文章编号:1672-3961(2026)01-0049-14 DOI:10.6040/j.issn.1672-3961.0.2025.003

基于非线性自适应的改进浣熊优化算法及应用

柳宗元[1,2],李小光[1,2*],侯宇翔[1,2],丁昊[1,2]
(1.青岛大学自动化学院, 山东 青岛 266071; 2.青岛大学智能无人系统研究院, 山东 青岛 266071)

摘要:针对浣熊优化算法(coati optimization algorithm, COA)全局搜索能力不足、易陷入局部最优和收敛速度慢的问题,提出一种基于非线性自适应的改进浣熊优化算法(improved coati optimization algorithm based on nonlinear adaptation, NACOA)。采用 Logistic-Tent 映射初始化浣熊种群,提升算法初始搜索空间覆盖度,生成更加分散且高质量的初始解;引入莱维飞行策略,利用其长跳跃特性,增强算法的全局搜索能力,有效避免算法陷入局部最优;利用非线性递减惯性权重提高种群的适应性与搜索效率,平衡全局搜索和局部搜索能力,并通过黄金正弦策略提高种群收敛精度。在基准测试函数上进行对比仿真试验,结果表明 NACOA 具有更好的收敛速度和寻优精度。将 NACOA 应用到工程问题设计中,证明了该算法的有效性和实用性。

关键词:浣熊优化算法;Logistic-Tent 映射;非线性递减惯性权重;黄金正弦策略;工程应用
中图分类号:TP301.6 **文献标志码**:A
引用格式:柳宗元,李小光,侯宇翔,等. 基于非线性自适应的改进浣熊优化算法及应用[J]. 山东大学学报(工学版),2026,56(1):49-62.
LIU Zongyuan, LI Xiaoguang, HOU Yuxiang, et al. Improved coati optimization algorithm based on nonlinear adaptation and applications [J]. Journal of Shandong University (Engineering Science), 2026, 56(1):49-62.

Improved coati optimization algorithm based on nonlinear adaptation and applications

LIU Zongyuan[1,2], LI Xiaoguang[1,2*], HOU Yuxiang[1,2], DING Hao[1,2]
(1. School of automation, Qingdao University, Qingdao 266071, Shandong, China; 2. Intelligent Unmanned Systems Research Institute, Qingdao University, Qingdao 266071, Shandong, China)

Abstract: Aiming to address the problems of insufficient global search capability, easily falling into local optima, and slow convergence speed of the coati optimization algorithm (COA), an improved coati optimization algorithm based on nonlinear adaptation (NACOA) was proposed. A Logistic-Tent mapping was used to initialize the coati population, which improved the initial search space coverage of the algorithm and generated more dispersed and high-quality initial solutions. The Levy flight strategy was introduced, which made use of its long-jump characteristic to enhance the global search capability of the algorithm and effectively avoided the algorithm from falling into local optima. The nonlinearly diminishing inertia weight was used to increase the adaptability of the population and the search efficiency, balance the global and local search capabilities, and improve the population convergence accuracy through the golden sine strategy. Comparative simulation experiments were conducted on benchmark test functions, and the results showed that NACOA had better convergence speed and optimization accuracy. The NACOA was applied to the design of engineering problems, which proved the effectiveness and practicality of this algorithm.

Keywords: coati optimization algorithm; Logistic-Tent mapping; nonlinearly diminishing inertia weight; golden sine strategy; engineering application

0 引言

元启发式算法在工程领域的应用非常广泛,尤其在解决传统优化方法难以处理的非线性和多目标问题时,展现出独特的优势。研究者提出许多新型元启发式算法方案,如几何平均优化器(geometric mean optimizer, GMO)算法[1]、切诺贝

收稿日期:2025-01-09
第一作者简介:柳宗元(2000—),男,山东聊城人,硕士研究生,主要研究方向为智能优化算法及应用。E-mail:liuzongyuan@qdu.edu.cn
*通信作者简介:李小光(1962—),男,江苏南京人,教授,博士生导师,博士,主要研究方向为智能无人系统。E-mail:lixiaoguang@qdu.edu.cn

利灾难优化器（Chernobyl disaster optimizer，CDO）算法[2]、淘金优化（gold rush optimizer，GRO）算法[3]、逻辑优化算法（incomprehensible but intelligible-in-time logics algorithm，ILA）[4]、消融优化（snow ablation optimizer，SAO）[5]算法等。

浣熊优化算法（coati optimization algorithm，COA）是一种基于长鼻浣熊觅食行为和群体协作模式的仿生优化算法[6]，能够有效处理非线性、非凸优化问题，尤其在传统方法难以处理的场景中表现出色。文献[7]将COA用于神经网络关键参数（即权值和阈值）的优化，提升神经网络收敛速度；文献[8]提出一种新的基于COA的无人机通信节能路由过程技术，有效减少能源利用和通信延迟；文献[9]将COA用于解决微调比例积分微分控制器的增益问题，并将其用于所提多区域电源系统模型的负荷频率控制（load frequency control，LFC）环路中；文献[10]采用基于改进的COA求解混合发电系统模型中的容量规划问题；文献[11]使用COA寻找变分模态分解参数的最优组合，实现快速准确选择参数。

在某些复杂的多峰问题中，COA会陷入局部最优解而无法跳出，在收敛过程后期，种群分布集中到某个区域时，算法可能过于依赖局部搜索，导致全局搜索能力减弱。文献[12]通过加入遗传算法的选择、交叉及变异算子帮助跳出局部最优，引入贪婪算子提高候选解的质量；文献[13]通过学习因子策略克服COA在搜索过程中易陷入局部最优的问题，提高勘探能力；文献[14]采用差分进化突变策略增强COA全局搜索能力；为增强COA在搜索空间中个体分布的均匀性，文献[15]引入拉丁超立方采样提高采样效率，使算法在搜索时获得更全面的样本信息；文献[16]使用指数加权移动平均策略提高COA的收敛速度，有效提高算法性能。

尽管上述改进策略对COA的寻优性能有所提升，但算法在求解精度方面仍存在明显不足，尚需进一步优化和完善。本研究提出一种基于非线性自适应的改进浣熊优化算法（improved coati optimization algorithm based on nonlinear adaptation，NACOA）。采用Logistic-Tent映射初始化种群，确保种群分布的均匀性和随机性，有助于算法全局搜索；引入莱维飞行策略，改进算法的随机性，提高勘探潜在最优区域的概率，避免陷入局部最优；采用非线性递减惯性权重动态控制算法步长，提高早期探索和后期开发中的平衡能力，在算法开发阶段引入黄金正弦函数与分割比例，使算法的更新步长与方向更加灵活，优化路径更新效率，提高收敛精度。将NACOA与6种新算法在基准测试函数的优化仿真试验中进行比较，结果表明，NACOA在收敛速度和优化精度方面表现优异。将NACOA用于解决焊接梁与悬臂梁设计问题，进一步证明算法在工程问题上的优越性和实用性。

1 改进浣熊优化算法

1.1 浣熊优化算法

浣熊优化算法的灵感来源于长鼻浣熊的两种自然行为：捕猎蜥蜴和躲避捕食者。算法通过数学建模将这些自然行为转化为优化问题求解的探索和开发过程，在搜索空间中通过模拟这些自然行为更新位置。浣熊的位置代表搜索空间中的一个候选解，候选解的每个坐标对应一个决策变量。搜索空间是一个多维空间，其维度由决策变量数 m 决定。第 i 个浣熊的初始位置向量为 $\boldsymbol{X}_i=(x_{i,1}\quad x_{i,2}\cdots\quad x_{i,m})$，其在第 $j(j=1,2,\cdots,m)$ 个决策变量上的位置

$$x_{i,j}=l_j+r(u_j-l_j)\,, \tag{1}$$

式中：l_j 为第 j 个决策变量的下界；u_j 为第 j 个决策变量的上界；r 为随机数，$r\in[0,1]$。

在勘探阶段（合作狩猎蜥蜴），一半的浣熊爬上树接近蜥蜴进行狩猎，此时第 i 个浣熊的位置向量为 $\boldsymbol{X}_i^{\mathrm{P1}}=(x_{i,1}^{\mathrm{P1}}\quad x_{i,2}^{\mathrm{P1}}\quad\cdots\quad x_{i,m}^{\mathrm{P1}})$，其在第 j 个决策变量上的位置

$$x_{i,j}^{\mathrm{P1}}=x_{i,j}+r(I_{\mathrm{guana},j}-Ix_{i,j})\,, \tag{2}$$

式中：$I_{\mathrm{guana},j}$ 为蜥蜴位置，表示当前最优解；I 为随机整数，$I\in\{1,2\}$。蜥蜴落地后，被放置在搜索空间的随机位置。在地面上的另一半浣熊根据蜥蜴落地位置进行移动。蜥蜴在第 j 个决策变量上的随机落地位置

$$I_{\mathrm{guana},j}^{\mathrm{G}}=l_j+r(u_j-l_j)\,。 \tag{3}$$

地面上的浣熊在第 j 个决策变量上位置更新为

$$x_{i,j}^{\mathrm{P1}}=\begin{cases}x_{i,j}+r(I_{\mathrm{guana},j}^{\mathrm{G}}-Ix_{i,j})\,, & F_{\mathrm{new}}<F_i\\ x_{i,j}+r(x_{i,j}-I_{\mathrm{guana},j}^{\mathrm{G}})\,, & F_{\mathrm{new}}\geqslant F_i\end{cases}, \tag{4}$$

