计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(3):150−158 [doi: 10.15888/j.cnki.csa.008363]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 偏移进化蜉蝣优化算法①

王克逸, 符　强, 陈嘉豪

(宁波大学科学技术学院 信息工程学院, 宁波 315300)
通信作者: 符　强, E-mail: fuqiang@nbu.edu.cn

**摘　要**: 蜉蝣算法是一种受蜉蝣飞行及交配行为启发的新型群智能优化算法, 具有良好的寻优性能, 但其在求解高维复杂问题时依然存在因失效蜉蝣而影响算法效率的问题. 鉴于此, 提出一种偏移进化蜉蝣算法 (migration evolutionary mayfly algorithm, MEMA). 针对蜉蝣种群进行个体能力评价, 剔除种群中生命周期较长但进化能力较弱的个体, 同时以其为据点进行全局位置偏移, 以获取新生个体. 对新个体进行指向性动态进化训练, 从而提升种群整体优化能力. 最后在 Matlab 环境下, 随机抽取了 6 个 benchmark 测试函数设计仿真实验以验证 MEMA 算法的有效性, 实验结果表明, 相比于其他 5 种对比算法, MEMA 算法在低维及高维函数测试中均能更好地实现最优解搜索, 在收敛精度、收敛速度以及鲁棒性等方面均具备一定优势.

**关键词**: 蜉蝣优化算法; 群智能优化算法; 生命周期; 偏移; 速度调节

引用格式: 王克逸,符强,陈嘉豪.偏移进化蜉蝣优化算法.计算机系统应用,2022,31(3):150–158. http://www.c-s-a.org.cn/1003-3254/8363.html

## Migration Evolutionary Mayfly Algorithm

WANG Ke-Yi, FU Qiang, CHEN Jia-Hao

(School of Information Engineering, College of Science & Technology Ningbo University, Ningbo 315300, China)

**Abstract**: The mayfly algorithm is a new type of swarm intelligence optimization algorithm inspired by mayfly flight and mating behavior. It has good optimization performance, but its efficiency is affected by failure mayflies when faced with high-dimensional and complex problems. In view of this, a migration evolutionary mayfly algorithm (MEMA) is proposed in this study. First, the individual ability of the mayfly population is evaluated, and individuals with a long life-cycle but weaker evolutionary ability are eliminated from the population. At the same time, with those eliminated ones as strongholds, a global position shift is performed on the mayfly population to obtain new individuals. Then, directional dynamic evolution training is carried out on new individuals to improve the overall optimization ability of the population. Finally, in the Matlab environment, six benchmark test functions are randomly selected to design simulation experiments for the effectiveness verification of the MEMA algorithm. The experimental results show that compared with the other five comparison algorithms, the MEMA algorithm outperforms in both low-dimensional and high-dimensional function tests for the optimal solution search, and it has advantages in convergence accuracy, convergence speed, and robustness.

**Key words**: mayfly optimization algorithm; swarm intelligence optimization algorithm; life cycle; migration; speed adjustment

在现实生活中存在大量需要优化的问题, 且随着时代的发展, 这些问题的特征也在不断改变, 从最简单的线性问题, 到现在的多模态问题, 问题的求解难度不断上升, 由此有学者提出群智能算法. 群智能算法通过模拟生物的群体行为, 群体中的个体通过学习不断改变自身位置和搜索方向, 使个体往更好的方向进化, 进

① 基金项目: 宁波市自然科学基金 (202003N4159); 国家级大学生创新创业训练计划 (202013277008)
收稿时间: 2021-05-20; 修改时间: 2021-06-14; 采用时间: 2021-06-21; csa 在线出版时间: 2022-01-24

而达到种群群体进化的效果. 经典的群智能算法有粒子群算法 (PSO)[1]、萤火虫算法 (FA)[2]、蚁群算法 (ACO)[3]、狼群算法 (WPA)[4]、混合蛙跳算法 (SFLA)[5] 和进化算法 (EAS)[6] 等, 群智能算法现已广泛应用于各个领域[7–10].

蜉蝣算法 (mayfly algorithm, MA)[11] 是 Zervoudakis 和 Tsafarakis 在 2020 年提出的一种新型的群智能算法. MA 从蜉蝣的飞行和交配过程中得到启发并遵循了达尔文进化理论中的交叉、变异、选择和适应决策. 对于求解简单问题具有较好的寻优能力, 但由于 MA 不具备良好的跳出局部最优的性能, 且在迭代前期, 惯性系数较小, 雄雌蜉蝣增速较慢导致种群整体进化效率较低, 而在迭代后期, 雄蜉蝣婚礼舞蹈系数、雌蜉蝣飞行系数和惯性系数的减小导致雄雌蜉蝣移动缓慢. 因此, 在高维复杂问题上, MA 体现出较为乏力的寻优能力. 在低维问题上, 收敛速度较慢, 收敛性较差. 为增强算法跳出局部最优的能力和提升种群整体进化速度, 本文提出偏移进化蜉蝣优化算法 (migration evolutionary mayfly algorithm, MEMA). MEMA 在 MA 基础上添加了一项新策略, 从种群的进化角度出发, 剔除种群中进化性能较差的蜉蝣, 并以特殊方式生成进化性能较好的蜉蝣以此来加快种群进化速度从而使算法快速趋向目标问题的最优解. 本文将引用 MA、PSO、ABC[12]、RPSO [13]、IOABC[14] 和本文所提算法, 通过不同函数特性随机抽取 benchmark 标准测试函数, 进行综合性能对比, 验证算法的有效性.

