The International Journal Of Engineering And Science (IJES)
|| Volume || 4 || Issue || 2 || Pages || PP.19-30|| 2015 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Adaptive Grouping Quantum Inspired Shuffled Frog Leaping Algorithm

Xin Li[1], Xu Huangfu[2], Xuezhong Guan[2], Yu Tong[2],
[1]*School of Computer and Information Technology, Northeast Petroleum University, Daqing, China*
[2]*School of Electrical and Information Engineering, Northeast Petroleum University, Daqing, China*

--**ABSTRACT**--
To enhance the optimization ability of classical shuffled frog leaping algorithm, a quantum inspired shuffled frog leaping algorithm with adaptive grouping is proposed. In this work, the frog swarms are adaptive grouped according to the average value of the objective function of child frog swarms, the frogs are encoded by probability amplitudes of Multi-Qubits system. The rotation angles of Multi-Qubits are determined based on the local optimum frog and the global optimal frog, and the Multi-Qubits rotation gates are employed to update the worst frog in child frog swarms. The experimental results of some benchmark functions optimization shows that, although its single step iteration consumes a long time, the optimization ability of the proposed method is significantly higher than the classical leaping frog algorithm.

KEYWORDS : *Quantum computing, Swarm intelligent optimization, Shuffled leaping frog algorithm, Adaptive grouping*

Date of Submission: 12 February 2015 ⟺ Date of Accepted : 28 February 2015

I. INTRODUCTION

Shuffled frog leaping algorithm (SFLA) is a new heuristic cooperative search algorithm, which simulates the foraging behavior of a group of frogs jumping in wetlands [1]. As a new bionic intelligent optimization algorithm, SFLA combines the advantages of memetic evolutionary algorithm and particle swarm optimization algorithm, with the concept of simple, less adjustment parameters, calculation speed, strong global search capability, easy implementation, currently used in the rough set attribute reduction [2], fuzzy-means clustering [3], drawing water bath to control [4], cooperative spectrum sensing cognitive radio [5- 6], speech recognition [7], etc. In terms of performance improvements SFLA, paper [8] by introducing in the frog population attraction and repulsion mechanism, effectively avoiding premature convergence; paper [9] by introducing the extreme dynamic optimization to improve the optimization efficiency of the algorithm; paper [10] through the integration, simulated annealing, immunizations, Gaussian mutation, chaotic disturbance, enhanced algorithm optimization capabilities; paper [11] by using a dispersion and fitness, improve the convergence speed and accuracy. However, these improvements do not consider the effect of the grouping SFLA for optimal performance. Since SFLA each group to update only a worst frog, in general, when a fixed number of frog, the fewer packets (the more the number of the group frog), the higher the computational efficiency, and optimization capability is relatively weak, so frog group grouping has an important influence on the optimization of the performance of SFLA. Taking into account the optimal number of packets SFLA usually related to specific issues as well as the optimization phase, this paper presents an Adaptive Grouping Quantum Inspired Shuffled Frog Leaping Algorithm. Quantum computing is an emerging interdisciplinary; combining information science and quantum mechanics, and its integration with intelligent optimization algorithms began in the 1990s. Currently, there are a lot more mature algorithms, such as quantum-behaved particle swarm optimization algorithm [12], quantum inspired evolutionary algorithm [13], quantum derivative harmony search algorithm [14], quantum inspired immune algorithm [15], quantum inspired genetic algorithm [16], quantum inspired differential evolution algorithm [17]. In addition to the literature [12] using the real-coded, the rest are encoded using a single bit probability amplitude. Single-bit probability amplitude coding disadvantage is that the adjustment of a qubit can only change one gene locus on the individual, while the probability amplitude for multi-bit coding, adjust a qubit, can change all of the ground state probability amplitude in multi-bit quantum superposition states , thereby changing the position of all genes on the individual. Therefore, this paper will also propose a new multi-bit probability amplitude coding mechanism to further improve the SFLA of optimizing capacity and optimize efficiency. The results of the optimization criterion function demonstrate the superiority of the proposed method.

II.SHUFFLED FROG LEAPING ALGORITHM

First, There is N frogs $x_i = (x_{i1}, x_{i2}, \cdots, x_{in})$, calculated the target value $f(x_i)(i = 1, 2, \cdots, N)$ of each frog, sort all Target value from good to bad. Order *N* = *k* × *m*, k is the number of sub-group, *m* is the number of subgroups frogs, the *N* sorted frogs cycle divided into k sub-groups. namely: the first sub-group divided into optimal frog, The second sub-group divided into Suboptimal frog, and so on, until all the frogs have been allocated. In each subgroup, the best and worst frogs were recorded as x_b and x_w, the whole frog group of the best frog recorded as x_g, in each iteration, the following formula is updated sub-group x_w.

$$D = r(x_b - w_b) \tag{1}$$

$$x'_w = x_w + D \tag{2}$$

where $-D_{\max} \le D_i \le D_{\max}$, D_i is the *i*th value of the vector D, i = 1, 2,…, n, $D_{\max}$ is the biggest leap frog step, r is a random number between 0 and 1.

If x'_w is better than x_w, then x'_w instead of x_w; otherwise, x_b instead of x_g, repeat the Eq.(1) and (2). If the x'_w inferior to x_w,, then randomly generate a solution to replace x_w. So the cycle until the termination condition is met.

III. ADAPTIVE GROUPING STRATEGY

In SFLA, by updating the group worst frog to achieve local optimization, by mixing and re-grouped in each group to achieve global optimization. From the optimization mechanism, the group number reflects the optimization of the global characteristics, while the number of frogs in the group reflects the optimization of the locality. So, how to achieve the balance of global and localized search, is worthy of in-depth study of the problem, and this balance by grouping decision. However, the balance of global and localized search is often also associated with specific issues and optimization phase, grouping of frog group cannot be fixed pattern. Therefore, this paper proposes an adaptive grouping strategy.
Let the total number of frogs be *N*, we rewritten *N* as following equation

$$N = n_1 \times m_1 = n_2 \times m_2 = \cdots = n_s \times m_s \tag{3}$$

where $n_1 < n_2 < \cdots < n_s,\ m_1 > m_2 > \cdots > m_s$.

Therefore, there are s kinds of groupings. Namely: n_1 group, each group has m_1 frogs; n_2 group, each group has m_2 frogs; …; n_s group, each group has m_s frogs. As an example of the minimum value optimization, recorded the average value of the objective function as f_{avg}, Frog group is divided into n_k groups, each with n_k frogs, the average value of the *i*th group objective function is f^i_{avg}, the $f^i_{avg} < f_{avg}$ number is n^1_k, the $f^i_{avg} > f_{avg}$ number is n^2_k. Adaptive grouping strategy proposed in this paper can be described as follows.