式中，F_{new} 为蜥蜴在新位置的目标函数，F_i 为第 i 个浣熊当前位置的目标函数。如果蜥蜴的新位置具有更优解，则浣熊向该位置靠近，否则维持当前位置不变。

在开发阶段（分散逃离捕食者），当天敌攻击浣熊时，浣熊会趋于安全地点进行逃离活动，此时第 i 个浣熊的位置向量为 $\boldsymbol{X}_i^{\mathrm{P2}}=(x_{i,1}^{\mathrm{P2}}\quad x_{i,2}^{\mathrm{P2}}\quad\cdots\quad x_{i,m}^{\mathrm{P2}})$，其在第 j 个决策变量上的位置

$$x_{i,j}^{P2}=x_{i,j}+(1-2r)\left[l_j^{local}+r(u_j^{local}-l_j^{local})\right], \quad (5)$$

式中:l_j^{local} 为第 j 个决策变量在当前局部搜索范围内的下界,$l_j^{local}=\frac{l_j}{g}$,其中 g 为当前迭代次数;u_j^{local} 为第 j 个决策变量在当前局部搜索范围内的上界,$u_j^{local}=\frac{u_j}{g}$。

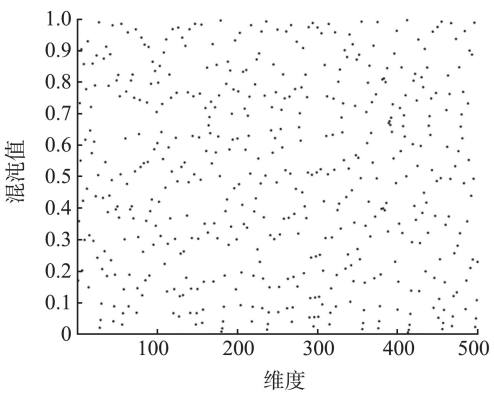
1.2　混沌映射种群初始化

种群初始化是算法运行的起点,直接影响算法的全局搜索能力、收敛速度及最终优化结果。COA 采用随机方式初始化种群位置,导致种群在搜索空间中分布不均匀,可能集中在搜索空间的某些区域,从而使算法在该区域局部收敛,影响算法的全局搜索能力。

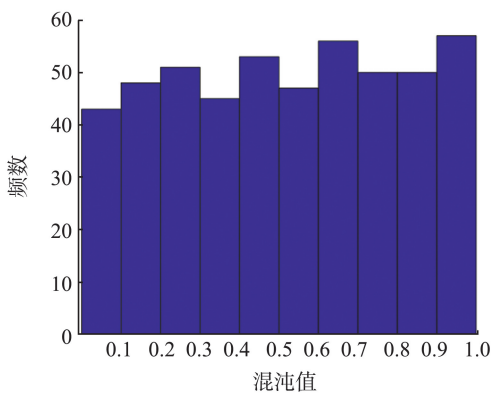
混沌映射在初始化阶段能够生成更加均匀和多样化的种群[17]。这种多样性对避免算法陷入局部最优解和提高全局搜索能力非常重要。Logistic-Tent 映射是一个结合 Logistic 映射[18]和 Tent 映射[19]的混合动态系统模型,通过动态演化提供非线性行为,有助于种群在广泛搜索空间内进行探索,避免算法过早收敛到某个解。Logistic-Tent 映射的计算式为

$$x_{i+1}=\begin{cases}\left[px_i(1-x_i)+\dfrac{(4-p)x_i}{2}\right]\bmod 1, & x_i<0.5\\ \left[px_i(1-x_i)+\dfrac{(4-p)(1-x_i)}{2}\right]\bmod 1, & x_i\geqslant 0.5\end{cases}, \quad (6)$$

式中:x_i 为第 i 次迭代值;x_{i+1} 为第 $i+1$ 次迭代值;mod 为取模运算;p 为映射参数,本研究设定 $p=0.3$。Logistic-Tent 映射初始种群分布图与分布直方图如图 1 所示,解的维度设定为 500。图 1(a)显示出种群的二维分布,其中每个点代表解空间中的每个个体,点的分布体现出 Logistic-Tent 映射初始化分布的均匀性。图 1(b)表明种群覆盖解空间的不同区间,且频率分布接近均匀,保证初始化的随机性和覆盖性。这种分布特性有助于算法在解空间中进行更全面的探索,有效避免过早收敛现象。



(a) Logistic-Tent分布图



(b) Logistic-Tent分布直方图

图 1　Logistic-Tent 映射种群分布

Fig.1　Logistic-Tent mapping of population distributions

1.3　引入莱维飞行策略更新蜥蜴位置

在求解复杂的多峰问题时,COA 算法由于蜥蜴落地位置策略设计不足,种群在局部区域容易过度集中,产生对局部搜索的过度依赖,限制全局搜索能力。为解决这一问题,本研究采用莱维飞行策略[20]对蜥蜴落地位置进行更新,可以显著增强算法的全局探索能力。该策略的核心特性在于更新步长遵循一种具有长尾特性的概率分布。这种分布使位置更新不同于高斯分布的常规随机游走,能够有效避免局部最优解问题,尤其在处理高维、复杂和多峰函数优化时具有明显的优越性。利用莱维飞行策略更新蜥蜴第 j 个决策变量上的落地位置

$$I_{guana,j}^{G}=l_j+s(u_j-l_j), \quad (7)$$

式中,s 为莱维飞行的步长,$s=\frac{u}{|v|^{1/\beta}}$,其中 u 为服从正态分布的随机数,$u\sim N(0,\sigma^2)$,$\sigma=\left[\frac{\Gamma(1+\beta)\cdot\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\cdot\beta\cdot 2^{\frac{\beta-1}{2}}}\right]^{\frac{1}{\beta}}$,$\Gamma(\cdot)$为伽马函数,$\beta$ 为控制分布的尾部特性,$\beta\in(0,2]$,v 为服从标准正态分布的随机数,$v\sim N(0,1)$。

二维莱维飞行轨迹如图 2 所示。个体在搜索空间中并非以规则或直线方式移动,而是以不确定的方向和距离跳跃。路径中既有短距离的连续跳跃,也有少数长距离的跨区域跳跃。这种方式既可以

进行局部搜索，又可以突破局部区域的限制，跳至未探索的区域。本研究通过引入莱维飞行长尾特性增强蜥蜴着陆过程的随机性，增大蜥蜴跃迁至潜在全局最优解区域的概率，确保算法在搜索过程中的多样性。

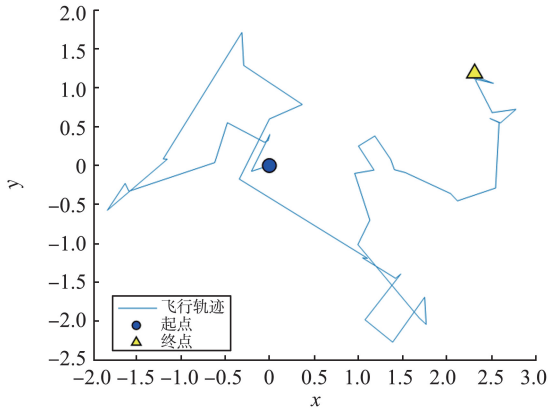


图 2 二维莱维飞行轨迹

Fig.2 Two-dimensional Levy flight trajectory

1.4 非线性递减惯性权重与黄金正弦策略

浣熊在树下捕杀蜥蜴过程中的步长由 r 和位置差决定，没有考虑算法迭代过程中的需求差异。在迭代后期，步长仍然过大，导致算法在最优解附近振荡，收敛速度变慢。通过引入非线性递减惯性权重 w 优化收敛速度，减少不必要的迭代次数，提高种群在早期探索和后期开发中的平衡能力。w 的计算式为

$$w=\begin{cases}\dfrac{G_{\max}}{G_{\max}+\mathrm{e}^{\frac{g}{0.1G_{\max}}}-1}, & g<\partial\\ \alpha\cdot\mathrm{e}^{-0.005\left(g-\frac{G_{\max}}{2}\right)}, & g\geqslant\partial\end{cases}, \tag{8}$$

式中，∂ 为分界参数，α 为调整系数，$G_{\max}$ 为最大迭代次数。w 随 g 改变而改变。当 $G_{\max}=500$ 时，w 的变化曲线如图 3 所示。

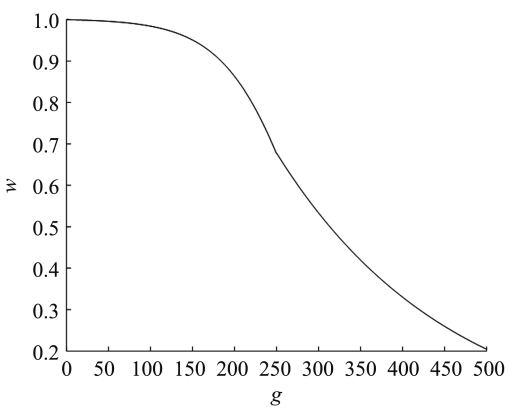


图 3 非线性递减惯性权重变化曲线

Fig.3 Nonlinear decline inertia weight change curve

由图 3 可以看出：在迭代初期，w 较大，且变化速度较慢，充分保持种群的全局寻优搜索能力；在迭代后期，w 较小，且变化速度较快，有效避免后期步长仍然过大，无法根据算法进程逐步缩小搜索范围的问题，确保种群能够聚焦于最优解附近区域进行精细化开发，显著提高种群在不同迭代阶段的适应性与搜索效率。改进后，浣熊捕食猎物时，在第 j 个决策变量上位置更新公式为

$$x_{i,j}^{\mathrm{P1}}=\begin{cases}x_{i,j}+wr\left(I_{\mathrm{guana},j}^{\mathrm{G}}-Ix_{i,j}\right), & F_{\mathrm{new}}<F_i\\ x_{i,j}+wr\left(x_{i,j}-I_{\mathrm{guana},j}^{\mathrm{G}}\right), & F_{\mathrm{new}}\geqslant F_i\end{cases}。 \tag{9}$$