## 1 蜉蝣算法简介

MA 的灵感来源于蜉蝣的飞行和交配过程. 雄蜉蝣皆群居于水面之上, 雄蜉蝣进行移动时, 将受到种群整体和自身的影响向更优方向进发. 每过一段时间, 雄蜉蝣会通过舞动来吸引雌蜉蝣, 雌蜉蝣则会随机飞向雄蜉蝣进行交配. 交配产生若干个子代蜉蝣, 极少数子代蜉蝣会发生变异. MA 中, 雄蜉蝣和子代蜉蝣参与寻优, 即作为解空间中的候选解决方案. 随后, 对种群中所有个体通过精英保留策略保留较好个体, 形成新种群进行下一轮迭代.

算法最初的时候会随机产生雄蜉蝣和雌蜉蝣, 每一只蜉蝣的位置都是随机的. 由 $D$ 维向量表示的蜉蝣的位置 $x_i=\{x_1,x_2,x_3,\cdots,x_D\}$, 速度表示为 $V_i=\{V_1,V_2,V_3,\cdots,V_D\}$, 每次迭代通过在当前位置上加上速度来进行位置更改.

### 1.1 雄性蜉蝣的位置更新

雄性蜉蝣是群居的生物, 每只雄蜉蝣的位置根据自己的情况以及种群整体的情况来调整, 雄蜉蝣一般在水面上进行上下移动, 假设蜉蝣不能快速移动, 雄蜉蝣的速度为:

$$v_{ij}^{t+1}=\begin{cases} g\cdot v_{ij}^{t}+a_1\mathrm{e}^{-\beta r_p^2}\left(pbest_{ij}-x_{ij}^{t}\right)+a_2\mathrm{e}^{-\beta r_g^2}\left(gbest_{ij}-x_{ij}^{t}\right), \\ \qquad \text{if } f(x_i)>f(gbest) \\ g\cdot v_{ij}^{t}+d\cdot r, \text{ if } f(gbest)\leqslant f(x_i) \end{cases} \tag{1}$$

其中, $v_{ij}^{t+1}$为蜉蝣$i$在$j$维度$t$时刻的速度, $x_i^t$为时间步长为$t$时, 蜉蝣$i$在搜索空间中的位置, $a_1$为种群学习系数, $a_2$为个体学习系数, $pbest_{ij}$为蜉蝣$i$曾经到过最好位置, $gbest_j$为全局最佳位置, $\beta$为蜉蝣的能见度系数, $g$为惯性权重, $d$为舞蹈系数, $r$为范围内的随机值$\in[-1,1]$, $r_p$代表当前位置与$pbest_{ij}$的笛卡尔距离. $r_g$代表当前位置与$gbest_j$的笛卡尔距离.

雄雌蜉蝣的移动, 通过在当前位置添加速度$v_i^{t+1}$来改变位置. 假设$x_i^t$是在时间步长为$t$时蜉蝣$i$在搜索空间中的位置, 公式如下:

$$x_i^{t+1}=x_i^t+v_i^{t+1} \tag{2}$$

笛卡尔距离公式为:

$$\|x_i-X_i\|=\sqrt{\sum_{j=1}^{n}\left(x_{ij}-X_{ij}\right)^2} \tag{3}$$

### 1.2 雌性蜉蝣的位置更新

雌蜉蝣会寻找适应值更好雄蜉蝣繁衍后代, 即雌蜉蝣将被适应值更好的雄蜉蝣所吸引, 所以雌蜉蝣会向更好雄蜉蝣靠近. MA 中种群雄雌蜉蝣的相互吸引定性为最优匹配, 即最优雄蜉蝣与最优雌蜉蝣相互吸引, 次优雄蜉蝣与次优雌蜉蝣相互吸引, 以此类推. 若对应雄蜉蝣的适应值更差, 雌蜉蝣将选择随机向周围飞行. 雌蜉蝣的速度的计算公式为:

$$v_{ij}^{t+1}=\begin{cases} g\cdot v_{ij}^{t}+a_2\mathrm{e}^{-\beta r_{mf}^2}\left(x_{ij}^{t}-y_{ij}^{t}\right), \text{ if } f(y_i)>f(x_i) \\ g\cdot v_{ij}^{t}+fl\cdot r, \text{ if } f(y_i)\leqslant f(x_i) \end{cases} \tag{4}$$

其中, $y_{ij}^t$为雌蜉蝣$i$对应的雄蜉蝣$i$的位置, $r_{mf}$是雌性蜉蝣与雄性蜉蝣的笛卡尔距离, $fl$是一个随机飞行系数.

### 1.3 蜉蝣交配行为

MA 采用最优匹配的机制来抽取雄雌蜉蝣进行交配产生后代, 公式如下:

$$\begin{cases} offs1 = L \cdot male + (1-L) \cdot female \\ offs2 = L \cdot female + (1-L) \cdot male \end{cases} \tag{5}$$

其中, $offs1$为第一个子代, $offs2$为第二个子代, $male$ 为雄蜉蝣, $female$为雌蜉蝣, $L$为$[-1,1]$ 的随机值.

### 1.4 子代蜉蝣突变

MA 引入了高斯变异[15], 假设种群数量为 $N$, 设定变异因子为 $m$, 则子代蜉蝣中发生变异个体为$N \cdot m$, 变异子代蜉蝣的随机一个维度将发生突变, 突变公式为:

$$offs_n = offs_n + \sigma \cdot N_n(0,1) \tag{6}$$

其中, $offs_n$为被选中的子代$offs$的 $n$ 维度, $\sigma$为正态分布的标准差, $N_n(0,1)$为均值为 0, 方差为 1 的标准正态分布.

### 1.5 动态惯性权重

为了平衡全局探索能力和局部搜索能力, MA 加入了动态惯性权重. 在迭代过程中, 蜉蝣的惯性权重将线性减少小, 公式如下:

$$g = g_{\max} - \frac{g_{\max} - g_{\min}}{iter_{\max}} \cdot iter \tag{7}$$

其中, $g$为惯性权重, $g_{\max}$是最大惯性权重, $g_{\min}$是最小惯性权重, $iter$为迭代次数, $iter_{\max}$为最大迭代次数, $iter$为迭代次数.