(1) If $\begin{cases} n^1_k \ge n^2_k \\ k < s \end{cases}$, than $\begin{cases} n_{k+1} \Rightarrow n_k \\ m_{k+1} \Rightarrow m_k \end{cases}$, (2) If $\begin{cases} n^1_k < n^2_k \\ k > 1 \end{cases}$, than $\begin{cases} n_{k-1} \Rightarrow n_k \\ m_{k-1} \Rightarrow m_k \end{cases}$.

For this strategy, we explain below. $n_{k1} \ge n_{k2}$ represents the sub-groups of f^i_{avg} less than f_{avg} have advantages, This means that the optimization approach tends more parallel sub-groups, and therefore need to increase the number of sub-groups; Conversely, $n_{k1} < n_{k2}$ represents the sub-groups of f^i_{avg} less than f_{avg} have disadvantages, this means that the optimization approach tends to parallel the minority carrier group, hence the need to reduce the number of sub-groups.

IV. MULTI-BIT QUANTUM SYSTEM AND THE MULTI-BIT QUANTUM ROTATION GATE

4.1. Qubits and single qubit rotation gate

What is a qubit? Just as a classical bit has a state-either 0 or 1- a qubit also has a state. Two possible states for a qubit are the state $|0\rangle$ and $|1\rangle$, which as you might guess correspond to the states 0 and 1 for a classical bit.

Notation like $|\rangle$ is called the *Dirac* notation, and we will see it often in the following paragraphs, as it is the standard notation for states in quantum mechanics. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form linear combinations of states, often called superposition.

$$|\phi\rangle = \cos\theta\,|0\rangle + \sin\theta\,|1\rangle = [\cos\theta \quad \sin\theta]^{\mathrm{T}}, \tag{4}$$

where θ is the phase of $|\phi\rangle$, $\cos\theta$ and $\sin\theta$ denote the probability amplitude of $|\phi\rangle$.
In the quantum computation, the logic function can be realized by applying a series of unitary transform to the qubit states, which the effect of the unitary transform is equal to that of the logic gate. Therefore, the quantum services with the logic transformations in a certain interval are called the quantum gates, which are the basis of performing the quantum computation. A single qubit rotation gate can be defined as

$$\boldsymbol{R}(\Delta\theta) = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{bmatrix}. \tag{5}$$

Let the quantum state $|\phi\rangle = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$, and $|\phi\rangle$ can be transformed by $\boldsymbol{R}(\Delta\theta) = \begin{bmatrix} \cos(\theta + \Delta\theta) \\ \sin(\theta + \Delta\theta) \end{bmatrix}$. It is obvious that $\boldsymbol{R}(\Delta\theta)$ shifts the phase of $|\phi\rangle$

4.2. The tensor product of matrix

Let the matrix ***A*** has *m* low and *n* column, and the matrix ***B*** has *p* low and *q* column. The tensor product of ***A*** and ***B*** is defined as.

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} A_{11}\boldsymbol{B} & A_{12}\boldsymbol{B} & \cdots & A_{1n}\boldsymbol{B} \\ A_{21}\boldsymbol{B} & A_{22}\boldsymbol{B} & \cdots & A_{2n}\boldsymbol{B} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}\boldsymbol{B} & A_{m2}\boldsymbol{B} & \cdots & A_{mn}\boldsymbol{B} \end{bmatrix}, \tag{6}$$

where $A_{i,j}$ is the element of matrix *A*.

4.3. Multi-bit quantum system and the multi-bit quantum rotation gate

In general, for an *n*-qubits system, there are 2^n of the form $|x_1 x_2 \cdots x_n\rangle$ ground states, similar to the single-qubit system, *n*-qubits system can also be in the a linear superposition state of 2^n ground states, namely

$$|\phi_1\phi_2 \cdots \phi_n\rangle = \sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n}^{1} a_{x1x2\cdots xn}|x_1 x_2 \cdots x_n\rangle = [a_{00\cdots0} \quad a_{00\cdots1} \quad \cdots \quad a_{11\cdots1}]^{\mathrm{T}}, \tag{7}$$

where $a_{x1x2\cdots xn}$ is called probability amplitude of the ground state $|x_1 x_2 \cdots x_n\rangle$, and to meet the following equation.

$$\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n}^{1}\left|a_{x1x2\cdots xn}\right|^2 = 1. \tag{8}$$

Let $|\phi_i\rangle = \cos\theta_i|0\rangle + \sin\theta_i|1\rangle$, according to the principles of quantum computing, the $|\phi_1\phi_2\cdots\phi_n\rangle$ can be written as

$$|\phi_1\phi_2 \cdots \phi_n\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots |\phi_n\rangle = \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos\theta_n \\ \sin\theta_n \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & \cos\theta_2 & \cdots & \cos\theta_n \\ \cos\theta_1 & \cos\theta_2 & \cdots & \sin\theta_n \\ \vdots & \vdots & \vdots & \vdots \\ \sin\theta_1 & \sin\theta_2 & \cdots & \sin\theta_n \end{bmatrix}. \tag{9}$$

It is clear from the above equation that, in an n-qubits system, any one of the ground state probability amplitude is a function *of n*-qubits phase $(\theta_1, \theta_2, \cdots, \theta_n)$, in other words, the adjustment of any θ_i can update all 2^n probability amplitudes.
In our works, the *n*-qubits rotation gate is employed to update the probability amplitudes. According to the principles of quantum computing, the tensor product of *n* single-qubit rotation gate $\boldsymbol{R}(\Delta\theta_i)$ is *n*-qubits rotation gate. Namely

$$\boldsymbol{R}(\Delta\theta_i\Delta\theta_2\cdots\Delta\theta_n) = \boldsymbol{R}(\Delta\theta_i)\otimes\boldsymbol{R}(\Delta\theta_2)\otimes\cdots\otimes\boldsymbol{R}(\Delta\theta_n), \tag{10}$$

where $\boldsymbol{R}(\Delta\theta_i) = \begin{bmatrix}\cos\Delta\theta_i & -\sin\Delta\theta_i\\ \sin\Delta\theta_i & \cos\Delta\theta_i\end{bmatrix}, \quad i = 1,2,\cdots,n.$

Taking *n*=2 as an example, the $\boldsymbol{R}(\Delta\theta_1\Delta\theta_2)$ can be rewritten as follows.

$$\boldsymbol{R}(\Delta\theta_1\Delta\theta_2) = \begin{bmatrix}\cos\Delta\theta_1\cos\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2 & -\sin\Delta\theta_1\cos\Delta\theta_2 & \sin\Delta\theta_1\sin\Delta\theta_2\\ \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2 & -\sin\Delta\theta_1\sin\Delta\theta_2 & \sin\Delta\theta_1\cos\Delta\theta_2\\ \sin\Delta\theta_1\cos\Delta\theta_2 & -\sin\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2\\ \sin\Delta\theta_1\sin\Delta\theta_2 & \sin\Delta\theta_1\cos\Delta\theta_2 & \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2\end{bmatrix}. \tag{11}$$