浣熊躲避天敌的过程是一个局部搜索阶段。使用黄金正弦函数[21]的波动性可以实现对解空间的灵活探索，黄金分割的比例控制能保证精细搜索。通过黄金分割策略调整搜索步长和方向，使浣熊在逃离天敌时的动作更加灵活，增强搜索的多样性。黄金正弦策略优化了逃跑路径效率，可以有效缩短收敛到最优解的时间。改进后，浣熊逃离天敌时，在第 j 个决策变量上的位置更新公式为

$$x_{i,j}^{\mathrm{P2}}=x_{i,j}+\frac{\sin(2\pi r_1)\cdot\left[l_j^{\mathrm{local}}+r_2\left(u_j^{\mathrm{local}}-l_j^{\mathrm{local}}\right)\right]}{\theta}, \tag{10}$$

式中：θ 为分割率，本研究 $\theta=\dfrac{(\sqrt{5}-1)}{2}$；$r_1$、$r_2$ 为随机数，$r_1,r_2\in[0,1]$。

1.5 NACOA 流程

NACOA 流程图如图 4 所示。算法具体步骤如下。

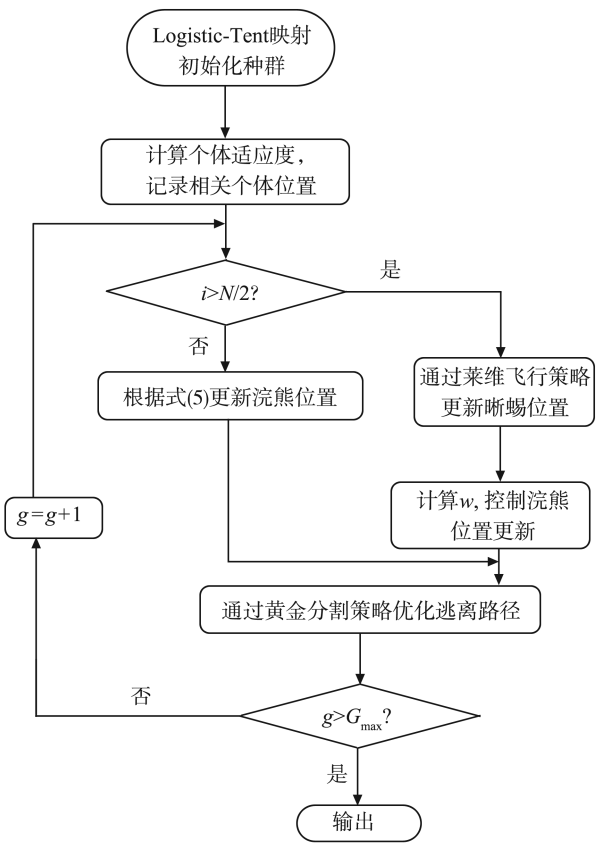


图 4 NACOA 流程图

Fig.4 Flow chart of NACOA

(1)对最大迭代次数、种群规模、上下界、维度等参数进行初始化。

(2)利用 Logistic-Tent 映射初始化种群。

(3)计算种群所有个体适应度,记录最优、次优和最差适应度个体位置及适应度。

(4)根据式(7)更新蜥蜴位置,计算 w,控制浣熊在捕杀蜥蜴过程中的位置更新。

(5)根据式(10)调整浣熊躲避天敌过程中的搜索步长和方向,优化逃跑路径。

(6)循环步骤(3)~(5),判断是否满足迭代条件。若满足,则跳出循环。

(7)算法结束,返回最佳方案。

1.6　NACOA 时间复杂度分析

设种群规模为 N,最大迭代次数为 $G_{\max}$,维度为 m。NACOA 在初始化阶段通过 Logistic-Tent 映射生成更加均匀和多样化的种群,时间为 t_1,此阶段算法时间复杂度为 $O_1(N\times m\times t_1)$。在主循环部分,按式(7)引入莱维飞行更新蜥蜴位置,时间为 t_2,时间复杂度为 $O_2(N\times m\times G_{\max}\times t_2)$;通过式(8)引入 w,求解时间为 t_3,时间复杂度为 $O_3(N\times G_{\max}\times t_3)$;采用式(10)黄金正弦策略更新浣熊逃离天敌的位置,时间为 t_4,时间复杂度为 $O_4(N\times m\times G_{\max}\times t_4)$。因此,主循环部分 NACOA 的时间复杂度为 $O_2+O_3+O_4=O_5(N\times G_{\max}\times(t_2\times m+t_3+t_4\times m))=O_5(N\times G_{\max}\times m)$。

综上,NACOA 总体时间复杂度为 $O_1+O_5=O(N\times m\times t_1+N\times G_{\max}\times m)=O(N\times G_{\max}\times m)$,与原算法 COA 的时间复杂度相同,改进策略并没有增加额外的时间复杂度。

2　仿真试验与结果分析

为验证改进算法的性能,本研究将 NACOA 与改进浣熊优化算法(improved coati optimization algorithm, ICOA)[22]、GMO[1]、CDO[2]、GRO[3]、ILA[4]、SAO[5] 6 种算法进行测试比较。为确保对比的公平性与结果的可信度,6 种算法参数设置均参照原文献,NACOA 设置 $r=0.3$,$\beta=1.66$,$\alpha=0.678$。算法的种群数设置为 50,最大迭代次数均为 500 次。各算法在基准测试函数上独立测试 30 次。选取最优解、平均解、标准差 3 种指标对算法性能进行对比分析。最优解用于评估算法的收敛能力;平均解反映算法的稳定性和鲁棒性,即算法在不同运行中是否能够一致地找到较好的解;标准差衡量算法在多次运行中找到的解的离散程度,反映算法的可靠性。通过 Wilcoxon 秩和检验对 7 种算法进行显著性分析,验证不同算法在性能上的差异是否具有显著性。通过高维函数测试进行对比,验证算法在处理复杂数据结构时的有效性与鲁棒性。

本研究选取 14 个基准测试函数对算法进行测试,如表 1 所示。单峰函数 f_1~f_7 有助于评估优化算法的收敛速度,即算法从初始解到最优解所需的时间或迭代次数;多峰函数 f_8~f_{11} 考察优化算法在面对复杂搜索空间时的全局搜索能力,有助于评估测试优化算法在高维复杂问题中的表现,尤其是在高维下是否能够有效地搜索和收敛;复合函数 f_{12}~f_{14} 结合单峰和多峰函数的特性,其搜索空间通常包含不同形状、规模和分布的局部最优解,优化难度增加。通过基准测试函数进行测试,智能算法的优势和局限性可以得到全面评估。

2.1　改进策略有效性分析

为证明本研究改进策略的有效性,将 NACOA、COA[6]、仅采用 Logistic-Tent 映射策略的 LTCOA、仅采用莱维飞行策略的 LCOA、仅采用非线性递减惯性权重的 NCOA、仅采用黄金正弦策略的 GCOA 在 f_2、f_3、f_6、f_9、f_{11} 基准函数上进行 30 次测试,维度设置为 30,测试结果如表 2 所示,其中最优结果加粗表示。

表 1　基准测试函数

Table 1　Benchmark function

类型	函数公式	维度	取值范围	最优解
单峰	$f_1=\sum_{i=1}^{n}x_i^2$	30	[-100,100]	0
	$f_2=\sum_{i=1}^{n}x_i+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]	0
	$f_3=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	[-100,100]	0
	$f_4=\max_i\{\lvert x_i\rvert,\ 1\leqslant i\leqslant n\}$	30	[-100,100]	0
	$f_5=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[-30,30]	0
	$f_6=\sum_{i=1}^{n}(\lvert x_i+0.5\rvert)^2$	30	[-100,100]	0
	$f_7=\sum_{i=1}^{n}ix_i^4+\mathrm{rand}[0,1]$	30	[-1.28,1.28]	0

表 1(续)

类型	函数公式	维度	取值范围	最优解
多峰	$f_8=\sum_{i=1}^{n}-x_i\cdot\sin(\sqrt{\lvert x_i\rvert})$	30	[-500,500]	-12 569.5
	$f_9=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}(\cos 2\pi x_i)\right)+20+\mathrm{e}$	30	[-32,32]	0
	$f_{10}=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	30	[-5.12,5.12]	0
	$f_{11}=\frac{1}{4\ 000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0
复合	$f_{12}=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	[-65.536,65.536]	1
	$f_{13}=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5,5]	0.000 307 5
	$f_{14}=-\sum_{i=1}^{7}[(\boldsymbol{x}-a_i)(\boldsymbol{x}-a_i)^{\mathrm{T}}+c_i]^{-1}$	4	[0,10]	-10

