计算机科学
COMPUTER SCIENCE

基于自适应反馈调节因子的阿基米德优化算法

陈俊，何庆，李守玉

引用本文

陈俊，何庆，李守玉. 基于自适应反馈调节因子的阿基米德优化算法[J]. 计算机科学, 2022, 49(8): 237-246

CHEN Jun, HE Qing, LI Shou-yu. Archimedes Optimization Algorithm Based on Adaptive Feedback Adjustment Factor[J]. Computer Science, 2022, 49(8): 237-246.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210700150

基于自适应反馈调节因子的阿基米德优化算法

陈 俊 何 庆 李守玉
贵州大学大数据与信息工程学院 贵阳 550025
(youngchen777@163.com)

摘 要 针对基础阿基米德优化算法收敛速度慢、容易陷入局部最优的问题,文中提出了一种基于自适应反馈调节因子的阿基米德优化算法。首先,通过佳点集初始化种群,增强初始种群的遍历性,提高初始解的质量;其次,提出自适应反馈调节因子,平衡算法的全局探索与局部开发能力;最后,提出了莱维旋转变换策略,增加种群的多样性,以防止算法陷入局部最优。将所提算法与主流算法在14个基准测试函数以及部分CEC2014函数上进行30次比较实验,结果表明,所提算法的平均寻优精度、标准差以及收敛曲线均优于对比算法。同时将所提算法分别与对比算法在14个基准函数上进行Wilcoxon秩和检验,检验结果显示所提算法与对比算法的差异性显著。将所提算法应用于焊接梁设计问题,其相比原始算法提升了2%,验证了所提算法的有效性。

关键词:阿基米德优化算法;佳点集;自适应反馈调节因子;旋转变换算子;莱维飞行
中图法分类号 TP301.6

Archimedes Optimization Algorithm Based on Adaptive Feedback Adjustment Factor

CHEN Jun, HE Qing and LI Shou-yu
College of Big Data & Information Engineering, Guizhou University, Guiyang 550025, China

Abstract Aiming at the problem of slow convergence speed of basic Archimedes optimization algorithm and is easy to fall into local optimum, this paper proposes an Archimedes optimization algorithm based on adaptive feedback adjustment factor. Firstly, initializing the population through the good point set to enhance the ergodicity of the initial population and improve the quality of the initial solution. Secondly, an adaptive feedback adjustment factor is proposed to balance the global exploration and local development capabilities of the algorithm. Finally, the Levy rotation transformation strategy is proposed, to increase the diversity of the population and prevent the algorithm from falling into a local optimum. Comparative experiments of the proposed algorithm and mainstream algorithms are carried on 14 benchmark functions and some CEC2014 functions for 30 times. The optimization results of the algorithm on the function show that the average optimization accuracy, standard deviation and convergence curve of the proposed algorithm are better than that of the comparison algorithm. At the same time, Wilcoxon rank sum test is performed on 14 benchmark functions between the proposed algorithm and comparison algorithms. The test results show that the proposed algorithm is significantly different from comparison algorithms. It will be applied to the design of welded beams, which is 2% higher than the original algorithm, which verifies the effectiveness of the proposed algorithm.

Keywords Archimedes optimization algorithm, Good point set, Adaptive feedback adjustment factor, Rotation transformation strategy, Levy fight

1 引言

近年来,全局优化方法在机械设计[1]、微电网调度[2]、无人机技术[3]、神经网络训练[4]、多目标优化[5]等众多领域中的应用越来越广泛。伴随着计算机性能和电子信息技术的飞速发展,大量的启发式算法被提出,如遗传算法[6]、粒子群

到稿日期:2021-07-14 返修日期:2021-12-06
基金项目:贵州省科技计划项目重大专项项目(黔科合重大专项字[2018]3002,黔科合重大专项字[2016]3022);贵州省教育厅青年科技人才成长项目(黔科合KY字[2016]124);贵州大学培育项目(黔科合平台人才[2017]5788);贵州省公共大数据重点实验室开放课题(2017BDKFJJ004);贵州省科技计划项目(黔科合基础-ZK[2021]一般335)
This work was supported by the Guizhou Province Science and Technology Plan Project Major Special Project(Qiankehe Major Special Project Word [2018] 3002, Qiankehe Major Special Project Word [2016] 3022), Guizhou Provincial Education Department Young Science and Technology Talent Growth Project(Qiankehe KY Word [2016] 124), Guizhou University Cultivation Project(Qiankehe Platform Talent [2017]5788), Guizhou Provincial Public Big Data Key Laboratory Open Project (2017BDKFJJ004) and Guizhou Science and Technology Plan Project(Qiankehe Foundation-ZK[2021] General 335).
通信作者:何庆(qhe@gzu.edu.cn)

算法[7]、鲸鱼优化算法[8]、黄金正弦算法[9]、萤火虫优化算法[10]以及人工蜂群优化算法[11]等。该类方法对目标函数性质要求低,容易实现且稳定性好,自提出以来就受到了国内外学者的关注。但是,随着优化问题规模越来越庞大,传统优化算法出现收敛精度低、收敛速度慢以及容易陷入局部最优等问题,越来越难以满足需求。对此,研究人员采用诸多方法对传统优化算法进行改进。Bodha 等[12]利用莱维步长短距离与偶尔较长距离步长相见的特性对粒子群算法中的速度更新公式进行改进,提高了算法的跳出局部极值点的能力。Ma 等[13]采用非线性收敛因子和自适应惯性权重,分别对鲸鱼算法中的参数 A 和捕食猎物时的位置更新公式进行改进,一方面平衡了算法全局搜索能力与局部探索能力,另一方面加快了算法的收敛速度。Xiao 等[14]利用精英反向学习策略,提高了种群的多样性,提升了鲸鱼优化算法的收敛速度;同时引入黄金正余弦优化算法的寻优方式,协调了算法的全局探索与局部开发能力。Zhang 等[15]采用精英主义策略、莱维飞行策略和反对学习理论来改进基本萤火虫算法,该算法在目标函数稀疏的问题上表现出了较好的搜索能力。综上所述,虽然上述改进策略在一定程度上提高了算法的搜索能力,但受限于算法本身,仍然存在收敛精度低、易受局部极值影响从而出现早熟现象的问题。

受到阿基米德定理的启发,Hashim 等[17]于 2020 年提出了阿基米德优化算法(Archimedes Optimization Algorithm, AOA)。与其他基于种群的优化算法类似,AOA 将流体中的物体视为种群,其中个体通过不断调节自身的密度和体积,从而使得整个种群达到平衡状态,此过程被视为 AOA 的寻优过程。相对于其他算法,阿基米德优化算法局部搜索能力极强,寻优精度高。但是该算法仍存在很多不足,如在迭代中后期算法种群多样性减少,导致算法容易陷入局部最优。

针对上述问题,学者们提出了诸多改进方案,譬如 Sun 等[18]将动态反向学习引入到阿基米德优化算法中,提高了候选解的质量;并引入基于混沌理论的局部搜索理论,提高了算法的搜索能力。Houssein 等[19]提出了一种基于正交学习和局部逃离算子的阿基米德优化算法,通过正交阵列对个体进行评估,从而加快算法的收敛速度。

本文提出了一种改进的阿基米德优化算法(An Improved Archimedes Optimization Algorithm, IAOA)。首先,引入佳点集初始化种群,提高初始解的质量,为算法的快速收敛奠定基础;其次,提出了自适应反馈调节因子,用于平衡算法的全局搜索与局部开发的能力;最后,提出了莱维旋转变换策略,解决了算法在迭代中后期种群多样性减少的问题,提高了算法跳出局部最优的能力。将 IAOA 在 14 个基准测试函数、Wilcoxon 秩和检验以及部分 CEC2014 函数上进行仿真实验,验证了 IAOA 的优越性和鲁棒性;并将 IAOA 应用于焊接梁设计问题,验证了 IAOA 的有效性。

2 阿基米德优化算法

阿基米德优化算法(AOA)是一种基于种群的优化算法,其设计灵感来自于阿基米德定理。该原理指出,当物体完全或部分浸入流体中时,液体给物体施加的浮力大小与排出液体的质量(体积)大小成正比[20]。若物体受到的浮力等于排出液体质量时,则视该物体处于平衡状态。

在 AOA 中,种群是浸透在液体中的物体,个体通过调整自身的密度(p)、体积(v)和加速度(a),使得自身达到平衡状态。根据浸透在液体中的物体是否发生碰撞,AOA 将其分为全局探索和局部搜索阶段。若未发生碰撞,则算法进入全局探索阶段;否则进入局部开发阶段。设置迁移算子(TF),用于两个阶段的切换,其定义如下:

$$TF=\exp\left(\frac{t-t_{\max}}{t_{\max}}\right) \tag{1}$$

其中,t 和 $t_{\max}$ 分别表示当前迭代次数和最大迭代次数。若 $TF\leqslant 0.5$,AOA 则进入全局探索阶段;否则进入局部开发阶段。

