Received June 24, 2019, accepted August 8, 2019, date of publication August 19, 2019, date of current version August 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936254

Lévy Flight Shuffle Frog Leaping Algorithm Based on Differential Perturbation and Quasi-Newton Search

XINMING ZHANG[1], **ZIHAO FU**[1], **HAIYAN CHEN**[2], **WENTAO MAO**[1], **SHANGWANG LIU**[1], **AND GUOQI LIU**[1]

[1]College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China
[2]Department of Gynecological Tumor, Hubei Cancer Hospital, Wuhan 430079, China

Corresponding author: Haiyan Chen (cleverbear88@126.com)

This work was supported by the Key Research Projects of Higher Education Institutions of Henan Province, China, under Grant 19A520026.

ABSTRACT Lévy flight Shuffle Frog Leaping Algorithm (LSFLA) is a SFLA variant and enhances the performance of SFLA largely, however, it still has some defects, such as poor convergence and low efficiency. So an improved LSFLA, namely, LSFLA based on Differential perturbation and Quasi-Newton search (DQLSFLA), is proposed in this paper. Firstly, the way of updating only one solution which is the worst one at every sub-iteration in LSFLA is replaced with an all-solution updating way in the subgroup to improve the probability of obtaining the best solution, to omit one sorting step at every sub-iteration and the sub-iteration number parameter setting, to reduce the computational complexity and to enhance the optimization efficiency. Secondly, a random differential perturbation approach and an improved Lévy flight updating one are created and used in the subgroup updating to form DLSFLA, that is, the first half frogs in the subgroup use the random differential perturbation updating approach to improve the global search ability, and the last half adopt the improved Lévy flight updating method to keep the advantage of the Lévy flight updating method and overcome its defects. Finally, quasi-Newton local search is integrated into DLSFLA near the end of the algorithm to improve the convergence speed. Experimental results on the complex functions from CEC-2014 show that DQLSFLA's convergence quality and optimization efficiency are much better than LSFLA's and that compared with quite a few other state-of-the-art algorithms, DQLSFLA has better performance. The results on medical image enhancement and Quadratic Assignment Problem (QAP) also show that DQLSFLA can solve the real word optimization problems better than LSFLA can.

INDEX TERMS Intelligent optimization algorithm, shuffled frog leaping algorithm (SFLA), Lévy flight, quasi-Newton search, image enhancement, quadratic assignment problem.

I. INTRODUCTION

The Intelligent Optimization Algorithm (IOA) is an important research field of artificial intelligence, which is designed by simulating the mechanism of natural phenomena or biological survival, competition and natural selection and so forth to solve optimization problems [1], [2]. IOA is good at solving many real-world optimization problems [3]. Shuffled Frog Leaping Algorithm (SFLA) proposed by Eusuff *et al.* in 2003 [4] is a popular IOA. It makes full use of the advantages of both Memetic Algorithm (MA) and Particle Swarm Optimization (PSO), so it has some good characteristics such as simple concept and easy implementation [5], and it has attracted much attention and is used widely in many applications[5], [6], [7]. However, SFLA also has some defects, such as easy entrapment into local optima, slow convergence and high computational complexity [7]. Thus, many scholars have done a lot of work to cope with the problems and applied SFLA and its variants to science and engineering fields, which mainly focuses on three aspects.

1) Integration with other strategies and operators. Li *et al.* [8] proposed a Modified SFLA with Extremal Optimization (MSFLA-EO), and it enhanced the global

The associate editor coordinating the review of this article and approving it for publication was Sotirios Goudos.

search ability of SFLA by the jump step and inertia component and introduced EO to obtain good exploration ability. Ahandani and Alavi-Rad [9] proposed a SFLA based on the Opposition-Based Learning (OBL) and OBL was used to initialize the population to improve the quality of the candidate solutions and the OBL strategy was also used to diversify search moves of SFLA and accelerated SFLA without premature convergence in the search process. Liu *et al.* [10] combined the chaotic operator and OBL operator to produce the initial population. In addition, an adaptive nonlinear inertia weight and a perturbation strategy based on Gaussian mutation for global (local) best frog were performed to ensure the balance of exploration and exploitation. Liu *et al.* [11] presented a continuous optimization algorithm based on SFLA, which combined with the excellent characteristics of cloud model that enhanced the accuracy and convergence speed of SFLA. Sharma *et al.* [12] embedded a centroid mutation strategy into the leaping operator of SFLA to increase the diversity of population.

2) Hybridizing SFLA with other algorithms. Sun and Zhao [13] proposed a hybrid swarm leaping optimization algorithm based on PSO. The algorithm overcame the shortcomings of SFLA, such as easy falling into local optima and premature convergence. Roy *et al.* [14] proposed a variant of SFLA, it embedded the crossover operation of Genetic Algorithm (GA) in the updating process of SFLA's subgroup to improve the population diversity and avoid falling into local optima. Tang *et al.* [15] summarized the advantages and disadvantages of SFLA and MA, and proposed a novel MA called memetic frog leaping algorithm based on SFLA, and it was applied to the parameter selection of SVM.
3) Applications of SFLA and its variants to real-world problems. Jiang *et al.* [16] presented a Cauchy oscillation SFLA for updating the premise parameters of the adaptive neuro-fuzzy inference system to improve the learning ability and prediction accuracy of the algorithm for electrical resistivity imaging inversion. Pérez-Delgado [17] propsed a SFLA for color image quantization. Computational results indicate that the method can generate a quantized image with low computational cost. Moreover, the quality of the image generated is better than that of the images obtained by several well-known color quantization methods. Jiang *et al.* [18] used a hybrid LC (Lévy and Cauchy mutation) attractor to enhance the exploitation ability and a differential updating rule was used to enhance the exploration ability to form a novel SFLA. Then the novel SFLA was adopted for improving the learning ability and inversion quality of wavelet neural network. Zhang *et al.* [19] proposed an improved SFLA for solving the local backlight dimming problem and preserving the image quality perception with a certain low backlight power consumption.

Through unremitting efforts made by a lot of experts and scholars in the past years, the optimization performance of SFLA has been improved largely. With the increasing scale of problems in the real life and the strict real-time limitations, however, the requirement for IOAs is getting higher and higher, and better algorithms are needed to solve these practical problems, so it's important and meaningful to improve SFLA and its variants further [20].

In 2016, Tang *et al.* [21] presented a new SFLA variant, that is, SFLA based on Lévy flight Shuffled Frog Leaping Algorithm (LSFLA). The subgroup search mode in LSFLA is a random way of short distance search and occasional long distance search, in which the random search path obeys the Lévy distribution. The short distance search improved the local search ability of SFLA, and the occasional long distance search reduced the possibility of falling into local optima. In order to further improve the global search ability of SFLA, the differential mutation strategy was used after global information exchange. Although LSFLA improved the optimization performance of SFLA largely, it still has some problems, such as slow convergence, low search efficiency, and high computational complexity especially when it solves many complex problems.

To deal with the above problems mentioned, this paper presents an improved LSFLA, that is, LSFLA based on Difference perturbation and Quasi-Newton search (DQLSFLA). Firstly, an all-solution updating way in the subgroup replaces the updating way of one solution at each sub-iteration in SFLA. The new updating way reduces the computational complexity, saves setting one parameter, the sub-iteration number, to improve the operability of LSFLA. Secondly, a random differential perturbation updating method is proposed and used in the first half frogs in the subgroup. Thirdly the Lévy flight updating method is improved to apply to the last half frogs in the subgroup to form DLSFLA. In this way, the population diversity and the global search ability of LSFLA are improved. Finally, the quasi-Newton local search is merged into DLSFLA, that is, the quasi-Newton local search is used near the end of the algorithm to improve the convergence quality and the search precision of DLSFLA, thus DQLSFLA is obtained. The above improvements balance exploration and exploitation to enhance the whole optimization performance. The graphical summary of this paper is shown in Fig.1.

The rest of this paper is organized as follows: Section 2 reviews the related work. The proposed algorithm (DQLSFLA) is explained in detail in Section 3. A lot of experiments are presented to verify DQLSFLA in Section 4. Sections 5 and 6 describe DQLSFLA's application to image enhancement and Quadratic Assignment Problem (QAP). Section 7 gives conclusions and future work.

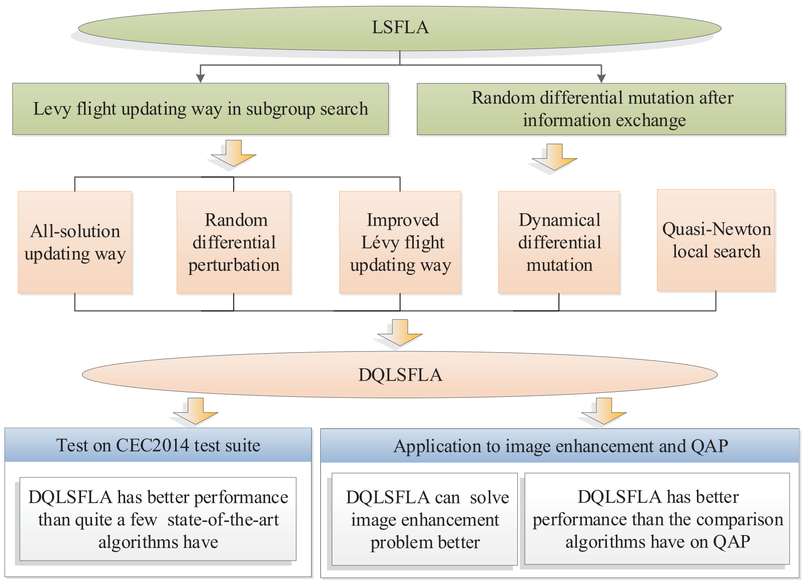

LSFLA
Levy flight updating way in subgroup search
Random differential mutation after information exchange
All-solution updating way
Random differential perturbation
Improved Lévy flight updating way
Dynamical differential mutation
Quasi-Newton local search
DQLSFLA
Test on CEC2014 test suite
DQLSFLA has better performance than quite a few state-of-the-art algorithms have
Application to image enhancement and QAP
DQLSFLA can solve image enhancement problem better
DQLSFLA has better performance than the comparison algorithms have on QAP

FIGURE 1. Graphical abstract of this paper.

II. RELATED WORK

A. SFLA

SFLA simulates the frog foraging process to search the best solution. In the optimization process, SFLA adopts a unique search way, and this way is the combination of intra-group search and global information exchange. The procedure of SFLA can be roughly divided into the following 4 steps.

Step 1: Parameter setting and population initialization

SFLA sets the parameters such as the number of subgroups m, the number of sub-iterations NE and so on, and randomly generates N frogs to form a population, and then calculate the fitness value of each frog.

Step 2: Grouping

According to the fitness values, all the frogs are sorted in descending order and divided into m subgroups. The first frog is assigned to the first group, the second frog to the second group, ..., the m-th frog to the m-th group, the m+1-th frog to the first group, and so on. Each group has n frogs.

Step 3: Intra-group search

The best frog and the worst one are found in the subgroup, denoted by $\boldsymbol{X}_b$ and $\boldsymbol{X}_w$, respectively. According to Eqs. (1) and (2), a new solution is generated and only one solution which is the worst in the subgroup is updated at a sub-iteration.

$$\boldsymbol{S} = rand * (\boldsymbol{X}_b - \boldsymbol{X}_w) \tag{1}$$

$$\boldsymbol{X}'_w = \boldsymbol{X}_w + \boldsymbol{S}, \quad \boldsymbol{S}_{min} \leq \boldsymbol{S} \leq \boldsymbol{S}_{max} \tag{2}$$

where $\boldsymbol{S}$ represents the updating step size of the frog; *rand* is a uniformly distributed random number in (0, 1); $\boldsymbol{S}_{min}$ and $\boldsymbol{S}_{max}$ are the maximum and minimum step sizes of the frog, respectively; $\boldsymbol{X}'_w$ represents the new position of the worst frog.

If the fitness value of $\boldsymbol{X}'_w$ is better than that of $\boldsymbol{X}_w$, $\boldsymbol{X}_w$is replaced by $\boldsymbol{X}'_w$; otherwise $\boldsymbol{X}'_w$ is generated by Eqs. (3) and (4).

$$\boldsymbol{S} = rand * \left(\boldsymbol{X}_g - \boldsymbol{X}_w\right) \tag{3}$$

$$\boldsymbol{X}'_w = \boldsymbol{X}_w + \boldsymbol{S}, \quad \boldsymbol{S}_{min} \leq \boldsymbol{S} \leq \boldsymbol{S}_{max} \tag{4}$$

where $\boldsymbol{X_g}$ is the best frog in the current population.

If the fitness value of $\boldsymbol{X}'_w$ is still not better than that of $\boldsymbol{X}_w$, then a new frog is randomly generated by Eq. (5) to replace $\boldsymbol{X}_w$.

$$\boldsymbol{X}'_w = \boldsymbol{a} + \boldsymbol{rand}(1, D) \otimes (\boldsymbol{b} - \boldsymbol{a}) \tag{5}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ represent the upper and lower boundary vectors of the decisive variables, respectively, and D represents the dimention of the optimization problem. $\boldsymbol{rand}(1, \boldsymbol{D})$ is a random vector of D components with each component between 0 and 1. $\otimes$ means an entrywise multiplication.