算法前期, 较大的$g$能更好地满足全局探索能力增加算法的收敛速度, 算法后期, 较小的$g$增加局部搜索能力提高算法的搜索精度.

### 1.6 婚礼舞蹈系数与随机飞行系数

为了寻找更好的解, 蜉蝣算法引入婚礼舞蹈系数, 当雄蜉蝣适应值大于全局最优时, 将进行随机移动以寻找更优解, 引入飞行系数, 当雌蜉蝣未被雄蜉蝣吸引时, 会向四周随机飞行寻找更好的雄蜉蝣. 婚礼舞蹈系数和随机飞行系数会随迭代次数减小以此减少蜉蝣步长, 增加局部搜索能力, 在算法后期提高收敛精度, 公式如下:

$$d_t = d_1 \cdot d_{\text{damp}}^t,\ 0 < d_{\text{damp}} < 1 \tag{8}$$

$$fl_t = fl_1 \cdot fl_{\text{damp}}^t, 0 < fl_{\text{damp}} < 1 \tag{9}$$

其中, $d_t$与$fl_t$为$t$时刻的婚礼舞蹈系数和随机飞行系数, $d_{\text{damp}}$与$fl_{\text{damp}}$为衰减参数.

## 2 偏移进化蜉蝣优化算法

MA 算法在迭代前期, 种群较快收束于当前全局最优位置, 不利于种群整体进化, 在迭代中后期, 易出现早熟收敛现象. 对这个问题提出 MEMA, 添加年龄到蜉蝣的属性中, 剔除群体中较为年长且进化程度较低的蜉蝣, 并在该个体附近重新生成新个体保证种群完整性, 同时对该蜉蝣进行指向性动态进化训练, 促进蜉蝣良性进化, 提升种群整体利用率, 提升算法寻优性能.

### 2.1 偏移进化机制

抽象蜉蝣良性进化的思想为算法的偏移进化机制, 在每次迭代后, 记录种群中雄蜉蝣的进化次数, 由于当蜉蝣进化次数超过一定次数时会存在适应值较差的个体, 为了排除进化 2 次到 3 次存在的偶然性, 而 4 次以上会使适应值较差的浮游在外徘徊次数过多浪费进化次数, 所以选取 4 次当蜉蝣数进化次数. 超过 4 次且适应值较差时, 说明该蜉蝣的进化速度始终在跟随整体的进化疲于奔向最优位置, 判定这样的解为失效解, 不具备较好进化能力对算法求解帮助较小. 剔除该蜉蝣并引入新蜉蝣, 从原蜉蝣的位置上进行一次偏移操作, 生成新蜉蝣. 新蜉蝣出现位置为:

$$mayfly_w = mayfly_w + W \tag{10}$$

其中, $mayfly_w$为被剔除蜉蝣的位置, $W$为拉伸因子, 公式为:

$$W = rand \cdot (Upper - Lower) \tag{11}$$

其中, $Lower$为解空间下限, $Upper$为解空间上限 $rand$ 为$(0,1]$ 的随机值.

为了保证新蜉蝣是优秀的个体且能有效地对种群进化起到帮助. 在加入种群前, 将对新蜉蝣进行随机次数的进化. 规定该蜉蝣会根据速度调节因子$\rho$进行指向性动态进化训练, 但不会超过当前种群进化次数, 其公式为:

$$\rho = [rand \cdot iter] \tag{12}$$

新蜉蝣在加入种群前的进化移动与种群进化移动相同, 通过在当前位置$z_{ij}^t$添加速度$v_i^{t+1}$来改变位置. 但更改速度的变化为:

$$v_{ij}^{t+1} = \begin{cases} g \cdot v_{ij}^t + \dfrac{1}{\rho} \cdot \left(gbest_j - z_{ij}^t\right) + \dfrac{1}{\rho} \cdot \left(pbest_{ij} - z_{ij}^t\right), \\ \quad \text{if } f(z_i) > f(gbest) \\ g \cdot v_{ij}^t + d \cdot r, \text{ if } f(gbest) \leqslant f(z_i) \end{cases} \tag{13}$$

根据速度公式, 在速度调节因子$\rho$的影响下, 蜉蝣将进行分段寻找, 重力因子$g$的存在, 可以减小前一次进化速度的影响, 同时蜉蝣的速度受到$gbest$和$pbest$两

个位置因子影响, 若蜉蝣找到比偏移后的初始位置更优的位置, 将受到上一次速度的影响继续向前前进一段距离, 之后受到两个位置因子的影响快速减速, 并反向加速回到更新后的全局最优位置附近, 否则将进行先加速后减速的分段寻找. 这一过程蜉蝣将仔细搜索更新后的全局最优位置的附近位置以期能将失效个体转换为引导种群进化的个体, 填补 MA 算法的缺陷, 增加算法收敛性能. 这些策略都是为了能更好地适应复杂的自然环境, 让其算法在保持多样性的条件下让其在全局搜索与局部搜索方面更为灵活.

### 2.2 MEMA 算法流程

Step 1. 初始化参数.

Step 2. 初始化雄雌蜉蝣位置与速度, 计算并找出当前最优解.

Step 3. 根据式 (1), 式 (2), 式 (4) 更新雄雌性蜉蝣速度与位置.

Step 4. 所有蜉蝣根据当前适应值排序.

Step 5. 根据式 (5) 生成后代, 并根据变异可能性和式 (6) 随机产生变异后将其后代随机变成雄蜉蝣或者雌蜉蝣.

Step 6. 根据种群的适应值, 对雄雌蜉蝣进行排序, 用当前更优解代替劣解.