It is clear that

$$\boldsymbol{R}_n(\Delta\theta_1\Delta\theta_2\cdots\Delta\theta_n)|\phi_1\phi_2\cdots\phi_n\rangle = |\hat{\phi}_1\rangle\otimes|\hat{\phi}_2\rangle\otimes\cdots\otimes|\hat{\phi}_n\rangle, \tag{12}$$

where $|\hat{\phi}_i\rangle = \cos(\theta_i + \Delta\theta_i)|0\rangle + \sin(\theta_i + \Delta\theta_i)|1\rangle.$

V. HUFFLED FROG LEAPING ALGORITHM ENCODING METHOD BASED ON MULTI-BITS PROBABILITY AMPLITUDES

In this paper, the frogs group is encoded by multi-qubits probability amplitudes. Let N denote the number of particles, D denote the dimension of optimization space. Multi-qubits probability amplitudes encoding method can be described as follows.

5.1. The number of qubits needed to code

For an n-bits quantum system, there are 2n probability amplitudes, which can be used directly as a result of an individual encoding. In the D-dimensional optimization space, it is clear that $D \le 2^n$. Due to the constraint relation between each probability amplitude (see to Eq.(10)), hence $D < 2^n$. For the D-dimensional optimization problem, the required number of qubits can be calculated as follows.

$$n = \log(D) + 1. \tag{13}$$

5.2. The encoding method based on multi-qubits probability amplitudes

First, generating randomly *N* *n*-dimensional phase vector $\boldsymbol{\theta}_i$, $i = 1,2,\cdots,N,$ as follows

$$\boldsymbol{\theta}_i = [\theta_{i1},\theta_{i2},\cdots,\theta_{in}], \tag{14}$$

where $\theta_{ij} = 2\pi \times rand$, *rand* is a random number uniformly distributed within the (0,1), $j = 1,2\cdots n$.

Let $|\phi_{ij}\rangle = \cos\theta_{ij}|0\rangle + \sin\theta_{ij}|1\rangle$, Using Equation (11), we can obtain following *N* *n*-qubits systems $|\phi_{11}\phi_{12}\cdots\phi_{1n}\rangle$, $|\phi_{21}\phi_{22}\cdots\phi_{2n}\rangle$,…, $|\phi_{N1}\phi_{N2}\cdots\phi_{Nn}\rangle$. In each of the quantum system, the first *D* probability amplitudes can be regarded as *a D*-dimensional particle code.

VI. THE UPDATE METHOD BASED ON MULTI-QUBITS PROBABILITY AMPLITUDES

In this paper, the multi-bit quantum rotation gates are employed to update particles. Let the phase vector of the global optimal frog be $\boldsymbol{F}_g = [\theta_{g1},\theta_{g2},\cdots\theta_{gn}]$, the phase vector of the group optimal frog is $\boldsymbol{F}_b = [\theta_{b1},\theta_{b2},\cdots,\theta_{bn}]$, the phase vector of the group worst frog $\boldsymbol{F}_w = [\theta_{w1},\theta_{w2},\cdots,\theta_{wn}]$. Similar to traditional SFLA, for each subgroup, we only just need to update $\boldsymbol{F}_w$.

By formula (9) it is clear that, once $\boldsymbol{F}_w$ has been updated, all its corresponding probability amplitudes will be updated. To improve the search capability, in an iteration, all phases $\boldsymbol{F}_w$ are updated in turn, which allows all particles are updated *n* times. Let $\Delta\theta_0$ denote the phase update step size, the specific update can be described as follows.

Step1. Set *j*=1, $\boldsymbol{F}_w(\theta)=[\cos\theta_{i1}\cos\theta_{i2}\cdots\cos\theta_{in},\cdots\cdots,\sin\theta_{i1}\sin\theta_{i2}\cdots\sin\theta_{in}]^{\mathrm{T}}$.

Step2. Set $\Delta\theta_{i1}=\Delta\theta_{i2}=\cdots=\Delta\theta_{in}=0$.

Step3: Determine the value of the rotation angle, where the sgn donates the symbolic function.

If $|\theta_{bj}^{i}-\theta_{ij}|\le\pi$, then $\Delta\theta_{ij}^{b}=\mathrm{sgn}(\theta_{ij}^{b}-\theta_{ij})\Delta\theta_0$.

If $|\theta_{bj}^{i}-\theta_{ij}|>\pi$, then $\Delta\theta_{ij}^{b}=-\mathrm{sgn}(\theta_{bj}^{i}-\theta_{ij})\Delta\theta_0$.

If $|\theta_{gj}-\theta_{ij}|>\pi$, then $\Delta\theta_{ij}^{b}=\mathrm{sgn}(\theta_{gj}-\theta_{ij})\Delta\theta_0$.

If $|\theta_{gj}-\theta_{ij}|>\pi$, then $\Delta\theta_{ij}^{g}=-\mathrm{sgn}(\theta_{gj}-\theta_{ij})\Delta\theta_0$.

Step4. $\Delta\theta_j=0.5\Delta\theta_j^{b}+0.5\Delta\theta_j^{g}$, $\boldsymbol{F}_w^{(1)}(\theta)=R_n(\Delta\theta_1,\Delta\theta_2,\cdots,\Delta\theta_n)\boldsymbol{F}_w(\theta)$.

Step5: Let $\Delta\theta_j=rnds\cdot\Delta\theta_0$, $rnds$ is a random number of -1 to 1. $\boldsymbol{F}_w^{(2)}(\theta)=R_n(\Delta\theta_1,\Delta\theta_2,\cdots,\Delta\theta_n)\boldsymbol{F}_w^{(1)}(\theta)$. If $\boldsymbol{F}_w^{(1)}(\theta)$ is better than $\boldsymbol{F}_w^{(2)}(\theta)$, then $\boldsymbol{F}_w(\theta)=\boldsymbol{F}_w^{(1)}(\theta)$, otherwise, $\boldsymbol{F}_w(\theta)=\boldsymbol{F}_w^{(2)}(\theta)$.

Step6: If $j<n$, then j = j + 1, back to step2, Otherwise, end.

VII. ADAPTIVE GROUPING QUANTUM INSPIRED SHUFFLED FROG LEAPING ALGORITHM

Suppose that, *N* denote the number of frogs, *D* denote the number of optimization space dimension. $n=l_1\times m_1=\cdots=l_s\times m_s$, l_i and m_i is positive integers and $l_1<l_2<\cdots<l_s$ $m_1>m_2>\cdots>m_s$. For adaptive grouping quantum-inspired shuffled frog leaping algorithm, called AGQISFLA, the optimization process can be described as follows.