The subgroup is sorted to determine the worst frog and the best frog. The subgroup search is executed repeatedly until the predefined number of sub-iterations in each subgroup. Then the next subgroup search is executed. An intra-group search stops when all the subgroups finish their searches.

Step 4: Global information exchange

The global information exchange is the reorganization of all the subgroups into a population of N frogs. The reorganized population is sorted and then divided into several subgroups according to the previous method. The above intra-group search and global information exchange execute alternately until the termination criterion is reached, and the best solution is obtained.

The pseudo code of SFLA is shown as in Algorithm 1. In Algorithm 1, *nfe* represents the current number of function evaluations, the Maximum Number of Function Evaluations (*MNFE*) is denoted by *MNFE*. l is the number of sub-iterations

From Algorithm 1, SFLA has the following good characteristics. On one hand, SFLA approaches the best position under the guidance of X_b or X_g, so SFLA has some local search ability. On the other hand, when the two updating methods do not get a better solution, a random solution is used to replace the worst solution in the subgroup so SFLA has some global search ability. In the framework of SFLA, the process of grouping and merging the subgroups makes the combination of intra group search and global information exchange. It makes SFLA obtain search ability. Compared with PSO, the SFLA's model has stronger global search ability owing to the good framework of SFLA [10].

But SFLA has the following problems. The worst frogs of subgroups in SFLA are guided by the best frogs in the current subgroup or the global best frog. The best frogs in each subgroup can be seen as an ''attractor'' that speeds up the search. But it also increases the possibility of falling into local optima. Moreover, SFLA only one solution which is the worst one in the subgroup is updated at a sub-iteration, so the updating efficiency (the efficiency of one-solution updating way) is not high. In addition, the 3 different updating methods need to be selected by some conditions, and each sub-iteration needs sorting, which increases the computational complexity of SFLA. The number of iterations of subgroups needs to do a lot of experiments for tuning, so that its operability is not strong. The way in which the frog generated randomly to replace the worst frog may degenerate the population. All these problems largely affect the optimization performance of SFLA.

Algorithm 1 SFLA

```
1:  Set the parameters and initialize the population randomly, then calculate the fitness value of each frog
2:  Set $nfe$=0
3:  **while** $nfe<MNFE$ **do**
4:      Group the population
5:      **for** $i = 1$ to $m$ **do**
6:          **for** $j = 1$ to $l$ **do**
7:              Update according to the judgment conditions
8:              using Eq.s (1) to (5)
9:              $nfe = nfe + 1$ at each function evaluation
10:             Sort the subgroup to determine $X_b$ and $X_w$
11:         **end for**
12:     **end for**
13:     Reorganize all the subgroups and sort the population by their fitness values
14: **end while**
15: return $X_g$
```

B. LSFLA

In order to deal with the problems of SFLA, LSFLA was proposed [21]. LSFLA made the following main modifications to SFLA: (1) a Lévy flight updating strategy was adopted to enhance local search ability of SFLA, and also improved the global search ability to a certain extent. The Lévy flight updating method adopted short distance search and occasional long distance search to generate a new solution. The short distance search was used with a large probability, which improved the local search ability of the algorithm, and the occasional long distance search reduced the possibility of falling into local optima. (2) A differential mutation operation was used to improve the global search ability. (3)LSFLA removed the cumbersome condition updating steps to simplify the search process.

The mathematical description of updating method in LSFLA is as follows:

$$\boldsymbol{step} = \boldsymbol{u}/|\boldsymbol{v}|^{1/beta} \tag{6}$$

$$\boldsymbol{LevyFlight} = rand * \boldsymbol{z} \otimes \boldsymbol{step} \tag{7}$$

$$\boldsymbol{X}'_w = \boldsymbol{X}_w + rand * (\boldsymbol{LevyFlight} \otimes \boldsymbol{X}_b - \boldsymbol{X}_w) \tag{8}$$

where $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{z}$ all obeys normal distribution with an average of 0 and a standard deviation of 1. $\boldsymbol{step}$ represents the step size of the Lévy distribution and *beta* is a manual parameter.

In order to enhance the diversity of the population and further improve the global information exchange intensity, a differential mutation method is used after the global information exchange in LSFLA. This method can be described as Eq. (9).

$$\boldsymbol{X}'_i = \boldsymbol{X}_i + scale * (\boldsymbol{X}_r - \boldsymbol{X}_s) \tag{9}$$

where $\boldsymbol{X}_i$ is updated based on the interaction of two other frogs randomly selected in the frog population. $\boldsymbol{X}'_i$ is the updated solution of $\boldsymbol{X}_i$. $\boldsymbol{X}_r$ and $\boldsymbol{X}_s$ are two different frogs which are selected randomly. *scale* is a parameter, namely, scaling factor, which is set to a random value between 0 and 1.

The steps of LSFLA are as follows:

Step 1: Set the number of frogs in the population N, the number of subgroups m, and the number of sub-iterations and etc. Initialize a frog population randomly.

Step 2: Calculate the fitness value of each frog and sort the population in descending order by their fitness values.

Step 3: Divide the population into m subgroups by the same way as SFLA.

Step 3.1: Find the worst frog $\boldsymbol{X}_w$ and the best frog $\boldsymbol{X}_b$ in the current subgroup and generate a new frog by Eq. (8).

Step 3.2: Calculate the fitness value of the new frog, adopt greedy selection to update the worst frog, and then sort the subgroup by their fitness values. If the number of sub-iterations is reached, jump to the next subgroup search, otherwise return **Step 3.1**.

Step 4: Reorganize the subgroups into a population, update every frog of the population according to Eq. (9) and calculate the fitness value of each frog. And the greedy selection is used to keep the better frogs.

Step 5: If the termination condition is satisfied, the best solution $\boldsymbol{X}_g$ is output; otherwise sort the population in descending order according to the fitness values and jump to **Step 3**.

III. DQLSFLA

A. LSFLA'S DEFECTS

LSFLA improves the performance of SFLA largely. From the above steps of LSFLA, however, LSFLA still has the following main problems: (1) LSFLA only updates a solution, i.e. the worst solution, at a sub-iteration like SFLA, so the updating method has low efficiency. (2) When the subgroups update at one sub-iteration, they need to sort the subgroups, thus it results in much time complexity. (3) The number of sub-iterations (the predefined parameter) requires a lot of experiments for tuning and the operability is very weak. (4) The differential mutation method in LSFLA affects the convergence speed.

Aiming at the defects of LSFLA, this paper proposes a series of new approaches, such as all-solution updating, differential perturbation, dynamic scaling factor, quasi-Newton local search and so on, to improve LSFLA.

B. ALL-SOLUTION UPDATING WAY

In LSFLA, only the worst solution of subgroups is updated at a time like SFLA. This may result in some frogs not being updated. If every solution in a subgroup needs to be updated, it is necessary to increase the sub-iteration number (the control parameter, needing careful tuning) in the subgroup and that increases the computational complexity, too.

Inspired by the updating way of other IOAs such as PSO and Differential Evolution (DE), the one-solution updating way of SFLA or LSFLA is replaced by an all-solution updating way in the subgroup. The all-solution updating way has the following advantages over those of LSFLA and SFLA: (1) All the solutions in the all-solution way may be updated to improve the solution updating efficiency and the probability of obtaining the better solution. (2) This way omits the sub-iteration parameter setting to improve the operability of LSFLA. (3) It also omits one sorting step at each sub-iteration to reduce the computational complexity and to improve the optimization efficiency. (4) This way may adopt parallel computing to fasten the running speed.

C. IMPROVED LÉVY FLIGHT UPDATING METHOD

In LSFLA, a new solution is generated by Eq. (8) to update the worst solution, but it is not good for the all-solution updating way to use Eq. (8) to update all the solutions of the subgroup.

In Eq. (8), the new frog is only influenced by $\boldsymbol{X}_b$. It is not make full use of the information of other frogs in the subgroup. Inspired by [22], an improved Lévy flight updating method is that $\boldsymbol{X}_b$ is replaced by a frog selected randomly in a new way. The new selection way can be described as follows.

$$b = ceil\left((j-1) * rand\right) \tag{10}$$

where *ceil* is rounded-up operation and j is the index of the current frog to be updated and $j = 1,2, \ldots, n$. Owing to the sorted subgroups, Eq. (10) can select the better frogs than the j-th frog. From Eq. (10), the maximum and minimum value of b is j-1 and 1, respectively, that is, the range of b is between 1 and j-1. What's more, the fitness value of $\boldsymbol{X}_b$ is not worse than that of $\boldsymbol{X}_j$, thus $\boldsymbol{X}_b$ is called as a demonstrator of $\boldsymbol{X}_j$. So the improved Lévy flight updating method is described as in Eq. (11).

$$\boldsymbol{X}_j' = \boldsymbol{X}_j + rand * \boldsymbol{LevyFlight} \otimes \left(\boldsymbol{X}_b - \boldsymbol{X}_j\right) \tag{11}$$

Compared with Eq. (8), the improved Lévy flight updating method has the following differences (see Eq. (11)): 1) Although both Eqs. (8) and (11) adopt $\boldsymbol{X}_b$ to generate new solutions, $\boldsymbol{X}_b$ in the former is the best frog of the subgroup, while that of the latter is the better frog selected randomly, and the new selection way used in Eq.(11) to generate new solutions can raise the diversity of the population to avoid falling into optima. 2) Owing to adoption of the all-updating way, Eq. (11) is used for every frog of the subgroup to have a high probability of obtaining better solutions.

D. DIFFERENTIAL PERTURBATION STRATEGY

Owing to the frogs sorted in descending order by their fitness values, the front frogs in the sorted queue have relatively better fitness values in each subgroup. According to Eq. (11), the difference between $\boldsymbol{X}_b$ and $\boldsymbol{X}_j$ (the frog to be updated) is relatively small. If Eq. (11) is adopted for the front frogs, the new frog may almost have the same features as the original frog, and even the new frog may remain unchanged. So that is not beneficial to exploration. In addition, from Eq. (10), j cannot be 1 and the front frogs have few demonstrators and the search efficiency is not high for the front frogs. In addition, although the improved Lévy flight updating method has improved the global search ability to some extent, there is still the possibility of falling into local optima. So a differential perturbation strategy is introduced to further increase the diversity of the population and improve the exploration ability.

Inspired by DE[23], a random differential perturbation is embedded in LSFLA. In the process of updating the subgroup solutions, the first half frogs in the subgroup use a random differential perturbation updating method, while the last half frogs adopt the improved Levy flight updating method.

Therefore, by using the random differential perturbation method with better global ability, the diversity of the population increases. Thus the global search ability of LSFAL is improved. The differential perturbation operation is as shown in Eq. (12).

$$\boldsymbol{X}_j' = \boldsymbol{X}_j + w * (\boldsymbol{X}_r - \boldsymbol{X}_s) \tag{12}$$

where $\boldsymbol{X}_j'$ is a new individual. $\boldsymbol{X}_r$ and $\boldsymbol{X}_s$ are frogs selected randomly. $\boldsymbol{X}_r$, $\boldsymbol{X}_s$ and $\boldsymbol{X}_j$ are different from each other. w is the scaling factor, in this paper expressed as Eq.(13).

$$w = (0.5 + 0.5 * rand) \tag{13}$$

where w takes a random number between [0.5, 1]. From Eq.(12), two individuals selected randomly from subgroups get differential vectors through difference calculation. Then a random w weight is given to the difference calculation, and it is added to the individual to generate diversified information.

In addition, the merits of the updating method of LSFLA remained unchanged in the last half of the subgroup. According to Eq.(11), every frog in the last half of the subgroup may approach $\boldsymbol{X}_b$, so the updating direction is toward the better solution. This allows the frogs in the subgroup to converge quickly. The above strategy improves the whole optimization ability of LSFLA. The corresponding pseudo codes are shown in **Algorithm 2**

The random differential perturbation strategy and the improved Lévy flight updating method increase exploration ability and some exploitation of LSFLA. According to intelligent optimization theory, the exploitation and exploration must be balanced well. So a quasi-Newton local search method is applied.

Algorithm 2 All-solution updating way

```
1: for $i = 1$ to $m$ do
2:     for $j = 1$ to $n$ do
3:         if $j<n/2$ then
4:             Perform the random differential perturbation
5:         else
6:             Perform the improved Lévy flight
7:         end if
8:     end for
9: end for
```

E. QUASI-NEWTON LOCAL SEARCH

Quasi-Newton search is one of the most effective methods of local search. It was proposed by Davidon in 1950s and has been developed rapidly since that.

The Newton search uses the curvature information provided by the Hesse matrix to solve the nonlinear optimization problem, but the computation complexity of the Hesse matrix is high. Since the Hesse matrix of some objective functions is difficult to calculate or even difficult to find out, the quasi-Newton algorithm was proposed [24]. The quasi-Newton method overcomes the defects of Newton's method. It constructs an approximate Hesse matrix without using the second-order partial derivative, and increases the one-dimentional search along the Newton direction and has the characteristics of fast convergence. The steps of the quasi-Newton search are as follows [25].

Step 1: Given initial value $\boldsymbol{X}_0$ and precision threshold ε, and set $\boldsymbol{D}_0 = \boldsymbol{I}$,$k$=0.

Step 2: Determine the search direction $d_k = -\boldsymbol{D}_k * g_k$.

Step 3: Calculate step length λ_k, and $S_k = \lambda_k * d_k$, $X_{k+1} := X_k + S_k$.