Step 7. 根据式 (8) 和式 (9) 更新舞蹈系数和随机飞行系数, 根据式 (7) 更新惯性权重.

Step 8. 若超过 4 次且能力较差, 则根据式 (10) 和式 (11) 产生新蜉蝣, 根据式 (12) 产生随机进化次数, 根据式 (13) 进化新蜉蝣, 将淘汰的个体替换.

Step 9. 判断是否到达迭代上限, 若是, 转 Step 10, 若否, 转 Step 3.

Step 10. 输出当前最优解.

## 3 实验及数据比对

### 3.1 实验环境

实验中所使用的软硬件参数如下:

(1) 操作系统: Windows 10;

(2) 硬件参数: Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz 2.21 GHz 8 GB RAM 内存;

(3) 编译环境及工具: Matlab 2018a.

### 3.2 测试函数

为了说明算法的普遍适用性, 本文为突出该算法的全面性随机抽取了标准 benchmark 测试函数中 6 个具有不同特点的函数 (见表 1).

表 1 测试函数

| 函数 | 函数名 | 公式 | Range | Min | Features |
|---|---|---|---|---|---|
| F1 | Sphere | $f_1(x)=\sum_{i=1}^{d}x_i^2$ | [−10, 10] | 0 | 单峰 |
| F2 | Rosenbrock | $f_2(x)=\sum_{i=1}^{d-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i^2\right)^2\right]$ | [−5, 10] | 0 | 病态单峰 |
| F3 | Powell Sum | $f_3(x)=\sum_{i=1}^{d}\lvert x_i\rvert^{i+1}$ | [−1, 1] | 0 | 底部平缓单峰函数 |
| F4 | Schwefel2.22 | $f_4(x)=\sum_{i=1}^{d}\lvert x_i\rvert+\prod_{i=1}^{d}\lvert x_i\rvert$ | [−100, 100] | 0 | 平滑多峰 |
| F5 | Rastrigin | $f_5(x)=10d+\sum_{i+1}^{d}\left(x_i^2-10\cos(2\pi x_i)\right)$ | [−5.12, 5.12] | 0 | 多模态多峰 |
| F6 | Ackley | $f_6(x)=20+\mathrm{e}-20\exp\left[-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right]-\exp\left[\sqrt{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}\right]$ | [−32, 32] | 0 | 梯度式单峰 |

首先抽取两个典型的单峰函数 Sphere 函数和 Ackley 函数, 其中 Sphere 函数常用于检验算法的收敛效率, 而 Ackley 函数是一个梯度式的函数, 在高维问题上, 算法收敛将会非常慢, 常用于检验算法的全局收敛速度.

其次抽取两种底部较平缓的单峰函数 Powell Sum 函数和 Rosenbrock 函数, 前一种单峰函数底部较平, 导致算法接近最优值附近区域时, 寻优较慢, 可以用于检验算法寻优效率. Rosenbrock 函数是病态单峰函数, 也称为香蕉函数. 函数面可以看作为陡坡, 底部较为平缓, 最优值位于陡坡底部, 底部区域很容易找到, 但由于底部的值变化不大, 要找到全局最优很困难, 一般用于检验算法的局部搜索能力.

最后抽取两个经典的多峰函数 Schwefel 2.22 函数和 Rastrigin 函数. Schwefel 2.22 函数较为平滑, 但全局最优值位于定义域的界限处. Rastrigin 函数是一个多

模态多峰函数, 具有大量的局部最小值, 尤其在高维问题上, 易使算法陷入局部最优, 常用于检验算法跳出局部最优的能力.

将 MEMA 与 MA[11]、ABC[12]、IQABC[14]、PSO[1]、RPSO[13] 5 种算法进行对比, 来测试算法的性能.

### 3.3 参数设置

设置迭代次数为 500 次, 所有算法的种群数量都为 40, MEMA 的参数设置为: 最大惯性权重为 1.5, 最小惯性权重为 0.4, 婚礼舞蹈系数为 1, 婚礼舞蹈系数的衰减系数为 0.8, 随机飞行系数为 1, 随机飞行系数的衰减系数为 0.99, 子代数量为 20, 变异可能性为 0.01, 种群学习参数 $a_1$ 为 1.0, 蜉蝣个体学习参数 $a_2$ 为 1.5, 其他算法的参数设置参考文献内容.

实验将偏移进化蜉蝣优化算法 (MEMA) 与蜉蝣算法 (MA), 蜂群算法 (ABC), 改进蜂群算法 (IQABC), 粒子群算法 (PSO) 和改进粒子群算法 (RPSO) 进行对比, 为避免因随机因素造成算法结果影响, 因此本文对每个测试函数独立运行 100 次实验取平均值作为算法的最终结果以降低随机数对结果的影响. 为了综合考虑算法的有效性, 记录了 6 种算法的平均值, 最优解, 最劣解和方差.

为了更全面地验证各算法在低维及高维空间的有效性, 本文将从低维和高维进行算法测试. 低维我们取 10 维, 数据见表 2, 高维我们取 100 维, 数据见表 3.

