计算机科学
COMPUTER SCIENCE

混沌自适应量子萤火虫算法

刘晓楠, 安家乐, 何明, 宋慧超

引用本文

刘晓楠, 安家乐, 何明, 宋慧超.混沌自适应量子萤火虫算法[J]. 计算机科学, 2023, 50(4): 204-211.

LIU Xiaonan, AN Jiale, HE Ming, SONG Huichao. Chaotic Adaptive Quantum Firefly Algorithm [J]. Computer Science, 2023, 50(4): 204-211.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220100242

混沌自适应量子萤火虫算法

刘晓楠　安家乐　何　明　宋慧超
数字工程与先进计算国家重点实验室(信息工程大学)　郑州 450000
(prof.liu.xn@foxmail.com)

摘　要　为提升量子萤火虫算法(Quantum Firefly Algorithm,QFA)的搜索性能,解决其在面对部分问题时易陷入局部最优等问题,文中提出了一种引入混沌映射、邻域搜索以及自适应随机扰动的改进量子萤火虫算法——混沌自适应量子萤火虫算法(Chaotic Adaptive Quantum Firefly Algorithm,CAQFA)。该算法将混沌映射应用于种群的初始化阶段,提高初始种群的质量;并在更新阶段对当前种群中的最优个体进行邻域搜索,增强算法跳出局部最优的能力;对其他个体引入自适应的随机扰动,增加算法的随机性,在对搜索空间的探索和开发之间寻找平衡,以此提升算法的性能。文中选取了18个不同类型的基准函数对算法的性能进行测试,并将其与萤火虫算法(Firefly Algorithm,FA)、QFA以及量子粒子群优化(Quantum Particle Swarm Optimization,QPSO)算法进行对比。实验结果表明,CAQFA具有更好的搜索能力和稳定性,表现出了较强的竞争力。

关键词:量子萤火虫算法;群体智能;全局优化;混沌映射;测试函数
中图法分类号　TP301

Chaotic Adaptive Quantum Firefly Algorithm

LIU Xiaonan,AN Jiale,HE Ming and SONG Huichao
State Key Laboratory of Mathematical Engineering and Advanced Computing,PLA Information Engineering University,Zhengzhou 450000,China

Abstract　In order to improve the search performance of quantum firefly algorithm(QFA) and solve the problem that it is easy to fall into local optimality when facing some problems,an improved QFA with chaotic map,neighborhood search and adaptive random disturbance is proposed,named chaos adaptive quantum firefly algorithm(CAQFA). In this algorithm,chaotic map is applied to the initialization stage of the population to improve the quality of the initial population. In the update stage,the neighborhood search is carried out for the optimal individual of the current population to enhance the ability of the algorithm to jump out of the local optimization. The introduction of adaptive random disturbance to other individuals increases the randomness of the algorithm and achieves a balance between the exploration and development of search space,so as to improve the performance of the algorithm. Eighteen different types of benchmark functions are selected to test the performance of the algorithm. The test results show that CAQFA has better search ability,stability and strong competitiveness compared with firefly algorithm(FA),QFA and quantum particle swarm optimization(QPSO).

Keywords　Quantum firefly algorithm,Swarm intelligence,Global optimization,Chaotic map,Test functions

1　引言

萤火虫算法是受到自然界中萤火虫发出荧光完成求偶、觅食等行为的启发发展而来的一种群智能优化算法。该算法有两个版本:Glowworm Swarm Optimization(GSO)算法,由印度学者Krishnanand等[1]于2005年提出;Firefly Algorithm(FA),由剑桥学者Yang[2]于2009年提出。GSO和FA的灵感来源同样是仿生原理,但在具体实现上存在一定的差异。本文提出的改进算法在本质上基于FA,因此对GSO不作赘述。经典FA的灵感来源于萤火虫的显著特征:闪烁的光。在自然界中,这些光是用来吸引伙伴或警告捕食者的,当两只萤火虫之间的距离增大时,光的强度就会降低,吸引力也会降低。萤火虫算法就是模拟这种发光机制,使个体不断向更具吸引力的萤火虫个体移动。FA具有一些很好的特性[3],如参数少、进化过程简单、全局搜索能力强等,因此已被广泛应用于复杂优化调度[4]、电机控制[5]、图像分割[6]等领域。同时它也存在很多不足,主要表现在前期收敛速度慢以及群智能优化算法容易陷入局部最优。针对这些不足,相关研究人员也在基础的FA上做了许多的改进,主要分为三大类:自适应的参数控制、吸引模型的改进和混合改进策略[7]。例如:1)Fister等[8]使用四元数表示萤火虫个体,如果种群的最佳位置没有改善,则降低最亮萤火虫个体的亮度以改变其运动

到稿日期:2022-01-25　返修日期:2022-08-17
基金项目:国家超算郑州中心创新生态系统建设专项(201400210200);国家自然科学基金(61972413,61701539)
This work was supported by the Special Project for the Construction of Innovation Ecosystem of Zhengzhou Center of National Supercomputer (201400210200) and National Natural Science Foundation of China(61972413,61701539).
通信作者:安家乐(cnlarryan@foxmail.com)

路线，从而防止陷入局部最优；2）Farahani 等[9]为了提高算法的收敛速度，提出了在每次迭代中利用高斯分布吸引所有萤火虫移动到全局最优的方法；3）Xia 等[10]提出了一种基于 FA 和粒子群算法的混合算法，将种群划分成两部分，分别执行 FA 和粒子群算法，在达到某个设定的情况时交换两边的最优解以防止算法停滞，同时引入了 BFGS 拟牛顿法，来提高算法的局部搜索能力。这些改进在测试与实际应用中都有着不错的表现。

量子萤火虫算法（QFA）是由 Zhang 等[11]于 2014 年提出的，简单而言，就是将量子计算的概念引入 FA，在 FA 的基础上采用量子信息对萤火虫个体进行编码，扩展寻优空间，并采用量子旋转门对个体位置进行更新，逐渐使群体趋于最优。量子计算天然的并行性以及量子信息的叠加特性，可以帮助算法在前期快速收敛，同时以少量个体便能对整个搜索空间进行很好的利用。QFA 结合了量子计算和 FA 的优点，能够在探索和开发搜索空间之间获得一个平衡的折衷方案[12]。在简单优化问题中 QFA 表现良好，具有很快的收敛速度和较强的寻优能力。然而，在面对高维多极值优化问题时，QFA 就会表现出明显的颓势，因为算法收敛速度过快会导致种群的多样性迅速降低，如果当前种群最优值并非全局最优解，多样性的迅速降低会导致算法容易陷入局部最优无法跳出，算法早熟无法得到预期结果。