在初始化阶段,AOA 会随机初始化每个对象的体积(vol)、密度(den)、加速度(acc)。在此过程中,AOA 将评估初始种群,选取当前最优个体(x_{best})、最优个体的密度(den_{best})、体积(vol_{best})、加速度(acc_{best}),用于其他个体密度、体积和加速度的更新。

$$\begin{aligned} den_i^{t+1}&=den_i^t+rand\times(den_{\text{best}}-den_i^t)\\ vol_i^{t+1}&=vol_i^t+rand\times(vol_{\text{best}}-vol_i^t)\end{aligned} \tag{2}$$

其中,den_i^t 和 den_i^{t+1} 为在 t 代中第 i 个和第 $i+1$ 个个体的密度,vol_i^t 和 vol_i^{t+1} 为在第 t 代中第 i 个和第 $i+1$ 个个体的体积。

当 $TF\leqslant 0.5$ 时,算法进行全局搜索,根据式(3)对个体的加速度进行更新。

$$acc_i^{t+1}=\frac{den_{mr}+vol_{mr}\times acc_{mr}}{den_i^{t+1}\times vol_i^{t+1}} \tag{3}$$

其中,acc_i^{t+1} 为在 $t+1$ 代中第 i 个体的加速度,den_{mr} 和 vol_{mr} 为在当前迭代中随机挑选个体的密度和体积。

当 $TF\geqslant 0.5$ 时,算法进行局部开发。根据式(4)对个体的加速度进行更新。

$$acc_i^{t+1}=\frac{den_{\text{best}}+vol_{\text{best}}\times acc_{\text{best}}}{den_i^{t+1}\times vol_i^{t+1}} \tag{4}$$

根据式(5),AOA 将对个体的加速度进行归一化操作,用于对个体位置的更新。

$$acc_{i\text{norm}}^{t+1}=u\times\frac{acc_i^{t+1}-\min(acc)}{\max(acc)-\min(acc)}+l \tag{5}$$

其中,$acc_{i\text{norm}}^{t+1}$ 为在 $t+1$ 代中第 i 个个体归一化后的加速度;u 和 l 用于调整归一化的范围。

在全局搜索阶段,个体位置根据式(6)进行位置更新:

$$x_i^{t+1}=x_i^t+C_1\times rand\times acc_{i\text{norm}}^{t+1}\times d\times(x_{rand}-x_i^t) \tag{6}$$

其中,$\boldsymbol{x}_i^{t+1}$ 和 $\boldsymbol{x}_i^t$ 为 $t+1$ 代和 t 代中第 i 个个体的位置,$\boldsymbol{x}_{rand}$ 为第 t 代中随机个体的位置,$rand\in(0,1)$ 之间的随机数,C_1 为固定常数。其中,d 为密度因子,更新公式为:

$$d^{t+1}=\exp\left(\frac{t_{\max}-t}{t_{\max}}\right)-\left(\frac{t}{t_{\max}}\right) \tag{7}$$

在局部开发阶段,个体位置根据式(8)进行位置更新。

$$\begin{aligned}\boldsymbol{x}_i^{t+1}=&\boldsymbol{x}_{\text{best}}^t+F\times C_2\times\ rand\times acc_{i\text{norm}}^{t+1}\times d\times\\&(T\times\boldsymbol{x}_{\text{best}}-\boldsymbol{x}_i^t)\end{aligned} \tag{8}$$

其中,C_2 为固定常数;$T=C_3\times TF$,且 $T\in[C_3\times 0.3,1]$;F 为方向因子,用于决定迭代的位置更新方向,其定义为:

$$F=\begin{cases}+1, & \text{if } P\leqslant 0.5\\ -1, & \text{if } P>0.5\end{cases} \tag{9}$$

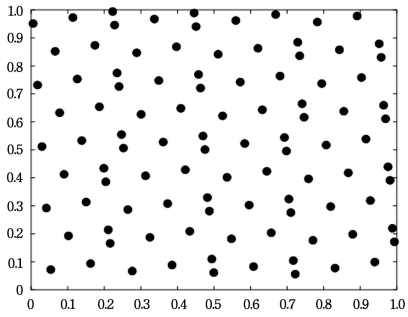
其中，$P=2\times rand-C_4$，C_4为固定常数。

3　改进阿基米德优化算法

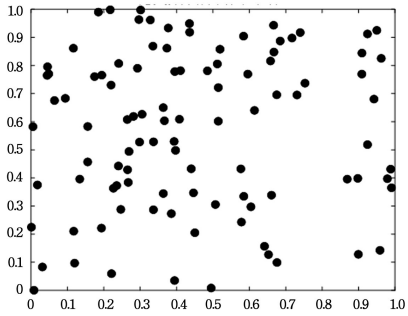
3.1　佳点集种群初始化

初始种群在解空间中的细微不同都可能影响算法的进化方向，最后产生完全不同的最终结果。在初始化阶段，AOA容易受到随机的干扰，发生聚集现象。本文通过引入佳点集初始化方法，来拟提高算法在解空间上的遍历能力。

佳点集最初由 Luo 等[21]提出，根据文献[22]中的佳点集产生原理得到点分布序列，再将初始化序列映射到解空间。该方法的构造与空间维数无关，能够很好地适应高维问题，且取点稳定。理论上，使用佳点集法生成的点的偏差远小于随机选择的点。分别采用佳点集法和随机法生成二维初始种群并对结果进行对比，对比结果如图 1 所示。在同取 100 个点的条件下，与随机初始化种群相比，佳点集方法能够使得初始化种群均匀地分布在搜索空间中。



(a)佳点集法产生的二维初始种群($N=100$)



(b)随机初始化的二维初始种群($N=100$)

图 1　二维初始种群分布图

Fig. 1　2D initial population distribution

3.2　自适应反馈调节因子

在 AOA 迭代后期进行局部开发时，根据位置更新式(8)可知，个体收敛于当前最优解的方式是直接跳跃到当前最优位置附近，而不是像粒子群优化算法那样向最优位置移动。这是一种盲目跟进的行为，导致 AOA 极容易陷入局部最优。又根据密度因子式(7)可知，随着迭代次数的增加，密度因子会呈现从 1 到 0 线性递减的趋势。密度因子过小会导致个体信息的缺失，从而导致了种群多样性的减少。因此，本文提出了自适应反馈调节因子对算法进行改进。本文将种群进化成功率[23]以及个体归一化后的适应度作为反馈调节参数对密度因子进行改进，使得个体根据种群所处的实际环境以及自身的适应度动态调节，具体操作如下。

以最小值问题为例，个体 i 在第 t 次迭代中的成功值 $S(i,t)$定义为：

$$S(i,t)=\begin{cases}1, & fit(pbest^i(t))<fit(pbest^i(t-1))\\ 0, & fit(pbest^i(t))=fit(pbest^i(t-1))\end{cases} \tag{10}$$

其中，$pbest^i(t)$表示迭代至第 t 次时，个体 i 的历史最优位置；$fit(*)$为适应度函数；根据个体 i 的成功值 $S(i,t)$，种群的成功率(P_s)定义为个体历史最优位置对应的进化成功个体数占种群大小的比例，计算式如下：

$$P_s(t)=\frac{1}{N}\sum_{i=1}^{N}S(i,t) \tag{11}$$

其中，N 为种群的大小。

在迭代初期，个体广泛分布在解空间中，进化成功的个体相对较多，种群进化成功率较高，侧面说明最优个体的引领方向正确。在迭代后期，种群个体逐渐聚集，若进化成功率降低，则应该保留更多的个体信息来保证算法的局部开发能力。将种群进化成功率 P_s和密度因子 d 结合，动态反馈算法的进化状态，以此来平衡算法的全局开发能力。自适应反馈调节因子如下：

$$d_i^{t+1}=2*\exp\left(\frac{t_{\max}-t}{t_{\max}}\right)-\left(\frac{t}{t_{\max}}\right)*(\alpha*P_s(t)+\beta*F_i) \tag{12}$$

其中，参数 α 表示种群进化成功率重要性系数，F_i定义为：

$$F_i=\lambda^{-(f(X_i)-f_{\min}+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)} \tag{13}$$

其中，F_i表示第 i 个个体归一化后的适应度系数；β 为该系数的重要性系数；λ 为衰减因子；ε 为一个非常小的有理数，以防止分母为 0 的情况。自适应反馈调节因子会根据适应度的变化情况动态调整大小，若当前个体适应度接近最优适应度，自适应反馈调节因子会保持原来的密度因子的性质，加快算法的收敛速度。而适应度较差的个体，其自适应调节因子接近最大值，保留了自身更多信息，保证了算法的局部开发能力。

3.3　莱维旋转变换策略

在保持 AOA 的种群多样性基础上，为了提高个体的探索能力，避免在迭代后期出现种群陷入局部最优的现象，本文提出了莱维旋转变换策略。以莱维步长[24]为半径、最优个体为中心对个体进行旋转变换操作，以提高种群的多样性和收敛速度，其中莱维旋转变换算子为：

$$\boldsymbol{x}_i^t=\boldsymbol{x}_{\text{best}}+s*\frac{1}{n\parallel\boldsymbol{x}_i^t\parallel_2}\boldsymbol{R}_r\boldsymbol{x}_i^t \tag{14}$$

