DOI:10.19651/j.cnki.emt.2415972

多策略改进蜣螂优化算法及其应用*

刘 薇 任腾腾 韩广雨 李 彤 严文律
(沈阳理工大学信息科学与工程学院 沈阳 110159)

摘 要:针对蜣螂优化算法全局探索能力差,容易陷入局部最优等问题,本文提出了一种基于正余弦算法和蜣螂优化算法的混合算法叫做 SCDBO 算法。该混合算法采用正余弦搜索算法代替蜣螂算法中滚球蜣螂的搜索机制,平衡了算法的全局搜索和局部开发能力。此外,在每次迭代过程中引入 t 分布扰动对蜣螂种群以一定概率进行更新的同时,引入 Levy-柯西变异算子,对最优位置进行突变。不仅加快了算法的收敛速度,也减少陷入局部最优的可能。最后采用混沌映射对蜣螂种群进行初始化增强了算法的种群多样性。采用 23 个基准函数 SCDBO 算法的有效性进行了研究,实验结果表明,该算法相对于其他对比算法表现出了更好的寻优能力。为了进一步评估 SCDBO 算法的实际应用性能,将该算法成功应用于 3 个工程设计问题。通过与其他算法的比较,结果表明,SCDBO 算法在解决实际工程问题方面具有很高的潜力。

关键词:蜣螂优化算法;正余弦算法;t 分布扰动;Levy-柯西变异;混沌映射
中图分类号:TP301.6;TN03 **文献标识码**:A **国家标准学科分类代码**:520.1040

Multi strategy improvement of dung beetle optimization algorithm and its application

Liu Wei Ren Tengteng Han Guangyu Li Tong Yan Wenlyu
(College of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China)

Abstract: Aiming at problems such as the dung beetle optimization algorithm's poor global exploration ability and its tendency to fall into local optimization, this paper proposes a hybrid algorithm based on the positive cosine algorithm and the dung beetle optimization algorithm called the SCDBO algorithm. The hybrid algorithm adopts the positive cosine search algorithm instead of the search mechanism of the rolling dung beetle in the dung beetle algorithm, which balances the global search and local exploitation ability of the algorithm. In addition, while introducing the t-distribution perturbation to update the dung beetle population with a certain probability during each iteration, the Levy-Corsi variation operator is introduced to mutate the optimal position. This not only accelerates the convergence speed of the algorithm but also reduces the possibility of falling into the local optimum. Finally, the population diversity of the algorithm is enhanced by initializing the dung beetle population with chaotic mapping. The effectiveness of the SCDBO algorithm is investigated using 23 benchmark functions, and the experimental results show that the algorithm exhibits a better ability to find the optimum compared with other comparative algorithms. To further evaluate the performance of the SCDBO algorithm for practical applications, the algorithm was successfully applied to three engineering design problems. By comparing with other algorithms, the results show that the SCDBO algorithm has high potential in solving practical engineering problems.

Keywords: dung beetle optimization; sine cosine algorithm; t-distribution perturbation; Levy-Cauchy variation; chaotic mapping

0 引 言

在现实实际应用中,优化问题是十分普遍的尤其是在物理,化学和生物领域。而元启发式算法得益于其结构简单,无需梯度信息等优势逐渐得到了众多研究者的广泛关注。它是通过对自然界中不同生物群体的行为机制或物理

收稿日期:2024-05-04
* 基金项目:辽宁省教育厅高等学校基本科研项目(JYTMS20230189)、沈阳理工大学引进高层次人才科研支持计划(1010147001131)资助

化学现象进行模拟而产生的一种随机优化技术。常见的算法有：遗传算法（genetic algorithm，GA）[1]、粒子群算法（particle swarm optimization，PSO）[2]、鲸鱼优化算法（whale optimization algorithm，WOA）[3]、灰狼优化算法（grey wolf optimizer，GWO）[4]、哈里斯鹰优化算法（Harris hawks optimization，HHO）[5]、正余弦优化算法（sine cosine algorithm，SCA）[6]等。

没有免费的午餐（no free lunch，NFL）理论[7]指出了一个观点，即不存在一种通用的元启发算法能够解决所有的优化问题。因此，学者们不断创新算法以应对不同的问题。为了解决特定问题，人们巧妙地将两种或多种元启发算法进行融合，从而提出了混合算法。混合算法不仅克服了传统单一算法的局限性，同时通过多种算法的协同工作，显著提升了算法优化过程的效率和效果。文献[8]提出了基于正交对立学习的改进麻雀搜索算法，与多个传统优化算法以及两个改进算法进行对比，表现出了优越的性能。此外，文献[9]引入了算数优化算子以及 t 分布扰动等策略对海洋捕食者算法进行了改进，实验结果表明，在基准测试函数上以及工程应用问题中均取得了优异的成绩。文献[10]提出了一种新的改进鲸鱼优化算法，采用多策略混合的方式。该文献运用了 23 个基准函数对改进的鲸鱼优化算法进行全面评估，验证了其卓越性能。文献[11]对 HHO 进行了改进，将其成功应用于旅行商问题（TSP）。通过对 80 个 TSP 数据集的测试，证明了这一算法的有效性。

蜣螂优化算法（dung beetle optimizer，DBO）[12]是近年来提出的一种新型群体智能优化算法，灵感来自于蜣螂种群的滚球、舞蹈、觅食、繁殖和窃取行为。与粒子群优化算法等传统算法相比，该算法具有快速收敛、高精度求解等优点，但同时也存在全局探索和局部开发能力不平衡的问题，容易陷入局部最优解，并且全局探索能力较弱。为此，许多研究者通过结合多种不同的优化策略对蜣螂优化算法进行了改进，同时对蜣螂优化算法的应用范围进行了扩充。为了实现移动机器人在复杂环境中路径规划的高效搜索能力，文献[13]提出了一种混合多策略的改进蜣螂算法。采用基本测试函数和路径优化方面仿真，将改进后算法与其他四种群体智能算法进行了比较。实验结果表明，改进的蜣螂优化算法显著提高了收敛速度和优化精度，具有良好的鲁棒性。受到 Sinh 和 Cosh 函数的数学性质的启发，文献[14]提出了一种新的蜣螂优化算法，该算法通过利用 Sinh 和 Cosh 函数扰乱 DBO 的初始分布，平衡滚球蜣螂的发育，通过非线性增强提高了 DBO 的搜索效率和全局探索能力。实验结果表明，该算法具有较好的优化性能。本文针对蜣螂优化算法所存在的现有问题，提出了一种新的优化算法，即 SCDBO 算法。本文结合正余弦算法，并将其嵌入 DBO 算法中，极大的提高了算法的全局探索能力。针对蜣螂种群初始化的问题，引入了一种新的混沌映射方法，有效的提高了初始解的质量，其次，结合 t 分布扰动和变异算子，极大的提升了算法跳出局部最优的能力。

为了验证 SCDBO 算法的性能，在 23 个基准测试函数上与多个元启发式算法以及改进的元启发式算法进行了对比。此外，SCDBO 还被应用于 3 个工程应用设计问题中去，结果表明 SCDBO 算法具有较强的寻优能力，是解决实际应用问题的一种有竞争力的算法。

1　蜣螂优化算法

蜣螂优化算法是一种新型群体智能优化算法，它借鉴了蜣螂在自然界中的社会行为。该算法将蜣螂种群按一定比例分为滚球蜣螂、繁育蜣螂、小蜣螂和偷窃蜣螂四种代理，具体比例划分如图 1 所示。

