	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
	worst	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
F_{17}	avg	**0.3979**	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	std	**0**	7.3750E-13	8.4140E-06	2.1005E-14	1.0879E-06	0.0000	1.82E-06
	best	**0.3979**	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	worst	**0.3979**	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
F_{18}	avg	**3.0000**	8.4000	3.0000	3.0000	3.0000	3.0000	3.0000
	std	4.11E-08	20.5503	7.0794E-05	6.2232E-13	7.7233E-06	**6.0036E-16**	3.07E-10
	best	**3.0000**	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	worst	**3.0000**	84.0000	3.0003	3.0000	3.0000	3.0000	3.0000
F_{19}	avg	**−3.8628**	−3.7288	−3.8582	−3.8628	−3.8628	−3.8628	−3.8628
	std	1.67E-10	0.3062	0.0054	2.4040E-13	2.2464E-05	**2.6402E-15**	2.95E-05
	best	**−3.8628**	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628
	worst	**−3.8628**	−2.7847	−3.8408	−3.8628	−3.8627	−3.8628	−3.8627
F_{20}	avg	−3.2863	**−3.2943**	−3.2364	−3.2621	−3.2819	−3.2783	−3.2295
	std	0.0554	**0.0511**	0.1007	0.0610	0.0577	0.0584	0.1062
	best	**−3.3220**	−3.3220	−3.3213	−3.3220	−3.3220	−3.3220	−3.3220
	worst	**−3.2031**	−3.2031	−3.0184	−3.1981	−3.1974	−3.1996	−3.0334
F_{21}	avg	**−8.2870**	−6.0675	−8.5287	−5.8753	−6.6400	−5.9796	−7.2158
	std	**2.4947**	3.6827	2.7585	3.0237	3.2604	3.3646	3.3058
	best	**−10.1532**	−10.1532	−10.1498	−10.1532	−10.1532	−10.1532	−10.1532
	worst	**−5.0552**	−2.6305	−2.6292	−2.6305	−2.6305	−2.6305	−2.6303
F_{22}	avg	**−9.3413**	−7.1190	−7.2448	−7.1785	−5.1249	−5.5091	−8.0724
	std	**2.1597**	3.6556	3.0850	3.3668	2.5737	3.1769	3.1972
	best	**−10.4029**	−10.4029	−10.4020	−10.4029	−10.4029	−10.4029	−10.4029
	worst	**−5.0877**	−1.8376	−1.8352	−1.8376	−2.7517	−1.8376	−2.7658
F_{23}	avg	−7.9281	−4.9279	−6.5554	−6.6569	−5.4597	−4.7287	**−8.0791**
	std	**2.8536**	3.2796	3.3755	3.7558	3.0183	3.2997	3.5845
	best	**−10.5364**	−10.5364	−10.5348	−10.5364	−10.5364	−10.5364	−10.5364
	worst	**−3.8354**	−2.4217	−1.6721	−2.4217	−2.4217	−2.4273	−2.4217

IGOA and GOA in F_{25} and F_{28} where GOA gets the best fitness and IGOA does not. Considering comprehensively, it can still be demonstrated that the proposed IGOA can significantly promote the performance of GOA.

5.6. **Evaluation of convergence rate.** The convergences rates of the algorithms are discussed in this part. The convergence curves of IGOA and all the other 6 algorithms for comparison over part of the benchmark functions are shown in Figure 2. The horizontal axis is the number of the current iterations, and the vertical axis is the best fitness value obtained so far. To make the curve more distinguishable, the logarithm is used in the vertical axis. The curve image in Figure 2 indicates that IGOA can keep a large slope relatively for the entire search process in most of the search processes. This phenomenon can mean that the proposed nonlinear comfort zone parameter can contribute to making