其中，$\boldsymbol{R}_r\in\boldsymbol{R}_{n*n}$是一个元素取值在$[-1,1]$之间均匀分布的随机矩阵；$\parallel\cdot\parallel$为向量 2-范数；$s$ 为结合莱维的旋转因子；理论上[25]能将位置旋转到以 s 为半径的任何位置，可用于控制解的搜索范围，其表达式为：

$$s=0.01*\frac{\mu}{|\nu|^{1/\beta}}(\boldsymbol{x}_i^t-\boldsymbol{x}_{\text{best}}) \tag{15}$$

其中，μ 和 ν 服从正态分布，$\mu\sim N(0,\sigma_\mu^2)$，$\nu\sim N(0,\sigma_\nu^2)$。

$$\sigma_\mu=\left\{\frac{\Gamma(1+\beta)\cdot\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left[\frac{(1+\beta)}{2}\right]\beta\cdot 2^{\frac{(\beta-1)}{2}}}\right\}^{1/\beta} \tag{16}$$

其中，β 的取值会影响搜索半径的大小，β 的取值越大，其局部开放能力就越强[26]。

莱维旋转变换算子可以对个体以一定的角度进行旋转，以最大限度地避免生成重复个体；并且其结合莱维步长的短距离步长和长距离步长相间的特点，扩大群体的搜寻范围，协助算法在必要时跳出局部最优，以提高算法的搜索性能。最后，为了兼顾种群搜索领域的广度与深度，本文根据个体的适应度值，计算个体的进化概率，如式(17)所示。个体的可行解质量越差，则该个体被选择进行莱维旋转变换操作的概率就越大，同时本文还采用贪婪选择更新解，在保留优势个体的同时兼顾个体的探索能力。

$$p_i=\frac{fit_i}{\sum_{i=1}^{N} fit_i} \tag{17}$$

其中，p_i是选择概率；fit_i的表达式为：

$$fit_i=\begin{cases}\dfrac{1}{1+f_i}, & f_i\geqslant 0\\ 1+|f_i|, & f_i<0\end{cases} \tag{18}$$

其中，f_i为第 i 个个体的适应度值。

3.4 IAOA

IAOA 如算法 1 所示。

算法 1 IAOA

```
procedure
初始化种群大小 N,最大迭代次数 M,算法内参数 C1,C2,C3,C4,β,α;
通过佳点集法生成初始种群 x;
随机生成个体密度(den),体积(vol);
评估初始种群适应度值,并选出最优个体 x_best;
while(t < M)do
    for i =1,…,N do
    根据式(2)更新个体密度(den_i^t),体积(vol_i^t);
    根据式(1)和式(12)更新转移算子(TF)和密度下降因子(d_{t+1}^t)。
      if TF ≤ 0.5 Then
        根据式(3)更新个体加速度(acc_i^t);
        根据式(5)归一化个体加速度(acc_{i-norm}^t);
        根据式(6)更新个体位置;
      else
        根据式(4)更新个体加速度(acc_i^t);
        根据式(5)归一化个体加速度(acc_{i-norm}^t);
        根据式(8)更新个体位置;
      end if
        根据式(10)更新成功值 S
        根据式(11)更新成功率 Ps
        根据式(17)、式(18)计算选择概率 pi
      if pi≤ rand Then
        根据式(14)、式(15)执行莱维旋转变换策略;
      end if
        评估当前种群适应度值,并选出当前最优个体;
    end for
      Set t=t+1
end while
return 最优个体适应度
end procedure
```

3.5 IAOA 的时间复杂度

时间复杂度从理论上反映了算法的收敛速度，假设算法的最大迭代次数为 M，种群规模为 N，空间维度为 D，则求解目标的适应度函数计算量为 $f(D)$，由文献[17]可知，标准的 AOA 时间复杂度为 $O(M(N*f(D)+N*D))$。IAOA 时间复杂度的分析如下。

由算法 1 的步骤可看出，在初始化阶段：假设初始化参数比之前多 x_1，且每一维度产生佳点时间为 x_2，所增加步骤对应的时间复杂度为 $O(x_1)$，$O(N*D*x_2)$，略去低阶项后，IAOA 在初始阶段的时间复杂度为：

$$O_1(x_1+N*D*x_2+M(N*f(D)+N*D))=O_1(M*(N*f(D)+N*D)) \tag{19}$$

在自适应权重调节因子阶段，假设计算种群成功率和归一化后的适应度系数需要增加 x_3 步，所增加步骤对应的时间复杂度为 $O(M*x_3)$，略去低阶项后，该阶段的时间复杂度为：

$$O_2(M*x_3+M*(N*f(D)+N*D))=O_2(M*(N*f(D)+N*D)) \tag{20}$$

在莱维旋转变换算子阶段，假设计算需要的参数的时间为 x_4，根据式(14)以及式(17)，增加步骤对应的时间复杂度最多为 $O(M*x_4+M*N*f(D))$，略去低阶项和系数项之后，该阶段的时间复杂度为：

$$O_3(M*x_4+M*(2N*f(D)+N*D))=O_3(M*(N*f(D)+N*D)) \tag{21}$$

综上所述，IAOA 的时间复杂度为：

$$O_1+O_2+O_3=O(M*(N*f(D)+N*D)) \tag{22}$$

将 IAOA 与标准的 AOA 的时间复杂度进行对比，理论分析上它们属于同一数量级，说明改进后的算法在理论上没有增加时间复杂度。

3.6 IAOA 算法的实现步骤

本文提出的改进算法的流程图如图 2 所示。

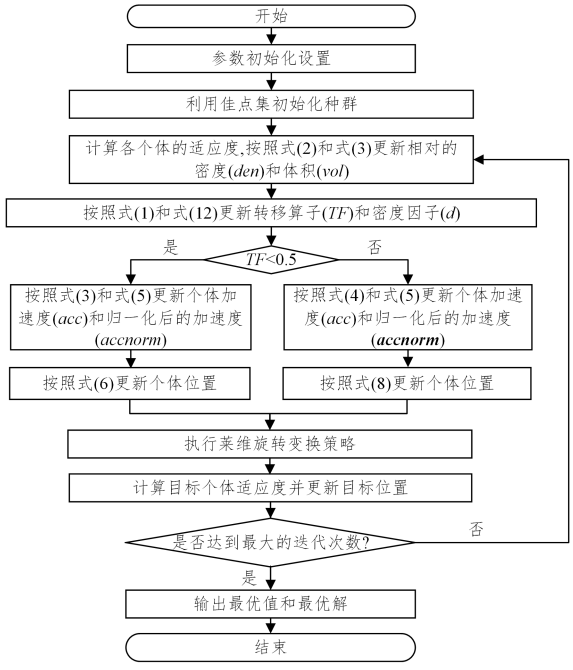


图 2 IAOA 流程图

Fig. 2 IAOA flow chart

4　数值实验与结果分析

为了全面验证本文改进策略的有效性和鲁棒性，以及改进后算法的性能，本文中的验证实验分为 4 个部分进行。

(1)将 AOA 与不同改进策略进行对比，验证改进策略的有效性。

(2)将 IAOA 与其他群智能算法进行对比实验，验证 IAOA 的优越性。

(3)通过 Wilcoxon 秩和检验验证本文算法与其他对比算法之间的差异显著性。

(4)在 CEC2014 基准函数中选取部分单峰值、多峰值、混合型以及复合型函数进行优化测试，验证本文算法的有效性和鲁棒性。

实验引入 14 个基准函数，其分为 3 类：第一类为单峰值函数，如表 1 中 F_1-F_6 所示；第二类为多峰函数，如表 1 中的 F_7-F_{11}所示；第三类为固定维度的多峰函数，如表 1 中的 $F_{12}-F_{14}$所示。

表 1　基准测试函数

Table 1　Benchmark test functions

NO	Function name	Solution space	Optimal value
F_1	Sphere	$[-100,100]^D$	$0.00\times10^{+00}$
F_2	Schwefel's 2.22	$[-10,10]^D$	$0.00\times10^{+00}$
F_3	Schwefel's 1.2	$[-100,100]^D$	$0.00\times10^{+00}$
F_4	Schwefel's 2.21	$[-100,100]^D$	$0.00\times10^{+00}$
F_5	Sumsquares	$[-10,10]^D$	$0.00\times10^{+00}$
F_6	Zakharov	$[-5,10]^D$	$0.00\times10^{+00}$
F_7	Quartic Niose	$[-1.28,1.28]^D$	$0.00\times10^{+00}$
F_8	Apline	$[-10,10]^D$	$0.00\times10^{+00}$
F_9	Schwefel's 2.26	$[-500,500]^D$	$-1.26\times10^{+04}$
F_{10}	Penalized 1	$[-50,50]^D$	$0.00\times10^{+00}$
F_{11}	Ackley's	$[-32,32]^D$	$0.00\times10^{+00}$
F_{12}	Griewank	$[-600,600]^D$	$0.00\times10^{+00}$
F_{13}	×10ggcrate	$[-2pi,2pi]^2$	$0.00\times10^{+00}$
F_{14}	Schaffer 2	$[-100,100]^2$	$0.00\times10^{+00}$