表 2　6 种算法对比数据表 (10 维)

| 函数 | 算法 | 平均值 | 最优解 | 最劣解 | 方差 |
|---|---|---|---|---|---|
| F1 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 1.5061E–03 | 2.0297E–45 | 9.2907E–37 | 1.2733E–78 |
| | ABC | 9.3861E–05 | 7.6471E–06 | 4.3027E–04 | 2.6085E–11 |
| | IQABC | 4.8188E–25 | 4.5909E–29 | 7.9601E–24 | 2.3039E–51 |
| | PSO | 7.5402E+00 | 4.8014E–01 | 1.9555E+01 | 1.2619E–02 |
| | RPSO | 1.2254E+02 | 4.6473E+01 | 1.9801E+02 | 2.6895E+00 |
| F2 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 7.7195E–02 | 1.4652E–09 | 3.9613E+00 | 6.0193E–05 |
| | ABC | 1.7663E+01 | 2.9422E+00 | 5.4406E+01 | 7.7687E–01 |
| | IQABC | 4.1022E+00 | 1.3869E–05 | 8.7366E+00 | 1.2959E–01 |
| | PSO | 5.7293E+03 | 1.2434E+02 | 1.0848E+05 | 2.3970E+05 |
| | RPSO | 9.5610E+04 | 1.6415E+04 | 2.6041E+05 | 1.0227E+08 |
| F3 | **MEMA** | **7.0136E–81** | **2.4494E–112** | **7.0128E–79** | **4.9688E–163** |
| | MA | 6.9568E–67 | 2.2300E–86 | 6.9259E–65 | 4.8886E–135 |
| | ABC | 5.9259E–09 | 2.9123E–11 | 3.9960E–08 | 3.0050E–19 |
| | IQABC | 2.5269E–38 | 1.1037E–50 | 1.8390E–36 | 6.4495E–78 |
| | PSO | 1.0381E–03 | 5.0956E–05 | 5.1857E–03 | 7.4611E–09 |
| | RPSO | 9.0422E–06 | 3.3285E–09 | 1.1347E–04 | 8.2526E–13 |
| F4 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 1.3831E–19 | 3.9040E–24 | 9.5683E–18 | 1.8157E–40 |
| | ABC | 5.0343E–03 | 1.4021E–03 | 1.8888E–02 | 2.5331E–08 |
| | IQABC | 1.6457E–13 | 2.4679E–16 | 2.3230E–12 | 1.7561E–28 |
| | PSO | 5.3454E+08 | 1.3816E+02 | 2.7704E+10 | 2.5518E+15 |
| | RPSO | 1.2103E+11 | 1.7046E+02 | 4.2386E+12 | 1.4796E+20 |
| F5 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 2.7568E+00 | **0.0000E+00** | 1.5923E+01 | 5.9406E–03 |
| | ABC | 3.1142E+01 | 1.0497E+01 | 1.0497E+01 | 2.1146E–03 |
| | IQABC | 6.9364E+00 | 9.9496E–01 | 1.8904E+01 | 8.8298E–02 |
| | PSO | 5.9702E+01 | 2.6111E+01 | 1.0826E+02 | 3.3832E–01 |
| | RPSO | 1.0998E+02 | 6.3085E+01 | 1.4615E+02 | 6.8349E+00 |
| F6 | **MEMA** | **2.6645E–15** | **2.6645E–15** | **2.6645E–15** | **0.0000E+00** |
| | MA | 2.7187E–01 | 2.6645E–15 | 1.6462E+00 | 7.4658E–04 |
| | ABC | 2.2946E–01 | 3.9484E–02 | 7.1919E–01 | 1.7888E–04 |
| | IQABC | 3.7450E–12 | 3.8192E–14 | 5.5880E–11 | 1.0543E–25 |
| | PSO | 1.0040E+01 | 4.7920E+00 | 1.9967E+01 | 2.0579E–02 |
| | RPSO | 1.9247E+01 | 1.5487E+01 | 1.9947E+01 | 4.8865E–03 |

表 3 6 种算法对比数据表 (100 维)

| 函数 | 算法 | 平均值 | 最优解 | 最劣解 | 方差 |
|---|---|---|---|---|---|
| F1 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 1.3101E+01 | 2.4939E+00 | 3.0428E+01 | 5.9070E–02 |
| | ABC | 7.8207E+02 | 5.3880E+02 | 9.7893E+02 | 1.3317E+00 |
| | IQABC | 7.6316E+02 | 5.4426E+02 | 9.6407E+02 | 1.3006E+01 |
| | PSO | 7.8951E+02 | 3.3011E+02 | 1.7256E+03 | 5.5353E+02 |
| | RPSO | 2.6836E+03 | 2.2261E+03 | 3.0418E+03 | 3.3261E+01 |
| F2 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 6.0388E+03 | 1.2257E+03 | 3.1071E+04 | 6.1801E+02 |
| | ABC | 1.2321E+06 | 7.9521E+05 | 1.7157E+06 | 1.9180E+08 |
| | IQABC | 1.2410E+06 | 8.4058E+05 | 1.6748E+06 | 1.1202E+08 |
| | PSO | 1.3851E+06 | 4.5746E+05 | 3.9867E+06 | 1.5122E+10 |
| | RPSO | 8.4834E+06 | 6.7097E+06 | 1.0247E+07 | 2.2569E+09 |
| F3 | **MEMA** | **5.4950E–41** | **1.9732E–59** | **5.4588E–39** | **3.0500E–83** |
| | MA | 1.4405E+00 | 1.8913E–10 | 2.8277E+00 | 1.3520E–04 |
| | ABC | 1.5965E+00 | 6.0477E–01 | 2.8277E+00 | 1.6409E–05 |
| | IQABC | 1.5800E+00 | 5.9825E–01 | 2.8277E+00 | 1.4799E–04 |
| | PSO | 1.4368E–02 | 1.9668E–04 | 6.9809E–02 | 1.9712E–06 |
| | RPSO | 1.3184E–01 | 1.7205E–06 | 1.6019E+00 | 1.7327E–04 |
| F4 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 2.1960E+03 | 1.3526E+03 | 3.4836E+03 | 2.1502E+02 |
| | ABC | 5.0556E+110 | 2.5292E+91 | 4.1399E+112 | 2.5487E+219 |
| | IQABC | 5.9587E+111 | 5.8189E+86 | 4.4762E+113 | 3.5865E+221 |
| | PSO | 3.6223E+129 | 6.8521E+89 | 3.5394E+131 | 1.3253E+257 |
| | RPSO | 3.1640E+149 | 8.1449E+136 | 1.8791E+151 | 1.0112E+297 |
| F5 | **MEMA** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** | **0.0000E+00** |
| | MA | 5.4858E+02 | 3.8700E+02 | 6.9564E+02 | 2.1845E+02 |
| | ABC | 1.2431E+03 | 1.1033E+03 | 1.3482E+03 | 1.0565E+01 |
| | IQABC | 1.2468E+03 | 1.1213E+03 | 1.3599E+03 | 1.4859E+02 |
| | PSO | 1.1583E+03 | 8.3986E+02 | 1.4492E+03 | 1.9952E+02 |
| | RPSO | 1.6136E+03 | 1.4663E+03 | 1.7307E+03 | 1.8418E+00 |
| F6 | **MEMA** | **8.0949E+00** | **6.2172E–15** | 1.9966E+01 | 6.6189E–01 |
| | MA | 1.6092E+01 | 7.9133E+00 | **1.9721E+01** | 5.1576E–04 |
| | ABC | 1.9719E+01 | 1.8579E+01 | 2.0690E+01 | 7.9455E–04 |
| | IQABC | 1.9706E+01 | 1.8346E+01 | 2.0609E+01 | 1.2764E–03 |
| | PSO | 1.8961E+01 | 1.4369E+01 | 1.9967E+01 | 9.6024E–04 |
| | RPSO | 1.9965E+01 | 1.9960E+01 | 1.9966E+01 | **2.5852E–10** |