QFA 提出的最初目的是快速精确求解应用于医学 DR 图像增强的增益 Beta 变换中的最佳变换参数[11]，如今在其基础上又提出了许多新的量子计算和萤火虫算法相结合的改进算法，并已经用于解决实际问题。Tao 等[13]将遗传算法中交叉和突变的概念引入 QFA，通过交换部分量子位来实现交叉，通过改变部分量子位来实现突变，以此获取新的信息来保持种群的多样性，并将改进的 QFA 用于寻找桥梁颤振临界风速和频率的双参数优化模型的最优解，该方法表现出了较强的竞争力；Yassine 等[14]提出了一种量子启发的二进制萤火虫算法，将传统的二进制萤火虫算法与量子遗传算法相结合，用于解决多播路由问题；Shareef 等[15]利用类似思想提出了 QBFA（Quantum Binary Firefly Algorithm），用以优化网络重构，以提高配电系统的功率质量和可靠性；Fehmi 等[16]将萤火虫算法扩展成为基于多种群的算法，用于解决多模动态优化问题，该算法中的量子粒子被用来监测每个子群最佳解的邻域，以解决种群多样性的损失问题，同时量子策略也被用来响应动态事件，在已知的移动峰值基准上进行实验对比，表现良好。Zhao[17]针对 QFA 存在的 3 点不足，分别做出了针对性的改进：为应对算法在最优值附近震荡的情况，采取自适应步长调整策略，根据算法当前的运行状况决定合适的步长；由于 QFA 对初始分布比较敏感，改进算法将广义精英反向学习策略应用到 QFA，使得种群初始分布合理；边界以外的个体集中于边界上会导致算法易陷入局部最优，改进算法采用边界控制策略来增加这部分萤火虫个体的随机性。

本文将混沌映射、邻域搜索以及自适应的随机扰动引入 QFA，提出了一种改进的量子萤火虫算法——混沌自适应量子萤火虫算法（CAQFA）。该算法利用混沌映射对种群进行初始化，并将邻域搜索和自适应的随机扰动分别作用于当前种群最优个体和其他个体的更新阶段，优化了算法初始种群的构成，并在探索和开发之间寻求平衡，很大程度上降低了算法陷入局部最优无法跳出的可能性，提高了算法的搜索性能。本文使用 18 个不同类型的基准函数对 CAQFA 的性能进行仿真测试，结果表明其搜索精度、探索和开发能力以及稳定性均优于对比算法 FA，QFA，QPSO[18]。

2 量子萤火虫算法

量子萤火虫算法的整体流程与萤火虫算法相似，都是将种群中的个体利用不同的初始化方法随机分散到定义域内，再利用适应度函数计算每个个体的适应度，适应度低的个体会朝着适应度高的个体移动，不断往复直到达到最大迭代次数，最后输出最优个体值。两者的关键不同在于萤火虫个体的编码方式，以及个体向适应度更优的个体移动的更新过程。本文前两节介绍了关于量子计算的一些基础知识，第 3 节针对量子萤火虫算法的算法流程进行详细讲解。

2.1 量子编码

与传统萤火虫采用经典的 0，1 比特进行个体编码的方式不同，基于量子特性的萤火虫算法个体编码的基本单位是量子比特（Qbit），即量子位。量子比特的 0 和 1 分别对应着量子的上旋和下旋，具体的对应与底层实现相关，而且并无规定 0 和 1 的对应不能互换。而且除了 0 态和 1 态，量子比特还可处于两者的叠加态，即：

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle=\cos\varphi|0\rangle+\sin\varphi|1\rangle \tag{1}$$

因此，一个量子位可以表示为$[\alpha,\beta]^{\mathrm{T}}$，$\alpha^2$和$\beta^2$分别代表该量子位处于 0 态和 1 态的概率，即对量子位进行观测后塌缩为普通 0，1 比特的概率，其中$|\alpha|^2+|\beta|^2=1$，可将 α 和 β 分别表示为 $\cos\varphi$ 和 $\sin\varphi$。具有 n 个量子位的量子态则可表示为：

$$\begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_n\\ \beta_1 & \beta_2 & \cdots & \beta_n\end{bmatrix} \tag{2}$$

其中，对于每一个 $i\in\{1,2,\cdots,n\}$，都有$|\alpha|_i^2+|\beta|_i^2=1$，处于 n 个量子比特的叠加态，如果其中每个$|\alpha|_i^2$都不等于 0 或者 1，那么这个叠加态在未被观测时就有概率处于2^n种状态中的任何一种状态。

2.2 量子逻辑门

忽略量子萤火虫算法迭代更新过程的具体实现细节，它与萤火虫算法本质的不同之处在于量子萤火虫个体是通过量子逻辑门作用于每个量子比特，使得量子位旋转而实现的更新，并非普通萤火虫进行的简单数值运算。

经典计算机使用逻辑门处理信息，使其从一种状态转换为另外一种状态。如逻辑非门，它的真值表为 0→1 和 1→0，即可以使得一个比特在 0，1 状态之间转换。与之类似，量子计算中同样拥有量子逻辑门，用于处理量子位信息。量子逻辑门分为单量子比特门和多量子比特门，其中多量子比特门的原型是 C-NOT 门，即分为控制位和受控位的受控非门，而量子萤火虫算法中不涉及此类量子逻辑门，因此不做过多介绍。单量子比特门是更常用的一种门，可以由 2×2 的矩阵表示。

量子旋转门在量子萤火虫算法中具有重要地位，算法流程中的更新过程便是由它实现的。下面以量子旋转门为例演示量子门作用于量子比特的过程。

$$|\psi\rangle=\begin{bmatrix}\alpha\\ \beta\end{bmatrix}=\begin{bmatrix}\cos\varphi\\ \sin\varphi\end{bmatrix} \tag{3}$$

$$|\psi'\rangle=R_Y(\theta)|\psi\rangle=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\varphi\\ \sin\varphi\end{bmatrix}$$

$$=\begin{bmatrix}\cos(\varphi+\theta)\\ \sin(\varphi+\theta)\end{bmatrix} \tag{4}$$

其中，绕 Y 轴旋转的量子旋转门 $R_Y(\theta)$ 作用于量子比特 $|\psi\rangle$ 得到新的量子比特 $|\psi'\rangle$，其中 $|\psi'\rangle$ 是由 $|\psi\rangle$ 的相位旋转大小为 θ 的角度而得。

2.3 算法的具体流程

目前不同研究对原始量子萤火虫算法的描述并未统一，对算法的具体实现在部分论文之间甚至有着较大出入。本文根据文献[12-14]中关于量子萤火虫算法的介绍，对未改进的原始算法的基本流程进行了总结，如算法 1 所示。

算法 1 量子萤火虫算法

输入：d(问题的维度，即问题解决方案 $X=(x_1,x_2,\cdots,x_d)$ 的维数)；$D^1=[x^1_{\min},x^1_{\max}],\cdots,D^d=[x^d_{\min},x^d_{\max}]$(各维度的定义域，即 $X=(x_1,x_2,\cdots,x_d)\in D^1\times D^2\times\cdots\times D^d$)；s(j)(各维度中量子染色体所含量子比特个数，与算法精度相关)，其中 $j\in\{1,2,\cdots,d\}$；N(种群规模，即候选解决方案的个数)；f(X)(目标函数)；T(最大迭代次数)