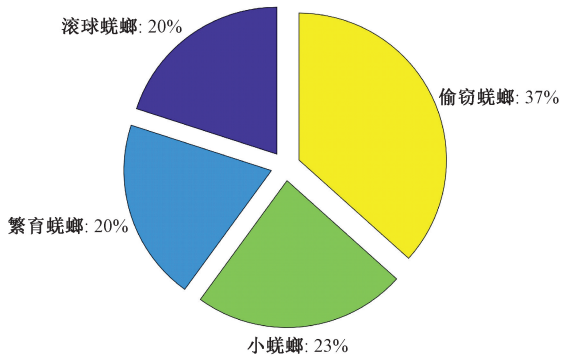

图 1　蜣螂种群分布图

Fig. 1　Distribution map of dung beetle populations

1.1　滚球蜣螂

蜣螂在滚球过程中需要根据天体信号来保持粪球在直线上滚动，在滚动过程中，蜣螂的位置也在不断变化。蜣螂的滚球数学模型可用式(1)表示。

$$x(t+1)=x_i(t)+\alpha\times k\times x_i(t-1)+b\times\Delta x,\quad \Delta x=|x_i(t)-X^w| \tag{1}$$

$x_i(t)$ 表示第 t 次迭代过程中第 i 只代理的位置，α 的取值为 -1 或 1。k 表示为偏转系数，且 $k\in(0,0.2]$。b 表示为一个随机常量，且 $b\in(0,1)$。X^w 为全局最差位置。

当蜣螂在滚球过程中遇到障碍物无法继续前行时，它需要通过跳舞来确定自己新的滚球方向，而这种跳舞行为被定义如下：

$$x_i(t+1)=x_i(t)+\tan(\theta)\,|x_i(t)-x_i(t-1)| \tag{2}$$

由上述公式中 $|x_i(t)-x_i(t-1)|$ 可以看出，蜣螂的位置更新受到当前和历史数据的影响。其中 $\theta\in[0,\pi]$，值得注意的是当 θ 取值为 0、$\pi/2$、π 时蜣螂的位置将不会进行更新。

1.2　繁育蜣螂

为了给后代提供安全的环境，蜣螂会将粪球移动至安

全地点并隐藏起来,然后进行产卵。因此,本文提出了一种模拟雌性蜣螂产卵区域的边界选择策略。

$$\begin{aligned} Lb^* &= \max(X^* \times (1-R), Lb) \\ Ub^* &= \min(X^* \times (1+R), Ub) \end{aligned} \tag{3}$$

X^* 表示局部最优值,Lb 和 Ub 分别表示问题求解的上下界,$R = 1 - \frac{t}{T_{\max}}$,$T_{\max}$ 为最大迭代次数。Lb^* 和 Ub^* 分别表示为产卵区域的上下边界。根据上述公式推断,蜣螂产卵区域的边界是根据 R 值的变化而动态决定的。因此,蜣螂繁殖位置会不断更新,具体的数学表达式如下:

$$X_i(t+1) = X^* + \boldsymbol{b}1 \times (X_i(t) - Lb^*) + \boldsymbol{b}2 \times (X_i(t) - Ub^*) \tag{4}$$

其中,$\boldsymbol{b}1$ 和 $\boldsymbol{b}2$ 代表两个独立的 $1 \times D$ 随机向量,D 是表示优化问题维度。

1.3　小蜣螂

为了模拟小蜣螂的觅食行为,首先需要设定一个引导小蜣螂进行觅食的最佳区域,并对觅食的边界进行定义:

$$\begin{aligned} Lb^b &= \max(X^b \times (1-R), Lb) \\ Ub^b &= \min(X^b \times (1+R), Ub) \end{aligned} \tag{5}$$

X^b 代表整体最佳位置,Lb^b 和 Ub^b 分别代表最佳觅食边界。因此,小蜣螂的位置更新如下:

$$x_i(t+1) = x_i(t) + C1 \times (x_i(t) - Lb^b) + C2 \times (x_i(t) - Ub^b) \tag{6}$$

$C1$ 是服从正态分布的随机变量,$C2$ 表示在区间(0,1)上的随机变量。

1.4　偷窃蜣螂

在自然界中会有一些蜣螂偷窃其他蜣螂的粪球。用以下公式模拟蜣螂的偷窃行为:

$$x_i(t+1) = X^b + S \times \boldsymbol{g} \times (|x_i(t) - X^*| + |x_i(t) - X^b|) \tag{7}$$

S 是一个常数,而 $\boldsymbol{g}$ 是一个 $1 \times D$ 的随机向量。

2　蜣螂算法的改进

在优化问题中,DBO 算法展现出了出色的优化性能,但是对于解决复杂问题方面,也面临极大的挑战。因此本文在现有 DBO 算法的缺陷上对其进行改进,通过引入正余弦算法策略,Logistic-tent 混沌映射策略,t 分布扰动和自适应 Levy-柯西变异策略对 DBO 算法的优化能力进行提升。

2.1　Logistic-tent 混沌映射

传统的 DBO 算法在启动阶段对蜣螂个体进行随机初始化,存在着群体多样性低、搜索结果不够理想等问题。对 DBO 算法的收敛性及准确性有一定的影响。Logistic-Tent 混沌映射[15],将 Logistic 映射与 Tent 映射相融合,大大确保了群体在空间中的遍历特性,增强了其整体寻优性能,极大的改进了 DBO 算法中群体多样性。Logistic-Tent 的混沌映射序列分布直方图如图 2 所示,它的具体表达式如式(8)所示。

$$X_{m+1} = \begin{cases} (a \times x_m \times (1-x_m) + (4-a) \times \frac{x_m}{2}) \bmod 1, & x_m < 0.5 \\ (a \times x_m \times (1-x_m) + (4-a) \times \frac{1-x_m}{2}) \bmod 1, & x_m \geqslant 0.5 \end{cases} \tag{8}$$

其中,a 为映射参数,且 $a \in (0,4)$,在本文中设置 $a = 0.3$,以达到最好的映射效果。

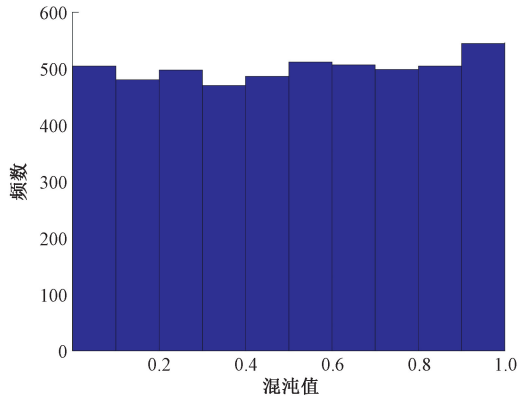

图 2　混沌映射序列分布直方图

Fig. 2　Histogram of the distribution of chaotic mapping sequences

2.2　正余弦算法

正余弦算法具备出色的全局搜索性能,但其缺点在于收敛精度不高且易于停滞于局部最优点。为了解决这些问题,本文引入了 Levy 飞行策略来增强算法的优化精确度。该策略通过干扰当前位置,有助于防止算法陷入局部最优的困境。以下是改进后余弦正弦算法的位置更新表达式:

$$X_i^{t+1} = \begin{cases} Levy \times X_i^t + c_1 \times \sin(c_2) \times |c_3 \times bestX - X_i^t|, & c_4 < 0.5 \\ Levy \times X_i^t + c_1 \times \cos(c_2) \times |c_3 \times bestX - X_i^t|, & c_4 > 0.5 \end{cases} \tag{9}$$

其中,c_1,c_3,c_4 为 3 个服从均匀分布的随机数,$c_2 \in [0,2\pi]$,$c_3 \in [0,2]$,$c_4 \in [0,1]$,c_1 为控制参数,$c_1 = a \cdot (1 - t/Tmax)$,a 为常数,一般取 2。

Levy 飞行策略[16]是一种用于模拟随机游走或搜索过程中的步长和方向的随机行为策略。Levy 飞行服从的是莱维分布,其计算公式如下:

$$\text{Levy} = 0.1 \cdot \frac{u}{|\nu|^{\frac{1}{\beta}}} \tag{10}$$

ν 服从标准正态分布,u 服从均值为 0,方差为 σ_μ 的高斯分布,σ_μ 计算公式如下:

$$\sigma_\mu = \left[\frac{\Gamma(1+\lambda) \times \sin\left(\pi \times \frac{\lambda}{2}\right)}{\Gamma\left(\frac{1+\lambda}{2}\right) \times \lambda \times 2^{\frac{\lambda-1}{2}}} \right]^{\frac{1}{\lambda}} \tag{11}$$