由表 2 中数据可见, 在 6 个 10 维的测试函数上, MEMA 所找到的最优值与最劣值均能达到测试函数的全局最优解, 说明算法的收敛精度相较于对比算法更强. 时可以看出 MEMA 方差有 5 个都趋于 0, 对于 F3 和 F6 函数未收敛完成的测试函数, 相比较其他 5 种算法, MEMA 在收敛精度和算法稳定性上也都体现了卓越的性能.

对于表 3 中的数据, 同样的, 对于 10 维的问题中可以收敛完成的测试函数, MEMA 在 100 维的问题上也完成收敛, 充分体现了 MEMA 对于解决高维问题的优异性能, 对于 F3 和 F6 函数未能有效完成收敛, 但是 MEMA 相较于其他的算法体现了较强的能力寻优能力和寻优稳定性. 因此表现了 MEMA 在求解高维问题时更能体现出算法的寻优能力.

综上数据结果可得, MEMA 相比较其他算法, 对于问题的处理能力更加出色, 说明在 MA 中加入生命周期的属性, 将生命周期较长且进化能力较弱的个体替换为更优个体, 可以有效地将蜉蝣的利用率提升, 提升种群整体进化速度. 为了清晰的体现算法的收敛效率, 同样采用图 1 与图 2 画出了在 10 维与 100 维的问

题上不同函数的收敛曲线. 同样采用低维和高维两种维度进行算法测试. 低维我们取 10 维, 数据见图 1, 高维我们取 100 维, 数据见图 2.

结合表 2、表 3 的数据和图 1、图 2 的收敛曲线可以得出, 对于对比函数, 其他 5 种算法在收敛速度上明显低于 MEMA. 在图 1 的低维问题上 F1 函数在迭代 25 次时完成收敛, F2 函数在迭代 50 次左右时完成收敛. 图 1 中 F5 函数在其他算法都没有找到最优解时, MEMA 大约 20 次的时候就找到了最优解. 在图 2 高维问题上难以在短时间内收敛完成, 而 MEMA 除 F6 都在迭代 50 次左右完成了收敛并达到全局最优值, 而 F6 也能在图中可以看出当其他对比算法都陷入局部最优时, MEMA 还能继续向下收敛, 且该函数为单峰函数, 函数面较平, 在最优值附近区域较小, 较为考验算法的收敛速度, 充分说明了在高维问题上 MEMA 求解效率快, 寻优精度高. 从图 2 中 F5 可以看出, 其他测试算法由于全局搜索能力较慢, 难以快速接近最优解, 而 MEMA 迭代 50 次左右就可以快速接近最优值. 对于 100 维的收敛曲线图可以看到在收敛速度上 MEMA 远超其他算法. 对于 F1、F2、F4、F5 函数, MEMA 在迭代 50 次内, 已经收敛完成, 证明算法充分利用了蜉蝣群体中的较劣蜉蝣, 提升了算法的收敛性能.

### 3.4 运行效率

为验证 MEMA 的寻优能力, 我们设计运行效率的实验, 因为篇幅原因, 我们随机抽取 4 个 benchmark 测试函数, 假设算法在求解精度达到$10^{-20}$时, 判定为算法寻找到最优解, 停止算法运行, 若未找到, 则运行算法直到迭代上限 500 次, 通过测试 CPU 运行时间, 得出算法运行时间, 最快运行时间标黑, 数据见表 4.

从数据表上看, MEMA 的 CPU 运行时间远小于其他 5 种算法, 也表明在实际优化方面找到最优值的概率会大幅增加. 即证明 MEMA 的寻优能力优于其他算法.