输出：全局最佳解决方案 X

```
1.  t←0;
2.  随机生成包含 N 个量子萤火虫的初始种群 Qf_i(t)=[Qc_1^i,…,Qc_d^i], i∈{1,2,…,N};
3. while 不满足终止条件 do
4.      观测种群中每个量子萤火虫 Qf_i(t)=[Qc_1^i,…,Qc_d^i], 通过式(8)塌缩为普通二进制萤火虫 Bf_i(t)=[Bc_1^i,…,Bc_d^i];
5.      利用式(10)将每个二进制萤火虫 Bf_i(t)=[Bc_1^i,…,Bc_d^i] 转换成一种候选解决方案 X^i(t)=[x_1^i,x_2^i,…,x_d^i];
6.      根据目标函数 f(X) 计算并存放每个量子萤火虫对应解决方案的目标函数值 f(X^i(t));
7.      for p=1:N do
8.        for q=1:N do
9.          if f(X^p(t)) 劣于 f(X^q(t)) then
10.           通过式(11)更新量子萤火虫个体 Qf_p;
11.         end
12.       end
13.     end
14.     t←t+1;
15. end;
16. 对最终种群进行观测、转换、计算候选解决方案 X^i(t) 对应的目标函数值 f(X^i(t))，并进行排序，返回最小值，算法结束。
```

具体解释如下：

算法 1 的第 1 行中，量子萤火虫算法的输入主要有 5 个。首先需要定义待求解问题的维数 d，也就是该问题解决方案 $X=[x_1,x_2,\cdots,x_d]$ 的维数；针对每个维度需要给出对应的定义域，即在维度 j 上 X 的分量 x_j 允许取值的范围 $D^j=[x^j_{\min},x^j_{\max}]$，其中 $j\in\{1,2,\cdots,d\}$；量子染色体是量子遗传算法中的概念，在这里表示多个量子比特构成的叠加态，算法需给定在每个维度 j 上量子染色体所含量子比特的个数 $s(j)$，满足不等式(5)，其中 $p(j)$ 指 X 在维度 j 上分量 x_j 的精度，即小数点后有 $p(j)$ 位；给出整个搜索空间中具有的候选解决方案的个数 N；评判每个候选解决方案优劣的目标函数 $f(X)$，直至达到算法的最大迭代次数 T。

$$2^{s(j)}\geqslant|x^j_{\max}-x^j_{\min}|\times10^{p(j)} \tag{5}$$

算法 1 的第 2 行中，基于量子编码，利用不同的规则随机生成初始种群，即 N 个量子萤火虫个体。每个个体 Qf_i 都由 d 个量子染色体 Qc^i_j 构成，Qc^i_j 在量子萤火虫算法中代表个体 Qf_i 在维度 j 上的分量，由 $s(j)$ 个量子位组成，可表示为 2 行 $s(j)$ 列的矩阵，其中 $j\in\{1,2,\cdots,d\}$。因此 Qc^i_j 可由式(6)表示，其中每个量子比特用式(7)实现初始化。

$$Qc^i_j=\begin{bmatrix}\alpha^{i,j}_1 & \cdots & \alpha^{i,j}_{s(j)}\\ \beta^{i,j}_1 & \cdots & \beta^{i,j}_{s(j)}\end{bmatrix} \tag{6}$$

$$\begin{bmatrix}\alpha^{i,j}_k\\ \beta^{i,j}_k\end{bmatrix}=\begin{bmatrix}\cos\theta^{i,j}_k & -\sin\theta^{i,j}_k\\ \sin\theta^{i,j}_k & \cos\theta^{i,j}_k\end{bmatrix}\begin{bmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{bmatrix} \tag{7}$$

其中，$k\in\{1,2,\cdots,s(j)\}$，$\theta^i_k\in[0,\pi/2]$。

算法 1 的第 3－15 行中，该部分是整个算法的核心计算过程，是算法的精髓所在，也是一个迭代过程。在每次循环中，将种群的量子萤火虫个体转换为解决方案，并计算其适应度，根据适应度对种群进行更新，以得到更好的种群，重复这一循环直到满足终止条件。需要强调的是，量子萤火虫的更新并非直接改变其位置，而是改变个体处在可能位置的概率。下面将更细致地对该部分进行说明。

算法 1 的第 4 行中，正常情况下种群中的量子萤火虫个体处于量子叠加态，无法通过目标函数直接对量子态的个体进行适应度的计算，因此需要首先利用式(8)对个体 Qf_i 每个维度分量 Qc^i_j 的每个量子位 $[\alpha^{i,j}_k,\beta^{i,j}_k]^{\mathrm{T}}$ 进行观测得到 $b^{i,j}_k$，其中 $\varepsilon\in[0,1]$，每次都是在范围内随机选取。使得量子萤火虫塌缩成为普通的二进制萤火虫 $Bf_i=[Bc^i_1,\cdots,Bc^i_d]$，$Bc^i_j$ 可由式(9)表示。

$$b^{i,j}_k=\begin{cases}0, & \varepsilon\leqslant|\alpha^{i,j}_k|^2\\ 1, & \varepsilon>|\alpha^{i,j}_k|^2\end{cases} \tag{8}$$

$$Bc^i_j=[b^{i,j}_1,\cdots,b^{i,j}_{s(j)}] \tag{9}$$

算法 1 的第 5 行中，在塌缩成为普通的二进制萤火虫之后，还需将二进制个体 Bf_i 在各个维度上的分量 Bc^i_j 分别对应到定义域 D^j 上，通过式(10)逐维度转换，得到候选解决方案 $X^i=[x^i_1,x^i_2,\cdots,x^i_d]$

$$x^j_j=x^j_{\min}+\frac{\sum_{k=1}^{s(j)}2^kb^{i,j}_k}{2^{s(j)}-1}(x^j_{\max}-x^j_{\min}) \tag{10}$$

算法 1 的第 6 行中，在得到候选解之后便可以将解 $X^i(t)$ 代入目标函数 $f(X)$ 中得到目标函数值 $f(X^i(t))$，即处于该位置萤火虫的亮度，并将其存入数组以供后续使用，其中当代最优个体为 $Qf_*(t)$，最优解为 $X^*(t)$。

算法 1 的第 7－13 行中，对于种群中每个量子萤火虫个体 Qf_p，根据第 7 行保存的目标函数值找出所有比该个体亮度更强的个体 Qf_q，即当前个体 Qf_p 被 Qf_q 吸引，需要通过式(11)对被吸引的 Qf_p 进行更新。其中 $\mathrm{e}^{-\gamma(\|R\|)^2}R$ 代表量子萤火虫 Qf_q 对 Qf_p 的吸引力强度，$\gamma\in[0.01,100]$ 决定算法探索时朝局部最优解的收敛速度。

