Computer Science and Application 计算机科学与应用, 2025, 15(4), 70-79
Published Online April 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.154079

Hans汉斯

基于精英反向学习策略的改进蜣螂优化算法

谷喜阳[1]，谢颖华[1]，杨　宁[2]

[1]东华大学信息科学与技术学院，上海
[2]山西大同大学计算机网络与工程学院，山西 大同

收稿日期：2025年3月1日；录用日期：2025年3月31日；发布日期：2025年4月8日

摘　要

为了提高蜣螂优化算法(DBO)全局搜索过程中的种群多样性和避免陷入局部最优的风险，利用混沌理论和精英反向学习策略提出了一种基于精英反向学习策略的改进蜣螂优化算法(EoDBO)。首先，在蜣螂初始化种群个体位置时引入Sinusoidal map混沌映射策略，以提高寻优前蜣螂种群整体质量，利于加快全局搜索速度；其次，在算法后期采用精英反向学习策略，对部分较优的蜣螂位置进行扰动以调高算法的局部开发能力。利用12个国际基准测试函数测试改进算法的性能，并于DBO算法、麻雀搜索算法(SSA)、灰狼优化算法(GWO)、鲸鱼优化算法(WOA)、黑猩猩优化算法(ChOA)进行对比分析，实验表明EoDBO算法在收敛精度和算法的稳定性方面均变现更优，且收敛速度更快。

关键词

蜣螂优化算法，局部最优，混沌映射，精英反向学习策略，群智能优化算法

Improved Dung Beetle Optimizer Based on Elite Opposition Learning Strategy

Xiyang Gu[1], Yinghua Xie[1], Ning Yang[2]

[1]College of Information Science and Technology, Donghua University, Shanghai
[2]School of Computer and Network Engineering, Shanxi Datong University, Datong Shanxi

Received: Mar. 1st, 2025; accepted: Mar. 31st, 2025; published: Apr. 8th, 2025

Abstract

In order to improve the population diversity and avoid the risk of falling into local optimum during the global search of Dung Beetle Optimizer (DBO), an improved Dung Beetle Optimizer based on Elite Opposition Learning Strategy (EoDBO) is proposed using chaos theory and elite opposition learning strategy. First, a Sinusoidal map chaotic mapping strategy is introduced in the initialization

文章引用: 谷喜阳, 谢颖华, 杨宁. 基于精英反向学习策略的改进蜣螂优化算法[J]. 计算机科学与应用, 2025, 15(4): 70-79. DOI: 10.12677/csa.2025.154079

of individual population positions of dung beetles to improve the overall quality of the dung beetle population before the search for optimality and to facilitate faster global search; second, an elite opposition learning strategy is used in the later stage of the algorithm to perturb some of the better dung beetle positions to tune up the local exploitation capability of the algorithm. The performance of the improved algorithm is tested using 12 international benchmarking functions and compared with DBO algorithm, Sparrow Search Algorithm (SSA), Gray Wolf Optimization Algorithm (GWO), Whale Optimization Algorithm (WOA), and Chimpanzee Optimization Algorithm (ChOA) for analysis. The experiments show that the EoDBO algorithm turns out to be superior in terms of convergence accuracy and stability of the algorithm, and converges faster.

Keywords

Dung Beetle Optimizer, Local Optimum, Chaotic Mapping, Elite Opposition Learning Strategy, Swarm Intelligence Optimization Algorithm