表 2 6 种算法测试结果对比

Table 2 Comparison of test results of six algorithms

算法	f_2				f_3			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
COA	2.55×10^{-115}	5.62×10^{-114}	4.56×10^{-114}	6	1.11×10^{-114}	4.41×10^{-114}	3.18×10^{-114}	5
LTCOA	2.08×10^{-124}	3.18×10^{-123}	1.33×10^{-123}	4	6.62×10^{-147}	5.28×10^{-146}	3.65×10^{-146}	3
LCOA	1.24×10^{-117}	5.33×10^{-117}	1.69×10^{-117}	5	3.52×10^{-115}	4.55×10^{-114}	7.34×10^{-114}	6
NCOA	9.16×10^{-142}	5.90×10^{-141}	2.65×10^{-141}	2	1.21×10^{-143}	4.53×10^{-143}	2.44×10^{-143}	4
GCOA	1.11×10^{-134}	5.30×10^{-134}	8.12×10^{-134}	3	2.66×10^{-146}	4.34×10^{-146}	7.32×10^{-146}	2
NACOA	**1.60×10^{-204}**	**5.73×10^{-202}**	**7.30×10^{-201}**	**1**	**0**	**0**	**0**	**1**

算法	f_6				f_9			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
COA	3.82×10^{-5}	4.41×10^{-5}	4.26×10^{-5}	5	1.12×10^{-13}	4.00×10^{-13}	1.43×10^{-13}	6
LTCOA	4.67×10^{-9}	7.11×10^{-9}	2.38×10^{-9}	3	2.19×10^{-15}	2.79×10^{-15}	2.66×10^{-15}	3
LCOA	7.60×10^{-6}	4.50×10^{-5}	9.32×10^{-5}	6	5.97×10^{-14}	8.93×10^{-14}	7.39×10^{-14}	5
NCOA	1.27×10^{-7}	3.22×10^{-6}	4.36×10^{-6}	4	4.99×10^{-15}	6.68×10^{-15}	4.79×10^{-15}	4
GCOA	7.33×10^{-13}	5.62×10^{-12}	3.43×10^{-12}	2	**3.79×10^{-16}**	8.44×10^{-16}	6.99×10^{-16}	2
NACOA	**0**	**0**	**0**	**1**	4.44×10^{-16}	**4.44×10^{-16}**	**0**	**1**

算法	f_{11}			
	最优解	平均解	标准差	Friedman 平均排名
COA	**0**	**0**	**0**	**1**
LTCOA	**0**	**0**	**0**	**1**
LCOA	**0**	**0**	**0**	**1**
NCOA	**0**	**0**	**0**	**1**
GCOA	**0**	**0**	**0**	**1**
NACOA	**0**	**0**	**0**	**1**

由表 2 可知,NACOA 在测试函数 f_2、f_3、f_6、f_{11} 上均能寻找到理论最优解,性能优于其他算法。在 f_2 中,NACOA 的最优解为 1.60×10^{-204},远低于排名第 2 的 NCOA 的最优解 9.16×10^{-142},表明 NACOA 在寻优精度上具有显著优势,且非线性递减惯性权重的平衡作用能够有效提升寻优效果。在 f_9 中,NACOA 与 GCOA 的求解结果数量级相同,表明黄金正弦策略在提高收敛精度方面具有显著优势,但 NACOA 的标准差为 0,与 GCOA 相比具有极高的稳定性。LTCOA 在 f_3 中取得较好结果,说明

Logistic-Tent 映射通过优化初始种群分布,显著提升初始解质量,有效增强算法的全局寻优能力与收敛精度。单一策略改进算法在不同测试函数上的表现存在较大差异。NCOA 在 f_2 中的最优解为 9.16×10^{-142},显著优于 LTCOA 与 GCOA,但在 f_6 与 f_9 中的求解结果不如 LTCOA 和 GCOA。这种差异性反映单一策略在不同问题场景下的适应性和局限性。单一策略在某种情况下能够提升算法性能,但无法在所有测试函数上实现稳定提升。所以,多策略融合作为一种更为全面的优化方法,能够提升算法整体性能,通过协同作用弥补单一策略的不足,从而实现更优的寻优精度、稳定性和鲁棒性。

2.2　求解精度的对比分析

在基准测试函数上,7 种算法的测试结果如表 3 所示,其中最优结果加粗表示。由表 3 可知,在单峰函数 $f_1\sim f_7$ 测试中,NACOA 展示出显著的寻优效果。在 f_1、f_3、f_5、f_6 中,NACOA 均能找到各函数的理论最优,展现出 NACOA 高效细化的搜索能力。在 f_2 中,NACOA 在 3 个指标上表现出绝对最优,ICOA 和 ILA 的最优解相对于 NACOA 具有一定差距,其余算法的性能显著低于 NACOA。在 f_4 中,NACOA 表现最优,ICOA 的寻优结果排名第 3,两者在标准差数量级上具有明显差异,NACOA 表现更稳定。在 f_2 和 f_4 中,尽管 NACOA 未能探索到函数的理论最优解,但其寻优结果在数量级上具有明显优势,结果精度明显高于其他 6 种算法,且标准差最小,展现出算法良好的精准性和稳定性。

在多峰函数 $f_8\sim f_{11}$ 测试中,NACOA 同样展示出显著的寻优效果。在 f_{10}、f_{11} 中,NACOA 均能找到各函数的理论最优,展现出良好的跳出局部最优的能力。在 f_8 中,NACOA 的结果最佳且最稳定,与 ICOA 相比,在求解精度和稳定性上均有一定优势;SAO 的标准差较高,解的波动较大。在 f_9 中,NACOA 和 ICOA、ILA 在高精度优化问题中具有明显优势,3 种算法的标准差均为 0,测试结果并列第 1。

在复合函数 $f_{12}\sim f_{14}$ 测试中,NACOA 能够稳定地找到近似最优解,并且在多次运行中保持一致的优化效果,表现出极高的收敛精度和稳定性。与其他算法相比,NACOA 的表现明显更好,标准差较低,表明结果的波动性较小,具有较好的稳定性,进一步证明 NACOA 具有较强的全局搜索能力和稳定性。

综上,NACOA 算法在函数收敛性、稳定性和适用性方面均展现出明显优势,是 7 种算法中最优选择,表明 NACOA 改进的搜索策略更好地避免陷入局部最优解,能够在复杂问题中探索更广泛的解空间,从而找到更优解。

表 3　7 种算法测试结果对比

Table 3　Comparison of test results of 7 algorithms

算法	f_1				f_2			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	**0**	**0**	**0**	**1**	1.45×10^{-166}	2.97×10^{-162}	9.16×10^{-166}	3
GMO	1.81×10^{-167}	2.83×10^{-132}	2.06×10^{-132}	2	7.08×10^{-84}	3.68×10^{-81}	1.33×10^{-84}	4
CDO	1.05×10^{-136}	3.44×10^{-116}	1.74×10^{-115}	4	1.75×10^{-72}	1.69×10^{-70}	1.69×10^{-70}	6
GRO	7.50×10^{-127}	5.97×10^{-121}	3.27×10^{-121}	3	9.16×10^{-77}	5.90×10^{-75}	2.65×10^{-74}	5
ILA	**0**	**0**	**0**	**1**	1.21×10^{-165}	1.30×10^{-164}	8.12×10^{-164}	2
SAO	1.80×10^{-7}	1.72×10^{-6}	1.32×10^{-6}	5	1.73×10^{-8}	1.68×10^{-5}	9.02×10^{-4}	7
NACOA	**0**	**0**	**0**	**1**	$\mathbf{1.60\times10^{-203}}$	$\mathbf{2.53\times10^{-202}}$	$\mathbf{1.30\times10^{-202}}$	**1**

算法	f_3				f_4			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	4.45×10^{-274}	7.35×10^{-261}	1.43×10^{-219}	2	2.95×10^{-161}	2.63×10^{-154}	5.94×10^{-154}	3
GMO	4.62×10^{-106}	8.28×10^{-104}	3.65×10^{-103}	3	9.88×10^{-89}	1.00×10^{-88}	9.77×10^{-89}	4
CDO	3.52×10^{-105}	3.59×10^{-99}	9.34×10^{-99}	4	3.07×10^{-66}	1.85×10^{-65}	3.43×10^{-65}	5
GRO	1.61×10^{-47}	4.72×10^{-38}	1.24×10^{-39}	5	1.32×10^{-41}	1.47×10^{-36}	3.81×10^{-36}	6
ILA	**0**	9.32×10^{-33}	4.32×10^{-21}	6	3.97×10^{-166}	1.13×10^{-163}	2.66×10^{-163}	2
SAO	1.07×10^{2}	2.57×10^{2}	1.37×10^{2}	7	3.07×10^{-1}	1.30×10^{1}	3.52×10^{1}	7
NACOA	**0**	**0**	**0**	**1**	$\mathbf{5.66\times10^{-217}}$	$\mathbf{8.76\times10^{-213}}$	$\mathbf{2.34\times10^{-212}}$	**1**

表 3(续)