其中,λ 的取值区间为(1,3),本文的取值为 1.5;$\Gamma(x)$ 为 gamma 函数。

滚球蜣螂在整个算法中具有重要的地位,其迭代产生

的当前最优位置极大的影响着繁育蜣螂和小蜣螂的寻优边界。或者也可以这样理解，滚球蜣螂在算法中起到了全局探索的作用，而其余几个种群则表现出局部开发的作用。因此，选择将上述改进的正余弦算法融入到蜣螂优化算法中去，具体表现为替代了原滚球蜣螂的更新方式。这种有机结合极大的提高了算法的全局探索能力，同时通过此方式所产生的当前最优值也会对下面几个种群产生比较积极的影响。

2.3　自适应 t 分布扰动

在蜣螂的位置更新完成后，一种创新的自适应 t 分布策略被引入[17]，旨在动态地调整种群位置，从而显著提升算法的收敛速率。特别地，把迭代次数（*iter*）巧妙地融合到该分布的自由度参数中，赋予了算法独特的特性：初期阶段，它展现出强大的全局搜索潜能；随着迭代深入，它又展现出优秀的局部优化技巧。这种策略极大地提升了算法的收敛效果和问题求解的执行效率。具体的，自适应 t 分布方法调整蜣螂个体位置的方式如下：

$$Xnew = X_i^t + X_i^t \cdot t(iter) \tag{12}$$

其中 $t(iter)$ 表示为以 *iter* 为自由度参数的自适应 t 分布函数，$Xnew$ 表示为自适应 t 分布产生的新的位置。

在迭代过程中，频繁地依赖 t 分布扰动无疑提升了算法的时间消耗。为此，本文引入了一种智能的决策机制，即动态地决定是否实施这种扰动。其决策逻辑基于一个随机概率阈值，如式(13)所示：当随机数 rand 小于预设概率 p 时，才会进行 t 分布扰动；反之则跳过。一旦选择了扰动，本文会进一步应用一个策略来确定是否更新当前位置，这就是所谓的贪婪策略，由式(14)详细阐述。

$$p = \frac{t}{T_{\max}} \tag{13}$$

$$X_i^{t-1} = \begin{cases} X_{\text{new}}, & f(X_{\text{new}}) < f(X_j^t) \\ X_i^t, & \text{其他} \end{cases} \tag{14}$$

2.4　Levy-柯西变异

在智能优化方法中，变异操作的引入对增强群体多样性及提升算法逃脱局部极值的性能起到了关键作用。为此，设计了一种结合 Levy 变异与 Cauchy 变异的复合自适应变异策略。此策略被应用于对全局最优位置的变异过程中，显著增强了算法逃避局部最优解的能力，同时拓宽了算法搜索的范围。其数学表述如下：

$$Y^b(t) = X^b(t) + X^b(t)\left(\left(1 - \frac{t}{T}\right) \times Levy + \left(\frac{t}{T}\right) cauchy(\sigma)\right) \tag{15}$$

$Y^b(t)$ 为在第 t 次迭代过程中最优位置 $X^b(t)$ 在 Levy-柯西变异扰动后的位置，$cauchy(\sigma)$ 为柯西变异算子。

在每次算法迭代过程中，通过 Levy-柯西变异对最优位置进行扰动，极大的避免了蜣螂个体可能发生的搜索停滞而导致陷入局部最优当中，同时通过迭代产生的新的蜣螂代理也提高了算法的种群多样性。与上述正余弦算法策略相呼应，实现了全局探索和局部开发能力的平衡。然而，再每次进行变异过后，所得到的新的最优位置不一定比原来的好，因此采用贪婪策略来决定是否对最优位置进行更新。贪婪策略公式如下所示：

$$X^b = \begin{cases} Y^b(t), & f(Y^b(t)) < f(X^b(t)) \\ X^b(t), & \text{其他} \end{cases} \tag{16}$$

2.5　SCDBO 算法流程图

SCDBO 算法流程图如图 3 所示。

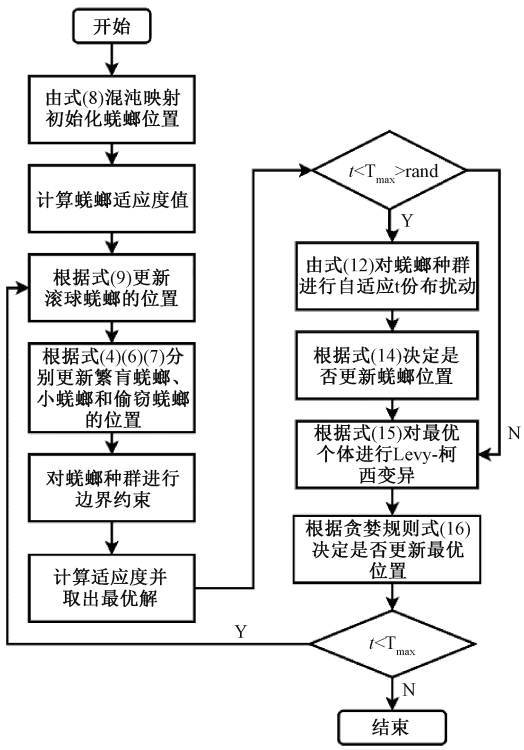

图 3　SCDBO 算法流程图

Fig. 3　Flowchart of SCDBO algorithm

3　SCDBO 算法在测试函数上的表现评估

在本文中，除了将 SCDBO 算法进行试验之外，还选取 DBO[12]算法、鲸鱼优化算法（WOA）[3]、哈里斯鹰优化算法（HHO）[5]、SCA[6]、北方苍鹰优化算法（northern goshawk optimization，NGO）[18]5 种原始算法以及改进黑寡妇优化算法（IBWOA）[19]、文献[14]中改进 DBO 算法（SCDBO1）作为对比算法。这两种混合算法均为最新提出的改进元启发式算法，其中文献[14]中改进蜣螂优化算法名称与本文冲突，故将其命名为 SCDBO1。在 23 个基准测试函数上比较平均值和方差来验证 SCDBO 算法的优越性。在本次对比研究中，对各类算法执行了 30 轮独立的试验，设定每个算法最大迭代次数为 500 次，所有种群的个体数量都设定为 30。其余与之对比的算法，其各项参数皆依照原始文献的设定保持不变。实验所采用的基准测试函数涵盖了单

峰、多峰以及固定维度多峰基准测试函数。函数公式及其　参数设置以及理论最优值如表 1～3 所示。

表 1　单峰基准测试函数

Table 1　Unimodal benchmark functions

函数	维度	范围	最优值
$F_1(x)=\sum_{i=1}^{n}x_i^2$	30	[−100,100]	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[−10,10]	0
$F_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	[−100,100]	0
$F_4(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	30	[−100,100]	0
$F_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[−30,30]	0
$F_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	[−100,100]	0
$F_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	[−1.28,1.28]	0

表 2　多峰基准测试函数

Table 2　Multimodal benchmark functions

函数	维度	范围	最优值
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[−500,500]	418.982 9×dim
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	[−5.12,5.12]	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[−32,32]	0
$F_{11}(x)=\frac{1}{4\ 000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	30	[−600,600]	0
$F_{12}(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$，$y_i=1+\frac{x_i+1}{4}$，$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	30	[−50,50]	0
$F_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[−50,50]	0