算法的基本参数如下：种群规模为 30，空间维度为 100/500 dim，最大迭代次数为 1 000。每种算法的具体参数如表 2 所列。

表 2　算法参数设置

Table 2　Algorithm parameter setting

Algorithm	Main parameters
AOA	$C_1=2,C_2=6$ $C_3=1,C_4=2$(基准测试函数) $C_3=2,C_4=0.5$(CEC2014，工程问题)
IAOA	$\beta=0.5,\alpha=0.5,C_1=2,C_2=6$ $C_3=1,C_4=2$(基准测试函数) $C_3=2,C_4=0.5$(CEC2014，工程问题)
PSO	$C_1=2,C_2=2,w_{max}=0.9,w_{min}=0.2$
DE	$F=0.4,CR=0.5$
ACO	$\alpha=1,\beta=7,\rho=0.3$
SCA	$a=2$
GWO	$a_{first}=2,a_{final}=0$
EO	$GP=0.5,a_1=2,a_2=1$
BOA	$P=0.8;a=0.1;b=0.025;c=0.01$

4.1　与不同的改进策略的对比

为了验证 IAOA 以及每个改进点的有效性，将改进的阿基米德优化算法(IAOA)与阿基优化米德算法(AOA)、加入佳点集初始化策略的阿基米德优化算法(GNS-AOA)、加入自适应反馈调节因子的阿基米德优化算法(AF-AOA)以及融入莱维旋转变换算子的阿基米德优化算法(LR-AOA)对 14 个经典基准测试函数在 30dim 条件下以及两个固定维度测试函数上进行对比实验，实验独立运行 30 次，采集基准测试函数的平均值和标准差，实验结果如表 3 所列。

通常采用平均值来反映算法的寻优能力，用标准差反映算法的稳定性。由表 3 可知，在单峰值测试函数中，IAOA 的收敛精度优于其他算法，尤其是对于函数 F_1，F_3，F_5，F_6，IAOA 均能达到最优值。对于函数 F_2，F_4，F_7，IAOA 虽然没有收敛到最优值，但是收敛精度高于其他算法，且标准差最小。在多峰值函数上，从表 3 中可以看出，IAOA 相比其他算法也表现出了较强的寻优能力。尤其是对于函数 F_9，虽然没有达到最优值，但是收敛精度提高至 10^{-200}。在标准差的对比中，IAOA 的标准差最小，表现出了较强的稳定性。最后在 F_{13}和 F_{14}上，IAOA 的收敛精度与其他算法结果值相近。从整体上看，IAOA 在多个函数上的平均寻优精度最高，且标准差最小，表现出了较强的鲁棒性。

表 3　基准测试函数的测试结果

Table 3　Test results of benchmark test function

	AOA		GNS-AOA		AF-AOA		LR-AOA		IAOA	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*
F_1	1.40×10^{-171}	$0.00\times10^{+00}$	4.80×10^{-200}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	2.84×10^{-221}	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_2	1.24×10^{-90}	3.92×10^{-90}	3.53×10^{-92}	1.12×10^{-91}	1.58×10^{-201}	$0.00\times10^{+00}$	2.23×10^{-120}	7.00×10^{-120}	$\mathbf{2.10\times10^{-299}}$	$\mathbf{0.00\times10^{+00}}$
F_3	1.18×10^{-161}	3.72×10^{-161}	2.00×10^{-161}	6.33×10^{-161}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	7.90×10^{-194}	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_4	2.05×10^{-87}	6.48×10^{-87}	6.59×10^{-83}	2.08×10^{-82}	5.07×10^{-207}	$0.00\times10^{+00}$	2.46×10^{-120}	7.10×10^{-120}	$\mathbf{2.48\times10^{-273}}$	$\mathbf{0.00\times10^{+00}}$
F_5	1.14×10^{-172}	$0.00\times10^{+00}$	7.67×10^{-177}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	4.40×10^{-229}	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_6	2.83×10^{-100}	8.95×10^{-100}	7.65×10^{-93}	2.42×10^{-92}	1.73×10^{-217}	$0.00\times10^{+00}$	9.48×10^{-208}	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_7	4.06×10^{-04}	2.43×10^{-04}	3.51×10^{-04}	2.37×10^{-04}	9.56×10^{-04}	6.15×10^{-04}	9.89×10^{-05}	1.19×10^{-04}	$\mathbf{8.59\times10^{-05}}$	$\mathbf{5.44\times10^{-05}}$
F_8	1.99×10^{-91}	6.29×10^{-91}	1.69×10^{-99}	5.34×10^{-99}	1.20×10^{-207}	$0.00\times10^{+00}$	2.30×10^{-109}	7.60×10^{-109}	$\mathbf{9.05\times10^{-251}}$	$\mathbf{2.84\times10^{+00}}$
F_9	$-3.85\times10^{+03}$	$5.20\times10^{+02}$	$-3.53\times10^{+03}$	$2.61\times10^{+02}$	$-4.35\times10^{+03}$	$3.71\times10^{+02}$	$-8.97\times10^{+03}$	$1.90\times10^{+03}$	$\mathbf{-8.57\times10^{+03}}$	$\mathbf{1.65\times10^{+03}}$
F_{10}	8.17×10^{-01}	3.69×10^{-01}	7.56×10^{-01}	1.08×10^{-01}	5.79×10^{-01}	4.83×10^{-01}	6.20×10^{-05}	8.08×10^{-05}	$\mathbf{5.15\times10^{-05}}$	$\mathbf{6.63\times10^{-05}}$
F_{11}	1.24×10^{-15}	1.12×10^{-15}	1.24×10^{-15}	1.12×10^{-15}	8.88×10^{-16}	$0.00\times10^{+00}$	3.73×10^{-15}	1.50×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{0.00\times10^{+00}}$
F_{12}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_{13}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$
F_{14}	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$0.00\times10^{+00}$	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$

横向来看，加入自适应反馈调节因子的阿基米德优化算法的表现优于其他两种改进策略。其次是融入莱维旋转变换

算子的阿基米德优化算法，加入佳点集初始化策略的阿基米德优化算法的表现次之。对于 F_1 函数的求解，GNS-AOA 和 LR-AOA 相比基本的 AOA 在求解精度上分别提升了 30 个数量级和 50 个数量级，并且在求解其他基准测试函数时，GNS-AOA 和 LR-AOA 的优化效果优于基本的 AOA。加入自适应反馈调节因子 AF-AOA 后，在求解 F_1，F_3，F_5，F_{12}，F_{13}，F_{14} 时均能找到理论最优值，并且标准差为 0。对于单峰值函数 F_2，F_4，F_6 以及多峰值函数 F_8，相比基本的 AOA，AF-AOA 在求解精度平均值上提升了近 110 个数量级，说明该改进策略对局部极值点具有一定的规避能力。并且，融合了 3 种改进策略的 IAOA 性能最好，说明了 3 种改进策略的互补性。综合来说，3 种改进策略都在一定程度上改进了算法的性能，改进策略可靠有效。

为了形象地观测 IAOA 的性能，对 IAOA，AOA，GNS-AOA，AF-AOA，LR-BOA，PSO，BOA 进行收敛性分析。图 3 为部分测试函数的平均收敛曲线图。因为寻优精度较高，为了便于观察平均曲线的收敛情况，本文的纵坐标取以 10 为底的对数。