算法 1 中种群的更新过程符合萤火虫算法的基本规则：两只萤火虫之间相距越远，吸引力强度越小。运算符 $\circ$ 是求两个矩阵间的 Hadamard 积，运算符 $\circ$ 后的矩阵中的 α_k 和 β_k 指 $\circ$ 前面部分计算得到的第 k 列的值，目的是为了保证 $Qf_p(t+1)$ 中 $|\alpha|^2_k+|\beta|^2_k=1$，其中 $k\in\{1,\cdots,\tau\}$。

$$Qf_p(t+1)=(Qf_p(t)+\mathrm{e}^{-\gamma(\|R\|)^2}R)\circ\begin{bmatrix}\frac{1}{\sqrt{\alpha^2_1+\beta^2_1}} & \cdots & \frac{1}{\sqrt{\alpha^2_\tau+\beta^2_\tau}}\\ \frac{1}{\sqrt{\alpha^2_1+\beta^2_1}} & \cdots & \frac{1}{\sqrt{\alpha^2_\tau+\beta^2_\tau}}\end{bmatrix} \tag{11}$$

$$R=Qf_q(t)-Qf_p(t) \tag{12}$$

$$\|R\|=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|r_{ij}|^2} \tag{13}$$

$$\tau=\sum_{j=1}^{d}s(j) \tag{14}$$

算法 1 的第 16 行中，跳出迭代后，最后一次对种群进行观测、转换并求目标函数值，整个函数返回算法寻找到的最佳解决方案。

3 混沌自适应量子萤火虫算法

相比经典的萤火虫算法，量子萤火虫算法在全局搜索方面表现得更加优秀，一定程度上弥补了算法易早熟的缺点。但由于群智能优化算法的特性，算法仍有许多不足，主要包含以下几点：1）算法对初始种群有一定的敏感性，如果初始分布过差，则会影响算法的性能；2）同时，随着迭代次数的增加，种群多样性会不可避免地降低，即使引入量子特性所带来的随机性，算法在部分情况下仍然会陷入局部最优，导致过早收敛。本文提出了几种改进措施，在算法的初始化阶段、更新阶段以及局部搜索阶段同时做出改进，以提高算法的搜索性能以及稳定性。

3.1 改进方法

3.1.1 混沌映射

在大多数量子群智能优化算法的实现过程中，对种群的初始化操作都是简单地使用伪随机数发生器生成随机数，利用这个随机数将个体随机散布到搜索空间中的各个位置。然而伪随机数并非真正的随机数，在初始化时会存在个体分布不均的问题。与其他量子群智能优化算法相似，量子萤火虫算法对初始种群的分布具有敏感性，初始化的不均匀在一定程度上会影响算法的收敛速度和寻优精度。

混沌系统具有随机性、规律性、遍历性以及对初值敏感等特性[19]，基于这些特性，混沌映射已经被大量应用于科学研究中的各个领域。在群智能优化领域，混沌映射可被用于种群的初始化以及搜索阶段。混沌映射能够在区间[0，1]中生成混沌序列，对其做出适当改进，即可以此代替简单的伪随机数对量子萤火虫种群进行初始化，更好地将个体分布到整个搜索空间，以提高算法的整体性能。

适用于群智能优化算法领域的混沌映射主要包括：Logistic 映射（虫口映射）[20]、PWLCM 映射、Chebyshev 映射、Tent 映射（帐篷映射）等。其中取值适当的 Tent 映射在迭代 10^5 次之后生成的混沌序列在区间[0，1]的分布较为均匀，更加符合量子萤火虫算法对初始种群的需求，因此算法在初始化方面的改进主要围绕其展开。Tent 映射的表达式如式(15)所示，可以看到当 μ 取 2 时，混沌序列在[0，1]的分布最为均匀，因此本文参数选择 $\mu=2$。针对本文中的算法，式(7)中 $\theta_k^{i,j}$ 的取值如式(16)所示。

$$z_{k+1}=\begin{cases}\mu z_k, & z_k\in\left[0,\dfrac{1}{2}\right)\\ \mu(1-z_k), & z_k\in\left[\dfrac{1}{2},1\right]\end{cases} \tag{15}$$

$$\theta_k^{i,j}=z_{k+\mu^{i,j}}\times\frac{\pi}{2} \tag{16}$$

$$\mu^{i,j}=(i-1)\tau+\sum_{t=1}^{j-1}s(t) \tag{17}$$

其中，i 指种群中的第 i 个量子萤火虫个体，j 代表维度，k 是个体 i 在维度 j 的第 k 个量子比特，$\mu^{i,j}$ 是混沌序列的偏移量，$s(t)$ 是个体在维度 t 所包含量子比特的个数，τ 如式(14)所示。

3.1.2 邻域搜索

随着迭代次数的增加，种群多样性会迅速降低，全局搜索能力下降，逐渐收敛到种群最优值。但是种群最优并不等于待解决问题的全局最优值，如果当前种族最优为搜索空间中的一个局部极值，算法极有可能会过早收敛，陷入局部最优无法跳出。因此，本文将邻域搜索的概念引入量子萤火虫算法。

邻域指在某种邻域规则的定义下的解的集合，本节中便指与指定量子萤火虫个体具有邻域关系的个体集合。邻域搜索就是基于邻域的一类算法，其本质仍是局部搜索，即在邻域范围内寻找比当前个体更优的个体来代替。其中最主要的邻域规则（邻域结构）的设计，也就是按何种规则生成新个体，并将其作为当前个体邻域中的一员，它决定了算法搜索的范围，在一定程度上可以说是决定了算法搜索性能的优劣。

本文提出的改进量子萤火虫算法将对当前种群全局最优值进行邻域搜索，针对当前最优个体 $Qf_*(t)$ 随机生成邻域个体 $Qf_*'(t)$，邻域范围逐代减小。若生成的新个体适应度（亮度）有所提升，便以 $Qf_*'(t)$ 替换 $Qf_*(t)$ 作为新的种群全局最优个体；否则以一定概率随机替换种群中的其他个体。邻域个体的计算式如下：

$$Qf_*'(t)=\begin{bmatrix}\alpha'_{*,1,1} & \cdots & \alpha'_{*,d,s(d)}\\ \beta'_{*,1,1} & \cdots & \beta'_{*,d,s(d)}\end{bmatrix} \tag{18}$$

$$\begin{bmatrix}\alpha'_{*,j,k}\\ \beta'_{*,j,k}\end{bmatrix}=\begin{bmatrix}\cos(\theta_t\varepsilon'_{*,j,k}) & -\sin(\theta_t\varepsilon'_{*,j,k})\\ \sin(\theta_t\varepsilon'_{*,j,k}) & \cos(\theta_t\varepsilon'_{*,j,k})\end{bmatrix}\begin{bmatrix}\alpha_{*,j,k}\\ \beta_{*,j,k}\end{bmatrix} \tag{19}$$

即对 $Qf_*(t)$ 的每一个量子位施加一个量子旋转门，角度为 $\theta_t\varepsilon'_{*,j,k}$，其中 θ_t 是随着迭代次数不断变化的一个角度，$\varepsilon'_{*,j,k}$ 是一个位于区间[−1/2，1/2]均匀分布的随机数。