算法	f_5				f_6			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	3.06×10^{-1}	5.16×10^{-1}	4.46×10^{-1}	2	2.11×10^{-7}	2.97×10^{-5}	9.76×10^{-5}	7
GMO	2.60×10^{1}	2.65×10^{1}	1.00×10^{0}	3	7.08×10^{-9}	8.91×10^{-9}	2.88×10^{-9}	2
CDO	2.70×10^{1}	2.77×10^{1}	2.25×10^{-1}	6	7.50×10^{0}	7.50×10^{0}	**0**	6
GRO	2.60×10^{1}	2.74×10^{1}	7.45×10^{-1}	4	1.80×10^{-1}	2.80×10^{-1}	9.69×10^{-2}	5
ILA	2.68×10^{1}	2.76×10^{1}	4.24×10^{-1}	5	1.90×10^{-1}	2.30×10^{-1}	1.22×10^{-1}	4
SAO	2.62×10^{1}	4.27×10^{1}	2.95×10^{1}	7	1.12×10^{-6}	1.52×10^{-6}	8.70×10^{-7}	3
]NACOA	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**

算法	f_7				f_8			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	4.85×10^{-5}	5.89×10^{-5}	2.47×10^{-5}	4	-5.60×10^{3}	-3.49×10^{3}	5.86×10^{3}	7
GMO	2.11×10^{-3}	2.72×10^{-3}	6.04×10^{-4}	5	-7.38×10^{3}	-6.05×10^{3}	3.19×10^{3}	5
CDO	3.82×10^{-5}	5.10×10^{-5}	1.52×10^{-5}	3	-3.99×10^{3}	-3.94×10^{3}	3.25×10^{2}	6
GRO	2.00×10^{-3}	6.81×10^{-3}	3.40×10^{-4}	6	-8.64×10^{3}	-8.10×10^{3}	6.02×10^{2}	3
ILA	4.05×10^{-5}	2.85×10^{-5}	$\mathbf{1.77\times10^{-5}}$	2	-7.93×10^{3}	-7.58×10^{3}	2.84×10^{2}	4
SAO	1.60×10^{-2}	2.40×10^{-2}	1.00×10^{-2}	7	-9.55×10^{3}	-9.13×10^{3}	6.36×10^{2}	2
NACOA	$\mathbf{5.91\times10^{-6}}$	$\mathbf{2.23\times10^{-5}}$	2.21×10^{-5}	**1**	$\mathbf{-1.31\times10^{4}}$	$\mathbf{-2.18\times10^{4}}$	$\mathbf{2.15\times10^{2}}$	**1**

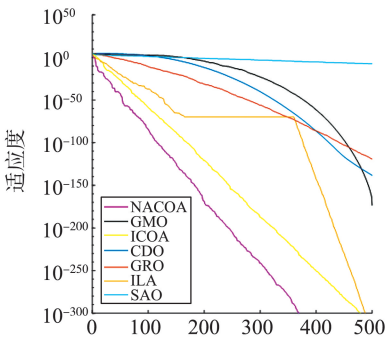
算法	f_9				f_{10}			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	$\mathbf{4.44\times10^{-16}}$	$\mathbf{4.44\times10^{-16}}$	**0**	**1**	**0**	**0**	**0**	**1**
GMO	3.99×10^{-15}	3.99×10^{-15}	**0**	2	**0**	1.21×10^{1}	9.91×10^{0}	2
CDO	3.99×10^{-15}	3.99×10^{-15}	**0**	2	2.03×10^{0}	5.34×10^{1}	2.14×10^{1}	4
GRO	3.99×10^{-15}	3.99×10^{-15}	**0**	2	**0**	**0**	**0**	**1**
ILA	$\mathbf{4.44\times10^{-16}}$	$\mathbf{4.44\times10^{-16}}$	**0**	**1**	**0**	**0**	**0**	**1**
SAO	6.15×10^{-5}	8.41×10^{-5}	4.29×10^{-5}	3	1.55×10^{1}	2.68×10^{1}	6.85×10^{0}	3
NACOA	$\mathbf{4.44\times10^{-16}}$	$\mathbf{4.44\times10^{-16}}$	**0**	**1**	**0**	**0**	**0**	**1**

算法	f_{11}				f_{12}			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	**0**	**0**	**0**	**1**	9.98×10^{-1}	1.20×10^{0}	4.50×10^{-1}	3
GMO	**0**	**0**	**0**	**1**	4.98×10^{0}	5.47×10^{0}	2.18×10^{0}	6
CDO	**0**	**0**	**0**	**1**	1.27×10^{1}	1.38×10^{1}	1.49×10^{0}	4
GRO	**0**	**0**	**0**	**1**	$\mathbf{9.98\times10^{-1}}$	$\mathbf{9.98\times10^{-1}}$	3.33×10^{-16}	2
ILA	**0**	**0**	**0**	**1**	$\mathbf{9.98\times10^{-1}}$	$\mathbf{9.98\times10^{-1}}$	**0**	**1**
SAO	1.52×10^{-2}	1.72×10^{-2}	1.48×10^{-2}	2	9.98×10^{-1}	1.48×10^{0}	7.64×10^{-1}	5
NACOA	**0**	**0**	**0**	**1**	$\mathbf{9.98\times10^{-1}}$	$\mathbf{9.98\times10^{-1}}$	**0**	**1**

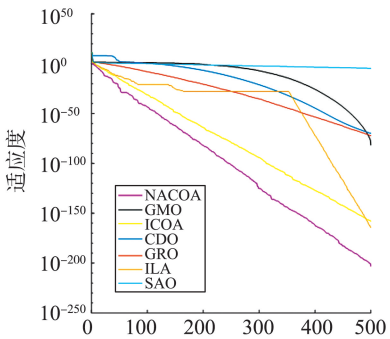
算法	f_{13}				f_{14}			
	最优解	平均解	标准差	Friedman 平均排名	最优解	平均解	标准差	Friedman 平均排名
ICOA	3.08×10^{-4}	3.52×10^{-4}	4.83×10^{-5}	7	-1.03×10^{1}	-6.44×10^{0}	2.29×10^{0}	4
GMO	3.12×10^{-4}	3.47×10^{-4}	4.43×10^{-5}	6	-1.04×10^{1}	-7.69×10^{0}	3.65×10^{0}	3
CDO	3.08×10^{-4}	3.28×10^{-4}	1.23×10^{-5}	5	-9.65×10^{0}	-7.79×10^{0}	1.10×10^{0}	2
GRO	$\mathbf{3.07\times10^{-4}}$	3.09×10^{-4}	1.90×10^{-6}	2	$\mathbf{-1.04\times10^{1}}$	$\mathbf{-1.04\times10^{1}}$	**0**	**1**
ILA	3.12×10^{-4}	3.16×10^{-4}	4.44×10^{-6}	4	$\mathbf{-1.04\times10^{1}}$	$\mathbf{-1.04\times10^{1}}$	**0**	**1**
SAO	$\mathbf{3.07\times10^{-4}}$	3.10×10^{-4}	5.70×10^{-6}	3	-5.09×10^{0}	-4.78×10^{0}	5.70×10^{-1}	5
NACOA	$\mathbf{3.07\times10^{-4}}$	$\mathbf{3.07\times10^{-4}}$	$\mathbf{3.33\times10^{-16}}$	**1**	$\mathbf{-1.04\times10^{1}}$	$\mathbf{-1.04\times10^{1}}$	**0**	**1**

2.3　收敛过程对比分析

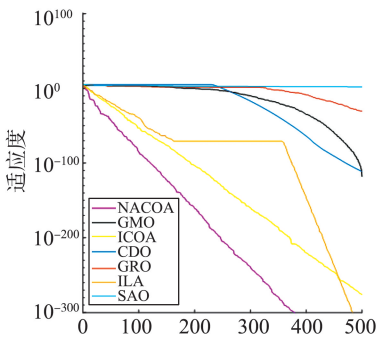
NACOA、GMO、ICOA、CDO、GRO、ILA、SAO 在基准测试函数 f_1~f_{14} 上的收敛曲线如图 5 所示。由图 5(a)~(d)可以看出:在单峰函数 f_1~f_4 收敛曲线中,NACOA 无论是在收敛精度还是迭代次数都明显优于其他 6 种函数;与 ICOA 相比,NACOA 利用 Logistic-Tent 映射进行种群初始化,增强初始种群在搜索空间中的全面性,有效探索搜索空间,减少迭代次数;ILA 在收敛过程中出现明显的陷入局部最优问题;其余算法收敛速度慢,且部分算法出现早熟情况。由图 5(e)、(f)可以看出:NACOA 在 f_5、f_6 收敛曲线中的收敛精度和收敛效率远超于 ICOA,在迭代 100 次内便可探寻到最优解,且寻优结果精度更高,主要得益于莱维飞行有助于 NACOA 在陷入局部最优时跳出,继续在搜索空间中寻找更好的解,同时非线性递减惯性权重有效解决 COA 在最优解附近振荡的问题;ICOA 在迭代后期明显出现振荡停滞现象。由图 5(g)可以看出,在 f_7 收敛曲线中,NACOA 虽在迭代初期的收敛性能不及 ILA,但在后期的收敛情况明显优于 ILA。由图 5(h)可以看出,在多峰函数 f_8 中,NACOA 同样取得最好的收敛效果,与其他算法相比,NACOA 通过黄金分割比的特性,能够以较少迭代次数缩小搜索区间,快速定位到极值点。由图 5(i)~(n)可以看出:在 f_9~f_{12}、f_{14} 收敛曲线中,NACOA 的收敛效率具有明显优势,仅在迭代初期便寻到最优解;在复合函数 f_{13} 中,NACOA 的测试结果与 ILA 相近,收敛结果与迭代次数远好于 ICOA。