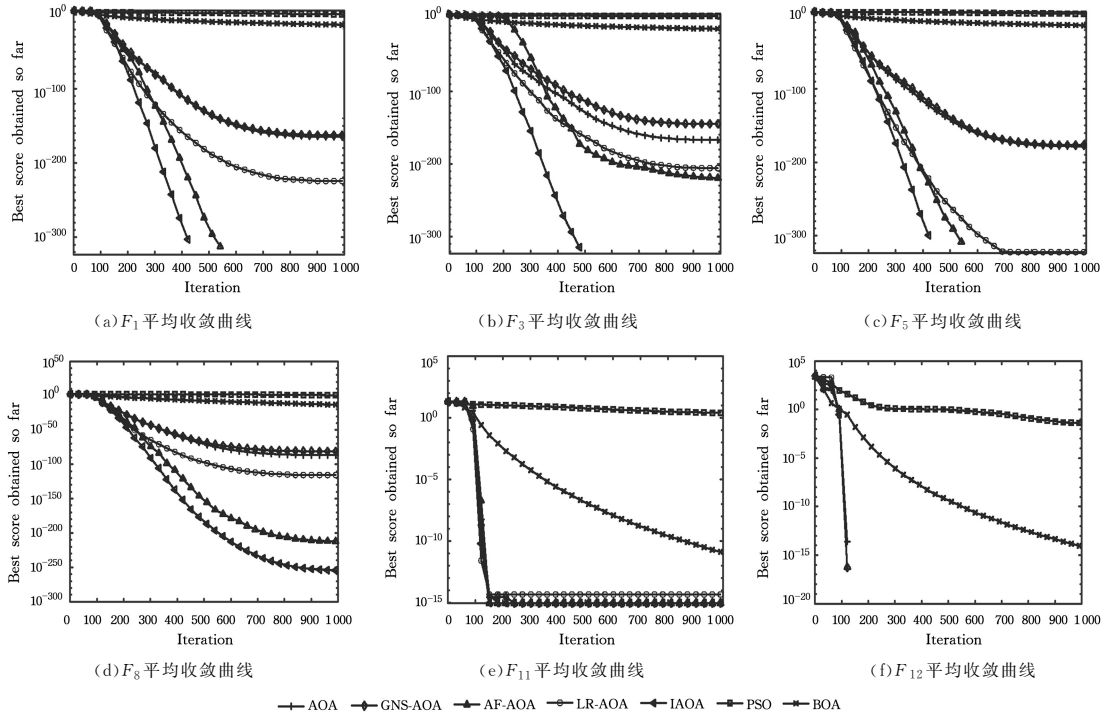


图 3 基准函数的平均收敛曲线

Fig. 3 Average convergence curve of benchmark function

通常采用单峰值函数来评价算法的局部开发能力，采用多峰函数来评价算法的全局探索能力。从图 3 可以明显看出，无论是单峰值函数还是多峰函数，IAOA 的收敛曲线下降最快，并且其收敛曲线均在其他算法的下方，这说明改进后的 IAOA 具有更快的收敛速度以及更高的收敛精度。具体来说，图 3(a)－图 3(c)为单峰值函数的收敛曲线，从图中可以看出 AF-AOA 具有较高的收敛精度以及更快的收敛速度。这得益于融入更多信息的自适应反馈调节因子增强了算法的局部开发能力，使得算法能随着迭代次数增加，持续地进行局部开发。图 3(d)－图 3(f)为多峰函数的收敛曲线，从图中可知，AF-AOA 和 LR-AOA 的平均收敛曲线均处于 AOA 收敛曲线的下方，且下降速度快，求解精度较高，使得 IAOA 在相同的迭代次数下具有更快的收敛速度以及更高的收敛精度。其次，相比 PSO 以及 BOA 算法，AOA 算法表现出了更强的寻优能力，这得益于 AOA 算法本身在设计上的优越性。

表 3 以及图 3 的分析结果证明了本文提出的改进策略的有效性。

4.2 与其他群智能优化算法的对比

为了进一步体现 IAOA 的优化性能，将 IAOA 与粒子群算法(Particle swarm optimization，PSO)、灰狼算法[27](Grey Wolf Optimizer，GWO)、正余弦算法[28](Sine Cosine Algorithm，SCA)、蝴蝶优化算法[29](Butterfly optimization algorithm，BOA)分别在 100/500 dim 的条件下独立运行 30 次，实验结果如表 4 和表 5 所列。由表 4 可知，在求解维度为 100 dim 时，IAOA 在求解单峰值函数(F_1－F_6)时精度最高，并且在求解 F_1，F_3，F_5 时平均寻优精度能达到理论最优值。对于 F_2，IAOA 的精度也达到了 10^{-240}，远优于其他算法。其次，在求解多峰函数时，相比 PSO，GWO，SCA，BOA 这 4 种算法，IAOA 也表现出了更强的寻优能力。随着维度的增加，算法的求解难度也呈指数级递增，因此算法在求解精度上有所降低，这属于正常现象。

在求解维度为 500 dim 的函数时，IAOA 在函数 F_1，F_3，F_5 以及 F_{12} 上，均能找到理论最优值，表现出了较强的寻优能力。并且在求解复杂多峰值函数 F_8 时，IAOA 的求解精度达到了 10^{-248}，远远优于其他算法。由此认为，融入自适应反馈调节因子以及莱维旋转变换策略的 IAOA 能有效地避免陷入局部最优，也保证了算法在进化过程中的种群多样性，使得 IAOA 获得了较强的寻优能力。

表 4　维度为 100 时各算法在 12 个测试函数上的实验结果

Table 4　Experimental results of different algorithms on 12 test functions with dimension of 100

	IAOABOA		PSOBOA		GWOBOA		SCABOA		BOA	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*
F_1	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$3.11\times10^{+00}$	$3.01\times10^{+00}$	2.87×10^{-29}	6.07×10^{-29}	$3.11\times10^{+00}$	$3.01\times10^{+00}$	2.87×10^{-29}	6.07×10^{-29}
F_2	$\mathbf{4.24\times10^{-240}}$	$\mathbf{0.00\times10^{+00}}$	$1.23\times10^{+01}$	$4.28\times10^{+00}$	6.29×10^{-18}	3.61×10^{-18}	$1.23\times10^{+01}$	$4.28\times10^{+00}$	6.29×10^{-18}	3.61×10^{-18}
F_3	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$1.08\times10^{+04}$	$2.34\times10^{+03}$	$7.18\times10^{+00}$	$1.56\times10^{+01}$	$1.08\times10^{+04}$	$2.34\times10^{+03}$	$7.18\times10^{+00}$	$1.56\times10^{+01}$
F_4	$\mathbf{3.05\times10^{-228}}$	$\mathbf{0.00\times10^{+00}}$	$9.32\times10^{+00}$	$1.28\times10^{+00}$	9.70×10^{-03}	4.45×10^{-02}	$9.32\times10^{+00}$	$1.28\times10^{+00}$	9.70×10^{-03}	4.45×10^{-02}
F_5	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$1.07\times10^{+02}$	$7.00\times10^{+01}$	1.85×10^{-30}	2.13×10^{-30}	$1.07\times10^{+02}$	$7.00\times10^{+01}$	1.85×10^{-30}	2.13×10^{-30}
F_6	$\mathbf{2.23\times10^{-49}}$	$\mathbf{1.22\times10^{-48}}$	$3.65\times10^{+03}$	$1.69\times10^{+03}$	$2.41\times10^{+00}$	$2.33\times10^{+00}$	$3.65\times10^{+03}$	$1.69\times10^{+03}$	$2.41\times10^{+00}$	$2.33\times10^{+00}$
F_7	$\mathbf{3.94\times10^{-04}}$	$\mathbf{4.99\times10^{-04}}$	$1.42\times10^{+03}$	$2.82\times10^{+02}$	2.28×10^{-03}	8.35×10^{-04}	$1.42\times10^{+03}$	$2.82\times10^{+02}$	2.28×10^{-03}	8.35×10^{-04}
F_8	$\mathbf{7.93\times10^{-236}}$	$\mathbf{0.00\times10^{+00}}$	$6.76\times10^{+00}$	$2.39\times10^{+00}$	1.44×10^{-04}	4.50×10^{-04}	$6.76\times10^{+00}$	$2.39\times10^{+00}$	1.44×10^{-04}	4.50×10^{-04}
F_9	$\mathbf{-2.38\times10^{+04}}$	$\mathbf{4.23\times10^{+03}}$	$-1.92\times10^{+04}$	$5.31\times10^{+03}$	$-1.72\times10^{+04}$	$1.67\times10^{+03}$	$-1.92\times10^{+04}$	$5.31\times10^{+03}$	$-1.72\times10^{+04}$	$1.67\times10^{+03}$
F_{10}	$\mathbf{2.07\times10^{-01}}$	$\mathbf{1.21\times10^{-01}}$	$1.72\times10^{+00}$	7.56×10^{-01}	2.98×10^{-01}	1.63×10^{-02}	$1.72\times10^{+00}$	7.56×10^{-01}	2.38×10^{-01}	6.63×10^{-02}
F_{11}	$\mathbf{8.89\times10^{-16}}$	$\mathbf{0.00\times10^{+00}}$	$2.65\times10^{+00}$	3.25×10^{-01}	1.12×10^{-13}	1.06×10^{-14}	$2.65\times10^{+00}$	3.25×10^{-01}	1.12×10^{-13}	1.06×10^{-14}
F_{12}	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	4.18×10^{-02}	1.87×10^{-02}	1.20×10^{-03}	4.56×10^{-03}	4.18×10^{-02}	1.87×10^{-02}	1.20×10^{-03}	4.56×10^{-03}

表 5　维度为 500 时各算法在 12 个测试函数上的实验结果

Table 5　Experimental results of different algorithms on 12 test functions with dimension of 500