Table 8. Results of composite functions

Function	Type	IGOA	GOA	WOA	DA	ALO	PSO	OBLGOA
F_{24}	avg	**94.0428**	135.9242	341.2250	1098.8600	189.5840	326.5899	189.6304
	std	130.6690	127.0964	164.2449	**99.4449**	117.6155	149.6609	83.2598
	best	**3.3965**	29.6057	163.2687	796.3368	72.0563	118.9426	92.1210
	worst	504.8393	514.7878	844.9761	1261.1685	554.4573	717.0723	**411.3461**
F_{25}	avg	324.3197	**284.6066**	521.9304	1211.2487	413.5431	393.6293	472.0768
	std	151.6351	163.1829	130.9867	**94.2232**	156.9269	126.4408	143.6411
	best	**21.2736**	28.9418	244.2571	1006.3367	72.6341	161.9665	66.0628
	worst	**497.3555**	510.3606	673.6729	1354.7747	662.1609	610.8645	606.4133
F_{26}	avg	**525.7384**	624.5092	1052.0733	1547.6195	884.9306	608.4939	875.4033
	std	128.5247	176.1401	155.6153	155.8894	157.8576	**103.4249**	151.2467
	best	317.4166	**200.8044**	849.0492	1064.4776	644.6867	326.5611	627.7263
	worst	900.011	1196.9114	1351.5909	1710.9641	1222.5593	**828.4274**	1217.9587
F_{27}	avg	894.6861	945.6744	902.6182	1421.5043	929.5948	**757.5836**	895.7587
	std	29.1104	101.1013	**15.0373**	46.9584	128.2259	114.6504	34.7088
	best	740.5569	638.304	896.3862	1319.5644	650.1341	**538.8135**	719.3590
	worst	**900.0053**	1030.5605	982.1588	1519.1409	1066.455	1012.7758	953.3374
F_{28}	avg	304.9727	**142.0615**	456.9052	1354.378	194.2336	303.9595	497.1264
	std	367.9136	**102.2128**	255.4945	127.0279	182.4486	143.7137	351.6947
	best	**46.3796**	54.2782	179.8317	992.2916	87.2393	113.7433	110.5681
	worst	900.0040	**435.4893**	900.0000	1505.6248	1025.2583	675.3012	900.0049
F_{29}	avg	900.0003	907.4664	**900.0000**	1371.4241	925.7089	931.1013	900.0004
	std	0.0001	5.1380	**0.0000**	56.4060	7.5901	19.0467	0.0002
	best	**900.0000**	901.0860	900.0000	1254.6648	913.0986	910.3009	900.0001
	worst	900.0006	920.7888	**900.0000**	1000.7252	1000.7252	1000.7252	900.0009

full utilization of every iteration. The curve span is ample in most of the functions, which can suggest that the local search mechanism based on Lévy flight makes the algorithm more creative. Some sudden drops of the curve occurring frequently in F_3, F_4, F_7, and F_{28} show that the random jumping strategy can help the search jump out of the local optimum. The convergence curves demonstrate that the proposed IGOA can make the search process converge more rapidly than others generally.

6. **Application on Task Scheduling Problems.**

Task scheduling problem is a common problem in cloud computing area [21], edge computing area [23] and SEA Service System [24]. Massive users produce massive tasks. Thus, the system should handle the tasks efficiently when a large number of users ask the service at the same time, which can be an essential problem of the computing system, especially in resources constrained conditions. In task scheduling problems, a series of tasks arriving at the scheduling center wait to be allocated to a particular execution node.

The execution node might be heterogeneous in this problem because of the difference in the resource condition, workload, processing speed and the type of task. A different solution to task scheduling could lead to different consequences, and an excellent solution can affect a lot. For service providers, allocating a task to the proper node can save energy and reduce the budget. For service consumers, scheduling the task to the efficient node can decrease the waiting time and improve the user experience.