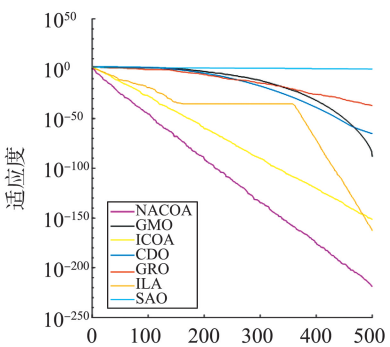
(a) f_1 收敛曲线



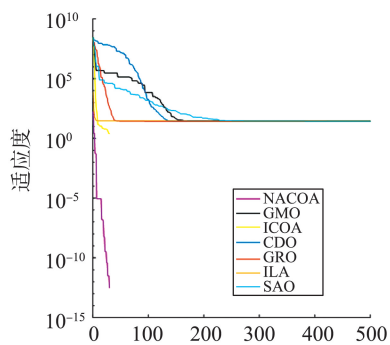
(b) f_2 收敛曲线



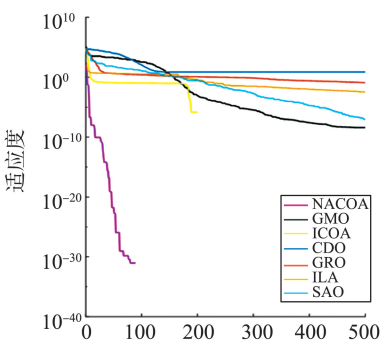
(c) f_3 收敛曲线



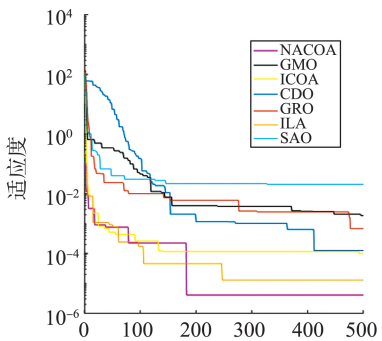
(d) f_4 收敛曲线



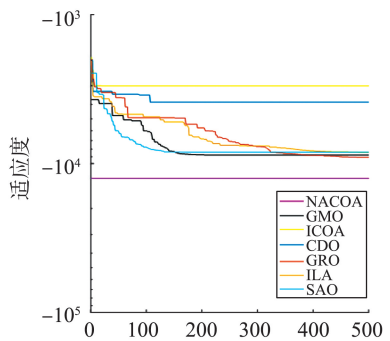
(e) f_5 收敛曲线



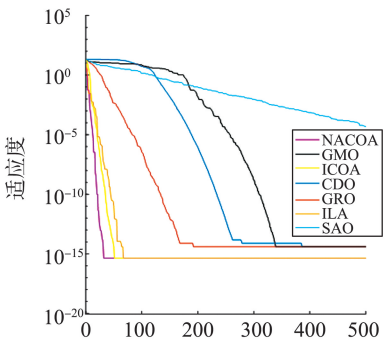
(f) f_6 收敛曲线



(g) f_7 收敛曲线



(h) f_8 收敛曲线



(i) f_9 收敛曲线

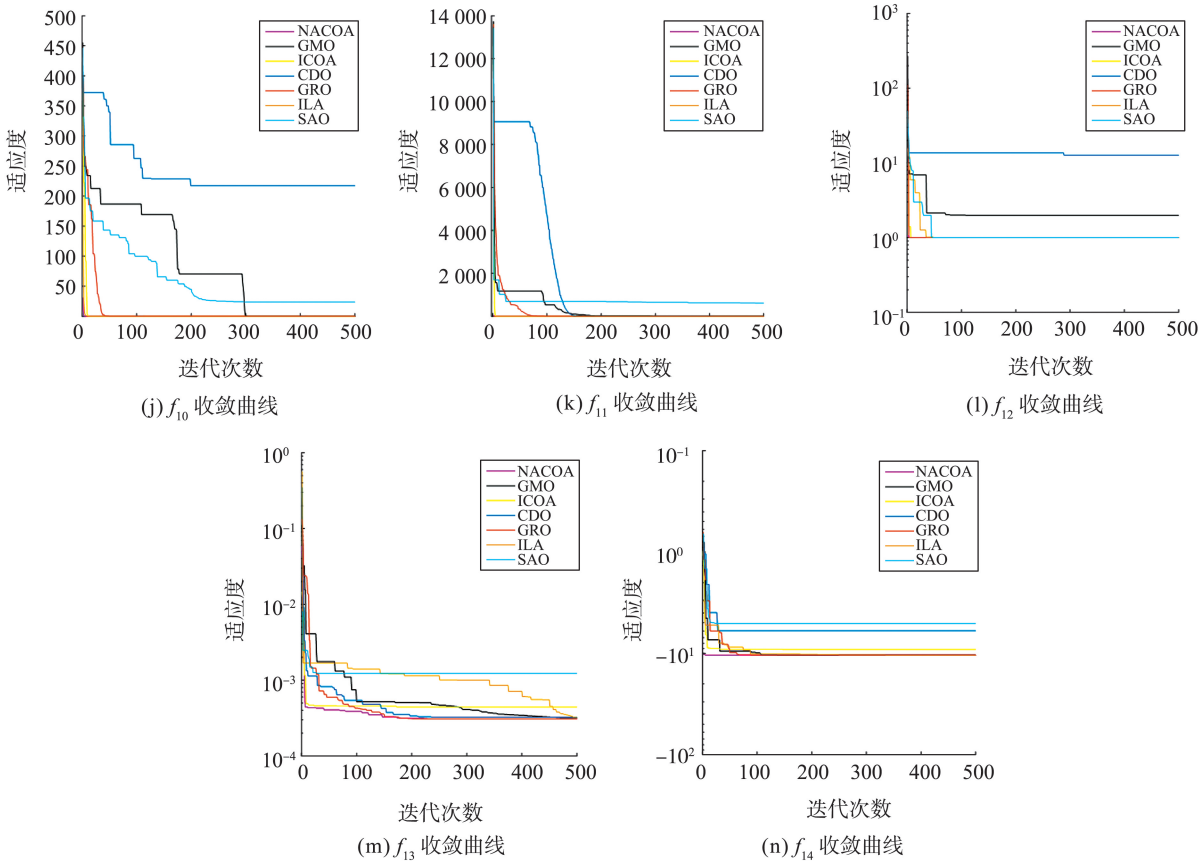


图 5 算法在基准测试函数上的收敛曲线

Fig.5 Convergence curve of the algorithm on the benchmark function

综上,在基准函数测试中,NACOA 在寻优精度与迭代次数方面均取得较好效果,与 ICOA 相比具有较强的全局搜索能力和较高的收敛效率,适用于解决不同类型的优化问题。

2.4 Wilcoxon 秩和检验

Wilcoxon 秩和检验是一种用于检验数据分布复杂性的非参数统计手段[23]。在检验水准 $\alpha=0.05$时,若 $P<0.05$,则提供充足的证据拒绝零假设,表明两种算法之间的差异存在统计学意义。当数据中出现 NaN 时,表明样本数据一致,各算法之间的差异无统计学意义。对 NACOA 与 6 种竞争算法在 30 次独立运行中的最优结果进行 Wilcoxon 秩和检验,揭示它们之间的显著性差异,结果如表 4 所示。由表 4 可以看出,虽然在 f_9、f_{11}上各算法的优化性能基本持平,但 NACOA 与 6 种算法在 Wilcoxon 秩和检验中的 P 大多小于 0.05,表明 NACOA 与其余 6 种算法的差异具有统计学意义。这一结果再次证明相较于其他算法,NACOA 算法在寻优能力上具有优势。

表 4 算法秩和检验的 P

Table 4 Algorithm rank sum test P

函数	P					
	ICOA	GMO	CDO	GRO	ILA	SAO
f_1	NaN	1.13×10^{-12}	1.13×10^{-12}	1.13×10^{-12}	NaN	1.13×10^{-12}
f_2	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	1.13×10^{-12}
f_3	1.43×10^{-12}	1.34×10^{-12}	1.34×10^{-12}	1.34×10^{-12}	1.34×10^{-12}	1.14×10^{-12}
f_4	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	2.85×10^{-11}	1.14×10^{-12}
f_5	1.13×10^{-12}	1.14×10^{-12}	1.14×10^{-12}	1.14×10^{-12}	1.14×10^{-12}	1.14×10^{-12}
f_6	1.13×10^{-12}	1.13×10^{-12}	1.57×10^{-14}	1.13×10^{-12}	1.13×10^{-12}	1.13×10^{-12}
f_7	6.00×10^{-3}	2.86×10^{-11}	1.85×10^{-4}	2.86×10^{-11}	1.85×10^{-4}	1.98×10^{-11}
f_8	2.22×10^{-11}	2.22×10^{-11}	2.22×10^{-11}	2.22×10^{-11}	1.93×10^{-11}	2.22×10^{-11}
f_9	NaN	1.13×10^{-12}	1.14×10^{-12}	NaN	NaN	1.14×10^{-12}

表 4(续)