(1) Initialize the frog group

According to Eq.(12) to determine the number of qubits *n*, according to Eq.(13) initialize phase of each particle, according to Eq.(9) to calculate the probability amplitude of 2^n each particle, where the first *D* probability amplitudes are the coding of the particles. Set the *j*th probability amplitude of the *i*th particle be x_{ij} , coding result can be expressed as the following equation.

$$\begin{cases} P_1=[x_{11},x_{12},\cdots,x_{1D}]^{\mathrm{T}} \\ P_1=[x_{21},x_{22},\cdots,x_{2D}]^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_n=[x_{N1},x_{N2},\cdots,x_{ND}]^{\mathrm{T}} \end{cases} . \tag{15}$$

Initialization phase update step $\Delta\theta_0$, the limited number of iteration *G*. Set the current iteration step *t*=1.

(2) Calculation of the objective function value

Set the *j*-dimensional variable range be $[MinX_j, MaxX_j]$, because of the probability amplitude x_{ij} values in the interval [0,1], it is need to make the solution space transformation. The transformation equation is below.

$$X_{ij}=\frac{1}{2}[MaxX_j(1+x_{ij})+MinX_j(1-x_{ij})], \tag{16}$$

where i = 1, 2, ⋯, *N*, j = 1, 2, ⋯, *D*.

With the above formula, calculate the objective function values of all frogs. And ascending objective function value, global optimal frog phase be $\hat{\boldsymbol{F}}_g=[\hat{\theta}_{g1},\hat{\theta}_{g2},\cdots,\hat{\theta}_{gn}]$, global optimal objective function value be $\hat{f}_g$,tLS is the total number of iteration.

(3) Frog Segmentation

The *N* sorted frogs cycle into *L* sub-groups, each group have *M* frogs, namely: Optimal frog divided into the first subgroup, suboptimal frog divided into the second subgroups, and so on, until all frog allocated.

(4) Update subgroup worst frog

The Each subgroup evolution times is *LS*, in each evolution, update subgroup worst frog , follow the steps in the previous section (1) to step (6). As such, the worst frog of each subgroup are updated *LS* × *n* times.

(5) Adaptive calculation of the number of subgroups

In each sub-group, the average value of the objective function is f_{avg}^{l} , where *l= 1, 2,.., L*. Mixing the subgroups, and arranged in ascending order according to the value of the objective function, remember the objective function is the average of the entire frog groups is f_{avg} , the number of $f_{avg}^{i} < f_{avg}$ is n_L^1 , the number of $f_{avg}^{i} > f_{avg}$ is n_L^2 .The following two steps to recalculate the number of sub-groups are *L* and the number of subgroups frog are *M*.

(a)If $\begin{cases} n_L^1 \geq n_L^2 \\ k < s \end{cases}$, than $\begin{cases} L = l_{k+1} \\ M = m_{k+1} \end{cases}$. (b) If $\begin{cases} n_L^1 < n_L^2 \\ k > 1 \end{cases}$,than $\begin{cases} l = l_{k-1} \\ M = m_{k-1} \end{cases}$.

(6) Update the global optimal solution

Let the optimal frog phase be $\boldsymbol{F}_g = [\theta_{g1}, \theta_{g2}, \cdots, \theta_{gn}]$, the corresponding objective function value be f_g . If $f_g < \hat{f}_g$, then $f_g = \hat{f}_g$, $\hat{\boldsymbol{F}}_g = \boldsymbol{F}_g$; otherwise $\hat{f}_g = f_g$, $\boldsymbol{F}_g = \hat{\boldsymbol{F}}_g$.

(7) Determine the termination condition

If t <G, t = t + 1, back to (3); otherwise, save the result, end.

VIII. COMPARATIVE EXPERIMENT

In this study, the 25 standard test functions are employed to verify the optimization ability of AGQISFLA, and compare with the traditional leapfrog algorithm (SFLA), adaptive grouping leapfrog algorithm (AGSFLA). All test functions are minimal value optimization, All functions belong to minimum optimization, where *D* is the number of independent variables, *Ω* is the solution space, $\boldsymbol{X}^*$ is the exact minimum point, $f(\boldsymbol{X}^*)$ is the corresponding minimum.

8.1. Test function

(1) $f_1(\boldsymbol{X}) = \sum_{i=1}^{D} x_i^2; \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(2) $f_2(\boldsymbol{X}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|; \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(3) $f_3(\boldsymbol{X}) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2; \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(4) $f_4(\boldsymbol{X}) = \max_{1 \leq i \leq D}(|x_i|); \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(5) $f_5(\boldsymbol{X}) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2); \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(6) $f_6(\boldsymbol{X}) = \sum_{i=1}^{D} [x_i + 0.5]^2; \Omega = [-100,100]^D; X^+ = [1,1,\mathrm{L}\ ,1]; f(X^+) = 0.$

(7) $f_7(\boldsymbol{X}) = \sum_{i=1}^{D} i x_i^4 (1 + random(0,1)); \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(8) $f_8(\boldsymbol{X}) = [x_i^2 - 10\cos(2\pi x_i) + 10]; \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(9) $f_9(\boldsymbol{X}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e;$

$\boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(10) $f_{10}(\boldsymbol{X}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1; \quad \boldsymbol{\Omega} = [-100,100]^D; \ \boldsymbol{X}^* = [0,0,\mathrm{L}\ ,0]; \ f(\boldsymbol{X}^*) = 0.$

(11) $f_{11}(\boldsymbol{X}) = \frac{1}{D}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i) + 78.3323314; \boldsymbol{\Omega} = [-100,100]^D; x_i^* = -2.903534; f(\boldsymbol{X}^*) = 0.$

(12) $f_{12}(\boldsymbol{X}) = \frac{\pi}{D}[10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2(1 + 10\sin^2(\pi y_{i+1})) + (y_D - 1)^2] + \sum_{i=1}^{D} u(x_i, 10, 100.4);$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, \quad -a \leq x_i \leq a \\ k(-x_i - a)^m, x_i, -a \end{cases} \quad y_i = 1 + \frac{1}{4}(x_i + 1);$$

$\boldsymbol{\Omega} = [-100,100]^D; \boldsymbol{X}^* = [-1,-1,\mathrm{L}\ ,-1]; f(\boldsymbol{X}^*) = 0.$

(13) $f_{13}(\boldsymbol{X}) = \frac{1}{D}[\sin^3(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2(1 + \sin^2(2\pi x_{i+1})) +$