3.1　SCDBO 算法在单峰基准测试函数上实验分析

表 4 为针对单峰测试函数的 SCDBO 算法和对照算法的试验结果(最好的结果用粗体显示)。从这些结果中可以看出 SCDBO 算法在 F1～F4 上均可以找到理论最优值,远远领先于其他对比算法。而 IBWOA 在 F1 上也可以找到最优值,但是结合图 4 中 F1 收敛结果来看,其收敛速度远不如 SCDBO 算法。所提算法在函数 F6 和 F7 上虽然没能找到理论最优值,但寻优结果的均值是所有对比算法中的最好的。而在 F5 上的测试结果也排在几种算法前列。从表 4 中的方差结果可以看出,SCDBO 算法具有很好的鲁棒性。SCDBO 算法和对比算法在单峰测试函数上的收敛图如图 4 所示。从图中不难看出 SCDBO 算法表现出了极快的收敛速度和较高的收敛精度。因此在单峰测试函数上表现出 SCDBO 算法较强的局部开发能力。

3.2　SCDBO 算法在多峰基准测试函数上实验分析

表 5 为 SCDBO 算法和其他对比算法在多峰测试函数

表 3　固定维多峰基准测试函数

Table 3　Fixed-dimension multimodal benchmark functions

函数	维度	范围	最优值
$F_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65,65]$	1
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.000 307 5
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]$	−1.031 628 5
$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$[-5,5]$	0.398
$F_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-5,5]$	3
$F_{19}(x)=\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	$[0,1]$	−3.86
$F_{20}(x)=\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	$[0,1]$	−3.32
$F_{21}(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10.153 2
$F_{22}(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10.402 8
$F_{23}(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10.536 3

表 4　各算法在单峰基准测试函数上的实验结果

Table 4　Experimental results of each algorithm on unimodal benchmark functions

	参数	SCDBO	DBO	NGO	SCA	HHO	WOA	SCDBO1	IBWOA
F1	Mean	**0**	8.21×10^{-110}	3.48×10^{-87}	9.65	2.35×10^{-97}	$3.00\times10-^{75}$	2.11×10^{-253}	**0**
	Std	**0**	4.49×10^{-109}	1.22×10^{-86}	1.69×10^{1}	7.48×10^{-97}	9.37×10^{-75}	**0**	**0**
F2	Mean	**0**	1.77×10^{-58}	1.09×10^{-45}	1.70×10^{-2}	8.59×10^{-50}	7.59×10^{-51}	2.04×10^{-127}	1.46×10^{-207}
	Std	**0**	8.94×10^{-58}	9.79×10^{-46}	1.87×10^{-2}	4.65×10^{-49}	2.37×10^{-50}	1.05×10^{-126}	**0**
F3	Mean	**0**	9.87×10^{-59}	1.96×10^{-21}	1.09×10^{4}	3.64×10^{-73}	4.52×10^{4}	6.30×10^{-257}	1.66×10^{-236}
	Std	**0**	5.31×10^{-58}	1.03×10^{-20}	7.32×10^{3}	1.99×10^{-72}	1.15×10^{4}	0	**0**
F4	Mean	**0**	1.35×10^{-44}	1.36×10^{-37}	3.90×10^{1}	1.12×10^{-47}	4.34×10^{1}	8.88×10^{-133}	3.35×10^{-192}
	Std	**0**	7.42×10^{-44}	1.54×10^{-37}	1.42×10^{1}	6.14×10^{-47}	3.14×10^{1}	4.83×10^{-132}	**0**
F5	Mean	2.52×10^{1}	2.57×10^{1}	2.60×10^{1}	2.47×10^{4}	$\mathbf{8.64\times10^{-3}}$	2.79×10^{1}	2.51×10^{1}	2.88×10^{1}
	Std	2.13×10^{-1}	2.14×10^{-1}	5.47×10^{-1}	6.23×10^{4}	$\mathbf{1.03\times10^{-2}}$	4.17×10^{-1}	3.21×10^{-1}	4.70×10^{-2}
F6	Mean	$\mathbf{5.38\times10^{-6}}$	7.35×10^{-4}	4.59×10^{-4}	1.25×10^{1}	1.91×10^{-4}	3.94×10^{-1}	1.70×10^{-5}	1.44
	Std	$\mathbf{6.35\times10^{-6}}$	2.13×10^{-3}	7.74×10^{-4}	1.01×10^{1}	2.12×10^{-4}	2.11×10^{-1}	3.95×10^{-5}	6.05×10^{-1}
F7	Mean	$\mathbf{1.10\times10^{-4}}$	1.21×10^{-3}	6.21×10^{-4}	1.48×10^{-1}	1.39×10^{-4}	4.72×10^{-3}	6.44×10^{-4}	1.27×10^{-4}
	Std	1.16×10^{-4}	1.00×10^{-3}	2.64×10^{-4}	1.61×10^{-1}	1.11×10^{-4}	4.98×10^{-3}	5.50×10^{-4}	$\mathbf{9.99\times10^{-5}}$

上的实验结果(最好的结果用粗体显示)。SCDBO 算法在 F8～F11 上均找到了理论最优值,在 F12 上相对于其他对比算法收敛精度更高,在 F13 上也只逊色于 HHO 算法。从表 5 中几种测试函数上的方差来看,DBO 具有更小的方差,展现出了较强的鲁棒性能。SCDBO 算法与对比算法在多峰测试函数上的迭代收敛图如图 5 所示,在 F9～F11 上有很多算法都可以找到最优值,但是所提算法 SCDBO 具有更快的收敛速度,领先于其他对比算法。通过以上分