Step 4: if $\|g_{k+1}\|<\varepsilon$,the algorithm is ended.

Step 5: Calculate $y_k = g_{k+1} - g_k$.

Step 6: Calculate $\boldsymbol{D}_{k+1} = \boldsymbol{D}_k + \frac{S_k S_k^T}{S_k^T y_k} - \frac{\boldsymbol{D}_k y_k y_k^T \boldsymbol{D}_k}{y_k^T \boldsymbol{D}_k y_k}$

Step 7: Set $k = k + 1$ and jump to **step 2**

where g_k represents gradient, λ_k is the step size of the one-dimentional search, and $\boldsymbol{I}$ is the unit matrix.

The quasi-Newton method has been integrated with many IOAs, for example, Chen *et al.* [26]combined it with PSO to improve the local search ability of PSO. But in [26] the Quasi-Newton method was used in PSO throughout the search process, which increased the time complexity of the algorithm. In this paper, the quasi-Newton method is used near the end of DLSFLA which is the ninety-seven percent of *MNFE*. Fast convergence is achieved on the basis of the obtained high quality solutions, which improves the search accuracy of the algorithm, too.

F. OTHER IMPROVEMENT

In addition to the improvements described above,the differential mutation of LSFLA is modified to enhance the performance. In this paper, the random differential mutation in LSFLA is changed to a dynamic one so that the search ability can be enhanced that is, the parameter *scale* is changed from the random value to a dynamic value. The expression is shown in Eq. (14).

$$scale = 1 - nfe/MNFE \tag{14}$$

The dynamic scaling factor strategy avoids the cumbersome steps of parameter setting. The scaling factor is linearly declined in Eq. (14), and *scale* is relatively large in the early stage, which enhances the exploration ability of the algorithm. In the later stage, *scale* is smaller and the exploitation ability of the algorithm is improved so that the whole performance of the algorithm is improved further.

G. DQLSFLA PROCESS

DQLSFLA adopts the all-solution updating method, the differential perturbation strategy and the quasi-Newton local search and so on. These modifications balance the exploration and exploitation better, and the better optimization performance is obtained.

The steps of DQLSFLA are as follows:

Step 1: Set the parameters, and initialize a population randomly.

Step 2: Calculate the fitness value of each frog, and the N frogs are sorted in descending order according to their fitness values.

Step 3: Divide the population into m subgroups by the same way as SFLA, each subgroup is composed of n frogs.

Step 4: In each subgroup the first half frogs are updated by the random differential perturbation strategy, and the last half frogs of subgroup are updated by the improved Lévy flight updating method. Calculate the fitness values of each frog in parallel and the greedy selection is utilized to retain the better solutions.

Step 5: Reorganize all the subgroups into a population, the differential scaling factor is calculated by Eq. (14) and all the individuals in the population adopt the differential mutation with a dynamic scaling factor for information exchange. Execute the boundary control, evaluate the fitness for each frog and sort the population in descending order.

Step 6: If the termination condition is satisfied, the iteration is stop; otherwise jump to **Step 3**.

TABLE 1. Detail of the state-of-the-art comparison algorithms.

Algorithms	Ref.	Years	Description	parameters
POBL-ADE	[28]	2014	a partial opposition-based learning based adaptive DE	The scaling factors (*F*) and crossover rate (*CR*) in ADE is adaptively tuned, when *D* is 10 and 30 *N* equals to 50 and 100, respectively.
LSFLA	[21]	2016	an Lévy Flight Shuffled Frog Leaping Algorithm	When *D* is 10 and 30, *N* is 40. When *D* is 50, *N* is 50.
LXBBO	[29]	2016	a Laplacian BBO	*N* is set 3 times the dimension of the problem, mutation probability=0.005, $\gamma_{\min}=0.1$, $\gamma_{\max}=1$
MOMPSO	[30]	2014	a membrane optimization algorithms of P-systems based on PSO	The acceleration constants(c_1 and c_2) is 2,the range of the weight(*w*) is [0.4, 0.9], the mutation probability(*pr*) is 0.1. *N* is fixed to 49
RW-GWO	[31]	2019	a modified algorithm GWO based on random walk	*N* is fixed to 40, α_i is linearly decreasing from 2 to 0 as the iterations proceed.
CO-ABC	[32]	2013	a modified algorithm called converge-onlookers ABC	*N* is fixed to 50, The update times of each employed bee (UTEB) are set to be 1, 2 and 3.
COGWO2D	[33]	2018	the combinations with the chaotic logistic map, the Opposition-Based Learning, the differential evolution, and the disruption operator is used to improve GWO	*N* is fixed to 30, p_i is proportional to its quality.
GLPSO	[34]	2016	a variant of PSO that mixes various elements from PSO and GA	*N* is fixed to 50, ω = 0.7298 *c* = 1.49618, *pm* = 0.01, *sg*=7.
cNrGA	[35]	2016	a continuous non-revisiting genetic algorithm.	*N* is fixed to 100, *CR*=0.5, *ROOT*=1.
IILPSO	[36]	2016	a multi-swarm PSO variant with specific scheme of information sharing.	*N* is fixed to 50, c_1 is changing from 2.5 to 0.5 and c_2 from 0.5 to 2.5 over the full range of the search. *w* lineally decreasing from 0.6 to 0.4
DNS-PSO	[37]	2016	a diversity-enhanced PSO with neighbourhood search	The acceleration constants is 1.49618, *w*=0.7298, the mutation probability is 0.9. *N* is fixed to 20

Step 7: Near the end of the algorithm, the quasi-Newton method is used to improve the search accuracy and then output the best solution.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT SETTING

In order to verify the optimization efficiency and convergence quality of DQLSFLA, a lot of experiments has been conducted on the complex functions from CEC2014 test set and some real-word problems. These complex functions can be classified into several different types. $f_1 - f_3$ are unimodal functions, $f_4 - f_{16}$ are multimodal functions, $f_{17} - f_{22}$ are hybrid functions, and $f_{23} - f_{30}$ are composite functions. More details of CEC2014 test set refer to [27]. All the experiments are performed on the PC with 3.10GHZ CPU and 4GB RAM memory under a Microsoft Windows 7 operating system. The software for Wilcoxon signed rank test is IBM SPSS Statistics 19, and the programming language for all the experiments is MATLAB R2014a.

In this paper, some statistical methods are used to evaluate the performance of an optimization algorithm. Through some independent runs, the average values (Mean), standard deviation (Std) and average running time (second/s) of the algorithm are recorded. For a minimum problem, the less the Mean value is, the better the performance of the algorithm is, the less the Std value is, the stronger the stability of the algorithm is, and the shorter the running time is, the faster the running speed is. The best values are in bold font in all the result tables.

DQLSFLA is compared with the comparison algorithms, which are described in Table 1. POBL-ADE is a variant of DE. MOMPSO, GLPSO, IILPSO and DNS-PSO are variants of PSO. RW-GWO and COGWO2D are variants of Grey Wolf Optimization (GWO). cNrGA is a variant of GA. LXBBO is a variant of Biogeography-Based Optimization (BBO). The data of POBL-ADE, LX-BBO MOMPSO, RW-GWO, COABC, COGWO2D, GLPSO, cNrGA, IILPSO, and DNS-PSO are from [28]–[37] respectively.

B. EFFECT OF EACH IMPROVEMENT OF DQLSFLA

To demonstrate the contribution of each improvement to the optimization performance of DQLSFLA, DQLSFLA is compared with the 4 incomplete versions of DQLSFLA. The comparison is made on the 10-dimentional CEC2014 test, the parameters of DQLSFLA and its incomplete versions are set the same: the number of frogs (*N*) is 50, *MNEF* = 10000*10, the number of subgroups(*m*) is 5. QLSFLA is an incomplete DQLSFLA without differential perturbation strategy. DQLSFLAuw is an incomplete DQLSFLA without dynamic weight. DLSFLA is an incomplete DQLSFLA without quasi-Newton local search. ILSFLA only uses the improved Levyflight strategy. The comparison results are shown in Fig.2.

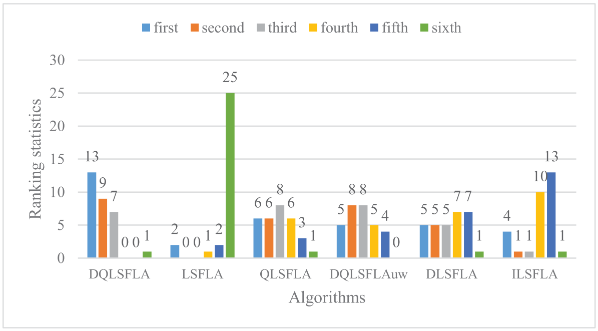


FIGURE 2. Comparison results with the incomplete algorithms ($D = 10$).

From Fig. 2, DQLSFLA obtained 13 times ranking the first, LSFLA obtains 2 times ranking the first, QLSFLA, DQLSFLA$_{uw}$, DLSFLA and ILSFLA obtain 6, 5, 5 and 4 times ranking the first, respectively. By calculation, DQLSFLA reachs the smallest average ranking value which is 1.93. The average ranking values of incomplete algorithms are better than that of LSFLA but inferior to DQLSFLA's. Compare with LSFLA and, ILSFLA obtains the better average ranking values, it proves that the novel Levyflight strategy of Subsection 3.3 is effective. Compared with the incomplete versions of DQLSFLA, the comparison results indicate that DQLSFLA has the best optimization performance and these improvements are indispensable and effective.

C. COMPARISON OF DQLSFLA WITH LSFLA

To be fair for comparison, DQLSFLA and LSFLA adopt the same values for the common parameters, such as dimensions (*D*), independent runs (*IR*) and *MNFE*. According to [27], *IR* = 51 and *MNFE* = 10000**D*. The other parameters of DQLSFLA is set as follows. When *D* is 30, *N* is 40. When *D* is 50, *N* is 50. And *m* is 5. In addition, the detailed settings of LSFLA refer to the corresponding references [21].The experimental results of DQLSFLA and LSFLA are listed in Tables 2 and 3.

TABLE 2. **Comparison results with LSFLA and the other state-of-the-art IOAs ($D = 30$).**