$(x_D - 1)^2(1 + \sin^2(2\pi x_D))] + \sum_{i=1}^{D} u(x_i, 5, 100, 4);$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, \quad -a \le x_i \le a \\ k(-x_i - a)^m, x_i, -a \end{cases}$$

$\Omega = [-100,100]^D;\ X^* = [-1,-1,L\ ,-1];\ f(X^+) = 0.$

(14) $f_{14}(\boldsymbol{X}) = \sum_{i=1}^{D-1}(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i)\cos(4\pi x_{i+1}) + 0.3);$

$\boldsymbol{\Omega} = [-100,100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ ,0];\ \ f(\boldsymbol{X}^*) = 0.$

(15) $f_{15}(\boldsymbol{X}) = \sum_{K=1}^{D}\sum_{j=1}^{D}(\frac{y_{jk}^2}{4000} - \cos(y_{jk}) + 1);$

$y_{jk} = 100(x_k - x_j^2)^2 + (1 - x_j)^2;$

$\Omega = [-100,100]^D;\ \boldsymbol{X}^* = [1,1,L\ ,1];\ f(\boldsymbol{X}^*) = 0.$

(16) $f_{16}(\boldsymbol{X}) = \sum_{i=1}^{D/4}\left[[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} + 10x_{4i})^4];\right.$

$\boldsymbol{\Omega} = [-100,100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ ,0];\ \ f(\boldsymbol{X}^*) = 0.$

(17) $f_{17}(\boldsymbol{X}) = \sum_{i=1}^{D-1} g(x_i, x_{i+1}) + g(x_D, x_1) g(x,y) = (x^2 + y^2)^{0.25} \times [\sin^2(50(x^2 + y^2)^{0.1}) + 1];$

$\boldsymbol{\Omega} = [-100,100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ ,0];\ \ f(\boldsymbol{X}^*) = 0.$

(18) $f_{18}(\boldsymbol{X}) = 10D + \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i));\quad y_i = \begin{cases} x_i & |x_i| < 1/2 \\ round(2x_i)/2, & |x_i| \ge 1/2 \end{cases};$

$\boldsymbol{\Omega} = [-100,100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ ,0];\ \ f(\boldsymbol{X}^*) = 0.$

(19) $f_{19}(\boldsymbol{X}) = \sum_{i=1}^{D}\left\{\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k(x_i + 0.5))]\right\} - D\sum_{k=0}^{k\max}(a^k\cos(\pi b^k));$

$a = 0.5; b = 0.3;\ \ k\max = 30;\ \ \boldsymbol{\Omega} = [-100,100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ \ 0];\ \ f(\boldsymbol{X}^*) = 0.$

(20) $f_{20}(\boldsymbol{X}) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4;\quad \boldsymbol{\Omega} = [-100.100]^D;\ \ X^* = [0,0,L\ \ 0];\ \ f(X^+) = 0.$

(21) $f_{21}(\boldsymbol{X}) = \sum_{i=1}^{D}|x_i|^{i+1};\Omega = [-100,100]^D;\ X^+ = [0,0,L\ ,0];\ f(X^+) = 0.$

(22) $f_{22}(\boldsymbol{X}) = \sum_{i=1}^{D}|x_i\ \sin(x_i) + 0.1x_i|;\Omega = [-100,100]^D;\ X^+ = [0,0,L\ ,0];\ f(X^+) = 0.$

(23) $f_{23}(\boldsymbol{X}) = -\sum_{i=1}^{D-1}\left[\exp\frac{-(x_i^2 + x_{i+1}^2 + 0.5x_ix_{i+1})}{8}) \times \cos(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_ix_{i+1}})\right] + D - 1;$

$\boldsymbol{\Omega} = [-100.100]^D;\ \ \boldsymbol{X}^* = [0,0,L\ \ 0];\ \ f(\boldsymbol{X}^*) = 0.$

(24) $f_{24}(\boldsymbol{X}) = 1 - \exp(-0.5\sum_{i=1}^{D}x_i^2);\Omega = [-1,1]^D;\ X^+ = [0,0,L\ ,0];\ f(X^+) = 0.$

(25) $f_{25}(\boldsymbol{X}) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{D}x_i^2}) + 0.1\sum_{i=1}^{D}x_i^2;\Omega = [-100,100]^D;\ X^+ = [0,0,L\ ,0];\ f(X^+) = 0.$

8.2. The experimental scheme and parameter design

The total number of frog group is set to N = 100, in order to achieve adaptive grouping, the N is decomposed into the following eight situations, as follows.

$$\boldsymbol{N} = 2 \times 50 = 4 \times 25 = 5 \times 20 = 10 \times 10 = 20 \times 5 = 25 \times 4 = 50 \times 2 = 100 \times 1 \tag{17}$$

Among them, each instance is a grouping scheme, the first number is the number of sub-groups, and the second number is the number of frogs in the group.

SFLA and AGSFLA limit the number of iteration steps is set to *G* = 100 and *G* = 1000, Biggest jump step taken $D_{\max} = 5$; limited iteration number of AGQISFLA is set to *G* = 100, The subgroups iterations of these three

algorithms is set to *LS*=15;phase update step is set to $\Delta\theta_0 = 0.05\pi$.
For SFLA, each function are used eight kinds of groupings were optimized 50 times, namely, each function is independent optimization .400 times. 400 times optimal results averaged and the average value of a single iteration of the running time as a comparative index; for AGSFLA and AGQISFLA, The initial number of frog groups is *L* = 10, the group number of frog is *M* = 10, Each function independently optimized 50 times, taking the average of 50 times optimal results, with the average value of a single iteration of the running time as a comparison index.

8.3. Comparative Experiment Results

Experiments conducted using Matlab R2009a. For comparison, the average time of a single iteration of the function f_i is set to T_i, the average optimal results for O_i, *i* = 1, 2,…,25. All test functions, Taking *G*=50 as an example (f_{16} for *D* = 52), the results of such comparison are shown in Table 1, he average optimization results for *D*=100, are shown in Table 2.

For the function f_i, the average time of the single iteration of SFLA, AGSFLA, AGQISFLA is set to T_i^A , T_i^F , T_i^Q .the average optimal results is set to Q_i^A , Q_i^F , Q_i^Q .To facilitate a further comparison, Taking AGSFLA and SFLA as example, there are the following two formulas. Eq.(18) is the ratio of the average running time. Eq.(19) is the ratio of the average optimal results

$$\frac{T^A}{T^F} = \frac{\sum_{i=1}^{20} T_i^A / T_i^F}{25} \tag{18}$$

$$\frac{Q^A}{Q^F} = \frac{\sum_{i=1}^{20} Q_i^A / Q_i^F}{25} \tag{19}$$

For these three algorithms, the ratio of average running time and the average optimal results is shown in Table 3.