Task scheduling problem can be an optimization problem, and it is an NP-hard problem [25]. Many algorithms are applied to solving the task scheduling problem. Some algorithms based on Best Resource Selection (BRS) such as Max-Min, Min-Min, and Sufferage are traditional methods to solve task scheduling problems [21]. The dimension of the tasks can also increase when the number of tasks increases, which can enlarge the

Table 9. Results of Wilcoxon rank-sum test

Function	GOA	WOA	DA	ALO	PSO	OBLGOA
F_1	3.02E-11	3.02E-11	3.02E-11	3.02E-11	9.53E-07	2.03E-07
F_2	3.02E-11	3.02E-11	3.02E-11	3.02E-11	7.12E-09	2.37E-10
F_3	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	5.57E-10
F_4	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.21E-10
F_5	1.86E-09	1.60E-07	3.02E-11	3.02E-11	0.5201	5.07E-10
F_6	3.02E-11	3.02E-11	3.02E-11	0.0099	3.02E-11	3.02E-11
F_7	3.16E-10	0.7618	3.02E-11	3.02E-11	8.15E-11	0.0594
F_8	0.0030	3.82E-09	0.7958	2.59E-06	0.0133	0.0002
F_9	3.02E-11	1.24E-09	3.02E-11	3.02E-11	2.86E-11	4.98E-11
F_{10}	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.43E-08	3.83E-05
F_{11}	3.02E-11	4.56E-11	3.02E-11	3.02E-11	1.34E-05	3.02E-11
F_{12}	3.02E-11	3.02E-11	3.02E-11	3.02E-11	0.0002	3.02E-11
F_{13}	3.02E-11	3.02E-11	3.02E-11	3.02E-11	0.9117	3.02E-11
F_{14}	1.67E-05	0.1578	0.0265	0.0082	0.7842	0.6951
F_{15}	1.33E-10	1.07E-09	2.87E-10	2.87E-10	0.3328	2.15E-10
F_{16}	0.0064	6.72E-10	3.02E-11	1.81E-05	4.08E-12	3.02E-11
F_{17}	3.20E-06	3.01E-11	3.01E-11	6.78E-06	4.56E-12	3.01E-11
F_{18}	1.73E-07	3.02E-11	3.02E-11	0.01911	1.55E-11	5.09E-08
F_{19}	0.5997	0.002266	2.13E-05	3.33E-11	5.14E-12	0.0003
F_{20}	0.5395	0.0003	0.00250	0.7958	0.0003	0.0001
F_{21}	0.0002	1.25E-05	6.74E-06	0.0850	0.2822	5.46E-06
F_{22}	0.0005	5.27E-05	3.83E-06	0.5493	0.0627	0.0013
F_{23}	6.05E-07	3.83E-06	1.73E-06	0.1858	0.0009	5.27E-05
F_{24}	0.0016	6.01E-08	0.0001	3.02E-11	3.81E-07	0.0001
F_{25}	0.7618	1.09E-05	0.0138	3.02E-11	0.1907	1.02E-05
F_{26}	0.0046	5.49E-11	5.57E-10	3.02E-11	0.0038	4.20E-10
F_{27}	7.74E-06	7.34E-09	0.0064	3.02E-11	3.08E-08	2.68E-06
F_{28}	0.3711	0.0042	0.0083	3.02E-11	0.0073	0.0001
F_{29}	3.02E-11	2.14E-11	3.02E-11	3.02E-11	3.02E-11	0.0024

search space and make the optimization problem more complicated. The traditional algorithms could not handle this situation. In recent years many meta-heuristic algorithms were applied to solving the task scheduling problem. Genetic Algorithm (GA) proposed by Goldberg in 1988 is a famous evolutionary algorithm, and it represents a scheduling scheme as a chromosome to solve the task scheduling problem. GA is complicated to calculate because of its operations of crossover and mutation. Particle Swarm Optimization (PSO) is a classical meta-heuristic algorithm proposed by Kennedy and Eberhard in 1995. PSO represents a scheduling solution as a discrete particle and evolves by the information of the personal best and the global best of the swarm. PSO is easy to fall into local optimum, and it does not perform well when dealing with multimodal problems.