		DQLSFLA	LSFLA	LX-BBO	POBL-ADE	MOMPSO	RW-GWO	B-BBO	COABC
f_1	Mean	**5.49E+00**	4.67E+06	1.01E+07	1.60E+04	1.98E+06	8.02E+06	6.50E+06	5.72E+07
	Std	**1.93E+01**	3.19E+06	3.81E+06	1.22E+04	1.25E+06	3.31E+06	1.30E+06	5.95E+07
	Rank	**1**	4	7	2	3	6	5	8
f_2	Mean	**1.24E+02**	1.29E+09	5.34E+04	3.14E+02	3.79E+02	2.23E+05	2.36E+04	1.09E+04
	Std	**4.17E+02**	3.79E+08	2.14E+04	7.52E+02	1.88E+02	5.51E+05	1.00E+04	1.14E+04
	Rank	**1**	8	6	2	3	7	5	4
f_3	Mean	3.37E+00	1.13E+04	1.64E+04	**6.43E-10**	1.20E+04	3.16E+02	6.04E+03	1.41E+04
	Std	5.67E+00	2.45E+03	1.71E+04	**4.59E-09**	1.18E+04	4.34E+02	3.16E+03	1.36E+04
	Rank	2	5	8	**1**	6	3	4	7
f_4	Mean	**4.69E-01**	2.47E+02	9.99E+01	6.34E+01	2.35E+02	3.41E+01	1.03E+02	1.67E+02
	Std	**1.30E+00**	4.91E+01	2.85E+01	2.63E+01	1.11E+03	1.80E+01	3.14E+01	5.83E+01
	Rank	**1**	8	4	3	7	2	5	6
f_5	Mean	2.00E+01	2.08E+01	**3.06E+00**	2.06E+01	2.06E+01	2.05E+01	3.75E+00	2.03E+01
	Std	3.46E-04	6.46E-02	**7.87E-01**	5.11E-02	2.64E-01	7.46E-02	4.91E-01	8.13E-02
	Rank	3	8	**1**	6	7	5	2	4
f_6	Mean	2.11E+01	2.51E+01	1.69E+01	**5.19E+00**	2.76E+01	9.84E+00	2.00E+01	2.28E+01
	Std	2.30E+00	1.51E+00	3.12E+00	**1.64E+00**	6.00E+00	3.49E+00	2.70E+00	3.70E+00
	Rank	5	7	3	**1**	8	2	4	6
f_7	Mean	**7.00E-03**	1.42E+01	1.76E-01	2.37E-02	1.28E-02	2.53E-01	7.82E-02	9.10E-01
	Std	**1.49E-02**	2.56E+00	8.56E-02	2.31E-02	1.41E-02	1.43E-01	4.44E-02	4.47E-01
	Rank	**1**	8	5	3	2	6	4	7
f_8	Mean	6.85E+01	1.52E+02	5.53E+01	5.59E+01	1.35E+02	4.38E+01	**4.71E-01**	9.91E+00
	Std	1.63E+01	1.12E+01	3.78E+02	1.10E+01	3.15E+01	8.48E+00	**6.80E-01**	6.45E+00
	Rank	6	8	4	5	7	3	**1**	2
f_9	Mean	7.84E+01	1.91E+02	7.66E+01	8.46E+01	1.64E+02	**6.33E+01**	9.11E+01	1.76E+02
	Std	2.09E+01	1.91E+01	1.61E+01	9.06E+00	2.35E+01	**1.30E+01**	1.54E+01	4.43E+01
	Rank	3	8	2	4	6	**1**	5	7
f_{10}	Mean	2.34E+03	4.10E+03	1.26E+04	2.17E+03	3.86E+03	9.61E+02	6.69E+03	**2.67E+02**
	Std	4.50E+02	2.74E+02	1.16E+02	4.92E+02	5.74E+02	2.72E+02	4.58E+02	**1.48E+02**
	Rank	4	6	8	3	5	2	7	**1**
f_{11}	Mean	2.97E+03	4.81E+03	1.23E+04	3.86E+03	3.81E+03	**2.68E+03**	6.72E+03	3.52E+03
	Std	4.74E+02	3.36E+02	3.42E+02	3.52E+02	4.80E+02	**3.68E+02**	5.18E+02	5.78E+02
	Rank	2	6	8	5	4	**1**	7	3
f_{12}	Mean	9.16E-01	1.44E+00	**1.11E-02**	9.51E-01	1.35E+00	5.45E-01	**1.11E-02**	6.82E-01
	Std	3.01E-01	2.28E-01	**1.75E-18**	1.35E-01	6.69E-01	1.66E-01	**1.75E-18**	1.10E-01
	Rank	5	8	**1**	6	7	3	**1**	4
f_{13}	Mean	3.14E-01	4.37E-01	6.55E-01	2.86E-01	4.34E-01	**2.80E-01**	6.75E-01	5.35E-01
	Std	6.19E-02	5.97E-02	1.56E-01	6.10E-02	1.01E-01	**6.30E-02**	7.98E-02	2.59E-01
	Rank	3	5	7	2	4	**1**	8	6
f_{14}	Mean	**1.94E-01**	2.42E-01	6.20E-01	2.26E-01	6.97E-01	4.23E-01	3.93E-01	3.96E-01
	Std	**3.55E-02**	3.30E-02	2.96E-01	4.28E-02	2.19E-01	2.15E-01	1.55E-01	1.69E-01
	Rank	**1**	3	7	2	8	6	4	5
f_{15}	Mean	8.78E+00	5.76E+01	1.55E+01	**7.73E+00**	1.03E+01	8.81E+00	1.88E+01	2.23E+01
	Std	4.27E+00	2.37E+01	5.50E+00	**1.04E+00**	3.24E+00	1.51E+00	5.64E+00	9.67E+00
	Rank	2	8	5	**1**	4	3	6	7
f_{16}	Mean	1.11E+01	1.20E+01	1.08E+01	1.04E+01	1.18E+01	**1.03E+01**	1.07E+01	1.17E+01
	Std	5.07E-01	2.78E-01	5.84E-01	4.58E-01	5.90E-01	**6.11E-01**	6.25E-01	6.07E-01
	Rank	5	8	4	2	7	**1**	3	6
f_{17}	Mean	**5.88E+02**	2.09E+03	1.46E+06	1.10E+03	1.13E+05	5.71E+05	1.28E+06	1.04E+07
	Std	**2.10E+02**	3.04E+02	9.34E+05	4.14E+02	3.34E+04	4.10E+05	5.47E+05	7.73E+06
	Rank	**1**	3	7	2	4	5	6	8
f_{18}	Mean	**5.38E+01**	1.87E+02	2.90E+03	1.10E+02	2.06E+04	6.52E+03	8.22E+02	1.10E+04
	Std	**1.32E+01**	3.31E+01	4.27E+03	3.81E+01	7.01E+03	4.62E+03	1.01E+03	2.88E+04
	Rank	**1**	3	5	2	8	6	4	7
f_{19}	Mean	9.16E+00	1.17E+01	5.19E+03	**8.88E+00**	3.15E+01	1.14E+01	7.81E+03	4.91E+01
	Std	1.10E+00	1.09E+00	5.67E+03	**1.21E+01**	3.22E+01	2.03E+00	4.67E+03	4.26E+01
	Rank	2	4	7	**1**	5	3	8	6
f_{20}	Mean	7.14E+01	2.17E+02	2.61E+04	**3.89E+01**	5.45E+02	6.27E+02	1.63E+04	3.19E+04
	Std	1.75E+01	4.31E+01	1.56E+04	**2.21E+01**	4.38E+02	1.12E+03	4.11E+03	1.82E+04
	Rank	2	3	7	**1**	4	5	6	8
f_{21}	Mean	**3.40E+02**	1.45E+03	1.11E+06	3.86E+02	5.16E+04	2.58E+05	1.23E+06	1.61E+06
	Std	**1.47E+02**	1.76E+02	7.95E+05	1.91E+02	2.89E+04	1.76E+05	7.97E+05	3.55E+06
	Rank	**1**	3	6	2	4	5	7	8
f_{22}	Mean	**1.53E+02**	3.15E+02	1.88E+03	2.31E+02	5.39E+02	2.08E+02	1.68E+02	3.59E+03
	Std	**7.39E+01**	9.14E+01	2.04E+02	8.16E+01	1.75E+02	1.29E+02	2.47E+02	2.88E+02
	Rank	**1**	5	7	4	6	3	2	8
f_{23}	Mean	**2.00E+02**	**2.00E+02**	4.11E+02	3.15E+02	3.39E+02	3.15E+02	3.43E+02	3.26E+02
	Std	**0.00E+00**	**0.00E+00**	6.43E+01	1.16E-07	4.31E+01	2.77E-01	2.84E+01	1.39E+01
	Rank	**1**	**1**	8	3	6	3	7	5
f_{24}	Mean	2.00E+02	**2.00E+02**	1.48E+04	2.22E+02	2.26E+02	2.00E+02	3.41E+04	2.50E+02
	Std	6.58E-06	**1.37E-06**	8.37E+03	7.48E+00	1.41E+01	3.04E-03	2.35E+04	1.19E+01
	Rank	2	**1**	7	4	5	3	8	6
f_{25}	Mean	**2.00E+02**	**2.00E+02**	5.29E+02	2.04E+02	2.06E+02	2.04E+02	6.54E+02	2.21E+02
	Std	**0.00E+00**	**0.00E+00**	4.37E+01	3.22E+00	1.87E+00	1.18E+00	6.01E+01	6.66E+00
	Rank	**1**	**1**	7	3	5	3	8	6
f_{26}	Mean	1.22E+02	1.94E+02	**2.13E+00**	1.39E+02	1.05E+02	1.00E+02	3.64E+01	1.10E+02
	Std	4.14E+01	2.37E+01	**3.46E+00**	4.91E+01	4.67E+00	7.36E-02	5.62E+01	3.02E+01
	Rank	6	8	**1**	7	4	3	2	5
f_{27}	Mean	2.00E+02	**2.00E+02**	**1.96E+02**	4.21E+02	1.10E+03	4.09E+02	3.05E+02	6.52E+02
	Std	2.71E-11	**0.00E+00**	**1.04E+02**	4.64E+01	2.41E+02	6.09E+00	1.60E+02	2.87E+02
	Rank	3	2	**1**	6	8	5	4	7
f_{28}	Mean	**2.00E+02**	**2.00E+02**	1.94E+03	9.16E+02	1.50E+03	4.34E+02	2.12E+03	2.25E+03
	Std	**0.00E+00**	**0.00E+00**	5.49E+02	1.63E+02	2.03E+02	8.45E+00	4.44E+02	7.44E+02
	Rank	**1**	**1**	6	4	5	3	7	8
f_{29}	Mean	1.79E+03	5.23E+02	1.98E+07	3.39E+05	3.61E+03	**2.14E+02**	3.09E+07	1.93E+03
	Std	1.38E+03	4.70E+02	3.96E+06	2.41E+06	1.34E+03	**2.37E+00**	6.92E+06	1.01E+03
	Rank	3	2	7	6	5	**1**	8	4
f_{30}	Mean	2.05E+03	4.34E+03	6.96E+06	1.29E+03	2.42E+04	**6.69E+02**	1.38E+07	8.27E+03
	Std	1.29E+03	2.26E+03	1.03E+07	5.14E+02	2.46E+04	**2.14E+02**	1.09E+07	5.56E+03
	Rank	3	4	7	2	6	1	8	5
Count		12	4	4	5	0	6	2	1
Avg.Rank		2.43	4.97	5.43	3.17	5.43	3.37	5.20	5.80
Total.Rank		1	4	6	2	7	3	5	8

TABLE 3. **Comparison results with LSFLA and the other state-of-the-art IOAs ($D = 50$).**