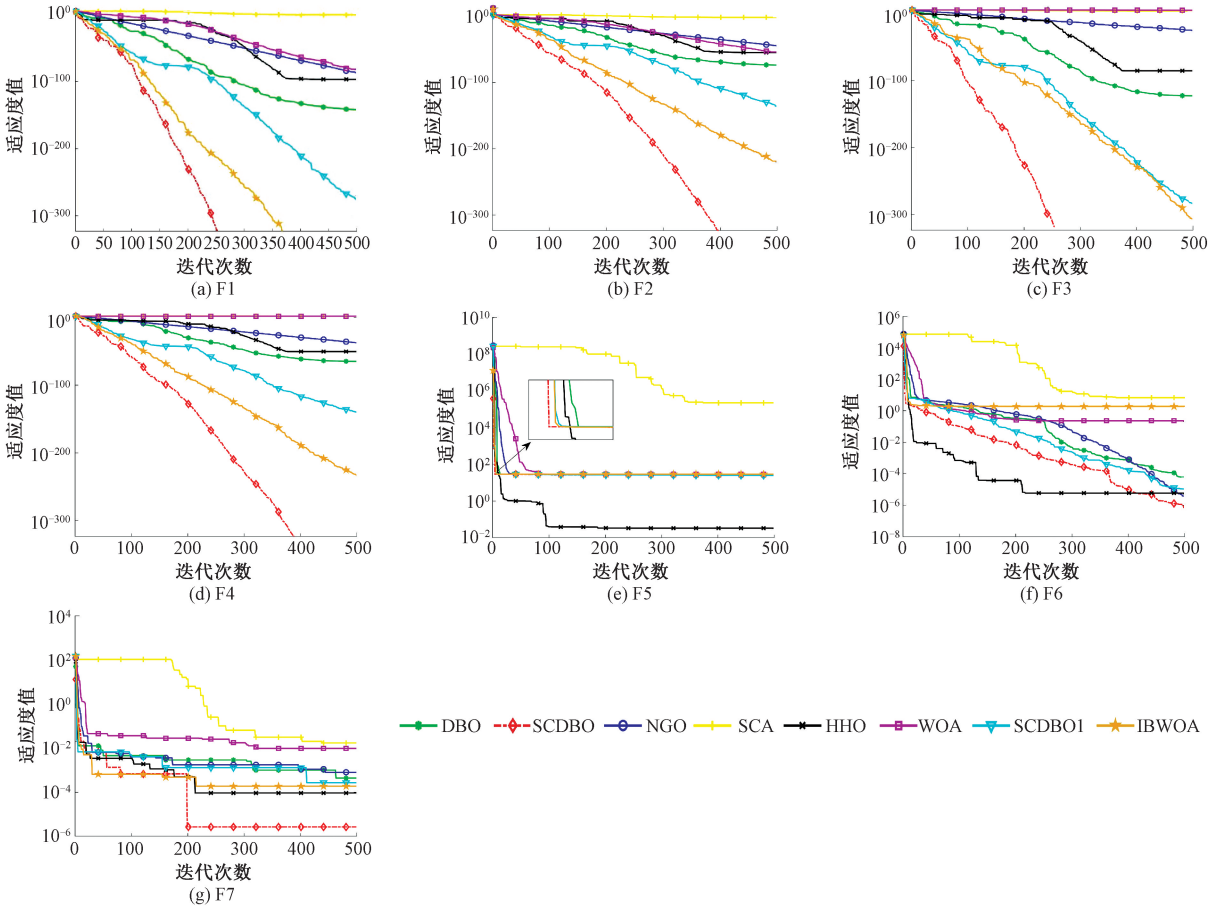

图 4　各算法在单峰基准测试函数上的收敛图

Fig. 4　Convergence plots of the algorithms on unimodal benchmark functions

表 5　各算法在多峰基准测试函数上的实验结果

Table 5　Experimental results of each algorithm on multimodal benchmark functions

	参数	SCDBO	DBO	NGO	SCA	HHO	WOA	SCDBO1	IBWOA
F8	Mean	$\mathbf{-1.26\times10^{4}}$	-8.88×10^{3}	-7.51×10^{3}	-3.75×10^{3}	-1.25×10^{4}	-1.07×10^{4}	-8.04×10^{3}	-1.16×10^{4}
	Std	$\mathbf{7.67\times10^{-1}}$	1.75×10^{3}	4.44×10^{2}	3.11×10^{2}	2.62×10^{2}	1.87×10^{3}	1.00×10^{3}	1.48×10^{3}
F9	Mean	**0**	**0**	**0**	4.00×10^{1}	**0**	1.89×10^{-15}	**0**	**0**
	Std	**0**	**0**	**0**	2.90×10^{1}	**0**	1.04×10^{-14}	**0**	**0**
F10	Mean	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	6.81×10^{-15}	1.22×10^{1}	$\mathbf{8.88\times10^{-16}}$	3.73×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$
	Std	**0**	**0**	1.70×10^{-15}	**0**	**0**	2.70×10^{-15}	1.00×10^{-31}	1.00×10^{-31}
F11	Mean	**0**	**0**	**0**	9.95×10^{-1}	**0**	1.26×10^{-2}	**0**	**0**
	Std	**0**	**0**	**0**	5.89×10^{-1}	**0**	4.89×10^{-2}	**0**	**0**
F12	Mean	$\mathbf{1.48\times10^{-6}}$	2.39×10^{-5}	2.31×10^{-5}	1.82×10^{5}	1.24×10^{-5}	1.53×10^{-2}	8.36×10^{-4}	8.73×10^{-2}
	Std	$\mathbf{3.50\times10^{-6}}$	8.32×10^{-5}	6.43×10^{-5}	6.66×10^{5}	1.60×10^{-5}	1.02×10^{-2}	4.58×10^{-3}	9.98×10^{-2}
F13	Mean	1.17×10^{-2}	5.82×10^{-1}	2.32×10^{-1}	7.75×10^{4}	$\mathbf{9.50\times10^{-5}}$	5.30×10^{-1}	6.83×10^{-1}	1.02
	Std	1.77×10^{-2}	5.67×10^{-1}	1.48×10^{-1}	2.21×10^{5}	$\mathbf{1.30\times10^{-4}}$	2.70×10^{-1}	3.91×10^{-1}	5.91×10^{-1}

析,证明了 SCDBO 算法具有更好的全局探索性能。

3.3　SCDBO 算法在固定维多峰基准测试函数上实验分析

SCDBO 算法及其对比算法在固定维多峰测试函数(如表 3 所示)上的实验结果如表 6 所示(最好的结果用粗体显示)。从表中可以分析出 SCDBO 算法在 F16～F19 以及 F21～F23 上均可以找到最优值,在众多对比算法中,展现出了更高的收敛精度。而对于 F14、F15、F20,虽然 SCDBO 算法没能在众多算法中保持首位,但是所寻最优

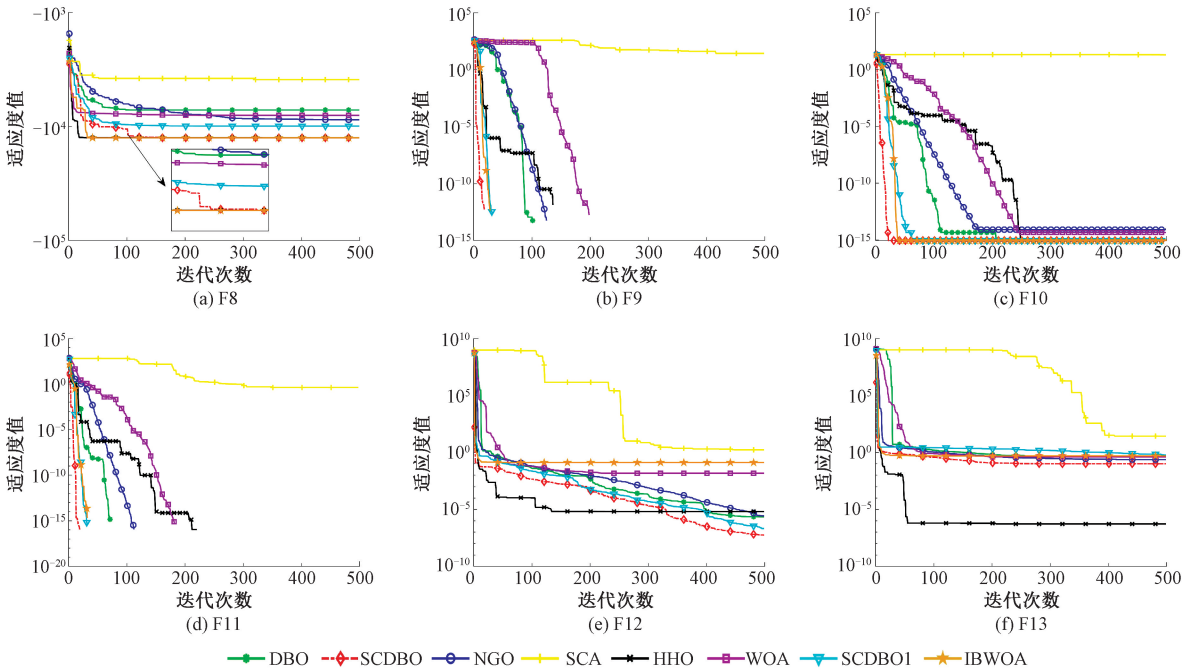

图 5　各算法在多峰基准测试函数上的收敛图

Fig. 5　Convergence plots of the algorithms on multimodal benchmark functions

表 6　各算法在固定维多峰基准测试函数上的实验结果

Table 6　Experimental results of each algorithm on fixed-dimension multimodal benchmark functions