Some recently proposed meta-heuristic algorithms are used such as DA, WOA, and GOA. In this work, the proposed IGOA is employed for task scheduling problems. A solution of N tasks scheduled to M nodes is abstracted as an N-dimension discrete vector which represents a position in the search space. Each solution can correspond to a makespan and a budget of the system, and a fitness value can be calculated with the

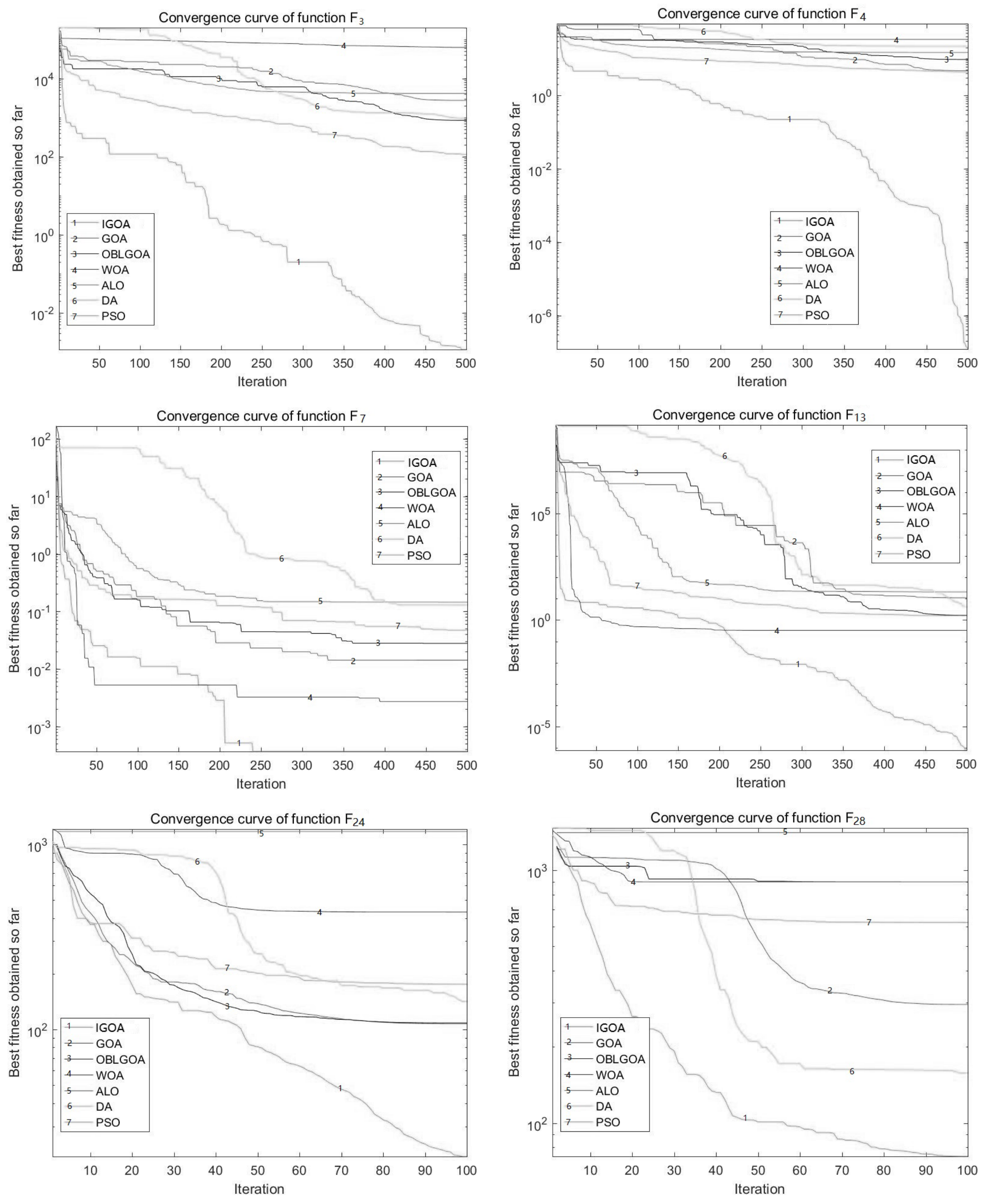

FIGURE 2. Convergence curves for all the 7 algorithms over some benchmark functions

makespan and budget. Several search agents search in the whole search space for the best fitness by the search strategy of IGOA.