		DQLSFLA	LSFLA	COGWO2D	GLPSO	cNrGA	IILPSO	DNS-PSO
f_1	Mean	**5.94E+03**	7.50E+07	1.10E+05	5.94E+05	8.74E+06	6.95E+05	8.48E+05
	Std	**3.64E+03**	1.92E+07	1.15E+05	2.96E+05	3.29E+06	3.75E+05	2.77E+05
	Rank	**1**	7	2	3	6	4	5
f_2	Mean	**3.22E+02**	8.89E+09	1.91E+03	7.41E+03	3.02E+05	3.47E+03	7.12E+03
	Std	**6.54E+02**	1.13E+09	2.40E+03	8.34E+03	1.90E+06	4.99E+03	8.69E+03
	Rank	**1**	7	2	5	6	3	4
f_3	Mean	**1.16E+01**	5.02E+04	4.74E+05	2.17E+03	3.16E+04	1.82E+03	1.58E+03
	Std	**1.85E+01**	5.58E+03	1.37E+00	2.06E+03	1.38E+04	1.11E+03	6.18E+02
	Rank	**1**	6	7	4	5	3	2
f_4	Mean	**5.98E-01**	1.21E+03	4.01E+02	7.22E+01	1.07E+02	1.10E+02	8.58E+01
	Std	**1.39E+00**	1.89E+02	1.22E+00	3.20E+01	2.99E+01	3.65E+01	4.70E+01
	Rank	**1**	7	6	2	4	5	3
f_5	Mean	**2.00E+01**	2.10E+01	5.21E+02	2.09E+01	2.00E+01	2.00E+01	2.11E+01
	Std	**8.42E-05**	4.38E-02	2.19E-02	3.82E-01	1.17E-02	8.92E-03	3.73E-02
	Rank	**1**	5	7	4	3	2	6
f_6	Mean	4.63E+01	5.28E+01	6.71E+02	**2.15E+01**	2.43E+01	3.77E+01	3.29E+01
	Std	2.56E+00	2.38E+00	1.02E+00	**3.26E+00**	3.57E+00	3.52E+00	5.91E+00
	Rank	5	6	7	**1**	2	4	3
f_7	Mean	**3.29E-03**	7.23E+01	7.00E+02	1.14E-02	1.91E-01	2.09E-02	1.04E-02
	Std	**8.08E-03**	7.75E+00	1.05E-01	1.30E-02	1.38E-01	4.11E-02	1.35E-02
	Rank	**1**	6	7	3	5	4	2
f_8	Mean	1.57E+02	3.46E+02	8.05E+02	**1.40E-13**	7.91E-01	9.48E-01	1.97E+02
	Std	3.37E+01	1.51E+01	2.64E+00	**1.06E-13**	9.19E-01	9.01E-01	3.38E+01
	Rank	4	6	7	**1**	2	3	5
f_9	Mean	1.62E+02	4.15E+02	1.61E+03	1.07E+02	**8.03E+01**	2.17E+02	2.30E+02
	Std	3.36E+01	2.30E+01	3.48E+00	2.17E+01	**1.59E+01**	3.54E+01	4.01E+01
	Rank	3	6	7	2	**1**	4	5
f_{10}	Mean	5.16E+03	8.89E+03	1.16E+03	1.59E+00	**5.94E-01**	6.20E+00	4.10E+03
	Std	7.45E+02	2.96E+02	4.31E+01	1.96E+00	**9.83E-01**	2.70E+00	7.33E+02
	Rank	6	7	4	2	**1**	3	5
f_{11}	Mean	5.79E+03	9.91E+03	**1.57E+03**	4.97E+03	4.39E+03	4.97E+03	6.22E+03
	Std	6.57E+02	4.08E+02	**2.09E+02**	7.25E+02	8.98E+02	6.77E+02	6.94E+02
	Rank	5	7	**1**	4	2	3	6
f_{12}	Mean	1.22E+00	1.99E+00	1.20E+03	**1.29E-01**	1.38E-01	1.95E-01	2.45E+00
	Std	3.73E-01	2.53E-01	5.05E-02	**5.28E-02**	3.60E-02	4.58E-02	1.01E+00
	Rank	4	5	7	**1**	2	3	6
f_{13}	Mean	5.01E-01	6.11E-01	1.31E+03	4.49E-01	**3.52E-01**	4.59E-01	5.61E-01
	Std	7.86E-02	6.41E-02	1.83E-02	7.78E-02	**7.21E-02**	8.56E-02	9.24E-02
	Rank	4	6	7	2	**1**	3	5
f_{14}	Mean	4.01E-01	1.79E+01	1.40E+03	3.76E-01	5.31E-01	**3.41E-01**	3.97E-01
	Std	2.41E-01	3.40E+00	2.23E-02	1.45E-01	2.55E-01	**1.23E-01**	2.37E-01
	Rank	4	6	7	2	5	**1**	3
f_{15}	Mean	2.55E+02	1.26E+03	8.07E+06	**1.36E+01**	2.27E+01	4.17E+01	2.11E+01
	Std	1.88E+02	6.49E+02	2.29E-01	**3.27E+00**	7.08E+00	1.86E+01	6.57E+00
	Rank	5	6	7	**1**	3	4	2
f_{16}	Mean	2.00E+01	2.14E+01	1.60E+03	**1.77E+01**	1.82E+01	1.85E+01	2.12E+01
	Std	4.31E+01	2.79E-01	3.82E-01	**9.34E-01**	8.17E-01	7.18E-01	4.03E-01
	Rank	4	6	7	**1**	2	3	5
f_{17}	Mean	4.45E+03	4.89E+04	**3.40E+03**	9.64E+04	3.59E+06	2.51E+05	1.54E+05
	Std	4.52E+03	2.12E+04	**4.18E+03**	4.63E+04	1.70E+06	3.32E+05	9.58E+04
	Rank	2	3	**1**	4	7	6	5
f_{18}	Mean	**2.28E+02**	7.52E+02	4.37E+03	1.39E+03	1.55E+03	6.73E+02	1.12E+03
	Std	**4.87E+01**	1.13E+02	6.34E+02	1.14E+03	1.12E+03	7.10E+02	1.12E+03
	Rank	**1**	3	7	5	6	2	4
f_{19}	Mean	4.95E+01	6.15E+01	3.96E+03	**1.36E+01**	2.27E+01	2.55E+01	1.65E+01
	Std	1.75E+01	1.22E+01	3.10E-01	**2.59E+00**	1.36E+01	1.17E+01	3.13E+00
	Rank	5	6	7	**1**	3	4	2
f_{20}	Mean	7.21E+02	4.28E+03	2.01E+03	1.57E+03	7.39E+04	2.38E+03	**4.32E+02**
	Std	2.25E+02	2.99E+03	7.01E+00	1.02E+03	3.45E+04	1.85E+03	**1.78E+02**
	Rank	2	6	4	3	7	5	**1**
f_{21}	Mean	**2.24E+03**	1.02E+04	2.15E+07	6.61E+04	3.64E+06	1.03E+05	1.14E+05
	Std	**1.41E+03**	2.65E+03	2.23E+01	4.10E+04	2.97E+06	5.26E+04	5.25E+04
	Rank	**1**	2	7	3	6	4	5
f_{22}	Mean	**5.76E+02**	1.12E+03	2.21E+03	8.97E+02	9.97E+02	1.18E+03	1.06E+03
	Std	**1.76E+02**	1.73E+02	7.90E+00	2.47E+02	3.56E+02	3.33E+02	3.96E+02
	Rank	**1**	5	7	2	3	6	4
f_{23}	Mean	**2.00E+02**	**2.00E+02**	2.50E+03	3.44E+02	3.44E+02	3.44E+02	2.00E+02
	Std	**0.00E+00**	**0.00E+00**	0.00E+00	1.81E-07	2.28E-02	7.02E-03	2.27E-08
	Rank	**1**	**1**	7	4	6	5	3
f_{24}	Mean	**2.00E+02**	2.00E+02	2.51E+03	2.70E+02	2.67E+02	2.64E+02	2.00E+02
	Std	**1.23E-06**	1.30E-06	0.00E+00	6.96E+00	6.55E+00	7.06E+00	6.49E-05
	Rank	**1**	2	7	6	5	4	3
f_{25}	Mean	**2.00E+02**	**2.00E+02**	2.69E+03	2.24E+02	2.18E+02	2.36E+02	**2.00E+02**
	Std	**0.00E+00**	**0.00E+00**	0.00E+00	6.69E+00	3.71E+00	8.06E+00	**0.00E+00**
	Rank	**1**	**1**	7	5	4	6	**1**
f_{26}	Mean	1.75E+02	1.94E+02	2.70E+03	1.59E+02	**1.41E+02**	1.53E+02	1.97E+02
	Std	4.38E+01	2.37E+01	2.18E-02	4.95E+01	**7.81E+01**	5.02E+01	1.82E+01
	Rank	4	5	7	3	**1**	2	6
f_{27}	Mean	**2.00E+02**	**2.00E+02**	2.90E+03	9.67E+02	9.79E+02	1.43E+03	2.31E+02
	Std	**0.00E+00**	**0.00E+00**	1.39E-12	9.97E+01	9.57E+01	2.86E+02	7.61E+01
	Rank	**1**	**1**	7	4	5	6	3
f_{28}	Mean	2.21E+02	**2.00E+02**	3.00E+03	1.76E+03	1.50E+03	4.39E+03	2.16E+02
	Std	1.48E+02	**0.00E+00**	1.39E-12	4.87E+02	1.16E+02	1.06E+03	6.76E+01
	Rank	3	**1**	6	5	4	7	2
f_{29}	Mean	5.01E+03	5.86E+03	**3.10E+03**	1.44E+06	4.17E+03	7.00E+05	1.85E+06
	Std	4.60E+03	2.69E+04	**0.00E+00**	7.17E+06	5.09E+03	4.98E+06	1.01E+07
	Rank	3	4	**1**	6	2	5	7
f_{30}	Mean	6.22E+03	2.89E+04	**3.20E+03**	1.24E+04	1.46E+04	1.43E+04	1.26E+04
	Std	3.60E+03	3.13E+04	**0.00E+00**	2.30E+03	2.81E+03	3.68E+03	2.60E+03
	Rank	2	7	**1**	3	6	5	4
Count		13	4	4	6	4	1	2
Avg.Rank		2.60	4.87	5.60	3.07	3.83	3.97	3.90
Total.Rank		1	6	7	2	3	5	4

From Table 2, on the 30-dimentional functions, DQLSFLA's Mean values are better on most cases. Just on f_{24} and f_{27}, DQLSFLA's Mean value is inferior to LSFLA's. LSFLA and DQLSFLA have the same Mean value and Std value only on f_{23}, f_{25} and f_{28}. On the other cases, the Mean values and Std values of DQLSFLA are less than those of

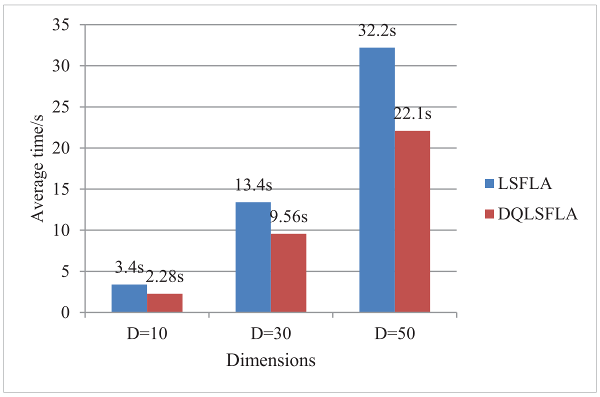


FIGURE 3. **Runtime chart of DQLSFLA and LSFLA.**

LSFLA, it proves that DQLSFLA has much better optimization performance than LSFLA has on these functions.

From Table 3, DQLSFLA also has better performance on the 50-dimentional functions. LSFLA obtains the better results on f_{28}, and has the same Mean values and Std values on f_{23}, f_{25}, and f_{27} as DQLSFLA. On the other cases, the Mean values and Std values of DQLSFLA are better than those of LSFLA, indicating that DQLSFLA has better optimization performance on these functions. From the experimental results on the 30- and 50-dimentional functions, show that the all-solution updating, differential perturbation, dynamic scaling factor, quasi-Newton local search of DQLSFLA improves the performance of LSFLA.

D. ANALYSIS OF TIME COMPLEXITY

In this section, the time complexity of DQLSFLA is analyzed. In an algorithm, the evaluation of the objective function is the most time-consuming part of the whole algorithm. In all the experiments, *MNFE* is used as the termination condition for DQLSFLA and LSFLA, so the part of the function evaluation is not discussed. It should be pointed out that although the evaluation of the objective function is the most time-consuming operation, the time-consuming of other operations also affects the speed of the whole algorithm.

Firstly, the subgroups need to be sorted after every sub-iteration in LSFLA. Suppose that the number of sub-iterations is l, N is the number of frogs in the population, m is the number of subgroups. If the simple sorting is used for subgroups, and the subgroup is already sorted and at most one element is updated, the computational complexity of simple sorting is $O\left(\log_2^n\right)$. Then the computational complexity of sorting for all the subgroups is $l{*}m{*}O\left(\log_2^n\right)$. So the totally computational complexity of all the subgroups is $MNEF/(l{*}m + N){*}\ l{*}m{*}O\left(\log_2^n\right)$, DQLSFLA omits the sorting step of all the subgroups, which reduces the computational complexity.

Secondly, the computational load of the Lévy flight updating method is much high, so the computational complexity of the algorithm will increase if the Lévy flight updating method is used frequently. Assume that the computational complexity of the Lévy flight updating method is O (Lévy). Then the computational complexity is $MNFE/(l{*}m + N){*}$O (Lévy) in LSFLA. In DQLSFLA only half of frogs use the improved Lévy flight update method with high computational complexity. The computational complexity of DQLSFLA is $MNFE/(2N)/2{*}$O (Lévy) and less than that of LSFLA. This improves the global search ability and reduces the computational complexity of LSFLA.

Finally, DQLSFLA uses quasi-Newton local search and it will increase the computational complexity of DQLSFLA, but quasi-Newton local search is used near the end of the algorithm. The average runtime comparison is showed as Fig. 3. From Fig. 3, DQLSFLA takes less runtime than LSFLA does, Although the quasi-Newton local search increases the computational complexity of DQLSFLA to a certain extent, this increase is a little bit, and from the convergence analysis in Section 4.5, the quasi-Newton local search greatly increases the convergence speed in the later stage of DQLSFLA.

From Fig. 3, When D is 10, LSFLA's average runtime is 3.4 seconds, while DQLSFLA's is 2.28 seconds. They cost 13.4 and 9.56 seconds on the 30-dimentional functions, respectively, and LSFLA's (32.2 seconds) is larger than DQLSFLA's (22.1 seconds) on the 50-dimentional functions. It indicates that DQLSFLA reduces the computational complexity of LSFLA and fastens the running speed.

E. CONVERGENCE PERFORMANCE

To investigate the convergence quality of DQLSFLA, the experiments for convergence are conducted to compare DQLSFLA and LSFLA in this section. To speak concisely, some representative 10-dimentional functions of different types are selected to explain in detail. Fig. 4 shows the comparison results. On f_1-f_4, f_{25} and f_{26}, the convergence speed of DQLSFLA is faster than that of LSFLA obviously. The initial convergence speed on f_5-f_{12} lags behind LSFLA's, but DQLSFLA converges quickly after quasi-Newton search near the end of DQLSFLA, thus DQLSFLA obtains better convergence performance than LSFLA does. DQLSFLA and LSFLA have the same convergence speed on f_{23}. On f_{28}, the convergence quality of LSFLA is better than DQLSFLA's. Shown as in Fig. 4, owing to adoption of strategies such as quasi-Newton search, the convergence performance of DQLSFLA is better than that of LSFLA in general.

From the above analysis, DQLSFLA can solve the problems of low efficiency and high computational complexity of LSFLA through the all-solution updating method, improved Lévy flight updating way and differential perturbation strategy.

F. COMPARISON WITH THE OTHER IOAS

In order to further verify the performance of DQLSFLA, it is compared with the state-of-the-art algorithms in this section. These state-of-art algorithms are described in Table 1. The data of B-BBO are from [29].The parameters setting of DQLSFLA is referred to Section 4.3. The detailed parameters settings of the comparison algorithms are referred to the corresponding reference. For these comparison algorithms,