	参数	SCDBO	DBO	NGO	SCA	HHO	WOA	SCDBO1	IBWOA
F14	Mean	1.13	1.32	**9.98E×10^{-1}**	2.15	1.79	2.54	1.10	4.39
	Std	5.03×10^{-1}	1.78	**0**	1.89	1.71	2.90	4.00×10^{-1}	4.18
F15	Mean	4.65×10^{-4}	8.77×10^{-4}	**3.08×10^{-4}**	1.01×10^{-3}	4.53×10^{-4}	7.78×10^{-4}	7.04×10^{-4}	4.48×10^{-4}
	Std	2.63×10^{-4}	4.96×10^{-4}	**2.05×10^{-6}**	3.98×10^{-4}	3.36×10^{-4}	4.50×10^{-4}	4.67×10^{-4}	1.86×10^{-4}
F16	Mean	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**
	Std	**0**	**0**	**0**	4.78×10^{-5}	1.18×10^{-9}	1.84×10^{-9}	**0**	
F17	Mean	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	4.00×10^{-1}	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**
	Std	**0**	3.40×10^{-16}	**0**	3.56×10^{-3}	4.55×10^{-5}	2.65×10^{-5}	1.13×10^{-16}	1.13×10^{-16}
F18	Mean	**3.00**	**3.00**	**3.00**	**3.00**	**3.0**	4.80	**3.00**	5.70
	Std	4.97×10^{-16}	5.56×10^{-16}	**1.81×10^{-16}**	1.32×10^{-4}	6.89×10^{-7}	6.85	4.81×10^{-15}	8.24
F19	Mean	**−3.86**	**−3.86**	**−3.86**	−3.85	**−3.86**	−3.85	**−3.86**	−3.84
	Std	2.00×10^{-3}	3.78×10^{-3}	**2.71×10^{-15}**	2.27×10^{-3}	2.60×10^{-3}	2.58×10^{-2}	1.44×10^{-3}	1.41×10^{-1}
F20	Mean	−3.23	−3.19	**−3.32**	−2.89	−3.09	−3.25	−3.26	−3.15
	Std	6.76×10^{-2}	1.81×10^{-1}	**9.97×10^{-12}**	2.67×10^{-1}	1.06×10^{-1}	1.17×10^{-1}	7.18×10^{-2}	1.64×10^{-1}
F21	Mean	**−10.15**	−7.06	−9.98	−2.04	−5.36	−8.60	−9.41	−8.96
	Std	**5.47×10^{-3}**	2.48	9.31×10^{-1}	1.82	1.17	2.38	2.29	2.47
F22	Mean	**−10.40**	−8.56	−10.18	−3.00	−5.25	−8.05	−9.86	−8.97
	Std	**6.63×10^{-3}**	2.68	1.2	1.71	9.48E−01	2.94	1.84	2.81
F23	Mean	**−10.53**	−8.07	**−10.53**	−4.00	−5.27	−6.99	−9.09	−9.04
	Std	2.08×10^{-3}	2.96	**8.69×10^{-12}**	1.59	8.08×10^{-1}	3.46	3.18	2.88

值仍位于几个算法的前列。在通过分析 SCDBO 算法在各个函数上的方差，不难看出，SCDBO 算法具有更强的稳定性。各个算法在固定维多峰测试函数上的收敛图如图 6 所示，SCDBO 算法在 F21、F22、F23 上的收敛速度略显逊

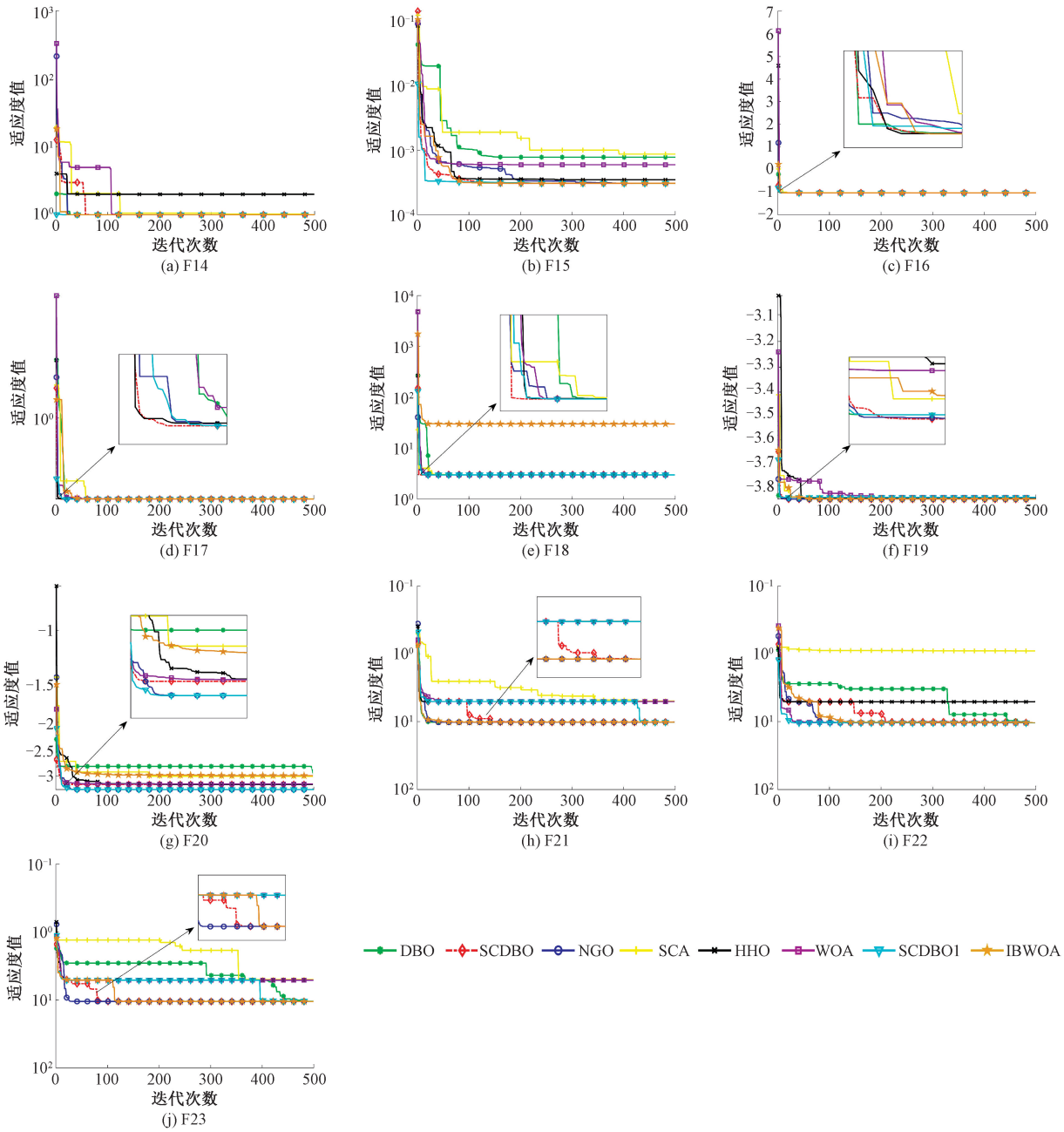

图 6　各算法在固定维多峰基准测试函数上的收敛图

Fig. 6　Convergence plots of the algorithms on fixed-dimension multimodal benchmark functions

色,但是其寻优结果以及寻优稳定性要高于其他对比算法。综上所述,所提算法 SCDBO 算法相对于 DBO 算法具有更好的平衡全局探索和局部开发的能力。

通过上述实验结果的分析,可以看出 SCDBO 算法无论是从寻优精度,收敛速度还是鲁棒性方面,在绝大多数函数寻优方面均领先于其他算法,表明了该算法具有更强的寻优性能。与最新的改进算法相比也表明本文改进算法更具有竞争力。

3.4　Wilcoxon 秩和检验

为了对比 SCDBO 算法与其他算法的差异,选择 Wilcoxon 秩和统计检验对算法进行检验。通过计算 p 值来确定统计结果的差异性,如果 p 值$<$0.05,则认为两种算法存在显著差异。统计结果如表 7 所示,其中 NAN 表示两种算法结果太相似而不显著。由表 7 可知,SCDBO 算法在大多数函数上与其他算法有明显差异。结合表 4～6,可以看出,SCDBO 算法在大多数情况下比其他算法有显著的优势,表现出了极强的寻优性能。