3.1.3 更新阶段的随机扰动

（1）随机扰动

原始量子萤火虫算法在更新阶段的实现如式(11)所示。由于更新操作仅与两个个体的位置相关，整个过程缺乏随机性，无法进行有效的全局搜索，可能会导致算法的收敛性能降低或陷入局部最优。面对这种情况，大多数群智能优化算法选择增加一个随机量来增加算法的随机性。因此，在本文提出对个体更新阶段的改进方法中，除了包含个体之间的吸引，还添加了随机扰动，以此增加种群的多样性。对式(11)的改进如式(20)所示，式中原有的变量代表的含义不变：

$$Qf_p(t+1)=(Qf_p(t)+\mathrm{e}^{-\gamma(\|R\|)^2}R+\alpha_t R_\varepsilon)\cdot\begin{bmatrix}\dfrac{1}{\sqrt{\alpha_1^2+\beta_1^2}} & \cdots & \dfrac{1}{\sqrt{\alpha_\tau^2+\beta_\tau^2}}\\ \dfrac{1}{\sqrt{\alpha_1^2+\beta_1^2}} & \cdots & \dfrac{1}{\sqrt{\alpha_\tau^2+\beta_\tau^2}}\end{bmatrix} \tag{20}$$

其中，α_t 是步长因子，控制随机扰动的步长；R_ε 可以被看作一个随机生成的个体，利用初始化种群时生成个体的方式创建，见式(6)和式(7)。

（2）动态调整步长因子

式(20)对原始算法的更新过程做出了改进，但同时也会引入一个新问题：步长因子 α_t 的值应该如何设置？如果设置得太大，在个体接近最优值时可能会出现震荡，即个体总的移动距离超过个体与最优值之间的距离，这种情况重复发生将

会出现运行中种群最优值在真正的最优值附近震荡的现象，导致算法性能降低甚至无法收敛；如果设置得太小，又将无法进行有效的全局搜索，添加扰动的意义将不复存在。因此，选择合适的α_t值对于随机扰动来说十分重要。

本文提出了一种动态调整步长因子 α 的方法：在算法前期，α_t的取值相对较大，即随机扰动占比较高，可以更好地进行全局搜索，防止错过真正的最优解；随着迭代次数的增加，以及吸引个体Qf_q和被吸引个体Qf_p之间的距离缩小，α_t同样会适当减小，在保证搜索能力的前提下，降低引入随机扰动导致的算法在最优值点震荡的可能性。α_t的取值方法如式(21)所示：

$$\alpha_t=(0.97)^{\frac{500t}{T}}\alpha\times\frac{\|R\|}{\tau} \tag{21}$$

其中，α_t随着迭代次数 t 的增加和两个体距离的缩短而降低，其中 α 为初始步长因子，根据具体的问题进行设置，$\|R\|$和 τ 的定义见式(13)和式(14)。0.97 的指数之所以不直接采用 t 转而使用 $500t/T$ 的原因是：$(0.97)^t$并不会因为最大迭代次数的改变而改变其函数曲线，总是会在较小的迭代次数时便已经接近于 0。可以算出，$(0.97)^n$在迭代进行到第 200 代时已经十分接近于 0，即失去了全局搜索能力，但这种情况与最大迭代次数无关。而对于不同的优化问题，算法得到较好结果所需的最大迭代次数会有较大不同。因此，对于一些复杂的优化问题，所需的迭代次数远超 200，算法将会过早地失去全局搜索能力，陷入局部最优；选择 500 这个数字则是由于 $0.97^{200}\approx0.0023$ 已经接近于 0，即在后续的迭代中全局搜索的能力也将接近于零，选择 500 使得在大约 2/5 的迭代周期中全局搜索能力较强，而在剩余的迭代周期中主要进行局部搜索，全局搜索和局部搜索的相对比例是在实验过程中不断优化得到的经验值。

3.2 算法流程

算法的流程大体与算法 1 中展示的相同，但在对应阶段应使用上文提出的改进措施，具体如下：

第 3 行：采用混沌映射改进中的式(16)计算式(7)中的$\theta_k^{i,j}$。

第 8－14 行：将式(11)替换成式(20)，同时对当前种群最优个体利用式(18)和式(19)进行邻域搜索。

4 仿真实验

为评估算法的性能，CAQFA 应在一组足够大的测试函数集上进行实验，并与其他算法进行对比，该测试套件应该包含各类问题，以确定算法的适用类型。文献[21]中有 140 种测试函数，包括 24 个单峰函数和 116 个多峰函数，其中又可分为低维测试函数和高维测试函数。CEC 测试函数集已经更新到了 2021 版本，从其最早版本至今也包含了大量的测试函数。然而，对于如何选择合适的测试函数来评估算法性能，研究者们仍未达成共识。

4.1 测试函数

本文选取 18 个不同类型的标准基准函数，根据它们的模态、维数、形状等特征，对改进算法的探索能力、开发能力、收敛速度等性能进行仿真测试，并将其与萤火虫算法 FA、原始量子萤火虫算法 QFA、量子粒子群算法 QPSO 进行比对。测试函数的相关信息如表 1 所列，不同维度自变量的定义域相同，函数f_8，f_9，f_{11}，f_{12}，f_{14}，f_{15}是单峰函数，f_1-f_7，f_{10}，f_{13}，$f_{16}-f_{18}$是多峰函数，包含可分离函数与不可分离函数，同时表现为 Bowl-Shaped，Plate-Shaped，Valley-Shaped，Steep Ridges/Drops 等多种函数形式。

表 1 测试函数

Table 1 Test functions

Function	Definition	Dim	Range	$f_{\min}$
Ackley	$f_1(X)=-20e^{\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)}+e^{\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)}+20+e$	20	$[-32.768,32.768]$	0
Drop-Wave	$f_2(X)=-\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$	2	$[-5.12,5.12]$	−1
Griewank	$f_3(X)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]$	0
Levy N. 13	$f_4(X)=\sin^2(3\pi x_1)+(x_1-1)^2[1+\sin^2(3\pi x_2)]+(x_2-1)^2[1+\sin^2(2\pi x_2)]$	2	$[-10,10]$	0
Rastrigin	$f_5(X)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$	30	$[-5.12,5.12]$	0
Schaffer N. 2	$f_6(X)=0.5+\frac{\sin^2(x_1^2-x_2^2)-0.5}{[1+0.001(x_1^2+x_2^2)]^2}$	2	$[-100,100]$	0
Schwefel 2.26	$f_7(X)=418.9829d-\sum_{i=1}^{d}x_i\sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]$	0
Sphere	$f_8(X)=\sum_{i=1}^{d}x_i^2$	20	$[-5.12,5.12]$	0
Sum Squares	$f_9(X)=\sum_{i=1}^{d}ix_i^2$	30	$[-10,10]$	0
Booth	$f_{10}(X)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	2	$[-10,10]$	0
Matyas	$f_{11}(X)=0.26(x_1^2+x_2^2)-0.48x_1x_2$	2	$[-10,10]$	0
Zakharov	$f_{12}(X)=\sum_{i=1}^{d}x_i^2+(\sum_{i=1}^{d}0.5ix_i)^2+(\sum_{i=1}^{d}0.5ix_i)^4$	20	$[-5,10]$	0
Three-Hump Camel	$f_{13}(X)=2x_1^2-1.05x_1^4+\frac{x_1^6}{6}+x_1x_2+x_2^2$	2	$[-5,5]$	0
Rosenbrock	$f_{14}(X)=\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-5,10]$	0