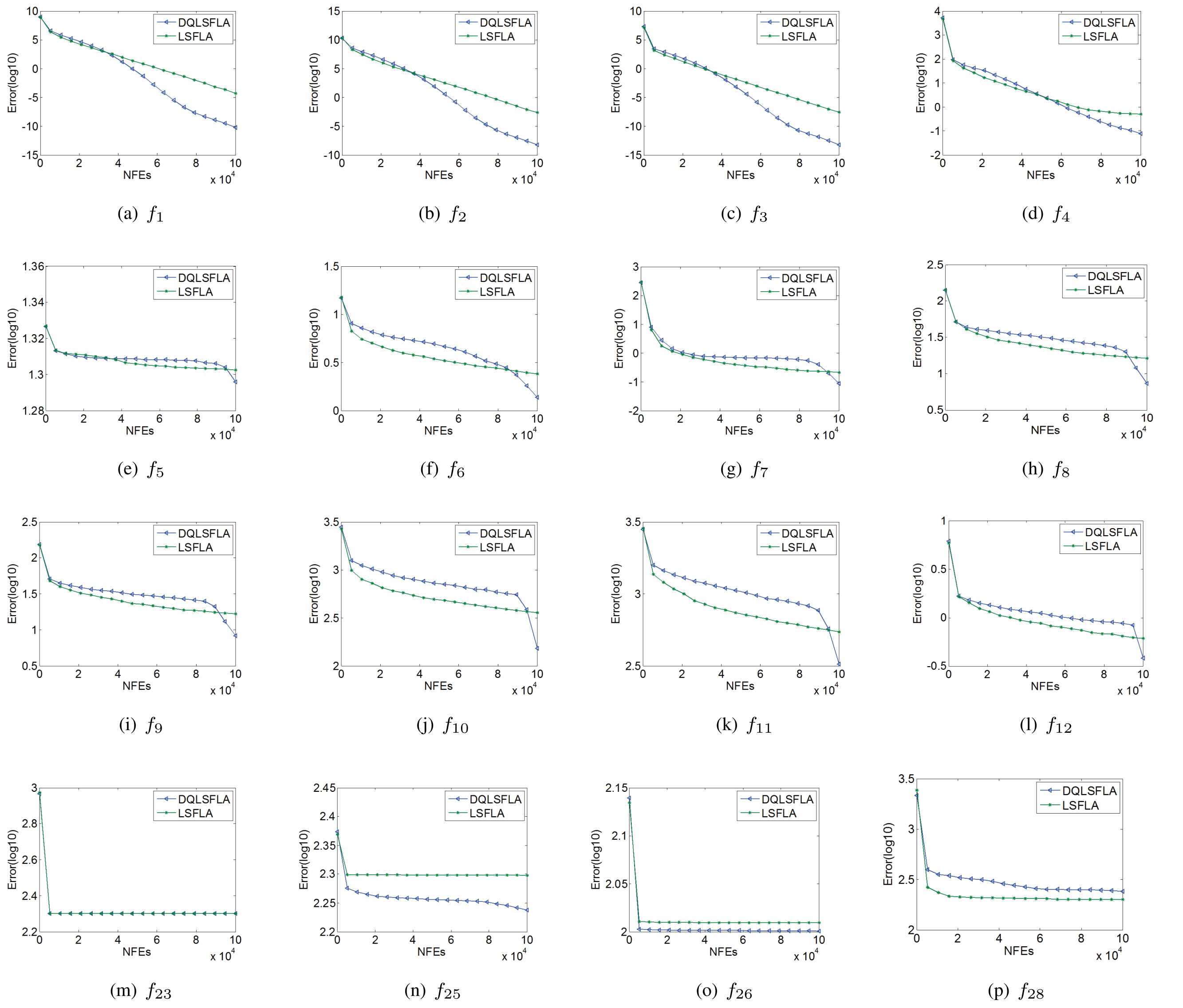


FIGURE 4. **Convergence curves of DQLSFLA and LSFLA.**

MNEF and *IR* are the same as the recommended experiment settings in [27] for fair comparison. The comparison results of these algorithms on the 30- and 50-dimentional functions are shown in Tables 2 and 3, respectively.

To intuitively compare the optimization performance of DQSFLA and the comparison algorithms, these algorithms are ranked according to the results obtained. The ranking criterion is that, these algorithms' Mean values are compared on each function, the less the Mean value is, the higher the ranking is. If some algorithms obtain the same Mean values, then their Std values are compared, the less the Std value is, the higher the ranking is. If some algorithms obtain the same Mean and Std values, their rankings are equal.

From Table 2, on the 30-dimentional functions, DQLSFLA obtains ranking the first on f_1 and f_2 (unimodal functions). The Mean values DQLSFLA obtains on f_1 and f_2 are 5.49E + 00 and 1.93E + 01 respectively, and they are better than all the comparison algorithms' obviously. On f_3 POBL-ADE obtains the better Mean value than the other algorithms do. On f_4-f_{16} (multimodal functions), DQLSFLA obtains ranking the first on f_4, f_7 and f_{14}. It is the second only to POBL-ADE on f_6, f_9 and f_{11}. DQLSFLA still has outstanding performance on f_{17}-f_{30} (composite and hybrid functions). On f_{17}, f_{18}, f_{21}-f_{23}, f_{25} and f_{28}, DQLSFLA obtains ranking the first, and the Mean values of LSFLA are equal to those of DQLSFLA on f_{23}, f_{25} and f_{28}. On the 30-dimentional functions, the results of Avg Rank show that DQSLFA obtains ranking the first(2.43) and the most significant optimization performance among all the algorithms. From Tables 3, DQLSFLA still performs better than the comparison algorithms do on the 50-dimentional functions. From the comparison results on the 50-dimentional functions, the same

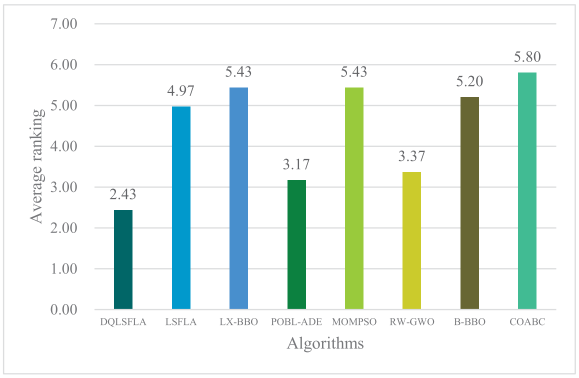


FIGURE 5. Average ranking graph of DQSFLA and the comparison algorithms.

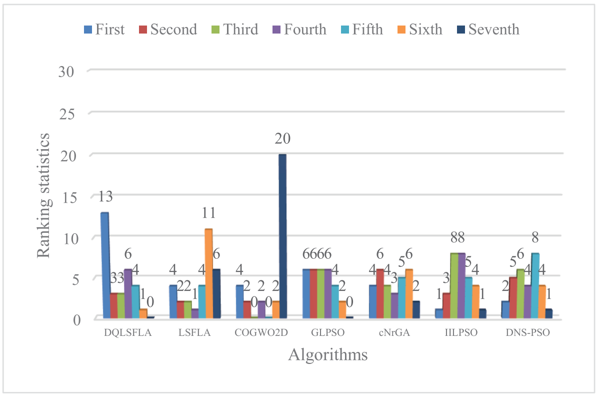


FIGURE 6. Ranking statistics of DQSFLA and the comparison algorithms on $D = 50$.

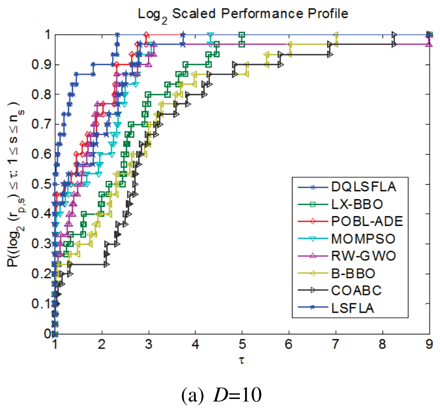
conclusion also can be obtained as on the 30-dimentional functions.

Fig. 5 shows the average rank statistics on the 30-dimentional functions. It can be seen intuitively from Fig. 5 DQLSFLA is ranking the first, with the best values 2.43 on the 30-dimentional functions. Fig. 6 shows ranking statistics of DQSFLA and the comparison algorithms on the 50-dimentional functions. From Fig. 6, DQLSFLA obtains 13 times ranking the first, 3 times ranking the second, 3 times ranking the third, 6 times ranking the fourth, 4 times ranking the fifth, 1 time ranking the sixth and no ranking the seventh. LSFLA obtains 4 times ranking the first, 2 times ranking the second, 2 times ranking the third, 1 times ranking the fourth, 4 times ranking the fifth, 11 time ranking the sixth and 6 times ranking the seventh and so on.

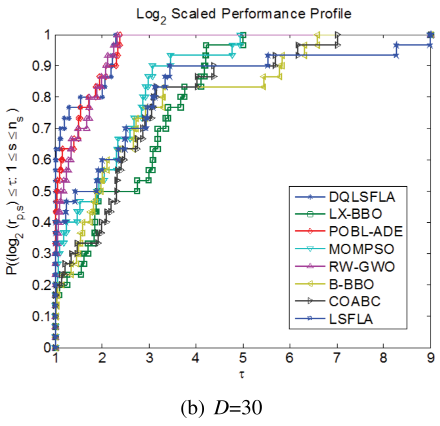
Speaking generally, DQLSFLA's ranking results are better than those of the comparison algorithms, indicating that DQSFLA has better performance than the comparison algorithms have on the 30- and 50-dimentional functions.

G. ANALYSIS OF PERFORMANCE

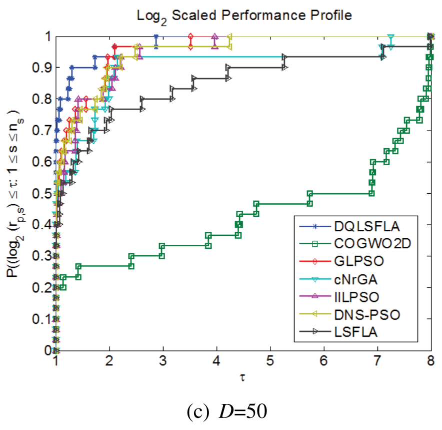
To intuitively compare the performance of all the algorithms, the performance profiles are plotted for the 10-, 30- and 50-dimentional functions in Fig. 7. The performance profile



(a) D=10



(b) D=30



(c) D=50

FIGURE 7. Performance profile on the 10 −, 30 − and 50-dimentional functions.

was introduced as a tool for evaluating and comparing the performance of the set of solvers S on a test set P [38]. In this paper, the ratio of the Mean value and the minimum Mean value of any solver in S is used as the performance metric. The performance ratio can describe as follows:

$$r_{p,s} = \frac{t_{p,s}}{min\{t_{p,s} : s \in S\}} \tag{15}$$

The performance of solver s on any $p \in P$ may be obtained, but we would like to obtain an overall assessment of the performance of the solver. Then $p_s(\tau)$ is defined as:

$$p_s(\tau) = \frac{1}{n_p} size \left\{ p \in P : r_{p,s} \leq \tau \right\} \tag{16}$$

where $p_s(\tau)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor $\tau \in R$ of the best possible ratio. The function p_s is the (cumulative) distribution function for the performance ratio. $R \rightarrow [0,1]$ for a solver is a no decreasing, piecewise constant function, continuous from the right at each breakpoint. The value of $P_s(1)$ is the probability that the solver will win over the rest of the solvers. The robustness of the solver is measured by $P_s(\tau)$ for τ sufficiently large [39]. If the number of wins is only interested, thus we need only to compare the values of $P_s(1)$ for all of the solvers [38].

Form Fig. 7, DQLSFLA has the most wins compared with the comparison algorithms. When $D = 10$ and $D = 30$, the probability that DQLSFLA is the winner on the given functions is about 0.65 for $\tau = 1$. When $D = 50$, the probability that DQLSFLA is the winner on the given functions is about 0.7 for $\tau = 1$. For large values of τ, DQLSFLA can solve a large percentage of the tested functions.

H. WILCOXON SIGNED RANK TEST

Wilcoxon signed rank test is a nonparametric test method [40]. It is used to test whether the differences between the two samples are significant. In this section, it is used to test the performance of DQLSFLA, LSFLA and the other comparison algorithms. The data on the 30- and 50-dimentional are from Tables 2 and 3. Where R^+ refers to the sum of ranks for the problems in which the first algorithm outperforms the second, and R^- refers to the sum of ranks for the opposite. When the performance of the first algorithm is the same as the second algorithm, the corresponding ranks are split evenly to R^+ and R^-. The p values can be computed according to the R^+ and R^- values. *w/t/l* means DQLSFLA wins on w functions, ties on t functions and loses on l functions. The results of the Wilcoxon signed rank test on the 10-, 30- and 50-dimentional functions are shown in Table 4.

From Table 4, DQLSFLA shows a significant performance over LSFLA with a level of significance $\alpha = 0.01$. In general, DQLSFLA shows a significant performance over the comparison algorithms with a level of significance $\alpha = 0.05$, except for RW-GWO and cNrGA. The two algorithms provide strong competition to DQLSFLA on the 30- and 50-dimentional functions, respectively.

TABLE 4. Results of wilcoxon signed rank tests.

D=10	*p*-value	R^+	R^-	*n/w/t/l*
DQLSFLA vs LSFLA	2.863E-05	426	39	28/1/1
DQLSFLA vs LX-BBO	4.196E-04	404	61	25/0/5
DQLSFLA vs POBL-ADE	3.634E-03	370	95	21/1/8
DQLSFLA vs MOMPSO	6.044E-05	418	47	24/1/5
DQLSFLA vs RW-GWO	2.136E-03	374.5	90.5	21/1/8
DQLSFLA vs B-BBO	1.114E-03	391	74	24/0/6
DQLSFLA vs COABC	3.460E-05	424	41	26/1/3
D=30	*p*-value	R^+	R^-	*n/w/t/l*
DQLSFLA vs LSFLA	1.011E-04	377	88	24/5/1
DQLSFLA vs LX-BBO	4.534E-04	403	62	22/0/8
DQLSFLA vs POBL-ADE	2.183E-02	344	121	20/0/10
DQLSFLA vs MOMPSO	4.729E-06	455	10	29/0/1
DQLSFLA vs RW-GWO	1.329E-01	302	163	18/1/11
DQLSFLA vs B-BBO	1.477E-04	417	48	24/0/6
DQLSFLA vs COABC	1.150E-04	420	45	26/0/4
D=50	*p*-value	R^+	R^-	*n/w/t/l*
DQLSFLA vs LSFLA	2.305E-05	399	66	25/4/1
DQLSFLA vs COGWO2D	4.114E-03	372	93	25/0/5
DQLSFLA vs GLPSO	3.001E-02	338	127	18/0/12
DQLSFLA vs cNrGA	7.802E-02	314	151	17/1/12
DQLSFLA vs IILPSO	1.591E-02	344	121	18/1/11
DQLSFLA vs DNS-PSO	7.658E-03	339	117	20/3/7

V. APPLICATION TO MEDICAL IMAGE ENHANCEMENT

A. PRINCIPLE OF IMAGE ENHANCEMENT

Image enhancement is used to improve the visual effect of an image and it's beneficial to subsequent processing and analysis. Medical images generally have characteristics such as ambiguity, non-uniformity, anatomy complexity of the human body and irregularities of the shape of the tissues and organs. So it's very important and necessary for medical images to be enhanced before their subsequent processing. DQLSFLA is applied to medical image enhancement by increasing the contrast and sharpening the features in this section. The description of image enhancement is below.

$$I(i,j) = k * A * [f(i,j) - c * m(i,j)] * (\sigma(i,j) + b) + m(i,j)^a \tag{17}$$

where I represents the enhanced image. f is the input image. $m(i,j)$ is the local mean of the pixel of the coordinate (i,j) in the image over an $n \times n$ window, and σ is the local standard deviation of the grayscale image. A is the average value of all points of the grayscale image. a, b, c, and k are parameters of the enhancement function and any small variation in their values produces a large variation in the processed image and thus the value of these parameters should be precisely set. The approximate ranges of a, b, c and k are defined as [0, 1.5], [0, $(A/2)$], [0, 1], and [0.5, 1.5], respectively [41].