表 7　统计分析结果

Table 7　Results of statistical analysis

函数	DBO	NGO	SCA	HHO	WOA	SCDBO1	IBWOA
F1	1.21×10^{-12}	4.57×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN
F2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	8.42×10^{-3}
F4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F5	2.67×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	6.15×10^{02}	3.02×10^{-11}
F6	4.20×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	2.49×10^{-6}	3.02×10^{-11}	6.52×10^{-1}	3.02×10^{-11}
F7	8.10×10^{-10}	1.96×10^{-10}	3.02×10^{-11}	2.77×10^{-1}	7.39×10^{-11}	6.01×10^{-8}	2.17×10^{-1}
F8	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.76×10^{-1}	4.98×10^{-11}	3.02×10^{-11}	1.02×10^{-6}
F9	NaN	NaN	1.21×10^{-12}	NaN	3.34×10^{-1}	NaN	NaN
F10	NaN	NaN	1.21×10^{-12}	NaN	7.46×10^{-7}	NaN	NaN
F11	NaN	NaN	1.21×10^{-12}	NaN	8.15×10^{-2}	NaN	NaN
F12	1.25×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.09×10^{-6}	3.02×10^{-11}	3.92×10^{-2}	3.02×10^{-11}
F13	7.12×10^{-09}	3.02×10^{-11}	3.02×10^{-11}	2.15×10^{-2}	3.34×10^{-11}	4.98×10^{-11}	3.02×10^{-11}
F14	6.77×10^{-1}	1.54×10^{-2}	1.17×10^{-10}	5.83×10^{-10}	2.76×10^{-10}	6.44×10^{-1}	1.55×10^{-6}
F15	9.78×10^{-5}	1.09×10^{-5}	5.09×10^{-8}	6.15×10^{-2}	4.08×10^{-5}	1.27×10^{-2}	4.92×10^{-1}
F16	NaN	NaN	1.21×10^{-12}	4.57×10^{-12}	1.21×10^{-12}	NaN	NaN
F17	3.34×10^{-1}	NaN	1.21×10^{-12}	4.57×10^{-12}	1.21×10^{-12}	NaN	NaN
F18	6.90×10^{-1}	3.31×10^{-1}	1.41×10^{-11}	4.69×10^{-11}	1.41×10^{-11}	6.07×10^{-1}	1.87×10^{-1}
F19	2.26×10^{-2}	8.15×10^{-2}	4.90×10^{-12}	1.15×10^{-9}	2.63×10^{-10}	3.38×10^{-1}	3.38×10^{-1}
F20	9.04×10^{-1}	4.88×10^{-4}	3.47×10^{-10}	5.15×10^{-8}	9.35×10^{-1}	1.20×10^{-2}	6.84×10^{-3}
F21	8.29×10^{-4}	6.83×10^{-2}	3.01×10^{-11}	3.01×10^{-11}	1.69×10^{-8}	8.94×10^{-1}	1.14×10^{-2}
F22	4.79×10^{-1}	7.81×10^{-2}	2.80×10^{-11}	2.80×10^{-11}	7.34×10^{-99}	3.90×10^{-1}	8.80×10^{-2}
F23	1.53×10^{-1}	2.98×10^{-2}	3.01×10^{-11}	3.01×10^{-11}	6.68×10^{-11}	6.77×10^{-1}	4.15×10^{-1}

4　SCDBO 算法在工程设计中的应用

在本研究中，选择了 3 个典型的工程设计问题——涉及压力容器设计问题[20]，拉伸/压缩弹簧设计问题[21]，三杆桁架设计问题[22]，旨在全面评估 SCDBO 算法在实际问题求解中的优化效能。采用罚函数来处理不等式约束，为了保证公平比较，所有算法，包括 SCDBO 和对比算法，都设置了相同的参数：种群规模固定为 $N=30$，而最大迭代次数设定为 $T_{\max}=1\ 000$。

4.1　压力容器设计问题

压力容器的设计可被视为一个综合优化任务，其核心目标是在确保功能需求的前提下，最小化容器的生产成本。这个问题涉及四个决策参数和四个限制条件。参数涵盖了壳体的厚度(T_s)、封头的厚度(T_h)、容器内部的半径(R)以及不考虑封头的圆柱截面长度(L)。SCDBO 算法和其他对比算法寻优结果如表 8 所示。此问题的数学表述可以归纳为：

$$f(x)=0.622\,4x_1x_3x_4+1.781x_2x_3^2+3.166\,1x_1^2x_4+19.84x_1^2x_3,\tag{17}$$

表 8　压力容器设计问题各算法优化结果

Table 8　Optimization results of each algorithm for the pressure vessel design problem

算法	设计变量				最优成本
	Th	Ts	R	L	
SCDBO	0.778 169	0.384 649	40.319 619	200	5 885.332 8
DBO	0.790 507	0.400 050	40.778 197	193.771 680	5 959.495 1
NGO	0.849 501	0.413 433	42.637 891	170.161 217	6 171.810 0
SCA	0.801 260	0.416 110	41.420 661	187.955 002	6 061.577 4
HHO	0.832 124	0.431 377	42.905 333	168.397 260	6 112.607 1
WOA	0.843 401	0.508 490	43.638 379	158.445 163	6 324.007 0
SCDBO1	0.783 380	0.386 928	40.554 794	196.829 561	5 899.771 9
IBWOA	0.790 291	0.390 621	40.945 189	191.471 146	5 906.652 9

约束条件:

$$\begin{aligned}&g_1(X)=-x_1+0.019\ 3x_3\leqslant 0,\\&g_2(X)=-x_2+0.009\ 54x_3\leqslant 0,\\&g_3(X)=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3+1\ 296\ 000\leqslant 0,\\&g_4(X)=x_4-240\leqslant 0.\end{aligned}\tag{18}$$

变量的取值范围为 $0\leqslant x_1,x_2\leqslant 99,10\leqslant x_3,x_4\leqslant 200$

4.2　拉伸/压缩弹簧设计问题

张力/压缩弹簧的设计问题是通过选择合适的参数使张/压弹的重量最小,该问题中包含 4 个约束条件和 3 个参数变量。变量为:导线直径 d、平均线圈直径 D 和活动线圈数 N。SCDBO 算法及其对比算法计算结果如表 9 所示。该问题的数学模型表示如下:

$$\min f(x_1,x_2,x_3)=(x_3+2)x_1^2x_2\tag{19}$$

约束条件为:

$$\begin{aligned}&g_1(X)=1-\frac{x_2^3x_3}{71\ 785x_1^4}\leqslant 0,\\&g_2(X)=\frac{x_2(4x_2-x_1)}{12\ 566x_1^3(x_2-x_1)}+\frac{1}{5\ 108x_1^2}-1\leqslant 0,\\&g_3(X)=1-\frac{140.45x_1}{x_2^2x_3}\leqslant 0,\\&g_4(X)=\frac{2(x_1+x_2)}{3}-1\leqslant 0.\end{aligned}\tag{20}$$

变量的范围为 $0.05\leqslant x_1\leqslant 2$, $0.25\leqslant x_2\leqslant 1.3$, 和 $2.0\leqslant x_3\leqslant 15.0$.