1. 引言

优化问题普遍存在于社会科学、工程和经济等各个领域中，群智能(Swarm Intelligent, SI)优化算法作为求解优化问题的有效工具且因其寻优能力强、操作简单等特点近些年来备受学者们的关注[1]-[5]。Kennedy 等人[6]通过对鸟群捕食行为的研究提出了一种粒子群优化算法(Partical Swarm Optimization, PSO)；针对萤火虫和萤火虫群的行为，Krishnanand 等人[7]提出了一种萤火虫优化算法(Glowworm Swarm Optimization, GSO)；Yang 等人[8]通过研究布谷鸟独特的寄生方式以养育幼鸟的行为提出了布谷鸟搜索算法(Cuckoo Search Via Vévy Flights, CS)；2014 年，Mirjalili 等人[9]通过模拟狼群中头狼引导群体捕食的行为提出了灰狼优化算法(Grey Wolf Optimizer, GWO)；2016 年，Mirjalili 等人[10]通过模拟自然界中座头鲸群体狩猎行为提出了鲸鱼优化算法(Whale Optimization Algorithm, WOA)；Khishe 等人[11]通过模拟攻击黑猩猩、驱赶黑猩猩、拦截黑猩猩和追逐黑猩猩的协同狩猎行为提出了黑猩猩优化算法(Chimp Optimization Algorithm, ChOA)；针对麻雀的觅食行为和反捕食行为，Xue 等人[12]在 2020 年提出了麻雀搜索算法(Sparrow Search Algorithm, SSA)。

为了提升算法的性能，不断有学者提出改进的 SI 算法。王乐洋等人[13]通过采用分段惯性因子调整粒子速度，修改局部和全局最优解的加速因子从而降低算法后期陷入局部最优的风险(Dynamic Particle Swarm Optimization, DPSO)；张文胜等人[14]提出一种自适应递减的收敛因子，并在灰狼位置更新公式引入惯性权重从而协调算法的全局领导和开发能力；吴泽忠等人[15]在初始化种群时采用反向学习策略及随机调整控制参数策略，利用正态变异算子与改进螺旋更新位置对鲸鱼种群进行干扰从而提高全局搜索速度(Improved Whale Optimization Algorithm, ImWOA)；兰周新等人[16]针对 ChOA 算法在寻优过程中求解精度低、收敛速度慢以及易陷入局部极值点的问题，提出了一种新型的柯西扰动黑猩猩优化算法(CP-ChOA)；钱敏等人[17]利用反向学习策略和混沌理论提出了一种改进的麻雀搜索算法(Improved Sparrow Search Algorithm, ISSA)，该算法在收敛精度、算法稳定性以及收敛速度上存在优势。

蜣螂优化算法(Dung Beetle Optimizer, DBO)是 2022 年 Xue 等人[18]提出的一种新的 SI 优化算法，该算法通过模拟蜣螂的生物活动来进行寻优。由于算法同时考虑了全局搜索和局部搜索，因此与其他

现有算法相比，DBO 算法在求解精度、收敛速度和鲁棒性上均具有较好的性能。本文在 DBO 算法的基础上提出了新的改进方案——基于精英反向学习策略的改进蜣螂优化算法(Improved Dung Beetle Optimizer based on Elite Opposition Learning Strategy, EoDBO)，在 DBO 算法搜索前期，利用混沌理论丰富种群多样性加快全局收敛速度，后期引入精英反向学习策略进行干扰以降低局部最优的风险。通过与其余五种 SI 算法相比，改进的在求解 12 个基本测试函数上寻优性能更好，且能够有效提高算法的收敛精度及稳定性。

2. 蜣螂优化算法

DBO 算法受自然界中蜣螂生物活动的影响，将蜣螂种群的生物活动抽象为五种过程：滚球、跳舞、繁殖、觅食和偷窃。蜣螂会利用天体线索，如阳光、月光和偏振光进行导航，使其所滚的粪球沿着直线滚动。一旦失去光源，蜣螂的路径就不再是直线，而是弯曲的，有时甚至是圆形。除了光源外，许多自然因素也会导致蜣螂偏离原来的方向，如风或者不平坦的地面。当蜣螂在滚球的过程中遇到障碍物无法前进时，则会爬到粪球上面跳舞(包括一系列的旋转和停顿)来决定它们前进的方向。因此，蜣螂的滚球行为可以描述为：

$$\begin{aligned} x_i(t+1) &= x_i(t) + \alpha \times k \times x_i(t-1) + b \times \Delta x \\ \Delta x &= \left| x_i(t) - x^w \right| \end{aligned} \tag{1}$$