	IAOABOA		PSOBOA		GWOBOA		SCABOA		BOA	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Std*
F_1	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$2.48\times10^{+03}$	$2.10\times10^{+02}$	1.46×10^{-12}	8.51×10^{-13}	$2.48\times10^{+03}$	$2.10\times10^{+02}$	1.46×10^{-12}	8.51×10^{-13}
F_2	$\mathbf{7.61\times10^{-231}}$	$\mathbf{0.00\times10^{+00}}$	$1.10\times10^{+38}$	$6.02\times10^{+38}$	5.95×10^{-08}	1.60×10^{-08}	$1.10\times10^{+38}$	$6.02\times10^{+38}$	5.95×10^{-08}	1.60×10^{-08}
F_3	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$4.69\times10^{+05}$	$1.04\times10^{+05}$	$1.34\times10^{+05}$	$6.22\times10^{+04}$	$4.69\times10^{+05}$	$1.04\times10^{+05}$	$1.34\times10^{+05}$	$6.22\times10^{+04}$
F_4	$\mathbf{1.31\times10^{-222}}$	$\mathbf{0.00\times10^{+00}}$	$2.67\times10^{+01}$	$1.42\times10^{+00}$	$5.92\times10^{+01}$	$6.72\times10^{+00}$	$2.67\times10^{+01}$	$1.42\times10^{+00}$	$5.92\times10^{+01}$	$6.72\times10^{+00}$
F_5	$\mathbf{0.00\times10^{+00}}$	$0.00\times10^{+00}$	$2.49\times10^{+05}$	$2.20\times10^{+04}$	8.70×10^{-13}	5.02×10^{-13}	$2.49\times10^{+05}$	$2.20\times10^{+04}$	8.70×10^{-13}	5.02×10^{-13}
F_6	$\mathbf{3.98\times10^{-02}}$	$\mathbf{1.64\times10^{-01}}$	$9.72\times10^{+04}$	$2.27\times10^{+04}$	$3.22\times10^{+03}$	$4.45\times10^{+02}$	$9.72\times10^{+04}$	$2.27\times10^{+04}$	$3.22\times10^{+03}$	$4.45\times10^{+02}$
F_7	$\mathbf{3.77\times10^{-04}}$	$\mathbf{2.91\times10^{-04}}$	$5.67\times10^{+04}$	$2.35\times10^{+03}$	1.12×10^{-02}	2.74×10^{-03}	$5.67\times10^{+04}$	$2.35\times10^{+03}$	1.12×10^{-02}	2.74×10^{-03}
F_8	$\mathbf{1.70\times10^{-231}}$	$\mathbf{0.00\times10^{+00}}$	$4.46\times10^{+02}$	$3.13\times10^{+01}$	3.49×10^{-03}	3.09×10^{-03}	$4.46\times10^{+02}$	$3.13\times10^{+01}$	3.49×10^{-03}	3.09×10^{-03}
F_9	$-6.05\times10^{+04}$	$\mathbf{7.77\times10^{+03}}$	$\mathbf{-8.67\times10^{+04}}$	$2.92\times10^{+04}$	$-5.93\times10^{+04}$	$9.71\times10^{+03}$	$-8.67\times10^{+04}$	$2.92\times10^{+04}$	$-5.93\times10^{+04}$	$9.71\times10^{+03}$
F_{10}	$1.02\times10^{+00}$	4.50×10^{-02}	$2.39\times10^{+04}$	$1.39\times10^{+04}$	$\mathbf{7.51\times10^{-01}}$	$\mathbf{3.38\times10^{-02}}$	$2.34\times10^{+04}$	$1.39\times10^{+04}$	7.51×10^{-01}	3.28×10^{-02}
F_{11}	$\mathbf{8.89\times10^{-16}}$	$\mathbf{0.00\times10^{+00}}$	$9.76\times10^{+00}$	3.16×10^{-01}	5.15×10^{-08}	1.52×10^{-08}	$9.76\times10^{+00}$	3.16×10^{-01}	5.15×10^{-08}	1.52×10^{-08}
F_{12}	$\mathbf{0.00\times10^{+00}}$	$\mathbf{0.00\times10^{+00}}$	$2.09\times10^{+00}$	1.07×10^{-01}	4.47×10^{-03}	1.20×10^{-02}	$2.09\times10^{+00}$	1.07×10^{-01}	4.47×10^{-03}	1.20×10^{-02}

4.3　Wilcoxon 秩和检验

本文采用 Wilcoxon 秩和检验[30]验证了 IAOA 与其他算法的显著性差异。每次运行结果在 $p=5\%$ 的显著性水平下，当 p 值小于 5%时，拒绝 H0 假设，说明两种算法的差异性显著，否则就是接受 H0 假设，说明两种算法在整体上是相同的。由于算法不能与自身进行比较，因此表中的标记“N/A”代表不适用，表示无法进行显著判断，“*R*”为显著性判断结果，“+”“−”“=”分别表示 IAOA 算法的性能优于、劣于和相当于对比算法。从表 6 中可以观察到，大部分的 p 值小于 5%，即“R”为“+”表示拒绝 H0 假设。其中 IAOA 与 GNS-AOA，AF-AOA，LR-AOA，PSO，GWO，BOA 分别有 10，11，8，12，14，12，14 个函数存在显著差异，这表明算法的优越性在统计上是显著的，即认为 IAOA 比其他算法表现出了显著的差异性，且 IAOA 显著更优。

表 6　基准函数 Wilcoxon 秩和检验的 p 值

Table 6　p value for Wilcoxon's rank-sum test on benchmark function

函数	AOA		GNS-AOA		AF-AOA		LR-AOA		PSO		GWO		BOA	
F_1	1.212×10^{-12}	+	1.212×10^{-12}	+	N/A	=	6.247×10^{-10}	+	1.212×10^{-12}	+	1.212×10^{-12}	+	1.212×10^{-12}	+
F_2	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+
F_3	1.212×10^{-12}	+	1.212×10^{-12}	+	6.617×10^{-04}	+	1.212×10^{-12}	+	1.212×10^{-12}	+	1.212×10^{-12}	+	1.212×10^{-12}	+
F_4	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+
F_5	1.212×10^{-12}	+	1.212×10^{-12}	+	N/A	=	6.247×10^{-10}	+	1.212×10^{-12}	+	1.212×10^{-12}	+	1.212×10^{-12}	+
F_6	7.695×10^{-08}	+	4.444×10^{-07}	+	2.813×10^{-02}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+
F_7	8.500×10^{-02}	-	1.297×10^{-01}	-	4.060×10^{-02}	+	9.521×10^{-04}	+	3.020×10^{-11}	+	4.616×10^{-10}	+	1.041×10^{-04}	+
F_8	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+
F_9	3.020×10^{-11}	+	3.020×10^{-11}	+	3.020×10^{-11}	+	2.755×10^{-03}	+	2.433×10^{-05}	+	3.020×10^{-11}	+	3.020×10^{-11}	+
F_{10}	1.657×10^{-11}	+	1.657×10^{-11}	+	1.657×10^{-11}	+	5.061×10^{-10}	+	3.018×10^{-11}	+	7.362×10^{-09}	+	1.657×10^{-11}	+
F_{11}	7.664×10^{-02}	-	7.664×10^{-02}	-	4.276×10^{-1}	-	1.174×10^{-06}	+	2.655×10^{-10}	+	2.094×10^{-10}	+	2.654×10^{-10}	+
F_{12}	3.337×10^{-01}	-	1.608×10^{-01}	+	N/A	=	N/A	=	1.212×10^{-12}	+	3.337×10^{-01}	-	1.206×10^{-12}	+
F_{13}	N/A	=	N/A	=	N/A	=	N/A	=	1.212×10^{-12}	+	N/A	=	1.212×10^{-12}	+
F_{14}	1.212×10^{-12}	+	1.212×10^{-12}	+	N/A	=	6.247×10^{-10}	+	1.212×10^{-12}	+	1.212×10^{-12}	+	1.212×10^{-12}	+
+/=/−	10/1/3		11/1/2		8/5/1		12/2/0		14/0/0		12/1/1		14/0/0	

4.4　CEC2014 基准函数测试

为了进一步验证 IAOA 的有效性和鲁棒性，本文在 CEC2014 目标优化函数中选取部分单峰值（Unimodal Functions，UN）、多峰值（Multimodal Functions，MF）、混合型（Hy-

brid Function, HF)和复合型(Composition Function, CF)函数进行优化求解。由于CEC2014函数具有复杂的特征,大多数算法都很难找到函数最优解,因此本文选取部分函数进行测试,结果如表7所列。实验中最大迭代次数设置为1 000,种群规模设置为30,维度为30,算法内其他参数设置与前面保持一致。表8列出了各算法在部分CEC2014测试函数上独立运行30次后的平均值以及标准差。由表8可知,IAOA在7个CEC测试函数上求得了比其他6种算法更好的结果,在CEC5和CEC9上,IAOA的平均值略高于LR-AOA以及AF-AOA,但是寻优效果优于AOA,且标准差远低于两者,表现出了较强的寻优能力。从整体来看,IAOA在包含4种类型的CEC2014测试函数上的平均寻优精度最高,且标准差最小,表现出了较强的鲁棒性。

表7 CEC2014基准函数

Table 7 CEC2014 benchmark function

Function name	Dim	Feature	Domain	Optimal value
CEC01	30	UN	[−100,100]	100
CEC03	30	UN	[−100,100]	300
CEC05	30	MF	[−100,100]	500
CEC07	30	MF	[−100,100]	700
CEC12	30	MF	[−100,100]	1 200
CEC14	30	MF	[−100,100]	1 400
CEC19	30	HF	[−100,100]	1 900
CEC27	30	CF	[−100,100]	2 900
CEC30	30	CF	[−100,100]	3 000