B. IMAGE ENHANCEMENT EVALUATION CRITERIA

There are different methods and criteria for evaluating the quality of enhanced images. In this paper, the evaluation function *F(Z)* of image enhancement quality in [42] is adopted. *F(Z)* is a function defined by factors such as the performance measures entropy value, sum of the edge intensities, and edge pixels. The enhanced quality of the image is evaluated by comparing *F(Z)* values. The larger the value of *F(Z)* is, the richer the image details are, and the better the image enhancement is [42]. The description is below.

$$F(\mathbf{Z}) = log\left(log\left(E\left(I\left(\mathbf{Z}\right)\right)\right)\right) * ne\left(\left(I\left(\mathbf{Z}\right)\right)\right) * H\left(I\left(\mathbf{Z}\right)\right) \big/ (M * N) \tag{18}$$

$$H\left(I\left(\mathbf{Z}\right)\right) = -\sum_{i=0}^{255} e_i \tag{19}$$

where the parameters to be optimized a, b, c and k are given by the species $Z = (a, b, c, k)$. $E(I_s(Z))$ is the intensity of the edge detected by the edge detection detector. ne is the number of pixels detected by the edge detector. $H(I(Z))$ is the entropy of the $I(Z)$. If h_i is not equal to 0, $e_i = h_i log^2(h_i)$,

TABLE 5. Comparison results of DQLSFLA and LSFLA on F(Z).

	LSFLA				DQLSFLA			
	Mean	*Std*	*Vmin*	*Vmax*	*Mean*	*Std*	*Vmin*	*Vmax*
Chest	180.34	0.71	178.99	181.17	**181.92**	**0.60**	181.23	183.28
Blood	129.23	2.64	125.02	132.55	**133.60**	**3.73**	126.24	137.88

otherwise $e_i = 0$. h_i is the probability of occurrence of the i_{th} intensity value of the enhanced image. M and N are the number of pixels in the horizontal and vertical direction of the image.

C. EXPERIMENTAL RESULTS

The optimization problem considered in this section is to solve the image enhancement problem by DQLSFLA. Our goal is to maximize the objective function in order to enhance the contrast by maximizing the number of pixels in the edges, increasing the overall edges intensity and the entropy measure[43]. DQLSFLA and LSFLA are used to parameterize the image enhancement function optimally. The experimental environment is the same as in Section 4.1. The performance of image enhancement using DQLSFLA is validated by applying it to a human chest bone fluoroscopic image and a human blood vessel image, and two images are chosen from a lot of experimental images as two examples to illustrate the effectiveness of DQLSFLA. The parameters of the two algorithms is $N = 100$, the maximum number of iterations (*MAXIter*) is 500, and $IR = 10$. Table 5 lists the values of the evaluation function $F(Z)$.

From Table 5, the Mean values of DQLSFLA on the two images are better than that of LSFLA. The Mean value of DQLSFLA on Blood tissue is 133.60, while the Mean value of LSFLA on the same image is 129.23. The difference between the Mean values is 4.37. In Chest, the minimum value of DQLSFLA is 181.23, while the maximum value of LSFLA is 181.17. The minimum value of DQLSFLA is larger than the maximum value of LSFLA. Comparative experimental results on medical images demonstrate the efficiency and effectiveness of DQLSFLA. The comparison between the original and enhanced images is shown in Fig. 8. From Fig. 8, compared with the original images, the backgrounds of enhanced images become weaker, and the textures of bone and blood vessel are more clearly highlighted. This indicates that adopting the parameter optimization algorithm enhances the contrast and detail of the medical image by maximizing $F(Z)$.

VI. QUADRATIC ASSIGNMENT PROBLEM

Quadratic Assignment Problem (QAP) was proposed by Koopmans *et al.* in 1957. Due to its wide range of applications, many scholars are committed to the study on it. Many real-world problems include hospital layout problem, traveling salesman problem and campus planning problem and etc. can be transformed into QAP, so it is very important and meaningful to solve QAP.

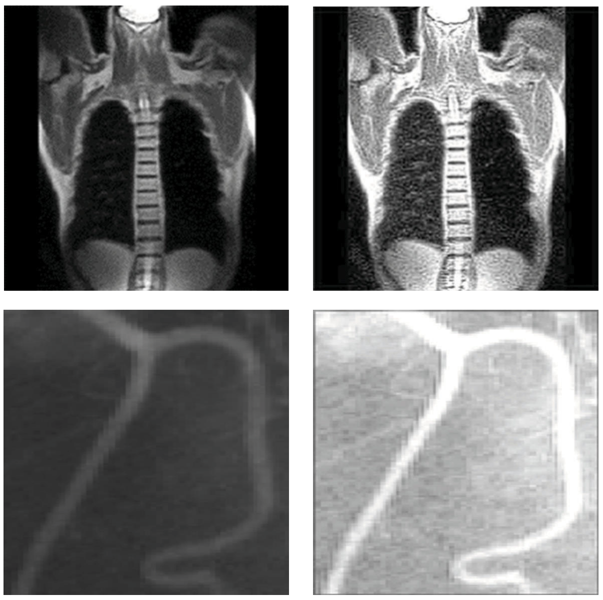

(a) Orginal image (b) Enhanced image

FIGURE 8. Original images and enhanced images.

A. MATHEMATICAL DESCRIPTION OF QAP

QAP can be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given task flows between the facilities. A facility corresponds to only one location and vice versa, and all facilities must be assigned.

The distance matrix between locations of QAP is denoted by $D = [d_{ij}]n* n$. The distance between each pair of locations is denoted by d_{ij} which represents the distance from location i to location j. $W = [w_{\pi(i)\pi(j)}]$ represents the task flow matrix between facilities. $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ represents an allocation scheme. Where $\pi(i)$ and$\pi(j)$ represent the facilities place at position i and j, respectively. Then QAP can be described below.

$$\underset{\pi\in\Pi}{minmize} : f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} w_{\pi(i)\pi(j)} \tag{20}$$

where Π is a set of all allocation schemes.

B. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, DQLSFLA is used to solve QAP. Many experiments are conducted on a lot of data. To illustrate the points concisely, two sets of locations (Set1 and Set2) and a set of flow matrix data are selected as examples to explain the ability of DQLSFLA to solve QAP. The comparison algorithms are LSFLA, IPSO and IFA. Set1, codes used to solve this problem, and IPSO and IFA are directly from [44]. Set2 is generated randomly. The coordinates of Set1 and Set2 are presented in Table 6. The experimental environment is the same as Section 4.1 and the common parameters of the 4 algorithms are set as follows: $N = 20$, $MAXIter = 4000$, $IR = 50$ and $D = 40$. In our experiment, there are 40 locations and 20 facilities, and a 20*20 flow matrix is presented

TABLE 6. The coordinate of the locations (Set1 and Set2).

Set1	(70, 89)	(63, 5)	(11, 29)	(5, 57)	(43, 50)	(94, 73)	(10, 47)	(2, 93)	(68, 96)	(74, 67)
	(24, 51)	(89, 80)	(59, 22)	(59, 89)	(41, 72)	(7, 83)	(73, 11)	(86, 56)	(34, 88)	(88, 82)
	(83, 6)	(19, 55)	(66, 99)	(8, 79)	(64, 23)	(56, 39)	(92, 12)	(83, 10)	(36, 33)	(77, 75)
	(74, 41)	(8, 86)	(51, 59)	(30, 68)	(60, 54)	(51, 9)	(17, 39)	(81, 45)	(65, 5)	(54,72)
Set2	(21, 71)	(15, 68)	(76, 96)	(38, 18)	(40, 12)	(47, 93)	(3, 76)	(92, 50)	(9, 6)	(74, 8)
	(15, 95)	(33, 44)	(47, 48)	(95, 92)	(28, 49)	(8, 51)	(48, 42)	(94, 25)	(68, 31)	(36,65)
	(61, 4)	(6, 70)	(26, 33)	(42, 45)	(33, 15)	(96, 87)	(47, 28)	(67, 97)	(66, 93)	(90, 40)
	(55, 25)	(18, 83)	(76, 3)	(85, 30)	(36, 80)	(49, 7)	(27, 22)	(61, 79)	(2, 94)	(62, 38)

TABLE 7. Flow matrix of 20 facilities.

w=[	4	9	9	6	7	3	9	7	7	7	5	4	w	5	9	2	4	1	4	7
	9	3	7	1	5	7	7	9	5	5	3	5	6	6	5	8	5	1	2	4
	9	7	6	4	4	1	6	6	8	9	4	3	3	5	7	1	7	6	7	2
	6	1	4	5	3	7	4	4	7	6	6	7	7	4	3	1	4	5	8	1
	7	5	4	3	1	5	6	7	4	7	3	4	4	4	2	1	6	5	2	7
	3	7	1	7	5	9	3	6	6	7	5	3	6	8	6	7	6	4	2	1
	9	7	6	4	6	3	3	4	6	4	3	5	6	4	2	5	5	9	6	6
	7	9	6	4	7	6	4	4	2	3	6	5	7	3	1	6	9	4	1	3
	7	5	8	7	4	6	6	2	7	3	7	8	5	5	8	4	4	3	7	5
	7	5	9	6	7	7	4	3	3	9	9	7	4	6	2	4	5	3	9	5
	5	3	4	6	3	5	3	6	7	9	4	3	3	5	7	w	6	6	5	4
	4	5	3	7	4	3	5	5	8	7	3	6	5	7	9	5	8	6	4	3
	w	6	3	7	4	6	6	7	5	4	3	5	1	8	4	7	5	5	7	5
	5	6	5	4	4	8	4	3	5	6	5	7	8	6	2	4	9	5	3	2
	9	5	7	3	2	6	2	1	8	2	7	9	4	2	7	4	7	7	9	5
	2	8	1	1	1	7	5	6	4	w	4	5	7	4	4	4	4	7	5	4
	4	5	7	4	6	6	5	9	4	5	6	8	5	9	7	4	7	4	8	6
	1	1	6	5	5	4	9	4	3	3	6	6	5	5	7	7	4	4	4	5
	4	2	7	8	2	2	6	1	7	9	5	4	7	3	9	5	8	4	6	w
	7	4	2	1	7	1	6	3	5	5	4	3	5	2	5	4	6	5	w	9]

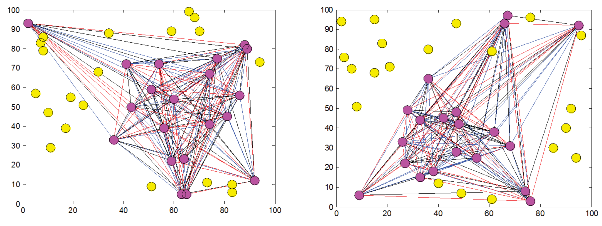
w(1,13)=-10000, w(13,1)=-10000, w(11,16)=20000
w(16,11)=20000, w(19,20)=10000, w(20,19)=10000

TABLE 8. Comparison results with the other IOAs on QAP.

Algorithms		Set1	Set2	Algorithms		Set1	Set2
DQSLFA	Mean	**-1089820.49**	**-1013515.99**	IPSO	Mean	-1019730.33	-978468.12
	Std	**15633.68**	16526.16		Std	57315.22	29297.57
	Vmax	**-1037268.00**	**-965198.49**		Vmax	-904468.22	-880180.49
	Vmin	**-1109337.54**	**-1040778.94**		Vmin	-1109119.40	-1040698.16
	Time	**3.2127**	**3.1991**		Time	6.2342	6.2409
LSFLA	Mean	-1066485.12	-1011461.32	IFA	Mean	-1059311.13	986148.88
	Std	32990.89	**15599.99**		Std	33326.3	25281.81
	Vmax	-972235.61	-962571.04		Vmax	-958942.32	-909487.47
	Vmin	-1109199.80	-1038825.12		Vmin	-1102329.72	-1029158.25
	Time	3.6617	3.6575		Time	5.8093	5.7939

in Table 7. Our goal is to select 20 locations out of 40 locations and determine an allocation scheme to minimize the objective function. In order to make full use of the advantages of DQLSFLA and comparison algorithms for solving continuous optimization problems, the search space of these algorithm is still defined in continuous space when solving QAP, the search range is limited to [0, 1]. The following mapping about the continuous search space and the solution space of QAP is defined. Assume that a solution generated by DQLSFLA is $X = [x_1, x_2 \ldots x_{39}, x_{40}]$, sort the components and get the index number of the top 20 components. That is, the first facility is assigned to the location of the first index number, and the second facility is assigned to the location of the second index number, the twentieth facility is assigned to the location of the twentieth index number. More details for this model can be seen form [44]. The comparison results are entered in Table 8.