（续表）

Function	Definition	Dim	Range	$f_{\min}$
Easom	$f_{15}(X)=-\cos(x_1)\cos(x_2)e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}$	2	$[-100,100]$	-1
Michalewicz	$f_{16}(X)=-\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	5	$[0,\pi]$	-4.6877
Beale	$f_{17}(X)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1x_2^3)^2$	2	$[-4.5,4.5]$	0
Perm Function d, Beta	$f_{18}(X)=\sum_{i=1}^{d}\left(\sum_{j=1}^{d}(j^i+\beta)\left(\left(\frac{x_j}{j}\right)^i-1\right)\right)^2$	3	$[-d,d]$	0

4.2 参数设置及测试结果

FA 和 QPSO 算法的代码思路及参数设置原则分别来自文献[2]和文献[18]，种群大小 N 设置为 50 和 100，迭代次数 T 设置为 1 000 和 3 000，其余主要参数值如表 2 所列。为减小误差，每种组合均进行 50 次实验，取实验的平均值进行比较。仿真结果如表 3 所列，其中 Mean 代表最优解的平均值，Std 是标准差，两者分别用于衡量算法的搜索能力和稳定性。

表 2 主要参数设置

Table 2 Settings of main parameters

Algorithm	Parameters
FA	$\beta_0=1,\gamma=1$
QFA	$\gamma=1$
QPSO	$\beta=1$
CAQFA	$\alpha=1,\gamma=1$

表 3 测试结果

Table 3 Test results

Func	N	T	FA		QFA		QPSO		CAQFA	
			Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	50	1 000	2.00×10^{-13}	2.19×10^{-13}	1.89×10^{-13}	1.99×10^{-13}	1.01×10^{-13}	1.39×10^{-10}	7.44×10^{-16}	5.11×10^{-17}
	100	1 000	2.31×10^{-13}	2.55×10^{-13}	1.92×10^{-13}	2.07×10^{-13}	8.37×10^{-13}	1.04×10^{-12}	9.13×10^{-16}	4.86×10^{-16}
	50	3 000	1.47×10^{-13}	1.62×10^{-13}	1.66×10^{-13}	1.81×10^{-13}	4.33×10^{-11}	6.75×10^{-11}	5.94×10^{-16}	3.57×10^{-17}
	100	3 000	1.73×10^{-13}	1.91×10^{-13}	1.59×10^{-13}	1.77×10^{-13}	2.46×10^{-13}	3.11×10^{-13}	8.33×10^{-16}	8.69×10^{-17}
f_2	50	1 000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
	100	1 000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
	50	3 000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
	100	3 000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
f_3	50	1 000	6.94×10^{-3}	1.13×10^{-3}	2.46×10^{-6}	4.13×10^{-6}	2.21×10^{-2}	3.37×10^{-2}	7.07×10^{-8}	8.19×10^{-8}
	100	1 000	4.67×10^{-3}	1.45×10^{-4}	1.63×10^{-6}	2.33×10^{-6}	1.38×10^{-2}	2.77×10^{-2}	2.84×10^{-8}	4.96×10^{-8}
	50	3 000	6.30×10^{-3}	4.87×10^{-4}	5.82×10^{-7}	8.23×10^{-7}	6.48×10^{-4}	1.21×10^{-3}	4.11×10^{-8}	5.54×10^{-8}
	100	3 000	4.11×10^{-3}	6.74×10^{-5}	3.06×10^{-7}	5.89×10^{-7}	2.93×10^{-4}	6.04×10^{-4}	8.13×10^{-8}	1.03×10^{-8}
f_4	50	1 000	7.86×10^{-31}	3.59×10^{-31}	2.49×10^{-31}	1.45×10^{-31}	9.41×10^{-17}	4.19×10^{-17}	0.0	0.0
	100	1 000	5.34×10^{-31}	9.67×10^{-31}	1.35×10^{-31}	9.64×10^{-31}	6.88×10^{-19}	8.41×10^{-18}	0.0	0.0
	50	3 000	2.96×10^{-31}	8.94×10^{-31}	2.43×10^{-31}	1.37×10^{-31}	6.45×10^{-23}	2.36×10^{-23}	0.0	0.0
	100	3 000	2.35×10^{-31}	2.37×10^{-31}	1.35×10^{-31}	5.41×10^{-31}	1.38×10^{-23}	9.84×10^{-24}	0.0	0.0
f_5	50	1 000	1.52×10	3.73	2.53×10^{-2}	4.68×10^{-2}	3.87×10^{2}	1.16×10	1.31×10^{-6}	3.45×10^{-6}
	100	1 000	7.64	1.65	9.09×10^{-3}	9.68×10^{-4}	3.46×10^{2}	1.35×10	1.57×10^{-8}	5.97×10^{-9}
	50	3 000	4.17	1.56	4.37×10^{-3}	6.94×10^{-3}	4.25×10	7.33	0.0	0.0
	100	3 000	1.07	7.66×10^{-1}	2.10×10^{-4}	2.39×10^{-4}	2.04×10	8.49	0.0	0.0
f_6	50	1 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	1 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	50	3 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	3 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
f_7	50	1 000	7.69×10	4.65×10	6.51×10^{-2}	1.59×10^{-1}	5.17×10	1.64×10^{2}	9.18×10^{-6}	1.46×10^{-6}
	100	1 000	5.79×10	6.48×10	4.38×10^{-2}	5.64×10^{-2}	7.09	2.36×10	6.31×10^{-6}	6.63×10^{-7}
	50	3 000	3.48×10	4.89	1.64×10^{-2}	4.37×10^{-2}	4.66×10	8.11×10	2.64×10^{-6}	8.94×10^{-7}
	100	3 000	3.03×10	1.65	1.21×10^{-2}	1.68×10^{-2}	5.63	2.47×10	6.15×10^{-7}	3.01×10^{-7}
f_8	50	1 000	1.06×10^{-54}	1.64×10^{-54}	1.02×10^{-54}	6.13×10^{-55}	1.34×10^{-44}	9.46×10^{-44}	0.0	0.0
	100	1 000	8.16×10^{-55}	8.72×10^{-55}	8.28×10^{-55}	4.79×10^{-55}	2.83×10^{-51}	5.18×10^{-51}	0.0	0.0
	50	3 000	2.31×10^{-107}	3.49×10^{-107}	2.37×10^{-107}	5.39×10^{-107}	9.84×10^{-48}	6.15×10^{-49}	0.0	0.0
	100	3 000	1.85×10^{-107}	1.87×10^{-107}	1.82×10^{-107}	2.37×10^{-107}	1.67×10^{-51}	3.24×10^{-51}	0.0	0.0
f_9	50	1 000	3.77×10^{-53}	6.51×10^{-53}	3.68×10^{-53}	4.81×10^{-52}	7.67×10^{-39}	3.49×10^{-38}	0.0	0.0
	100	1 000	2.68×10^{-53}	8.94×10^{-53}	2.92×10^{-53}	6.13×10^{-53}	1.32×10^{-43}	7.12×10^{-43}	0.0	0.0
	50	3 000	8.80×10^{-105}	9.84×10^{-105}	8.68×10^{-106}	4.65×10^{-105}	5.78×10^{-43}	2.36×10^{-42}	0.0	0.0
	100	3 000	5.65×10^{-105}	6.51×10^{-105}	6.49×10^{-106}	6.63×10^{-106}	1.39×10^{-50}	4.66×10^{-50}	0.0	0.0
f_{10}	50	1 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	1 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	50	3 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	3 000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
f_{11}	50	1 000	8.47×10^{-32}	1.33×10^{-31}	4.76×10^{-43}	1.65×10^{-42}	7.93×10^{-29}	4.86×10^{-28}	4.36×10^{-60}	6.41×10^{-60}
	100	1 000	2.63×10^{-34}	4.65×10^{-34}	7.96×10^{-44}	8.17×10^{-44}	3.43×10^{-36}	8.79×10^{-36}	8.70×10^{-61}	8.96×10^{-61}
	50	3 000	1.38×10^{-60}	6.43×10^{-60}	5.53×10^{-81}	6.13×10^{-81}	6.68×10^{-55}	4.89×10^{-54}	9.02×10^{-113}	5.13×10^{-112}
	100	3 000	5.77×10^{-61}	3.37×10^{-60}	1.17×10^{-81}	3.94×10^{-81}	3.74×10^{-67}	3.51×10^{-67}	2.44×10^{-113}	2.79×10^{-113}