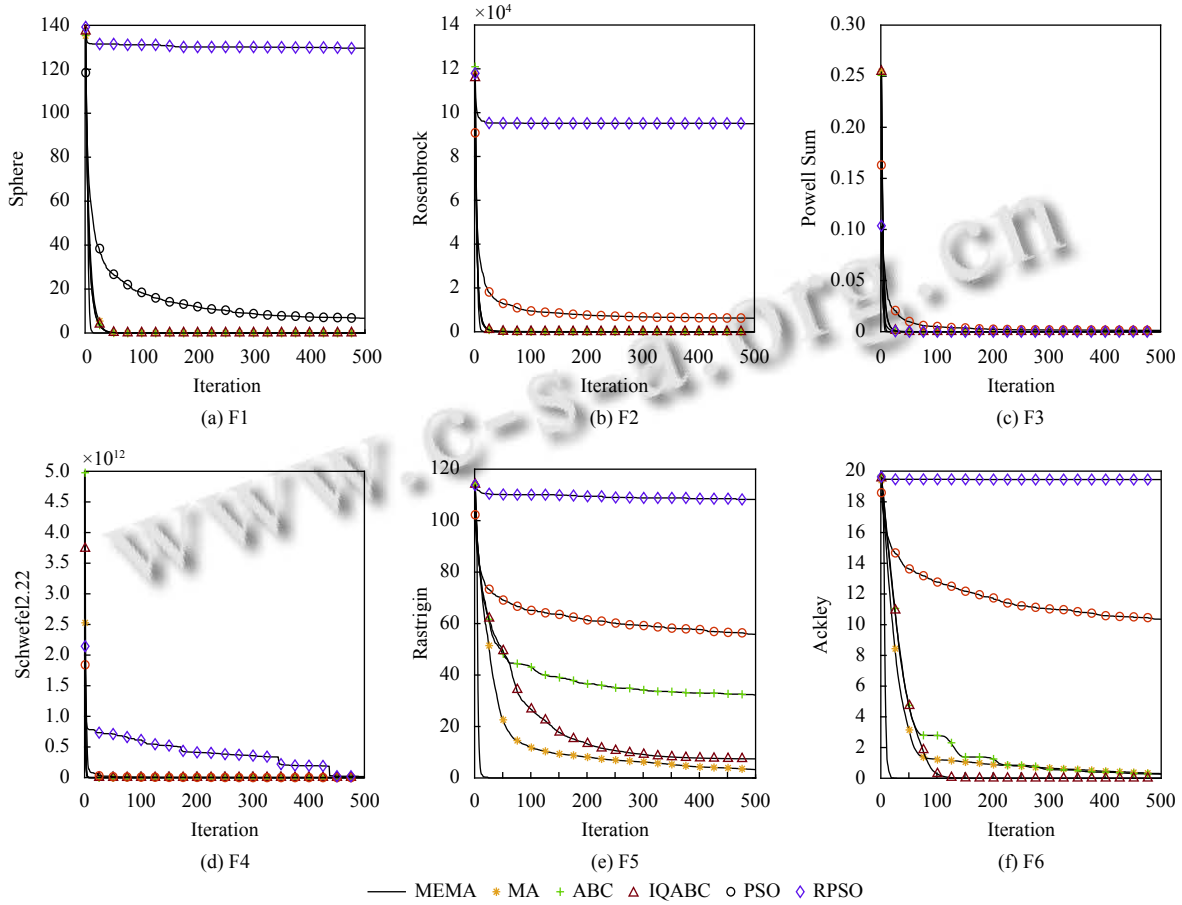


图 1 10 维问题函数收敛曲线图

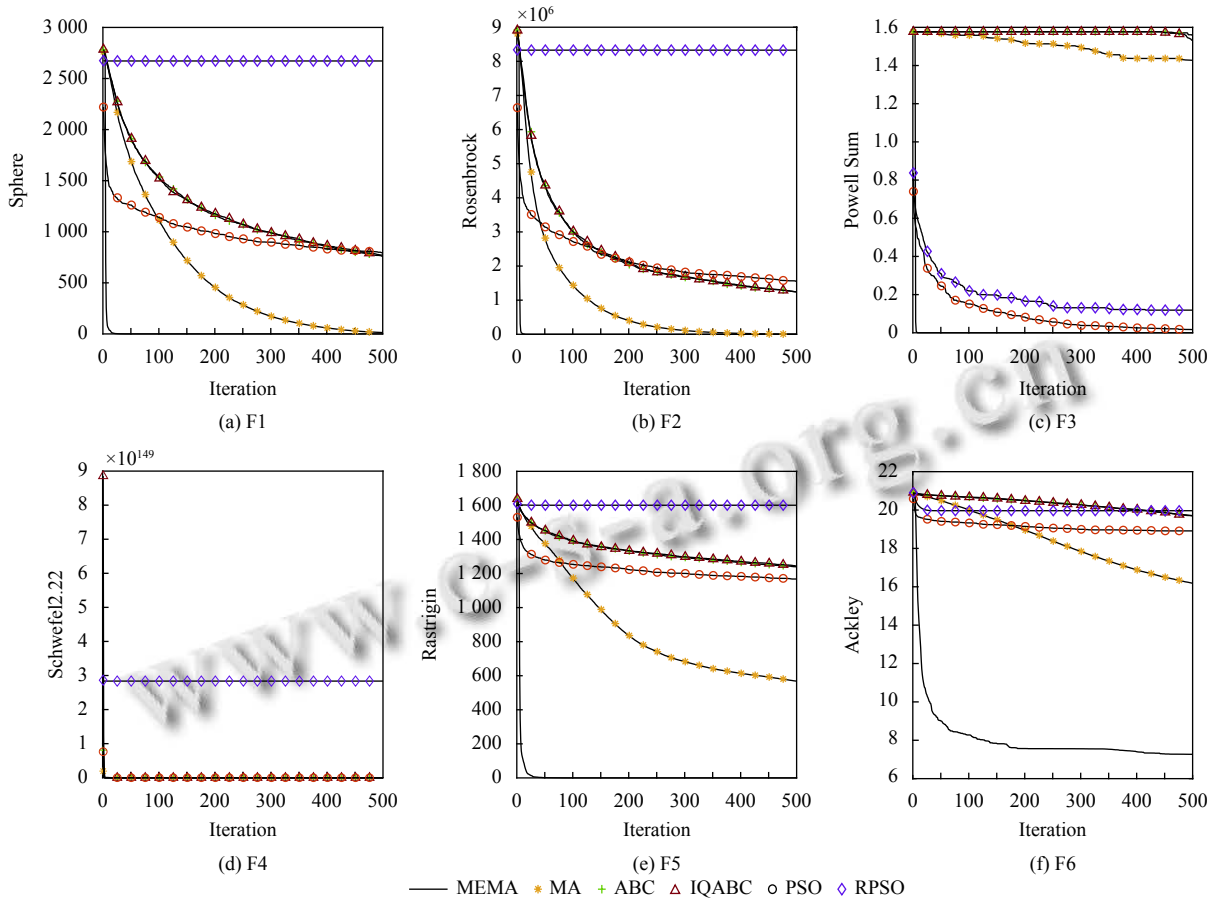


图 2 100 维问题函数收敛曲线图

表 4 运行时间 (s)

| 函数名 | MEMA | MA | ABC | IQABC | PSO | RPSO |
|---|---|---|---|---|---|---|
| Sphere | **9.3149E–03** | 2.0360E–01 | 3.2647E–02 | 1.7071E–01 | 7.6064E–02 | 7.4295E–02 |
| Rosenbrock | **9.2529E–03** | 1.8877E–01 | 3.3029E–02 | 1.6984E–01 | 7.5675E–02 | 7.3892E–02 |
| Sum quares | **9.2009E–03** | 1.8972E–01 | 3.2674E–02 | 1.6883E–01 | 7.5200E–02 | 7.3427E–02 |
| Powell Sum | **9.1309E–03** | 8.7900E–02 | 3.2399E–02 | 1.6790E–01 | 7.4758E–02 | 7.3033E–02 |

## 4 结束语

本文分析了蜉蝣算法难以有效利用失效蜉蝣的缺点, 引入了偏移进化后的蜉蝣去替换原有物种中失效个体, 在陷入局部极值时能够快速的跳出循环, 已趋于最优解空间, 以此来加快算法收敛效率并提高算法收敛性, 最终提出了基于偏移进化的改进型蜉蝣优化算法 MEMA. 本文从多方面对算法性能进行实验, 使用 6 个具有典型代表的 benchmark 标准测试函数, 将 MEMA 与 MA、ABC、PSO、IQABC、RPSO 五种算法进行对比, 证实了改进后的该算法无论在高纬度还是低维度反面都可以有效地进行全局收敛, 并且在图中也能更为直观地看出全局收敛能力能是远超于其他算法的. 为了证明该算法的效率方面的情况, 设计了运行时间比对实验, 实验结果表明, MEMA 算法不但结构简洁, 而且针对测试函数集表现出良好的寻优性能, 在收敛精度和收敛速度等方面均具有明显优势. 后期研究方向为提出相应的离散化算法及多目标优化算法.

## 参考文献

1 董红斌, 李冬锦, 张小平. 一种动态调整惯性权重的粒子群优化算法. 计算机科学, 2018, 45(2): 98–102, 139. [doi: 10.11896/j.issn.1002-137X.2018.02.017]

2 莫愿斌, 刘付永, 张宇楠. 带高斯变异的人工萤火虫优化算法. 计算机应用研究, 2013, 30(1): 121–123. [doi: 10.3969/j.issn.

1001-3695.2013.01.029]