表8 CEC2014测试函数的结果对比

Table 8 Results comparison of CEC2014 test functions

Algorithm	Index	CEC01	CEC03	CEC05	CEC07	CEC12	CEC14	CEC19	CEC27	CEC30
PSO	*Mean*	$1.985\times10^{+07}$	$9.624\times10^{+03}$	$5.210\times10^{+02}$	$7.000\times10^{+02}$	$1.202\times10^{+03}$	$1.400\times10^{+03}$	$1.921\times10^{+03}$	$3.392\times10^{+03}$	$1.320\times10^{+04}$
	Std	$1.459\times10^{+07}$	$6.759\times10^{+03}$	8.850×10^{-02}	3.108×10^{-02}	5.268×10^{-01}	1.387×10^{-01}	$2.542\times10^{+01}$	$1.151\times10^{+02}$	$7.310\times10^{+03}$
BOA	*Mean*	$1.439\times10^{+09}$	$7.719\times10^{+04}$	$5.211\times10^{+02}$	$1.461\times10^{+03}$	$1.203\times10^{+03}$	$1.694\times10^{+03}$	$2.421\times10^{+03}$	$3.439\times10^{+03}$	$3.200\times10^{+03}$
	Std	$3.303\times10^{+08}$	$7.063\times10^{+03}$	4.135×10^{-02}	$6.718\times10^{+01}$	4.378×10^{-01}	$2.191\times10^{+01}$	$6.656\times10^{+01}$	$1.403\times10^{+02}$	$0.000\times10^{+00}$
AOA	*Mean*	$8.717\times10^{+08}$	$6.402\times10^{+04}$	$5.210\times10^{+02}$	$1.229\times10^{+03}$	$1.203\times10^{+03}$	$1.592\times10^{+03}$	$2.239\times10^{+03}$	$3.860\times10^{+03}$	$2.040\times10^{+06}$
	Std	$2.671\times10^{+08}$	$5.361\times10^{+03}$	7.235×10^{-02}	$9.942\times10^{+01}$	3.386×10^{-01}	$3.147\times10^{+01}$	$7.274\times10^{+01}$	$2.590\times10^{+02}$	$8.431\times10^{+05}$
GNS-AOA	*Mean*	$2.248\times10^{+07}$	$2.753\times10^{+04}$	$5.208\times10^{+02}$	$7.130\times10^{+02}$	$1.202\times10^{+03}$	$1.402\times10^{+03}$	$1.941\times10^{+03}$	$3.331\times10^{+03}$	$6.958\times10^{+03}$
	Std	$2.485\times10^{+07}$	$8.640\times10^{+03}$	1.484×10^{-01}	$1.508\times10^{+01}$	8.499×10^{-01}	$5.171\times10^{+00}$	$3.463\times10^{+01}$	$2.853\times10^{+02}$	$7.851\times10^{+03}$
AF-AOA	*Mean*	$1.596\times10^{+07}$	$2.491\times10^{+04}$	$5.210\times10^{+02}$	$7.023\times10^{+02}$	$1.203\times10^{+03}$	$1.401\times10^{+03}$	**$1.916\times10^{+03}$**	$3.477\times10^{+03}$	$9.261\times10^{+03}$
	Std	$7.981\times10^{+06}$	$1.495\times10^{+04}$	7.121×10^{-02}	$3.815\times10^{+00}$	6.727×10^{-01}	4.069×10^{-01}	**$1.072\times10^{+01}$**	$8.745\times10^{+01}$	$3.041\times10^{+03}$
LR-AOA3	*Mean*	$1.349\times10^{+07}$	$4.786\times10^{+03}$	**$5.207\times10^{+02}$**	$7.010\times10^{+02}$	$1.202\times10^{+03}$	$1.400\times10^{+03}$	$1.919\times10^{+03}$	$3.768\times10^{+03}$	$1.363\times10^{+04}$
	Std	$6.785\times10^{+06}$	$2.823\times10^{+03}$	**1.416×10^{-01}**	1.112×10^{-01}	4.998×10^{-01}	9.858×10^{-02}	$3.114\times10^{+00}$	$3.874\times10^{+02}$	$7.033\times10^{+03}$
IAOA	*Mean*	**$9.789\times10^{+06}$**	**$4.267\times10^{+03}$**	$5.210\times10^{+02}$	**$7.000\times10^{+02}$**	**$1.201\times10^{+03}$**	**$1.400\times10^{+03}$**	$1.921\times10^{+03}$	**$3.228\times10^{+03}$**	**$1.264\times10^{+04}$**
	Std	**$5.902\times10^{+06}$**	**$1.976\times10^{+03}$**	8.101×10^{-02}	**2.899×10^{-02}**	**7.109×10^{-01}**	**6.618×10^{-02}**	$1.290\times10^{+01}$	**$3.046\times10^{+02}$**	**$1.207\times10^{+04}$**

5 焊接梁设计问题

为了验证IAOA算法在工程应用中的有效性和可行性,选取焊接梁设计问题[31]进行分析。本文使用文献[32]中的惩罚系数将带约束优化问题转换为无约束优化问题,并且将优化结果与粒子群算法(PSO)、引力搜索算法[33](Gravitational Search Algorithm, GSA)、生物地理学优化算法[34](Biogeography-based optimization, BBO)、差分进化算法[35](Differential Evolutionary Algorithm, DE)、蚁群优化算法[36](Ant Colony Optimization, ACO)、樽海鞘群算法[37](Salp Swarm Algorithm, SSA)、正余弦优化算法(SCA)、灰狼优化算法(GWO)、阿基米德优化算法(AOA)进行对比实验,算法内参数与表2保持一致,并设置种群规模 $N=30$,最大迭代次数 $M=1\,000$,算法独立运行30次。

本文以图4所示的焊接梁为研究对象,优化目标是使制造总成本最低。

$$\min f(X)=1.10471x_1^2x_2+0.04811x_3x_4(14+x_2) \tag{23}$$

焊接梁设计问题包含4个决策变量,即焊缝宽度 h、长度 l、横梁宽度 d、厚度 b,分别用 x_1,x_2,x_3,x_4 来表示;包含7个约束条件,即剪切应力 τ、横梁弯曲应力 σ、屈曲载荷 P_c、横梁挠度 δ 以及各设计变量之间的尺寸约束,其具体数学约束方程表示如下:

$$\begin{cases}
\text{s.t. } g_1(X)=\sqrt{(\tau')^2+2\tau'\tau''\dfrac{x_2}{2R}+(\tau'')^2}-\tau_{\max}\leqslant 0\\
g_2(X)=\dfrac{6PL}{x_3^2x_4}-\sigma_{\max}\leqslant 0\\
g_3(X)=x_1-x_4\leqslant 0\\
g_4(X)=0.10471x_1^2+0.04811x_3x_4(14+x_2)-5\leqslant 0\\
g_5(X)=0.125-x_1\leqslant 0\\
g_6(X)=\dfrac{4PL^3}{Ex_3^3x_4}-\delta_{\max}\leqslant 0\\
g_7(X)=P-\dfrac{4.013Ex_3x_4^3}{6L^2}\left(1-\dfrac{x_3}{2L}\sqrt{\dfrac{E}{4G}}\right)\leqslant 0\\
0.1\leqslant x_i\leqslant 2,i=1,4;0.1\leqslant x_i\leqslant 10,i=2,3
\end{cases} \tag{24}$$

其中,

$$\begin{cases}
\tau'=\dfrac{P}{\sqrt{2}x_1x_2}\\
\tau''=\dfrac{MR}{J}\\
M=P\left(L+\dfrac{x_2}{2}\right)\\
J=2\left\{\sqrt{2}x_1x_2\left[\dfrac{x_2^2}{12}+\left(\dfrac{x_1+x_3}{2}\right)^2\right]\right\}\\
R=\sqrt{\dfrac{x_2^2}{4}+\left(\dfrac{x_1+x_3}{2}\right)^2},P=6\,000\text{lb}\\
L=14\text{ in},E=30\times10^6\text{ psi},G=12\times10^6\text{ psi}\\
\tau_{\max}=13\,600\text{ psi},\sigma_{\max}=30\,000\text{ psi}\\
\delta_{\max}=0.25\text{ in}
\end{cases} \tag{25}$$

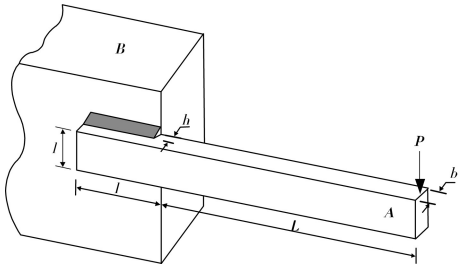


图 4　焊接梁设计问题

Fig. 4　Welded beam design problem

IAOA 与其他 9 种算法的实验结果如表 9 所列。从表中可知，IAOA 对该问题的优化效果优于其他 9 种算法。

表 9　10 种不同算法焊接梁设计问题对比

Table 9　Comparison of 10 different algorithms for solving welded beam design problems

Algorithm	*Best*	*Worst*	*Mean*	*Std*
GSA	$2.66916\times10^{+00}$	$3.83572\times10^{+00}$	$3.15094\times10^{+00}$	3.80909×10^{-01}
PSO	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	1.64702×10^{-02}
BBO	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	1.58576×10^{-01}
D×10	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	$1.08900\times10^{+14}$	1.64702×10^{-02}
ACO	$1.69163\times10^{+05}$	$1.69163\times10^{+05}$	$1.69163\times10^{+05}$	$0.00000\times10^{+00}$
SSA	$1.69560\times10^{+00}$	$2.23673\times10^{+00}$	$1.82490\times10^{+00}$	1.81073×10^{-01}
SCA	$1.79957\times10^{+00}$	$1.87293\times10^{+00}$	$1.84392\times10^{+00}$	1.90456×10^{-02}
GWO	$1.69597\times10^{+00}$	$1.70228\times10^{+00}$	$1.69850\times10^{+00}$	2.16486×10^{-03}
AOA	$1.69525\times10^{+00}$	$1.71591\times10^{+00}$	$1.69773\times10^{+00}$	6.46620×10^{-03}
IAOA	$\mathbf{1.69524\times10^{+00}}$	$\mathbf{1.69660\times10^{+00}}$	$\mathbf{1.69557\times10^{+00}}$	$\mathbf{5.52567\times10^{-04}}$