Table 1 The 50 times contrast of the average results for three algorithms optimization (*D* = 50)

	SFLA			AGSFLA			AGQISFLA	
f_i	$T_i(s)$	O_i		$T_i(s)$	O_i		$T_i(s)$	O_i
		$G=10^2$	$G=10^3$		$G=10^2$	$G=10^3$		$G=10^2$
f_1	0.0146	2.03E+03	3.09E+02	0.1452	2.61E+02	2.59E-008	0.1700	3:26E-015
f_2	0.0163	4.23E+08	3.41E+02	0.1103	5.30E+05	5.64E+02	0.2091	5:19E-008
f_3	0.0984	5.16E+04	8.11E+03	0.5064	5.55E+03	3.35E+02	1.2738	1:17E-015
f_4	0.0190	14.9593	10.4918	0.1055	15.3773	12.8575	0.2239	0:514605
f_5	0.0372	7.04E+07	7.73E+05	0.2028	7.42E+05	2.03E+02	0.3585	48:24962
f_6	0.0279	1.58E+03	43.23333	0.1434	85.70000	34.30000	0.2793	0
f_7	0.0341	2.84E+07	3.24E+03	0.1532	4.92E+04	0.001770	0.2827	3:55E-028
f_8	0.0249	2.40E+03	9.78E+02	0.1377	1.09E+03	9.71E+02	0.2221	3:16E-013
f_9	0.0316	17.18656	16.07727	0.1465	14.31819	13.61492	0.4036	0:1850413
f_{10}	0.0308	1.62E+02	3.149318	0.0740	4.434261	0.002976	0.3502	1:16E-015
f_{11}	0.0317	1.35E+04	1.21E+02	0.1606	1.45E+02	12.72304	0.3003	6:1248557
f_{12}	0.0773	8.40E+06	11.55566	0.4046	2.94E+02	17.54648	0.6654	2:99E-005
f_{13}	0.0772	2.59E+07	8.99E+03	0.3996	5.32E+04	1.00E+02	0.7448	0:0322664
f_{14}	0.0366	6.05E+03	9.57E+02	0.1735	7.94E+02	11.75275	0.4062	1:42E-013
f_{15}	0.3756	3.33E+13	2.35E+09	0.7196	1.41E+10	2.52E+03	3.1635	2:90E+002
f_{16}	0.0515	3.76E+07	3.98E+05	0.2787	1.97E+05	2.26E+03	0.6207	9:26E-023
f_{17}	0.0515	1.93E+02	1.62E+02	0.3485	1.77E+02	1.65E+02	0.5558	0:0016304
f_{18}	0.0447	2.42E+03	9.61E+02	0.3161	1.24E+03	9.68E+02	0.3490	1:26E-012
f_{19}	0.4529	58.03894	41.94107	0.8332	60.97842	44.34244	3.7331	6:8459530

f_{20}	0.0202	1.55E+07	1.19E+04	0.1772	3.08E+04	7.66E+03	0.2160	5:43E-015
f_{21}	0.0338	4.17E+65	2.50E+47	0.2809	4.13E+41	1.53E+15	0.3652	1:27E-206
f_{22}	0.0167	1.47E+02	74.83271	0.0969	71.211607	46.331449	0.1750	6:76E-009
f_{23}	0.0348	44.76880	43.062524	0.1367	45.373445	42.675330	0.2528	0:1789964
f_{24}	0.0158	0.073135	0.0317157	0.0996	0.0041286	1.01E-14	0.1133	0
f_{25}	0.0173	1.92E+02	15.005397	0.1037	24.769981	4.357839	0.1543	4:72E-014

Table2 The 50 times contrast of the average results for three algorithms optimization (D = 100)

f_i	SFLA			AGSFLA			AGQISFLA	
	$T_i(s)$	O_i		$T_i(s)$	O_i		$T_i(s)$	O_i
		$G=10^2$	$G=10^3$		$G=10^2$	$G=10^3$		$G=10^2$
f_1	0.0226	4.99E+03	1.02E+03	0.2278	2.16E+03	0.061430	0.2606	8:91E-015
f_2	0.0246	3.64E+51	6.45E+04	0.1723	1.27E+35	1.15E+03	0.3227	1:15E-007
f_3	0.1544	2.10E+05	3.12E+04	0.7998	1.41E+04	5.87E+03	1.9394	1:61E-013
f_4	0.0292	18.15317	13.80720	0.1683	17.63319	11.72408	0.3488	0:559791
f_5	0.0589	1.48E+08	2.98E+06	0.3147	2.58E+07	4.78E+02	0.5650	69:8203
f_6	0.0442	3.68E+03	3.46E+02	0.2277	8.68E+02	3.80E+02	0.4337	0
f_7	0.0531	1.43E+08	2.38E+05	0.2324	6.12E+06	6.29E+03	0.4421	1:39E-027
f_8	0.0385	5.66E+03	2.59E+03	0.2200	4.01E+03	2.95E+03	0.3377	013
f_9	0.0502	18.44872	17.75022	0.2237	17.56241	12.90171	0.6207	0:404291
f_{10}	0.0487	3.37E+02	6.909176	0.1128	27.02232	0.077702	0.5527	9:11E-014
f_{11}	0.0495	1.62E+04	2.06E+02	0.2549	5.62E+02	13.19321	0.4708	9:089264
f_{12}	0.1222	1.80E+07	17.88713	0.6367	3.51E+05	16.03746	1.0287	1:23E-004
f_{13}	0.1208	5.76E+07	7.37E+04	0.6048	1.03E+07	2.06E+02	1.1594	5:01E-002
f_{14}	0.0561	1.48E+04	3.04E+03	0.2604	5.33E+03	32.15902	0.6415	2:04E-013
f_{15}	0.5742	1.95E+14	2.05E+10	1.1437	6.90E+12	1.47E+04	4.7639	3:99E+002
f_{16}	0.0790	5.23E+07	7.03E+05	0.4235	1.03E+06	1.72E+04	0.9684	4:32E-022
f_{17}	0.0800	4.16E+02	3.54E+02	0.5331	3.73E+02	1.43E+02	0.8364	0:091603
f_{18}	0.0703	5.37E+03	2.70E+03	0.4950	4.30E+03	1.88E+03	0.5379	5:11E-012
f_{19}	0.6933	1.39E+02	1.16E+02	1.2734	1.44E+02	21.29647	5.7135	9:935560
f_{20}	0.0319	1.17E+05	4.09E+04	0.2741	1.15E+05	1.89E+04	0.3428	3:88E-014
f_{21}	0.0526	7.74E+84	1.71E+55	0.4231	1.87E+60	1.63E+17	0.5483	9:36E-187
f_{22}	0.0256	3.56E+02	1.98E+02	0.1549	2.64E+02	1.58E+02	0.2759	8:34E-008
f_{23}	0.0546	93.59604	92.42014	0.2130	94.61277	90.05981	0.4037	0:9899643
f_{24}	0.0245	0.183484	0.096426	0.1536	0.029085	1.12E-08	0.1811	0
f_{25}	0.0267	4.92E+02	91.40583	0.1608	2.06E+02	14.42596	0.2436	2:67E-013