From Table 8, the Mean value of DQLSFLA is -1089820.49 on Set1. The Mean value of DQLSFLA is better than that of the comparison algorithms. The Std value of DQLSFLA is 15633.68. It's also the best value among these algorithms' Std values. On Set2, DQLSFLA obtain the best Mean value, and the Std value is only slightly worse than LSFLA's. All in all, DQLSFLA has better performance than the comparison algorithms have on QAP.



(a) Set1 (b) Set2

FIGURE 9. Contribution graphs corresponding to the best results of DQLSFLA.

The contribution graphs corresponding to the best results of DQLSFLA on Set1 and Set2 are shown in Fig. 9. The circles represent the locations on QAP, where the red ones represent the best locations selected from 40 locations. The lines represent the relationship of two locations, namely the task flows of facilities form one to another.

VII. CONCLUSION AND FUTURE WORK

In this paper, an improved LSFLA(DQLSFLA) is proposed to solve the defects of LSFLA, such as slow convergence speed and low optimization efficiency. Firstly an all-solution updating way is created and replaces the one-solution updating way of LSFLA to improve the search efficiency, avoid sorting each subgroup and reduce the computational complexity. Secondly a differential perturbation strategy and is adopted to improve the exploration ability. Thirdly an improved Lévy flight updating method is formulated to keep the advantages of Lévy flight updating way and apply it to the all-solution updating approach. Finally, the quasi-Newton local search method is merged into the improved algorithm near the end of the iterations to obtain better search accuracy and convergence quality. The experimental comparisons not only prove the effectiveness of the improvements, but also prove that DQLSFLA is better than quite a few state-of-the-art algorithms. Moreover, the results on medical image enhancement and QAP also prove that DQLSFLA can solve the problems better. In future, DQLSFLA will be applied to other real-world optimization problems.

REFERENCES

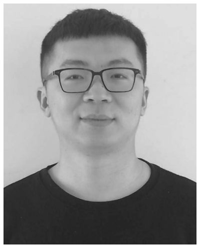
[1] M. Z. Ali, N. Awad, R. G. Reynolds, and P. N. Suganthan, "A balanced fuzzy cultural algorithm with a modified Lévy flight search for real parameter optimization," *Inf. Sci.*, vol. 447, no. 10, pp. 12–35, Mar. 2018.

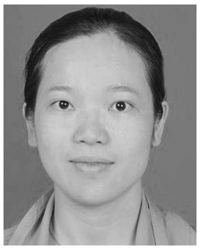
[2] G. Wang and Y. Tan, "Improving metaheuristic algorithms with information feedback models," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 542–555, Feb. 2019.

[3] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl.-Based Syst.*, vol. 148, pp. 115–130, May 2018.

[4] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J. Water Sources Planning Manage.*, vol. 129, no. 3, pp. 210–225, 2003.
[5] K. Vaisakh and A. S. Reddy, "MSFLA/GHS/SFLA-GHS/SDE algorithms for economic dispatch problem considering multiple fuels and valve point loadings," *Appl. Soft Comput.*, vol. 13, no. 11, pp. 4281–4291, Nov. 2013.
[6] W. Ding and J. Wang, "A novel approach to minimum attribute reduction based on quantum-inspired self-adaptive cooperative co-evolution," *Knowl. Based Syst.*, vol. 50, pp. 1–13, Sep. 2013.
[7] H.-B. Wang, K.-P. Zhang, and X.-Y. Tu, "A mnemonic shuffled frog leaping algorithm with cooperation and mutation," *Appl. Intell.* , vol. 43, no. 1, pp. 32–48, Jul. 2015.
[8] X. Li, J. Luo, M.-R. Chen, and N. Wang, "An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimisation," *Inf. Sci.*, vol. 192, no. 6, pp. 143–151, Jun. 2012.
[9] M. A. Ahandani and H. Alavi-Rad, "Opposition-based learning in shuffled frog leaping: An application for parameter identification," *Inf. Sci.*, vol. 291, no. 291, pp. 19–42, Jan. 2015.
[10] C. Liu, P. Niu, G. Li, Y. Ma, W. Zhang, and K. Chen, "Enhanced shuffled frog-leaping algorithm for solving numerical function optimization problems," *J. Intell. Manuf.*, vol. 29, no. 5, pp. 1133–1153, Jun. 2015.
[11] H. Liu, F. Yi, and H. Yang, "Adaptive grouping cloud model shuffled frog leaping algorithm for solving continuous optimization problems," *Comput. Intell. Neurosci.*, vol. 2016, Nov. 2016, Art. no. 5675349.
[12] S. Sharma, T. K. Sharma, M. Rajpurohit, J. Rajpurohit, and B. Naruka, "Centroid mutation embedded shuffled frog-leaping algorithm," *Procedia Comput. Sci.*, vol. 46, pp. 127–134, Jan. 2015.
[13] H. Sun and J. Zhao, "Application of particle sharing based particle swarm frog leaping hybrid optimization algorithm in wireless sensor network coverage optimization," *J. Inf. Comput. Sci.*, vol. 8, no. 14, pp. 3181–3188, Dec. 2011.
[14] P. Roy, P. Roy, and A. Chakrabarti, "Modified shuffled frog leaping algorithm with genetic algorithm crossover for solving economic load dispatch problem with valve-point effect," *Appl. Soft Comput.*, vol. 13, no. 11, pp. 4244–4252, Nov. 2013.
[15] D. Tang, Z. Liu, J. Yang, and J. Zhao, "Memetic frog leaping algorithm for global optimization," *Soft Compt.*, vol. 22, pp. 1–29, Dec. 2018. doi: 10.1007/s00500-018-3662-3.
[16] F. Jiang, L. Dong, Q. Dai, and D. C. Nobes, "Using wavelet packet denoising and ANFIS networks based on COSFLA optimization for electrical resistivity imaging inversion," *Fuzzy Set. Syst.*, vol. 337, pp. 93–112, Apr. 2018.
[17] M.-L. Pérez-Delgado, "Color image quantization using the shuffled-frog leaping algorithm," *Eng. Appl. Artif. Intell.*, vol. 79, pp. 142–158, Mar. 2019.
[18] F. Jiang, L. Dong, and Q. Dai, "Electrical resistivity imaging inversion: An ISFLA trained kernel principal component wavelet neural network approach," *Neural Netw.*, vol. 104, pp. 114–123, Aug. 2018.
[19] T. Zhang, X. Zhao, X. Pan, and X. Li, "Optimal local dimming based on an improved shuffled frog leaping algorithm," *IEEE Access*, vol. 6, pp. 40472–40484, 2018.
[20] A. Sarkheyli, A. M. Zain, and S. Sharif, "The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: A review," *Soft Comput.*, vol. 19, no. 7, pp. 2011–2038, Jul. 2015.
[21] D. Tang, J. Yang, S. Dong, and Z. Liu, "A Lévyflight-based shuffled frog-leaping algorithm and its applications for continuous optimization problems," *Appl. Soft Comput.*, vol. 49, pp. 641–662, Dec. 2016.
[22] X. Zhang, X. Wang, Q. Kang, and J. Cheng, "Differential mutation and novel social learning particle swarm optimization algorithm," *Inf. Sci.*, vol. 480, pp. 109–129, Apr. 2019.
[23] X. Zhang, Q. Kang, J. Cheng, and X. Wang, "A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer," *Appl. Soft Comput.*, vol. 67, pp. 197–214, Jun. 2018.
[24] P. S. B. Nigro, M. Anndif, Y. Teixeira, P. M. Pimenta, and P. Wriggers, "An adaptive model order reduction with Quasi-Newton method for non-linear dynamical problems," *Int. J. Numer. Methods Eng.*, vol. 106, no. 9, pp. 740–759, Jun. 2016.
[25] E. Guerrout, S. Aitaoudia, D. Michelucci, and R. Mahiou, "Hidden Markov random field model and Broyden–Fletcher–Goldfarb–Shanno algorithm for brain image segmentation," *J. Exp. Theor. Artif. Intell.*, vol. 30, no. 3, pp. 415–427, 2018.
[26] Y. Chen, L. Li, H. Peng, and Q. T. Wu, "Dynamic multi-swarm differential learning particle swarm optimizer," *Swarm Evol. Comput.*, vol. 39, pp. 209–221, Apr. 2018.
[27] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Nanyang Technol. Univ., Singapore, Tech. Rep. 201311, 2013.
[28] Z. Hu, Y. Bao, and T. Xiong, "Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Beijing, China, Jul. 2014, pp. 2259–2265.
[29] V. Garg and K. Deep, "Performance of laplacian biogeography-based optimization algorithm on CEC 2014 continuous optimization benchmarks and camera calibration problem," *Swarm Evol. Comput.*, vol. 27, pp. 132–144, Apr. 2016.
[30] G. Singh, K. Deep, and A. K. Nagar, "Cell-like P-systems based on rules of particle swarm optimization," *Appl. Math. Comput.*, vol. 246, pp. 546–560, Nov. 2014.
[31] S. Gupta and K. Deep, "A novel random walk grey wolf optimizer," *Swarm Evol. Comput.*, vol. 44, pp. 101–112, Feb. 2019.
[32] J. Luo, Q. Wang, and X. Xiao, "A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization," *Appl. Math. Comput.*, vol. 219, no. 20, pp. 10253–10262, Jun. 2013.
[33] R. A. Ibrahim, M. A. Elaziz, and S. Lu, "Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization," *Expert Syst. Appl.*, vol. 108, pp. 1–27, Oct. 2018.
[34] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.
[35] Y. F. Lou and S. Y. Yuen, "Non-revisiting genetic algorithm with adaptive mutation using constant memory," *Memetic Comput.*, vol. 8, no. 3, pp. 189–210, 2016.
[36] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2238–2251, Oct. 2016.
[37] A. P. Piotrowski and J. J. Napiorkowski, "Searching for structural bias in particle swarm optimization and differential evolution algorithms," *Swarm Intell.* , vol. 10, no. 4, pp. 307–353, Dec. 2016.
[38] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, 2002.
[39] E. Bergou, Y. Diouane, and S. Gratton, "On the use of the energy norm in trust-region and adaptive cubic regularization subproblems," *Comput. Optim. Appl.*, vol. 68, no. 3, pp. 533–554, Dec. 2017.
[40] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
[41] N. Singh, M. Kaur, and K. V. Singh, "Parameter optimization in image enhancement using pso," *Amer. J. Eng. Res.*, vol. 2, no. 5, pp. 84–90, 2013.
[42] J. Jasper, S. B. Shaheema, and S. B. Shiny, "Natural image enhancement using a biogeography based optimization enhanced with blended migration operator," *Math. Problems Eng.*, vol. 2014, Feb. 2014, Art. no. 232796.
[43] K.-L. Chung, W.-J. Yang, and W.-M. Yan, "Efficient edge-preserving algorithm for color contrast enhancement with application to color image segmentation," *J. Vis. Commun. Image Represent.*, vol. 19, no. 5, pp. 229–310, Jul. 2008.
[44] Yarpiz. (2015). *Quadratic Assignment Problem Using GA, PSO and FA*. [Online] Avaliable:http://yarpiz.com/359/ypap104-quadratic-assignment-problem

XINMING ZHANG is currently a Full Professor and a Supervisor of the master's degree with the College of Computer and Information Engineering, Henan Normal University, Henan, Xinxiang, China. He has published over 100 papers on the important journals and conferences, such as *Information Sciences and Applied Soft Computing*. His current research interests include swarm intelligence, evolutionary computation, and digital image processing.


ZIHAO FU received the B.Sc. degree from Huanghuai University, Henan, China, in 2016. He is currently pursuing the M.Sc. degree with the College of Computer and Information Engineering, Henan Normal University. His research interests include intelligent optimization algorithm and medical image processing.


HAIYAN CHEN is currently a Doctor with the Department of Gynecological Tumor, Hubei Cancer Hospital, Hubei, Wuhan, China. Her research interests include cancer identification and medical image processing.

WENTAO MAO received the M.S. degree in computer science from the Chongqing University of Posts and Telecommunications, in 2006, and the Ph.D. degree from Xi'an Jiaotong University, China, in 2011. He is currently an Associate Professor with Henan Normal University, China. His current research interests include soft computing, machine learning, and data mining.

SHANGWANG LIU received the master's degree from Northwest Agriculture, in 2009, and the Ph.D. degree from Forestry University, in 2012. He is currently an Associate Professor with Henan Normal University. His main research interests include computer vision and image processing.

GUOQI LIU received the M.S. degree from the Department of Applied Mathematics, South China University of Technology (SCUT), Guangzhou, China, in 2010, and the Ph.D. degree from SCUT, in 2013. He is currently an Associate Professor with the College of Computer Science and Information Engineering, Henan Normal University. His research interests include digital image processing and partial differential equation.