结束语　为了改善 AOA 收敛速度慢、容易陷入局部最优等缺陷，本文提出了 IAOA。在算法初期，引入佳点集法构造初始种群位置，以此保证初始种群分布均匀，为算法的迭代奠定了良好的基础。在迭代过程中，本文提出自适应反馈调节因子，有效地增强了个体的学习效率，提高了 AOA 的寻优精度。并且，本文还针对 AOA 容易陷入局部最优的问题，提出了莱维旋转变换策略，提高了个体的搜索能力，有效地避免了 AOA 在迭代后期陷入局部最优的现象。最后将本文提出的 IAOA 应用于焊接梁设计实验，实验结果表明 IAOA 取得了优于其他优化算法的优化结果，进一步验证了 IAOA 的可行性以及有效性。

将 IAOA 应用于更为复杂的机械工程参数优化问题中是下一步工作开展的热点。

参 考 文 献

[1] WANG Z M, DAI Y. A New Chaotic Genetic Hybrid Algorithm and Its Applications in Mechanical Optimization Design[J]. Defence Technology, 2010, 6(3): 220-224.

[2] MA Y, PING Y, GUO H, et al. Dynamic Economic Dispatch and Control of a Stand-alone Microgrid in DongAo Island[J]. Journal of Electrical Engineering & Technology, 2015, 10(4): 1433-1441.

[3] WILBURN B K, PERHINSCHI M G, WILBURN J N. A modified genetic algorithm for UAV trajectory tracking control laws optimization[J]. International Journal of Intelligent Unmanned Systems, 2014, 2(2): 58-90.

[4] VEKKOT S, GUPTA D, ZAKARIAH M, et al. Emotional Voice Conversion Using a Hybrid Framework With Speaker-Adaptive DNN and Particle-Swarm-Optimized Neural Network[J]. IEEE Access, 2020, 8(1): 74627-74647.

[5] WANG Y, DU T, LIU T, et al. Dynamic multiobjective squirrel search algorithm based on decomposition with evolutionary direction prediction and bidirectional memory populations[J]. IEEE Access, 2019, 7: 115997-116013.

[6] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[7] LI D, GUO W, LERCH A, et al. An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization[J]. Swarm and Evolutionary Computation, 2021, 60(7): 100789-100721.

[8] ALJARAH I, FARIS H, MIRJALILI S. Optimizing connection weights in neural networks using the whale optimization algorithm[J]. Soft Computing, 2018, 22(1): 1-15.

[9] TANYILDIZI E. A novel optimization method for solving constrained and unconstrained problems: modified golden sine algorithm[J]. Turkish Journal of Electrical Engineering & Computer Sciences, 2018, 26(6): 3287-3304.

[10] ARORA S, SINGH S. The Firefly Optimization Algorithm: Convergence Analysis and Parameter Selection[J]. International Journal of Computer Applications, 2014, 69(3): 48-52.

[11] JHAC D, LL B, YZC D. An improved multi-cores parallel artificial Bee colony optimization algorithm for parameters calibration of hydrological model-ScienceDirect[J]. Future Generation Computer Systems, 2018, 81(22): 492-504.

[12] BODHA K D, BODHA K. A Levy Flight Based Voltage Particle Swarm Optimization for Multiple-Objective Mixed Cost-Effective Emission Dispatch[C]// 2018 8th International Conference on Cloud Computing, Data Science & Engineering(Confluence). IEEE, 2018: 82-87.

[13] MA C, ZHOU D Q, ZHANG Y. BP neural network water resources demand forecasting method based on improved whale algorithm [J]. Computer Science, 2020, 47(S2): 496-500.

[14] XIAO Z Y, LIU S. Research on elite reverse golden sine whale algorithm and its engineering optimization [J]. Acta Electronica Sinica, 2019, 47(10): 2177-2186.

[15] ZHANG J, LI X G. Research on intelligent production line scheduling problem based on levy firefly algorithm [J]. Computer Science, 2021, 48(S1): 668-672.

[16] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 67-82.

[17] HASHIM F A, HUSSAIN K, HOUSSEIN E H, et al. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems[J]. Applied Intelligence, 2020, 21(1): 1-21.

[18] SUN X, WANG G, XU L, et al. Optimal estimation of the PEM fuel cells applying deep belief network optimized by improved archimedes optimization algorithm[J]. Energy, 2021, 237(1): 121532-121544.

[19] HOUSSEIN E H, HELMY B E, REZK H, et al. An enhanced Archimedes optimization algorithm based on Local escaping operator and Orthogonal learning for PEM fuel cell parameter identification[J]. Engineering Applications of Artificial Intelligence, 2021, 103(1): 104309-104321.

[20] LI Y, ZHU H, WANG D, et al. Comprehensive optimization of distributed generation considering network reconstruction based on Archimedes optimization algorithm[C]// IOP Conference Series: Earth and Environmental Science. IOP Publishing, 2021, 647(1): 012031- 012043.

[21] CHEN W W, NIE Y F, ZHANG W W, et al. A fast local mesh generation method about high-quality node set[J]. Jisuan Lixue Xuebao: Chinese Journal of Computational Mechanics, 2012, 29(5): 704-709.

[22] XIAO C, CAI Z, WANG Y. Incorporating good nodes set principle into evolution strategy for constrained optimization[C]// Third International Conference on Natural Computation (ICNC 2007). IEEE, 2007, 4: 243-247.

[23] NICKABADI A, EBADZADEH M M, SAFABAKHSH R. A novel particle swarm optimization algorithm with adaptive inertia weight[J]. Applied Soft Computing, 2011, 11(4): 3658-3670.

[24] KAMARUZAMAN A F, ZAIN A M, YUSUF S M, et al. Levy Flight Algorithm for Optimization Problems A Literature Review[J]. Applied Mechanics & Materials, 2013, 421(1): 496-501.

[25] ZHOU X J, YANG C H, GUI W H. Principle and development of state transition algorithm[J]. Acta Automatica Sinica, 2020, 46(11): 2260-2274.

[26] TARKHANEH O, ISAZADEH A, KHAMNEI H J. A new hybrid strategy for data clustering using cuckoo search based on Mantegna levy distribution, PSO and k-means[J]. International Journal of Computer Applications in Technology, 2018, 58(2): 137-149.

[27] GUPTA S, DEEP K. Random walk grey wolf optimizer for constrained engineering optimization problems[J]. Computational Intelligence, 2018, 34(4): 1025-1045.

[28] WANG J, YANG W, PEI D, et al. A novel hybrid forecasting system of wind speed based on a newly developed multi-objective sine cosine algorithm[J]. Energy Conversion and Management, 2018, 163(1): 134-150.

[29] TONG L, DONG M, AI B, et al. A Simple Butterfly Particle Swarm Optimization Algorithm with the Fitness-based Adaptive Inertia Weight and the Opposition-based Learning Average Elite Strategy[J]. Fundamenta Informaticae, 2018, 163(2): 205-223.

[30] LIN Y L. Robust estimation of parameter for fractal inverse problem[J]. Computers & Mathematics with Applications, 2010, 60(7): 2099-2108.

[31] ALMGREN A S, AGOGINO A M. A Generalization and Correction of the Welded Beam Optimal Design Problem Using Symbolic Computation[J]. Journal of Mechanical Design, 1989, 111(1): 137-140.

[32] ZHANG Z, MENG Q C, XUE R, et al. New algorithm for solving nonlinear constrained optimization problems with particle swarm optimizer[J]. Journal of Harbin Institute of Technology, 2006, 38(10): 1716-1718.

[33] HRELJA M, KLANCNIK S, BALIC J, et al. Modelling of a Turning Process Using the Gravitational Search Algorithm[J]. International Journal of Simulation Modelling, 2014, 13(1): 30-41.

[34] ERGEZER M, SIMON D. Oppositional biogeography-based optimization for combinatorial problems[C]// Evolutionary Computation. IEEE, 2011: 1496-1503.

[35] MAYER D G, KINGHORN B P, ARCHER A A. Differential evolution an easy and efficient evolutionary algorithm for model optimisation[J]. Agricultural Systems, 2005, 83(3): 315-328.

[36] DORIGO M, BIRATTARI M, STÜTZLE T. Ant Colony Optimization[J]. IEEE Computational Intelligence Magazine, 2006, 1(4): 28-39.

[37] BABAEI F, LASHKARI Z B, SAFARI A, et al. Salp swarm algorithm-based fractional-order PID controller for LFC systems in the presence of delayed EV aggregators[J]. IET Electrical Systems in Transportation, 2020, 10(3): 259-267.

CHEN Jun, born in 1996, postgraduate. His main research interests include evolutionary computation and deep learning.

HE Qing, born in 1982, Ph.D, associate professor. His main research interests include big data application and evolutionary computation.

(责任编辑:喻藜)