(续表)

Func	N	T	FA		QFA		QPSO		CAQFA	
			Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_{12}	50	1000	7.16×10^{-23}	1.68×10^{-22}	3.79×10^{-26}	9.48×10^{-26}	4.15×10^{-23}	1.61×10^{-22}	4.51×10^{-47}	6.41×10^{-47}
	100	1000	4.64×10^{-23}	4.61×10^{-23}	6.33×10^{-27}	6.74×10^{-27}	9.78×10^{-31}	1.39×10^{-30}	3.56×10^{-47}	7.96×10^{-47}
	50	3000	5.43×10^{-54}	6.36×10^{-54}	5.72×10^{-54}	1.65×10^{-54}	1.86×10^{-50}	7.35×10^{-50}	1.27×10^{-106}	2.97×10^{-106}
	100	3000	3.84×10^{-54}	4.31×10^{-54}	3.92×10^{-54}	6.15×10^{-55}	6.79×10^{-51}	4.61×10^{-51}	8.76×10^{-107}	1.36×10^{-106}
f_{13}	50	1000	5.31×10^{-37}	4.87×10^{-36}	6.89×10^{-46}	6.30×10^{-46}	4.16×10^{-33}	9.04×10^{-34}	2.96×10^{-76}	4.24×10^{-76}
	100	1000	2.77×10^{-37}	5.12×10^{-37}	4.42×10^{-46}	1.49×10^{-46}	3.51×10^{-34}	2.85×10^{-34}	7.13×10^{-77}	1.00×10^{-76}
	50	3000	1.16×10^{-59}	9.63×10^{-59}	1.45×10^{-59}	2.85×10^{-59}	1.61×10^{-59}	6.31×10^{-59}	3.73×10^{-112}	3.21×10^{-112}
	100	3000	3.29×10^{-60}	2.85×10^{-59}	3.31×10^{-60}	3.61×10^{-60}	3.27×10^{-60}	4.97×10^{-59}	6.69×10^{-113}	9.14×10^{-113}
f_{14}	50	1000	5.98×10	1.34×10	5.19×10	4.38×10^{-1}	2.85×10	2.22	6.31×10^{-3}	9.86×10^{-3}
	100	1000	2.00×10	8.40×10	4.78×10	3.71×10^{-1}	4.14×10	3.70	1.45×10^{-3}	5.43×10^{-3}
	50	3000	1.99×10	8.91×10	2.39×10	5.47×10^{-1}	2.78×10	1.17×10^{-1}	5.40×10^{-5}	4.85×10^{-4}
	100	3000	1.19×10	6.37×10	1.20×10	4.90×10^{-1}	2.78×10	9.91×10^{-2}	6.14×10^{-5}	1.73×10^{-4}
f_{15}	50	1000	-9.30×10^{-1}	2.65×10^{-1}	-9.66×10^{-1}	2.00×10^{-1}	-9.73×10^{-1}	3.11×10^{-1}	-1.0	0.0
	100	1000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
	50	3000	-9.57×10^{-1}	2.24×10^{-1}	-1.0	0.0	-1.0	0.0	-1.0	0.0
	100	3000	-1.0	0.0	-1.0	0.0	-1.0	0.0	-1.0	0.0
f_{16}	50	1000	-4.3721	1.65×10^{-3}	-4.5907	4.61×10^{-3}	-4.0417	5.61×10^{-3}	-4.6877	4.77×10^{-14}
	100	1000	-4.3885	3.34×10^{-4}	-4.6113	2.47×10^{-4}	-4.1567	4.72×10^{-3}	-4.6877	4.35×10^{-14}
	50	3000	-4.6877	4.53×10^{-11}	-4.6877	0.0	-4.2643	4.66×10^{-3}	-4.6877	0.0
	100	3000	-4.6877	3.71×10^{-11}	-4.6877	0.0	-4.3764	3.34×10^{-3}	-4.6877	0.0
f_{17}	50	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	50	3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100	3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
f_{18}	50	1000	4.20×10^{-3}	7.41×10^{-3}	4.96×10^{-5}	1.75×10^{-4}	5.61×10^{-3}	7.49×10^{-3}	6.43×10^{-7}	6.76×10^{-7}
	100	1000	4.13×10^{-3}	9.85×10^{-3}	2.73×10^{-5}	6.14×10^{-5}	1.64×10^{-3}	5.68×10^{-3}	1.65×10^{-7}	2.17×10^{-7}
	50	3000	3.99×10^{-3}	2.06×10^{-3}	3.54×10^{-8}	7.98×10^{-8}	4.83×10^{-5}	3.47×10^{-5}	3.13×10^{-15}	2.98×10^{-15}
	100	3000	3.86×10^{-3}	9.48×10^{-4}	8.94×10^{-9}	5.61×10^{-8}	4.71×10^{-5}	1.67×10^{-5}	2.67×10^{-15}	2.42×10^{-15}

4.3 性能分析

函数f_8,f_9,f_{11},f_{12},f_{14},f_{15}是单峰函数,只有一个全局最优解,不存在其他局部极值点,因此可以利用它们观察算法对搜索空间的开发能力;函数f_1-f_7,f_{10},f_{13},$f_{16}-f_{18}$是多峰函数,具有多个局部最优值,并且局部最优值的数量随着维度的增加呈指数增长,因此主要用于检验算法在搜索空间中的探索能力。