函数	*P*					
	ICOA	GMO	CDO	GRO	ILA	SAO
f_{10}	NaN	1.57×10^{-14}	1.57×10^{-14}	1.57×10^{-14}	1.13×10^{-12}	9.35×10^{-13}
f_{11}	NaN	NaN	NaN	1.13×10^{-12}	NaN	1.13×10^{-12}
f_{12}	1.27×10^{-3}	2.32×10^{-12}	1.08×10^{-12}	NaN	NaN	NaN
f_{13}	3.68×10^{-9}	2.40×10^{-11}	3.48×10^{-10}	3.87×10^{-4}	2.11×10^{-11}	4.61×10^{-3}
f_{14}	6.07×10^{-13}	1.21×10^{-3}	1.74×10^{-12}	NaN	NaN	3.01×10^{-13}

2.5　高维函数测试对比

高维问题通常计算量较大,能够更好地评估算法在复杂问题中的全局搜索能力。为更好地验证各算法在高维问题上的寻优性能,本研究选取 ICOA、GMO、CDO、GRO 及 NACOA 共 5 种算法,基于测试函数在不同维度(维度为 300、500)条件下进行对比试验。各算法在典型单峰函数和多峰函数中的寻优结果如表 5 所示,以平均值和标准差为评价指标,反映算法的寻优精度与鲁棒性,其中最优结果加粗表示。

表 5　高维函数测试对比

Table 5　Comparison of multidimensional function tests

函数	维度	ICOA		GMO		CDO	
		平均值	标准差	平均值	标准差	平均值	标准差
f_2	300	9.46×10^{-153}	1.54×10^{-152}	2.26×10^{-69}	5.05×10^{-69}	1.46×10^{-57}	1.18×10^{-57}
	500	1.39×10^{-143}	2.36×10^{-143}	1.23×10^{-63}	2.74×10^{-63}	1.17×10^{-47}	8.86×10^{-48}
f_3	300	1.51×10^{-252}	1.78×10^{-252}	6.60×10^{-65}	1.46×10^{-64}	1.50×10^{-39}	3.36×10^{-39}
	500	3.54×10^{-242}	8.98×10^{-241}	1.19×10^{-59}	1.77×10^{-59}	5.84×10^{-37}	1.27×10^{-37}
f_5	300	1.39×10^{1}	7.36×10^{0}	2.98×10^{2}	2.35×10^{-1}	2.97×10^{2}	6.43×10^{-2}
	500	1.48×10^{1}	1.52×10^{1}	4.98×10^{2}	4.02×10^{-1}	4.97×10^{2}	6.05×10^{-2}
f_8	300	-2.68×10^{3}	3.74×10^{3}	-4.88×10^{3}	6.78×10^{3}	$\mathbf{-2.04\times10^{4}}$	7.01×10^{2}
	500	-1.11×10^{3}	9.17×10^{3}	-2.09×10^{3}	7.05×10^{4}	-4.18×10^{3}	1.17×10^{4}
f_9	300	4.44×10^{-16}	**0**	1.67×10^{-14}	1.77×10^{-15}	3.99×10^{-15}	**0**
	500	3.44×10^{-14}	1.77×10^{-14}	1.12×10^{-12}	1.87×10^{-14}	3.99×10^{-15}	**0**
f_{10}	300	**0**	**0**	1.21×10^{2}	1.34×10^{2}	6.74×10^{1}	8.24×10^{1}
	500	**0**	**0**	1.91×10^{2}	1.44×10^{3}	2.23×10^{2}	3.64×10^{0}

函数	维度	GRO		NACOA	
		平均值	标准差	平均值	标准差
f_2	300	2.25×10^{-39}	2.04×10^{-39}	$\mathbf{3.42\times10^{-205}}$	$\mathbf{1.34\times10^{-204}}$
	500	8.43×10^{-33}	1.06×10^{-33}	$\mathbf{5.96\times10^{-204}}$	$\mathbf{3.22\times10^{-204}}$
f_3	300	2.11×10^{5}	5.73×10^{4}	**0**	**0**
	500	5.97×10^{5}	1.13×10^{4}	**0**	**0**
f_5	300	2.80×10^{6}	2.02×10^{6}	**0**	**0**
	500	2.45×10^{7}	1.62×10^{7}	**0**	**0**
f_8	300	-4.60×10^{3}	1.97×10^{3}	-1.18×10^{4}	$\mathbf{6.07\times10^{2}}$
	500	-3.51×10^{3}	6.74×10^{4}	$\mathbf{-1.26\times10^{4}}$	$\mathbf{9.20\times10^{3}}$
f_9	300	3.99×10^{-15}	**0**	$\mathbf{4.44\times10^{-16}}$	**0**
	500	2.11×10^{-14}	3.12×10^{-14}	$\mathbf{4.44\times10^{-16}}$	**0**
f_{10}	300	**0**	**0**	**0**	**0**
	500	6.73×10^{0}	2.13×10^{-1}	**0**	**0**

由表 5 可以看出,在单峰函数的高维测试中,NACOA 表现出卓越的性能,在 300 维和 500 维下的测试结果充分体现出该算法在单峰优化问题中的收敛精度与稳定性。与 NACOA 相比,ICOA 随着维度增加,其收敛精度与稳定性均产生较大变化。在 f_2 测试中,与其他算法相比,NACOA 能够非常接近理论最优解,具有极高的收敛精度,且在高维情况下仍保持极高的稳定性。在 f_3、f_5 测试中,只有 NACOA 的收敛精度和稳定性均达到理论最优解。在多峰函数 f_{10} 测试中,NACOA 具有极高的稳

定性和准确性。在f_9测试中，NACOA虽然未能取得理论最优解，但其收敛精度与稳定性仍明显优于其他对比算法。

综上所述，NACOA在高维函数优化问题中表现出良好的全局搜索与局部开发能力，显著优于其他算法，展示出该算法在解决复杂优化问题时的优秀能力。

3 工程应用

3.1 焊接梁设计问题

在焊接梁设计问题中涉及一个最小化目标，即通过优化算法减少制造过程中的成本。该优化任务旨在确定4个关键设计参数——梁的长度l、宽度b、高度d和焊接缝宽度h，以确保它们满足剪切应力τ、弯曲应力σ、梁的弯曲载荷P_c、端部偏差δ和边界条件等限制，同时达到成本最低。该设计问题为非线性规划任务，涉及多个设计变量和复杂的约束条件。选择该问题作为验证NACOA的工程应用场景，将l、b、d和h设为模型的4个变量，σ、P_c、δ和边界条件等限制设为模型的约束条件，在焊接梁设计问题中寻找更优的设计方案，降低制造成本。问题的数学模型如下。

变量为

$$x=\{x_1,x_2,x_3,x_4\}=\{h,l,d,b\}。\tag{11}$$

目标函数为

$$f=1.104\ 71x_1^2x_2+0.048\ 11x_3x_4(14+x_2)。\tag{12}$$

约束条件为

$$\begin{cases}\tau(x)-\tau_{\max}\leqslant 0\\ \sigma(x)-\sigma_{\max}\leqslant 0\\ \delta(x)-\delta_{\max}\leqslant 0\\ x_1-x_4\leqslant 0\\ 6\ 000-P_c(x)\leqslant 0\\ 0.125-x_1\leqslant 0\end{cases},\tag{13}$$

式中：$\tau(x)=\sqrt{(\tau')^2+\tau'\tau''\dfrac{x_2}{R}+(\tau')^2}$，其中$\tau'$为焊接缝直接剪切应力，$\tau'=\dfrac{6\ 000}{\sqrt{2}x_1x_2}$，$\tau''$为焊接缝弯曲剪切应力分量，$\tau''=\dfrac{3\ 000\left(14+\dfrac{x_2}{2}\right)R}{\sqrt{2}x_1x_2\left[\dfrac{x_2^2}{12}+\left(\dfrac{x_1+x_3}{2}\right)^2\right]}$，$R$为焊接缝到梁中性轴的几何距离，$R=\sqrt{\dfrac{x_2^2}{4}+\left(\dfrac{x_1+x_3}{2}\right)^2}$；$\sigma(x)=\dfrac{504\ 000}{x_3^2x_4}$；$\delta(x)=\dfrac{2.4\times10^4\times14^3}{3\times10^6x_3^3x_4}$；$P_c(x)=\dfrac{1.203\ 9\times10^7x_3x_4^3}{1\ 176}\left(1-\dfrac{x_3}{28}\sqrt{\dfrac{3\times10^6}{4.8\times10^7}}\right)$；$\tau_{\max}$为最大剪切应力，$\tau_{\max}=136\ 000$；$\sigma_{\max}$为最大弯曲应力，$\sigma_{\max}=30\ 000$；$\delta_{\max}$为最大端部偏差，$\delta_{\max}=0.25$。

变量边界约束为

$$0.1\leqslant x_1\leqslant 2,\ 0.1\leqslant x_2\leqslant 10,$$
$$0.1\leqslant x_3\leqslant 10,\ 0.1\leqslant x_4\leqslant 2。$$

为验证算法之间的差异是否存在统计学意义，本研究使用Wilcoxon秩和检验对试验结果进行非参数假设检验。NACOA与COA、GMO、ILA、ICOA对比算法在该工程问题中独立迭代寻优30次的对比结果如表6所示。由表6可知，与其他算法相比，NACOA在该工程问题中表现最为突出，设计成本最低，表明NACOA在该工程问题中是一个有效的优化算法。

表6 焊接梁设计问题结果对比

Table 6 Comparison of results of welded beam design problems

算法	h	l	d	b	最优解	P
NACOA	0.205 7	3.235 6	9.036 0	0.205 7	**1.692 8**	
COA	0.144 0	5.366 3	9.011 1	0.207 6	1.866 2	2.44×10^{-11}
GMO	0.185 1	5.003 0	9.150 5	0.205 1	1.905 8	2.48×10^{-11}
ILA	0.202 4	3.107 9	9.506 4	0.203 6	1.733 8	2.06×10^{-11}
ICOA	0.205 0	3.244 7	9.051 5	0.205 7	1.695 6	1.70×10^{-5}