Table 3 the ratio of average running time and the average optimal results

D	T^A / T^F	Q_i^A / Q_i^F		T^Q / T^A	$Q_{G=100}^Q / Q^A$		T^Q / T^F	$Q_{G=100}^Q / Q^F$	
		$G=10^2$	$G=10^3$		$G=10^2$	$G=10^3$		$G=10^2$	$G=10^3$
50	5.495	0.2784	0.5133	2.107	0.0819	0.0418	10.06	0.0067	0.0111
100	5.486	0.4143	0.3492	2.117	0.0602	0.0567	10.04	0.0054	0.0081
AVG	5.491	0.3463	0.4312	2.112	0.0710	0.0492	10.05	0.0060	0.0096

From Tab.1-Tab.3, the introduction of adaptive grouping strategy and quantum computing makes AGQISFLA single-step running time of about 10 times that of traditional SFLA. Therefore, in order to enhance the fairness of the comparison results, we not only need to look at the same contrast iteration number, and must be further investigated algorithm to optimize the results of comparison in the same time. This is the fundamental reason for the SFLA and AGSFLA the iteration number is set to G = 100 and G = 1000. According to the optimization results of f1 ~ f25, when G = 100, AGSFLA is about 0.35 times of SFLA; when G = 1000, AGSFLA is about 0.43 times of SFLA. This shows that the introduction of adaptive grouping strategy can indeed enhance the ability to optimize the algorithm. When G = 100, AGQISFLA (G = 100) is about 0.07 times of AGSFLA; when G = 1000, AGQISFLA (G = 100) of approximately 0.05 times of AGSFLA. This shows that the use of multi-bit probability amplitude coding and evolutionary mechanisms can indeed improve the algorithm optimization capabilities. From

Table 3, in the same iteration steps, optimization results of AGQISFLA only 6/1000 of SFLA; at the same time optimization, optimization results of AGQISFLA only one percent of SFLA. Experimental results show that the adaptive grouping and multi-bit probability amplitude coding can indeed significantly improve the ability to optimize the traditional leapfrog algorithm.
In this paper, when D = 50 and D = 100, the average results of the algorithm AGQISFLA is show in Fig.1, the average results of 50 times each function after optimization, the figure shows that when the dimension of AGQISFLA increases, there is a good stability.

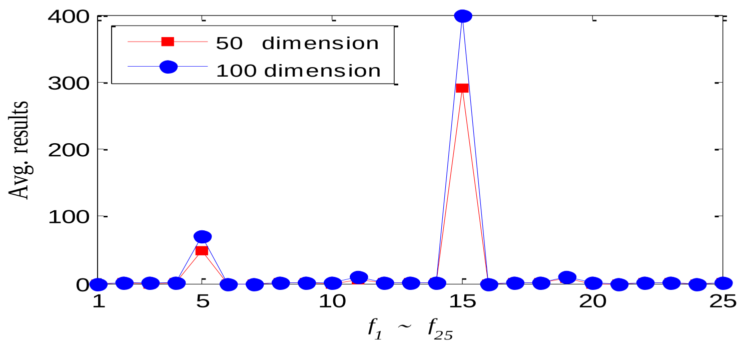

Figure 1 When D = 50 and D = 100, contrast optimization results of AGQISFLA

8.4 Analysis of experimental results

About adaptive grouping strategy, when f1 ~ f25 is 100-dimensional. Average number of iteration steps show in different groups AGQISFLA and AGSFLA, as shown in Figure 2 ~ 4.

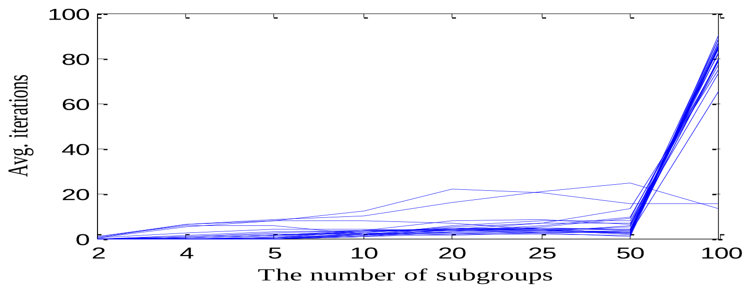

Figure 2 when G = 100, the average value of the iteration number of different groups of AGSFLA

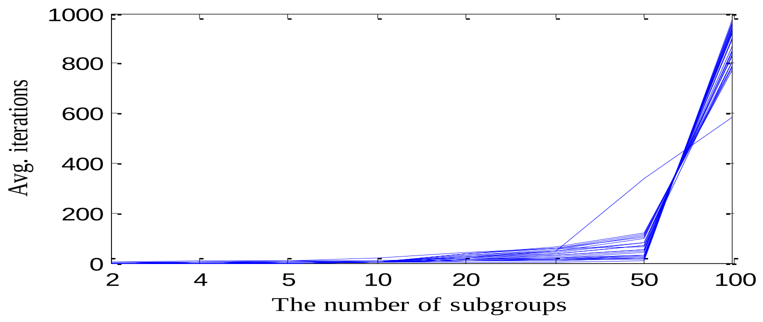

Figure 3 when G = 1000, the average value of the iteration number of different groups of AGSFLA

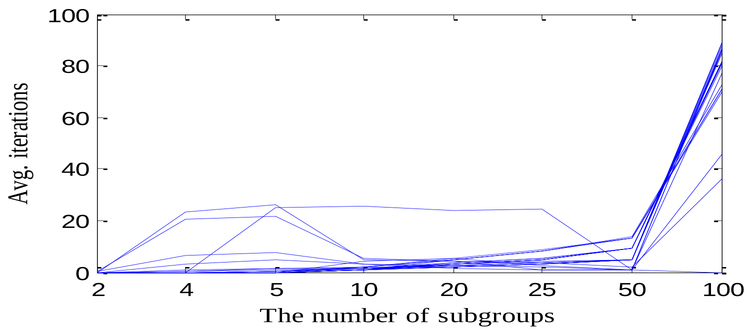

Figure 4 when *G* = 100, the average value of the iteration number of different groups of AGQISFLA

After adaptive grouping, the percentages of the number iterations for each subgroup in total iterations are shown in Table 4.