6.1. **Task scheduling model.** In this paper, we assume that users start N tasks which are $\{T_1, T_2, \ldots, T_N\}$ and M nodes which are $\{S_1, S_2, \ldots, S_M\}$ are waiting to handle them. The task scheduling model is described as follows.

1) The set of N tasks is $\{T_1, T_2, \ldots, T_N\}$, and T_i represents the resource that the i-th task consumes.
2) The set of M nodes is $\{S_1, S_2, \ldots, S_M\}$ and S_j represents the maximum amount of resources that the j-th node can provide.
3) A task can be operated at any node as long as the node can provide enough resource, which means $T_i \leq S_j$.
4) Every node can handle only one task at the same time. Multiple tasks can be run sequentially on the same node.
5) There are K types of all the tasks. The vector of types of tasks is $tot[N]$, and tot_i which is an integral value in $[1, K]$ represents the type of the i-th task.
6) The ability of every node to handle different types of tasks is different. The matrix of operating speed is $mips[M, K]$. $mips_j^k$ represents the resource that the type k task can consume on the j-th node in a unit time. The working time for the i-th task running on the j-th node which is et_{ij} is calculated as follows.

$$et_{ij} = \begin{cases} \dfrac{T_i}{mips_j^{tot_i}} & \text{task } i \text{ runs on node } j \\ 0 & \text{task } i \text{ does not run on node } j \end{cases} \tag{16}$$

7) The makespan of the j-th node which is st_j is calculated by adding all the operating time of the tasks handled on the j-th node. The equation is shown as follows.

$$st_j = \sum_{i=1}^{N} et_{ij} \tag{17}$$

   The total *Makespan* is the longest time for all the nodes. *Makespan* is calculated as follows.

$$Makespan = \max\{st_j | j = 1, 2, 3, \ldots, M\} \tag{18}$$

8) The work node can produce the extra budget when it is working. The budget is only related to the time of working, regardless of the type of task running on it. The vector of the price of the nodes is $bps[M]$ which represents the budget that the j-th node generates in a unit time.
9) The total *Budget* is calculated by adding all the budget of all the nodes. *Budget* is calculated as follows.

$$Budget = \sum_{j=1}^{M} (st_j \times bps_j) \tag{19}$$

10) Makespan and budget can describe the cost of the problems in two aspects. A unified evaluation fitness is required. In this module, the fitness of the problem is defined as the mixture of makespan and budget with weight parameters of α and β. The *fitness* is calculated as follows.

$$fitness = \alpha Makespan + \beta Budget \tag{20}$$

11) The search process is continuous, but the solution to the problem is discrete. When a step of search is finished, the search agent should adjust itself to the nearest integral location.
12) The switching time between two tasks on the same node is ignored in this module. The transmission time among the edge nodes is also ignored.

6.2. **Results of IGOA on task scheduling problems.** In this task scheduling module, N tasks are divided into K types, and M working nodes are preparing to handle them. The specifics of the parameter selection of the task scheduling module depend on the specific problem and will not be discussed here. In this work, without loss of generality, K, N and M are set to 4, 10 and 5 respectively. To balance the impact of *Makespan* and *Budget* on *fitness*, the coefficients α and β are set to 0.7 and 0.3. Those parameters mentioned above are set as follows.

$$K = 4;\ N = 10;\ M = 5;\ \alpha = 0.7;\ \beta = 0.3 \tag{21}$$

The vector T, tot, S and bps, and the matrix mips are set as Tables 10-12.