表 9　拉伸/压缩弹簧设计问题各算法优化结果

Table 9　Optimization results of each algorithm for the tension/compression spring design problem

算法	设计变量			最优成本
	w	d	L	
SCDBO	0.051 897 32	0.361 748 67	11.377 592 06	0.012 666 02
DBO	0.050 111 26	0.319 272 78	14.206 045 81	0.012 827 81
NGO	0.052 680 56	0.380 979 85	10.091 926 28	0.012 687 73
SCA	0.050 650 89	0.331 340 55	12.504 359 04	0.012 750 88
HHO	0.052 675 13	0.380 906 75	9.822 828 96	0.012 682 68
WOA	0.050 599 53	0.330 910 31	12.500 000 01	0.012 708 51
SCDBO1	0.051 960 18	0.362 771 94	10.896 230 91	0.012 732 64
IBWOA	0.052 675 13	0.380 906 75	10.379 250 00	0.012 682 68

4.3　三杆桁架设计问题

三杆桁架设计问题是为了找到桁架杆件最合适的截面组合使得三杆桁架的体积最小。表 10 给出了算法的统计优化结果。该问题的数学模型表示如下:

$$\min f(x)=[A1,A2]=(2\sqrt{2}x_1+x_2)\times l\tag{21}$$

$$\begin{aligned}&g_1(x)=\frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0\\&g_2(x)=\frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0\ \text{where}\ l=100\ \text{cm},\\&P=2\ \frac{\text{KN}}{\text{cm}^2},\sigma=2\ \frac{\text{kN}}{\text{cm}^2}.\\&g_3(x)=\frac{1}{\sqrt{2}x_2+x_1}P-\sigma\leqslant 0\end{aligned}\tag{22}$$

其中 $0\leqslant x_1,x_2\leqslant 1$。

由表 8～10 可知,SCDBO 算法在 3 种工程设计问题上所得到的优化成本比其他对比算法更好,充分验证了算法在工程实践问题上具有较高的潜力。

表 10　三杆桁架设计问题各算法优化结果

Table 10　Optimization results of each algorithm for the three-bar truss design problem

算法	设计变量		优化成本
	A1	A2	
SCDBO	0.788 639 96	0.408 347 79	263.895 844 33
DBO	0.788 573 94	0.408 564 63	263.898 853 80
NGO	0.789 188 83	0.406 798 08	263.896 115 92
SCA	0.786 695 73	0.413 983 63	263.909 517 77
HHO	0.789 447 56	0.406 067 91	263.896 281 26
WOA	0.788 215 82	0.409 549 01	263.895 999 82
SCDBO1	0.789 283 11	0.406 546 03	263.897 580 12
IBWOA	0.787 849 97	0.410 587 22	263.896 345 18

5　结　　论

本文提出了一种创新的 SCDBO 算法,核心在于采用改进的正余弦算法对滚球蜣螂的位置进行更新,以此强化算法的全局搜寻能力。另外,通过引入 Logistic-tent 混沌映射进行初始种群配置,有效增强了种群的多样性。再

者，结合自适应 t 分布扰动和自适应 Levy-柯西变异策略，显著增强了算法逃离局部最优解的能力。这些策略的巧妙整合，使得算法在全局探索和局部开发之间达到良好的平衡，从而全面提高了其求解优化问题的性能。

在与多种算法在 23 个标准函数上的广泛对比分析中，SCDBO 方法证实了其在收敛速度和寻优能力等方面的优越性。经 Wilcoxon 秩和检验，该算法在大多数测试函数上与其他对比算法有明显的差异。进一步地，本文把 SCDBO 算法应用于 3 个实际工程设计场景，实验结果显示，提出的算法在工程实践中表现出极高的实用性。这些综合实验数据有力地证明了 SCDBO 算法的杰出性能。

基于此，本文预计将该算法应用到神经网络以及更多的实际应用当中，并尝试与其他算法相结合，得到更有效地算法。

参考文献

[1] ALHIJAWI B, AWAJAN A. Genetic algorithms: Theory, genetic operators, solutions, and applications [J]. Evolutionary Intelligence, 2024, 17(3): 1245-1256.

[2] MIRJALILI S, LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[3] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

[4] FARIS H, ALJARAH I, AL-BETAR M A, et al. Grey wolf optimizer: A review of recent variants and applications[J]. Neural Computing and Applications, 2018, 30: 413-435.

[5] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: Algorithm and applications [J]. Future Generation Computer Systems, 2019, 97: 849-872.

[6] MIRJALILI S. SCA: A sine cosine algorithm for solving optimization problems [J]. Knowledge-based Systems, 2016, 96: 120-133.

[7] ADAM S P, ALEXANDROPOULOS S A N, PARDALOS P M, et al. No free lunch theorem: A review [J]. Approximation and Optimization: Algorithms, Complexity and Applications, 2019: 57-82.

[8] 王天雷，张绮媚，李俊辉，等. 基于正交对立学习的改进麻雀搜索算法[J]. 电子测量技术，2022，45(10)：57-66.
WANG T L, ZHANG Q M, LI J H, et al. Improved sparrow search algorithm based on orthogonal dyadic learning [J]. Electronic Measurement Technology, 2022, 45(10): 57-66.

[9] 朱学敏，刘升，朱学林，等. 多策略混合改进的海洋捕食者算法及其工程应用[J]. 国外电子测量技术，2023，42(5)：125-134.
ZHU X M, LIU S, ZHU X L, et al. Multi-strategy hybrid improved marine predator algorithm and its engineering applications [J]. Foreign Electronic Measurement Technology, 2023, 42(5): 125-134.

[10] DENG H, LIU L, FANG J, et al. A novel improved whale optimization algorithm for optimization problems with multi-strategy and hybrid algorithm [J]. Mathematics and Computers in Simulation, 2023, 205: 794-817.

[11] GHAREHCHOPOGH F S, ABDOLLAHZADEH B. An efficient harris hawk optimization algorithm for solving the travelling salesman problem[J]. Cluster Computing, 2022, 25(3): 1981-2005.

[12] XUE J, SHEN B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization[J]. The Journal of Supercomputing, 2023, 79 (7): 7305-7336.

[13] 万怡华，张雪梅. 混合多策略改进蜣螂算法的避障路径规划[J]. 电子测量技术，2024，47(2)：69-78.
WAN Y H, ZHANG X M. Hybrid multi-strategy improved dung beetle algorithm for obstacle avoidance path planning [J]. Electronic Measurement Technology, 2024, 47(2): 69-78.

[14] WANG X, WEI Y, GUO Z, et al. A sinh-cosh-enhanced DBO algorithm applied to global optimization problems[J]. Biomimetics, 2024, 9(5): 271.

[15] GUPTA M, GUPTA K K, KHOSRAVI M R, et al. An intelligent session key-based hybrid lightweight image encryption algorithm using logistic-tent map and crossover operator for internet of multimedia things[J]. Wireless Personal Communications, 2021, 121(3): 1857-1878.

[16] SHARMA H, BANSAL J C, ARYA K V, et al. Lévy flight artificial bee colony algorithm [J]. International Journal of Systems Science, 2016, 47(11): 2652-2670.

[17] 郑婷婷，刘升，叶旭. 自适应 t 分布与动态边界策略改进的算术优化算法[J]. 计算机应用研究，2022，39(5)：1410-1414.
ZHENG T T, LIU SH, YE X. Arithmetic

optimization algorithm for adaptive t-distribution with dynamic boundary policy improvement [J]. Application Research of Computers Research,2022,39(5):1410-1414.

[18] DEHGHANI M, HUBÁLOVSKY Š, TROJOVSKY P. Northern goshawk optimization: A new swarm-based algorithm for solving optimization problems[J]. Ieee Access, 2021, 9: 162059-162080.

[19] WANG Y, HE Q, ZHANG D, et al. Improving Li-ion battery health: Predicting remaining useful life using IWBOA-ELM algorithm[J]. Journal of Energy Storage, 2023, 72: 108547.

[20] 潘劲成,李少波,周鹏,等.改进正弦算法引导的蜣螂优化算法[J].计算机工程与应用,2023,59(22):92-110.
PAN J CH, LI SH B, ZHOU P, et al. Dung beetle optimization algorithm guided by improved sinusoidal algorithm[J]. Computer Engineering and Applications, 2023,59(22):92-110.

[21] 梁昔明,李星,龙文.融合鲸鱼算法的混合灰狼优化算法[J].数学的实践与认识,2022,52(6):130-138.
LIANG X M, LI X, LONG W. A hybrid gray wolf optimization algorithm incorporating the whale algorithm[J]. Mathematics in Practice and Theory, 2022,52(6):130-138.

[22] ZHANG J, CHENG X, ZHAO M, et al. ISSWOA: Hybrid algorithm for function optimization and engineering problems [J]. The Journal of Supercomputing, 2023, 79(8): 8789-884.

作者简介

刘微(通信作者),博士,教授,主要研究方向为智能优化算法及其应用。

E-mail:LiuWei19781020@126.com