从表3可以看出,在与3种同类算法FA,QFA,QPSO的比较中,CAQFA在所有测试函数上的搜索能力和稳定性都是第一或并列第一。其中较为典型的有函数f_{14},该函数是一个单峰函数,全局最小值位于一个狭窄的抛物线山谷中,其他3种算法都落入了其陷阱,陷入了局部最优值无法跳出,导致算法过早收敛,而CAQFA则取得了不错的结果,表现出了较强的局部搜索能力。而在后续针对函数f_{14}进行的细化实验表明,3种改进措施中,自适应的随机扰动对算法优化起到了重要的作用,在与其他两种改进措施结合之后,算法的稳定性和准确性有了明显的提高,3种改进措施相辅相成;在对多峰函数的优化中,整体的搜索精度和稳定性都有很大的提升,表现出了较强的探索能力,即全局搜索能力。

利用IBM SPSS统计软件对4种算法进行Friedman检验[22],得到各钟算法的秩均值,如表4所列。

表4 算法的秩均值

Table 4 Rank mean of algorithms

Algorithm	Mean rank value
FA	3.01
QFA	2.29
QPSO	3.27
CAQFA	1.43

可以看到,CAQFA的秩均值最小,然后分别是QFA,FA,QPSO,秩均值越小,表示算法表现越好。因此,测试结果表明,相比其他3种对比算法,CAQFA的综合性能最好。

结束语 本文将混沌映射、邻域搜索以及自适应的随机扰动引入量子萤火虫算法的不同阶段,提出了CAQFA。混沌映射可以提高初始种群的质量,邻域搜索和自适应的随机扰动则可以增强算法跳出局部最优的能力,使算法在探索和开发之间取得平衡,从而提高算法的综合性能。在18个基准函数上的测试结果表明,CAQFA显著提升了QFA的性能,在与其他两种算法的对比中也表现出了很大的优越性。但是算法仍有许多可优化的问题以及进一步研究的方向,例如邻域搜索策略的选择、选取最优参数、引入分组优化或者其他策略、在实际问题上的应用等,未来可将对这些方面的研究作为当前工作的延伸。

参 考 文 献

[1] KRISHNANAND K N, GHOSE D. Detection of multiple source locations using a firefly metaphor with applications to collective robotics[C]// Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, 2005: 84-91.

[2] YANG X S. Firefly Algorithms for Multimodal Optimization [C]// International Symposium on Stochastic Algorithms. Berlin/Heidelberg: Springer, 2009.

[3] AN J L, LIU X N, HE M, et al. Survey of Quantum Swarm Intelligence Optimization Algorithm [J/OL]. Computer Engineering and Applications, 2022: 1-15. http://kns.cnki.net/kcms/detail/11.2127.tp.20211201.2057.008.html.

[4] ZHAO J, CHEN W P, XIAO R B, et al. Firefly algorithm with

division of roles for complex optimal scheduling[J]. Frontiers of Information Technology & Electronic Engineering, 2021, 22(10):1311-1333.

[5] BAZI S,BENZID R,BAZI Y,et al. A Fast Firefly Algorithm for Function Optimization:Application to the Control of BLDC Motor[J]. Sensors,2021,21(16):5267.

[6] CHEN K,CHEN F,DAI M,et al. Fast image segmentation with multilevel threshold of two-dimensional entropy based on firefly algorithm[J]. Optics and Precision Engineering,2014,22(2):517-523.

[7] WANG H,WANG W J,XIAO S Y. A survey of firefly algorithms[J]. Journal of Nanchang Institute of Technology,2019,38(4):71-77.

[8] FISTER I,YANG X S,BREST J,et al. Modified firefly algorithm using quaternion representation[J]. Expert Systems With Applications,2013,40(18):7220-7230.

[9] FARAHANI S M,ABSHOURI A A,NASIRI B,et al. [J]. International Journal of Machine Learning & Computing,2011,1(5):448-453.

[10] XIA X W,LING G,HEGl,et al. A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm[J]. Journal of Computational Science,2018,26:488-500.

[11] ZHANG J F,DU X X,WANG B. Adaptive enhancement of medical DR image based on quantum firefly and gain beta[J]. Microelectronics and Computer,2014,31(5):135-139.

[12] FAROUQ Z,SAAD H,RAMDANE M. A Novel Quantum Firefly Algorithm for Global Optimization[J]. Arabian Journal for Science and Engineering,2021,46(9):8741-8759.

[13] TAO S B,LIU D Z,TANG A P,et al. Bridge Critical State Search by Using Quantum Genetic Firefly Algorithm[J]. Shock and Vibration,2019,2019.

[14] YASSINE M,DALILA A,AMAR R,et al. A quantum-inspired binary firefly algorithm for QoS multicast routing[J]. International Journal of Metaheuristics,2017,6(4):309-333.

[15] SHAREEF H,MOHAMED W,LING A,et al. Power quality and reliability enhancement in distribution systems via optimum network reconfiguration by using quantum firefly algorithm[J]. International Journal of Electrical Power and Energy Systems,2014,58:160-169.

[16] FEHMI B O,ADIL B. Quantum firefly swarms for multimodal dynamic optimization problems[J]. Expert Systems with Applications,2019,115:189-199.

[17] ZHAO J L. Improvement of quantum firefly algorithm and its application in image threshold segmentation [D]. Yinchuan:Ningxia University,2019.

[18] SUN J,FENG B,XU W. Particle swarm optimization with particles having quantum behavior [C]// Proceedings of the 2004 Congress on Evolutionary Computation. Portland:IEEE Press,2004:325-331.

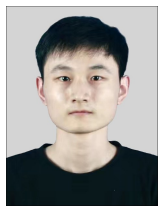
[19] TANG W,LI D P,CHEN X Y. Chaos theory and its application [J]. Automation of Electric Power Systems,2000,24(7):67-70.

[20] MAY R M. Simple mathematical models with very complicated-dynamics[J]. Nature,1976,261(5560):459-466.

[21] JAMIL M,YANG X S,ZEPERNICK H J. Test functions for global optimization:a comprehensive survey[J]. Swarm Intelligence and Bio-inspired Computation,Elsevier,2013:193-222. https://www.sciencedirect.com/science/article/pii/B9780124051638000089?via%3Dihub.

[22] ZIMMERMAN D W,ZUMBO B D. Relative Power of the Wilcoxon Test,the Friedman Test,and Repeated-Measures ANOVA on Ranks[J]. The Journal of Experimental Education,1993,62(1):75-86.

LIU Xiaonan,born in 1977,Ph.D,associate professor,master's supervisor,is a member of China Computer Federation. His main research interests include quantum algorithm and high-performance parallel computation.



AN Jiale,born in 1997,postgraduate. His main research interests include quantum algorithm and swarm intelligence optimization algorithm.

(责任编辑:喻藜)