3.2 悬臂梁设计问题

悬臂梁设计问题涉及结构工程领域的设计优化。该悬臂梁一端被固定支撑，另一端为自由端，受垂直方向力的作用，由5个等厚度（设为2/3）的空心方形截面构成。将截面高度h_1、h_2、h_3、h_4、h_5设为变量。问题的数学模型如下。

变量为

$$x=\{x_1,x_2,x_3,x_4,x_5\}=\{h_1,h_2,h_3,h_4,h_5\}。\tag{14}$$

目标函数为

$$f=0.062\ 4(x_1+x_2+x_3+x_4+x_5)。\tag{15}$$

约束条件为

$$\frac{61}{x_1^3}+\frac{37}{x_2^3}+\frac{19}{x_3^3}+\frac{7}{x_4^3}+\frac{1}{x_5^3}\leqslant 1。\tag{16}$$

变量边界约束为

$$0.01\leqslant x_i\leqslant 100。$$

使用Wilcoxon秩和检验对悬臂梁设计问题的试验结果进行非参数假设检验。在悬臂梁设计问题中，NACOA与其他算法独立迭代寻优30次的结果如表7所示。由表7可知：COA与ICOA在该问

题中优化能力较弱;Wilcoxon 秩和检验结果表明 NACOA 与 GMO 性能相当,但与其余算法的差异具有统计学意义;NACOA 的最优解为 1.339 96,在所有算法中最小,表明其在解决该工程设计问题时具有良好的优化性能。

表 7　悬臂梁设计问题结果对比

Table 7　Comparison of results of cantilever beam design problems

算法	h_1	h_2	h_3	h_4	h_5	最优解	P
NACOA	6.013 15	5.322 87	4.492 30	3.496 57	2.148 87	**1.339 96**	
COA	5.976 06	5.212 46	4.598 04	3.548 14	2.149 59	1.340 61	2.48×10^{-11}
GMO	6.032 40	5.315 34	4.481 03	3.499 02	2.146 08	1.339 97	NaN
ILA	6.023 58	5.316 04	4.490 48	3.502 97	2.141 43	1.340 01	4.37×10^{-11}
ICOA	6.522 96	5.262 81	4.278 66	3.276 10	2.280 66	1.349 26	2.43×10^{-11}

4　结论

为解决 COA 全局搜索能力不足、易陷入局部最优和收敛速度慢的问题,本研究使用 Logistic-Tent 映射实现种群分布的均匀性和随机性,提高算法搜索能力;通过莱维飞行改进算法的随机性,增加跳跃性,提升算法的收敛效率;利用非线性递减惯性权重增强算法早期探索与后期开发之间的平衡,同时解决算法振荡问题;结合黄金正弦函数与分割比例,优化逃避路径效率,提高收敛精度。在基准测试函数上与 6 种算法进行对比,通过结果分析与秩和检验证明 NACOA 在收敛速度和优化精度方面具有显著的优越性。将 NACOA 应用到工程设计问题中,展现出算法良好的实用性。在未来工作中,将把算法进行离散化改进,用于求解现实生活中的离散型问题,进一步提升算法的应用性。

参考文献:

[1] REZAEI F, SAFAVI H R, ABD ELAZIZ M, et al. GMO: geometric mean optimizer for solving engineering problems[J]. Soft Computing, 2023, 27(15): 10571-10606.

[2] SHEHADEH H A. Chernobyl disaster optimizer (CDO): a novel meta-heuristic method for global optimization[J]. Neural Computing and Applications, 2023, 35(15): 10733-10749.

[3] ZOLFI K. Gold rush optimizer: a new population-based metaheuristic algorithm[J]. Operations Research and Decisions, 2023, 33(1): 113-150.

[4] MIRRASHID M, NADERPOUR H. Incomprehensible but intelligible-in-time logics: theory and optimization algorithm [J]. Knowledge-Based Systems, 2023, 264: 110305.

[5] DENG L Y, LIU S Y. Snow ablation optimizer: a novel metaheuristic technique for numerical optimization and engineering design[J]. Expert Systems with Applications, 2023, 225: 120069.

[6] DEHGHANI M, MONTAZERI Z, TROJOVSKÁ E, et al. Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems [J]. Knowledge-Based Systems, 2023, 259: 110011.

[7] 苏仁斌, 熊卫红, 刘先珊, 等. 基于新型元启发式 BP 神经网络的 500 kV 覆冰输电线路力学响应预测研究[J]. 应用基础与工程科学学报, 2024, 32(1): 100-122.
SU Renbin, XIONG Weihong, LIU Xianshan, et al. Study on BP neural network based on a new metaheuristic optimization algorithm and prediction of mechanical response for 500 kV UHV transmission lines considering icing[J]. Journal of Basic Science and Engineering Sciences, 2024, 32(1): 100-122.

[8] ABDULLAH MENGASH A, ALQAHTANI H, MARAY M, et al. Coati optimization-based energy efficient routing protocol for unmanned aerial vehicle communication[J]. Computers, Materials & Continua, 2023, 75(3): 4805-4820.

[9] MEKHAMER A S, HASANIEN H M, ALHARBI M, et al. Coati optimization algorithm-based optimal frequency control of power systems including storage devices and electric vehicles[J]. Journal of Energy Storage, 2024, 93: 112367.

[10] HUAN J J, HE Y L, SUN K, et al. Capacity planning for wind, solar, thermal and energy storage in power generation systems considering coupled electricity-carbon markets[J]. IET Generation, Transmission & Distribution, 2024, 18(24): 4090-4104.

[11] LEI W, WANG G, WAN B Q, et al. High voltage shunt reactor acoustic signal denoising based on the combination of VMD parameters optimized by coati optimization algorithm and wavelet threshold[J]. Measurement, 2024, 224: 113854.

[12] 秦敏敏, 刘立芳, 齐小刚. 面向维修资源分配调度的遗传-长鼻浣熊混合优化算法[J]. 智能系统学报, 2023, 18(6): 1322-1335.

QIN Minmin, LIU Lifang, QI Xiaogang. Hybrid genetic long-nosed raccoon optimization algorithm for maintenance resource allocation and scheduling[J]. CAAI Transactions on Intelligent Systems, 2023, 18(6): 1322-1335.

[13] 杨世源. 基于自适应克里金模型的混合不确定性设计优化方法研究[D]. 成都: 电子科技大学, 2024: 14-15.
YANG Shiyuan. Design andoptimization method based on adaptive Kriging model under mixed uncertainty[D]. Chengdu: University of Electronic Science and Technology of China, 2024: 14-15.

[14] THIRUMOORTHY K, JEROLD JOHN BRITTO J. A two-stage feature selection approach using hybrid quasi-opposition self-adaptive coati optimization algorithm for breast cancer classification[J]. Applied Soft Computing, 2023, 146: 110704.

[15] WANG C, LIN H, YANG M, et al. A novel chaotic time series wind power point and interval prediction method based on data denoising strategy and improved coati optimization algorithm[J]. Chaos, Solitons & Fractals, 2024, 187: 115442.

[16] RAVINDRAN N, ANTO KUMAR R P. SECOA: serial exponential coati optimization algorithm for MANET routing with link lifetime prediction[J]. Engineering Science and Technology, an International Journal, 2024, 59: 101869.

[17] ARORA S, ANAND P. Chaotic grasshopper optimization algorithm for global optimization[J]. Neural Computing and Applications, 2019, 31(8): 4385-4405.

[18] 李鹏, 丁倩雯. 基于麻雀算法优化的 OSTU 分割算法[J]. 电子测量技术, 2021, 44(19): 148-154.
LI Peng, DING Qianwen. OSTU segmentation algorithm based on sparrow algorithm optimization[J]. Electronic Measurement Technology, 2021, 44(19): 148-154.

[19] 蔡娟. 混沌映射与精英高斯扰动的非线性灰狼优化算法[J]. 计算机工程与设计, 2022, 43(1): 186-195.
CAI Juan. Non-linear gray wolf optimization algorithm based on chaotic Tent mapping and elite Gauss perturbation[J]. Computer Engineering and Design, 2022, 43(1):186-195.

[20] WU L, WU J W, WANG T B. Enhancing grasshopper optimization algorithm (GOA) with levy flight for engineering applications[J]. Scientific Reports, 2023, 13: 124.

[21] 匡鑫, 阳波, 马华, 等. 多策略改进的蜣螂优化算法[J]. 计算机工程, 2024, 50(10): 119-136.
KUANG Xin, YANG Bo, MA Hua, et al. Multi-strategy improved dung beetle optimization algorithm[J]. Computer Engineering, 2024, 50(10): 119-136.

[22] 邱文浩. 基于改进浣熊优化算法的含抽水蓄能电力系统经济调度研究[D]. 南昌: 南昌大学, 2024: 14-16.
QIU Wenhao. Research on economic dispatch of pumped storage power systems based on improved coati optimization algorithm[D]. Nanchang: Nanchang University, 2024: 14-16.

[23] WILCOXON F. Individual comparisons by ranking methods[J]. Biometrics, 1945, 1: 80-83.

(编辑:孙亚彤)