Table 10. Resources and types about tasks

Task	1	2	3	4	5	6	7	8	9	10
T	40	20	30	40	25	35	45	10	5	10
tot	1	1	1	2	2	2	3	3	4	4

Table 11. Resource and bps about nodes

Node	1	2	3	4	5
S	100	50	150	100	80
bps	0.5	0.2	0.8	0.1	0.5

Table 12. Mips about tasks and nodes

$mips_j^k$	k:1-4			
j:1-5	5	2	10	8
	2	4	2	1
	8	10	10	5
	2	1	3	4
	5	5	8	8

In this work, 6 algorithms were compared with the proposed IGOA. 30 search agents were employed in each algorithm. In order to reduce the impact of accidental factors, every algorithm was tested for 30 times and the statistical data such as average value, standard deviation, best value and worst value were calculated and compared. The result of the experiment is listed as Table 13.

Table 13. Results about task scheduling

Algorithm	average	std	best	worst	promotion
IGOA	13.50	0.62	12.63	14.62	0
GOA	14.95	0.82	13.50	16.67	9.7%
WOA	14.08	0.75	13.02	15.93	4.1%
ALO	19.83	2.89	16.02	26.76	31.9%
DA	19.00	2.49	15.62	26.81	29.0%
PSO	17.93	1.68	13.96	22.29	24.7%
OBLGOA	14.85	0.74	13.30	16.27	9.1%

The result of the experiment demonstrated that the proposed IGOA outperformed the other algorithms in all fields. IGOA can do best in average values, and it can be more stable for the smallest standard deviation. It can have the ability to find the best solution over all the algorithms, and it can control the worst value which it can find. Compared with the other algorithms, IGOA can promote the average values respectively by 9.7%, 4.1%, 31.9%, 29.0%, 24.7%, and 9.1%.

7. Conclusion.

Grasshopper Optimization Algorithm (GOA) is a novel meta-heuristic algorithm inspired by the natural behavior of grasshoppers for solving single-objective numeric optimization problems. In this paper, an Improved Grasshopper Optimization Algorithm (IGOA) was proposed to enhance the performance of the original algorithm. The nonlinear comfort zone parameter can help the algorithm to make full utilization of all the iterations. A local search mechanism based on Lévy flight is introduced to give vision and creativity to the algorithm. The random jumping strategy can contribute to getting rid of local optimal.

29 benchmark functions were used to test the search ability of the proposed IGOA, and several statistical data were estimated. Then IGOA was used to handle the task scheduling problem. The performance of IGOA was compared with the original algorithm of GOA, OBLGOA, new meta-heuristic algorithms of DA, WOA and ALO and a classical meta-heuristic algorithm of PSO. The experimental results demonstrate that the proposed IGOA can efficiently improve the performance of the algorithm in aspects of accuracy, stability, convergence rate, the ability to find the best solution and the ability of controlling the worst solution. The p-value of the Wilcoxon rank-sum test can illustrate that the promotion is significant. When applied to task scheduling problems, IGOA can promote the performance of the original GOA and OBLGOA by 9.7% and 9.1% respectively.

Our future work will focus on further developing the ability of the algorithm to handle more complex optimization problems. Additionally, modifying the task scheduling module and adding more constrained conditions to make it more practical are also what concerns us.

Acknowledgment. This work is partially supported by Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research and System Development (Project No. XDC02010701) and Innovation Youth Talent Project of Institute of Acoustics of Chinese Academy of Sciences: Research on Key Technologies of Embedded Container File System (Project No. Y754061601). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] S. Saremi, S. Mirjalili and A. Lewis, Grasshopper optimisation algorithm, *Advances in Engineering Software*, vol.105, pp.30-47, 2017.

[2] S. Łukasik, P. A. Kowalski, M. Charytanowicz and P. Kulczycki, Data clustering with grasshopper optimization algorithm, *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp.71-74, 2017.

[3] N. Rajput, V. Chaudhary, H. M. Dubey and M. Pandit, Optimal generation scheduling of thermal system using biologically inspired grasshopper algorithm, *The 2nd International Conference on Telecommunication and Networks (TEL-NET)*, 2017.