3 蒋华, 张乐乾, 王鑫. 基于多维评价模型及改进蚁群优化算法的云计算资源调度策略. 计算机测量与控制, 2015, 23(7): 2559–2562.
4 郭立婷. 基于自适应和变游走方向的改进狼群算法. 浙江大学学报 (理学版), 2018, 45(3): 284–293.
5 崔文华, 刘晓冰, 王伟, 等. 混合蛙跳算法研究综述. 控制与决策, 2012, 27(4): 481–486, 493.
6 钱洁, 季敏. 基于文化知识的量子进化算法. 系统工程理论与实践, 2015, 35(1): 228–238. [doi: 10.12011/1000-6788(2015)1-228]
7 林诗洁, 董晨, 陈明志, 等. 新型群智能优化算法综述. 计算机工程与应用, 2018, 54(12): 1–9. [doi: 10.3778/j.issn.1002-8331.1803-0260]
8 陈婷婷, 殷贺, 江红莉, 等. 基于天牛须搜索的粒子群优化算法求解投资组合问题. 计算机系统应用, 2019, 28(2): 171–176. [doi: 10.15888/j.cnki.csa.006771]
9 左玉洁. 智能制造发展背景下物流系统优化研究与实践——评《群智能优化及其在物流中的应用》. 中国科技论文, 2019, 14(10): 132.
10 徐小平, 张东洁. 一种改进的猴群算法. 计算机系统应用, 2017, 26(6): 193–197. [doi: 10.15888/j.cnki.csa.005822]
11 Zervoudakis K, Tsafarakis S. A mayfly optimization algorithm. Computers & Industrial Engineering, 2020, 145: 106559.
12 Du ZX, Han DZ, Liu GZ, *et al*. Artificial bee colony algorithm with gradually enhanced exploitation. Journal of Shanghai Jiaotong University, 2018, 52(1): 96–102.
13 赵延龙, 滑楠, 于振华. 基于二次搜索的改进粒子群算法. 计算机应用, 2017, 37(9): 2541–2546. [doi: 10.11772/j.issn.1001-9081.2017.09.2541]
14 敖媛, 丁学明. 人工蜂群算法改进. 软件导刊, 2016, 15(11): 65–67.
15 Shao D, Xu SC, Du AM. Dynamic friction modelling and parameter identification for electromagnetic valve actuator. Journal of Central South University, 2018, 25(12): 3004–3020. [doi: 10.1007/s11771-018-3970-x]