Table 4 the percentages of the number iterations for each subgroup in total iterations (%)

Algorithm	Iterations	Number of Subgroups							
		2	4	5	10	20	25	50	100
AGSFLA	100	0.14	1.22	2.04	3.32	5.08	5.64	5.78	76.78
AGSFLA	1000	0.02	0.08	0.17	0.35	1.87	3.18	6.66	87.67
AGQISFLA	100	0.04	2.22	3.62	2.86	4.10	5.18	6.52	75.46

The experimental results demonstrate the AGQISFLA long run time, high capacity optimization features, we give the following analysis.

First, as previously described, the more subgroups, the more number of frogs need to update for each iteration, and so the Longer the time of single iteration. Secondly, the individual coding method based on multi-bit probability amplitude, if the number of qubits encoded set is n, each iteration subgroup evolutionary times for LS, then each iteration, each subgroup of the worst frog are updated $LS \times n$ times, which in the extended run time, but also greatly improve the number updates of worst frog; at the same time, this by individually adjusting the phase of qubits to cycle update individual approach, making individuals more elaborate update, which also enhances the solution space of ergodic. Third, in the update strategy based on multi-bit revolving door, and draws on the thinking of particle swarm optimization, taking the leading role of subgroups optimal frogs and global optimum frog, To some extent, to avoid the tendency to fall into premature convergence; at the same time, the use of multi-bit quantum revolving door, one operation can be achieved for all the updates on the individual probability amplitude, the characteristics of quantum rotation gates guaranteed probability amplitude of the "length" unchanged, effectively avoiding iterative sequence divergence, and thus improve the convergence capability. In summary, the adaptive grouping and multi-bit encoding probability amplitude of these two mechanisms, at the expense of time in exchange for the ability to optimize, which is consistent with the theorem no free lunch.

IX. **CONCLUSION**

In this paper, a quantum inspired shuffled frog leaping algorithm algorithms is presented which encoded by adaptive grouping and multi-bit probability amplitude. Frog group individual coding approach is to use multi-qubits system in the ground state of the probability amplitude, frog group of individuals update method is a multi-bit quantum revolving door. Function extreme optimization results show that under the same running time, the optimization ability of proposed algorithm has greatly superior to the conventional leapfrog algorithm. thus revealing, adaptive grouping and multi-bit encoding probability amplitude of these two mechanisms is indeed an effective way to greatly improve traditional leapfrog algorithm optimization capabilities.

X. **ACKNOWLEDGEMENTS**

This work was supported by the Youth Foundation of Northeast Petroleum University (Grant No.2013NQ119).

XI. REFERENCES

[1] Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management.* 2003, 129(3): 210-225.

[2] Ding Wei-Ping, Wang Jian-Dong, Chen Shen-Bo, Chen Xue-Yun, Sen Xue-Hua. Rough attribute reduction with cross-entropy based on improved shuffled frog-leaping algorithm. *Journal of Nanjing University (Natural Sciences),* 2014, 50(2): 159-166.

[3] Cui Wen-Hua, Liu Xiao-Bing, Wang Wei, Wang Jie-Sheng. Product family design method based on fuzzy Cmeans clustering method optimized by improved shuffled frog leaping algorithm. *Journal of Dalian University of Technology,* 2013, 53(5): 760-765.

[4] Xio Chun-Cai, Hao Kuang-Rong, Ding Yong-Sheng. An improved shuffled frog leaping algorithm for solving controlled optimization problems of water bath stretching slot. *Journal of East China University of Science and Technology (Natural Science Edition)*, 2014, 40(1): 102-106.

[5] Zheng Shi-Lian, Luo Cai-Yi, Yang Xiao-Niu. Cooperative spectrum sensing for cognitive radios based on a modified shuffled frog leaping algorithm. *Acta Physica Sinica*, 2010, 59(5): 3611-3617.

[6] Zheng Shi-Lian, Yang Xiao-Niu. Swarm initialization of shuffled frog leaping algorithm for cooperative spectrum sensing in cognitive radio. *Acta Physica Sinica*, 2013, 62(7): 078405.

[7] Zhang Xiao-Dan, Huang Cheng-Wei, Zhao Li, Zou Cai-Rong. Recognition of practical speech emotion using inproved shuffled frog leaping algorithm. *Acta Acustica*, 2014, 39(2): 271-280.

[8] Wang Jie-Sheng, Gao Xian-Wen. Design of multivariable PID controller of electroslag remelting process based on improved shuffled frog leaping algorithm. *Control and Decision*, 2011, 26(11): 1731-1734.

[9] Luo Jian-Ping, Li Xia, Chen Min-Rong. Improved Shuffled Frog Leaping Algorithm for Solving CVRP. *Journal of Electronics & Information Technology*, 2011, 33(2): 429-434.

[10] Zhang Xiao-Dan, Zhao Li, Zou Cai-Rong. An improved shuffled frog leaping algorithm for solving constrained optimization problems. *Journal of Shandong university (Engineering Science)*, 2013, 43(1): 1-8,21.

[11] Xiao Ying-Ying, Chai Xu-Dong, Li Bo-Hu, Wang Qiu-Sheng. Convergence analysis of shuffled frog leaping algorithm and its modified algorithm. *Journal of Huazhong University of Science & Technology (Natural Science Edition)*, 2012, 40(7): 15-18,28.

[12] Sun J,Wu X J, Vasile P, Fang W, Lai C H, XuWB. quantum-behaved particle swarm optimization. *Information Sciences*, 2012, 193: 81-103.

[13] Lu T C, Yu G R. An adaptive population multi-objective quantum-inspired evolutionary algorithm for multiobjective0/1 knapsack problems. *Information Sciences*, 2013, 243: 39-56.

[14] Abdesslem L. A hybrid quantum inspired harmony search algorithm for 0-1 optimization problems. *Journal of Computational and Applied Mathematics*, 2013, 253: 14-25.

[15] Jiaquan G, Jun W. A hybrid quantum-inspired immune algorithm for multi-objective optimization. *Applied Mathematics and Computation*, 2011, 217: 4754-4770.

[16] Han Kuk-Hyun and Kim Jong-Hwan. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*,2002, 6(6): 580-593.

[17] Liu Xian-De, Li Pan-Chi, Yang Shu-Yun, Pan Jun-Hui, Xiao Hong, Cao Mao-Jun. Design and implementation of quantum-inspired differential evolution algorithm. *Journal of Signal Processing*, 2014, 30(6): 623-633.

[18] Giuliano Benenti, Giulio Casati, Giuliano Strini. Principles of Quantum Computation and Information (Column I: Basic Concepts). Singapore: World Scientific Publishing Company. 2004, 99-187.

[19] Cheng Dai-zhan, Qi hong-sheng. Semi-tensor product of matrix: theory and application (Second Edition).Beijing: Science Press. 2011: 1-24.