[4] Z. Elmi and M. Ö. Efe, Multi-objective grasshopper optimization algorithm for robot path planning in static environments, *IEEE International Conference on Industrial Technology (ICIT)*, pp.244-249, 2018.

[5] C. M. Fonseca and P. J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, *Electronic Commerce*, vol.3, no.1, pp.1-16, 1995.

[6] D. Whitley, A genetic algorithm tutorial, *Statistics and Computing*, vol.4, no.2, pp.65-85, 1994.

[7] R. Tanese, *Distributed Genetic Algorithms for Function Optimization*, Ph.D. Thesis, Ann Arbor, MI, USA, 1989.

[8] N. A. AL-Madi, K. A. Maria, E. A. Maria and M. A. AL-Madi, A structured-population human community based genetic algorithm (HCBGA) in a comparison with both the standard genetic algorithm (SGA) and the cellular genetic algorithm (CGA), *ICIC Express Letters*, vol.12, no.12, pp.1267-1275, 2018.

[9] J. Kennedy and R. C. Eberhart, Particle swarm optimization, *IEEE International Conference on Neural Networks*, vol.4, pp.1942-1948, 1995.

[10] C.-J. Liao, C.-T. Tseng and P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, *Computers & Operations Research*, vol.34, pp.3099-3111, 2007.

[11] B. Gomathi and K. Krishnasamy, Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing environment, *International Journal of Advanced Computer Science and Applications*, vol.55, no.1, pp.12-16, 2013.

[12] M. Dorigo and L. M. Gambardella, Ant colonies for the travelling salesman problem, *BioSystems*, vol.43, no.2, pp.73-81, 1997.

[13] H. Hao, Y. Jin and T. Yang, Network measurement node selection algorithm based on parallel ACO algorithm, *Journal of Network New Media*, vol.7, no.1, pp.7-15, 2018.

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

[15] S. Mirjalili and A. Lewis, The whale optimization algorithm, *Advances in Engineering Software*, vol.95, pp.51-67, 2016.

[16] 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.

[17] A. A. Ewees, M. A. Elaziz and E. H. Houssein, Improved grasshopper optimization algorithm using opposition-based learning, *Expert Systems with Applications*, 2018.

[18] S. Arora and P. Anand, Chaotic grasshopper optimization algorithm for global optimization, *Neural Computing and Applications*, pp.1-21, 2018.

[19] X. Yang and S. Deb, Cuckoo search via Lévy flights, *Nature and Biologically Inspired Computing*, pp.210-214, 2009.

[20] L. Ying, Y. Zhou and Q. Luo, Lévy flight trajectory-based whale optimization algorithm for global optimization, *IEEE Access*, vol.5, no.99, pp.6168-6186, 2017.

[21] N. Qiao, J. You, Y. Sheng, J. Wang and H. Deng, An efficient algorithm of discrete particle swarm optimization for multi-objective task assignment, *IEICE Trans. Information and Systems*, vol.12, pp.2968-2977, 2016.

[22] J.-J. Liang, P. N. Suganthan and K. Deb, Novel composition test functions for numerical global optimization, *Proc. of Swarm Intelligence Symposium*, pp.68-75, 2005.

[23] J. Guo, Z. Song, Y. Cui, Z. Liu and Y. Ji, Energy-efficient resource allocation for multi-user mobile edge computing, *IEEE Global Communications Conference*, pp.1-7, 2017.

[24] J. Wang, J. Tian, J. You, X. Liu and H. Deng, Research and design of an on-site, elastic, autonomous network – SEA Service System, *Scientia Sinica Informationis*, vol.45, no.10, 2015.

[25] M. A. Tawfeek, A. El-Sisi, A. E. Keshk and F. A. Torkey, Cloud task scheduling based on ant colony optimization, *The 8th International Conference on Computer Engineering & Systems (ICCES)*, pp.64-69, 2013.