式中，t 表示当前迭代次数，$x_i(t)$ 表示第 t 次迭代时第 i 只蜣螂的位置信息，$k \in (0, 0.2]$ 是一个常量表示偏转系数，$b \in (0, 1)$，α 是自然系数，取 1 或者−1，x^w 是全局最差位置，Δx 用来模拟光强的变化。

当蜣螂遇到障碍物无法前进时，则通过跳舞的方式决定新的方向，此时蜣螂滚球的位置按式(2)进行更新：

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

式中，$\theta \in [0, \pi]$ 是偏转角。

蜣螂所获得的粪球主要有两个目的，第一个目的是将所获得的粪球用于产卵和养育下一代。为了给下一代提供一个安全的环境，将雌性蜣螂的产卵区域定义如下：

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

式中，X^* 表示当前局部最优位置，Lb^* 和 Ub^* 分别表示产卵区域的下边界和上边界。$R = 1 - t/T_{\max}$，$T_{\max}$ 表示最大迭代次数，Lb 和 Ub 分别表示优化问题的下边界和上边界。

一旦确定了产卵区域，雌性蜣螂就会在该区域进行产卵，假设每个雌性蜣螂在每次迭代过程中只会下一个卵，则卵球在每次迭代过程中，其位置更新如下式所示：

$$B_i(t+1) = X^* + b_1 \times \left(B_i(t) - Lb^*\right) + b_2 \times \left(B_i(t) - Ub^*\right) \tag{4}$$

式中，$B_i(t)$ 是第 i 个卵球在第 t 次迭代的位置信息，b_1 和 b_2 是两个独立的大小为 $1 \times D$ 的随机变量，D 是优化问题中的维度。小蜣螂会从地下钻出来进行觅食，定义最佳觅食区域的边界如下：

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

式中，X^b 表示全局最优位置，Lb^b 和 Ub^b 分别表示最佳觅食区域的下边界和上边界。小蜣螂的位置更新如下式所示：

$$x_i(t+1)=x_i(t)+C_1\times\left(x_i(t)-Lb^b\right)+C_2\times\left(x_i(t)-Ub^b\right) \tag{6}$$

式中，$x_i(t)$ 表示第 i 个小蜣螂第 t 次迭代的位置信息，C_1 表示服从正太分布的随机数，C_2 是 $(0,\ 1)$ 范围内的随机向量。还有一定比例的蜣螂会从其他蜣螂哪里偷窃粪球，因此，进行偷窃的蜣螂的位置更新如下所示：

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

式中，$x_i(t)$ 表示第 i 个小偷第 t 次迭代的位置信息，g 是大小为 $1\times D$ 的服从正太分布的随机变量，S 是常量。

3. 改进的蜣螂优化算法

3.1. 基于 Sinusoidal Map 混沌映射的种群初始化

混沌系统具有随机性、遍历性，因此将混沌理论和 SI 算法结合是改进优化算法的一种思路，研究表明其能有效逃离局部和提高收敛精度[19]-[22]。种群初始化的质量好坏是影响全局寻优快慢的重要因素，因此本文采用 Sinusoidal map 混沌映射[23]对种群进行初始化，这样既不改变 DBO 算法初始化时多具有的随机性本质，又利用混沌原理提高了种群的多样性和搜索的遍历性，在产生大量的蜣螂中欧给那群基础上，从中择优出初始群体。Sinusoidal 映射是混沌映射的典型代表，其表达式如下所示：

$$x_{k+1}=ax_k^2\sin\left(\pi x_k\right) \tag{8}$$

式中，x_k 为第 k 次迭代的位置信息，迭代过程中，$a=2.3$，$x_0=0.7$。

3.2. 精英反向学习策略

精英反向学习策略是 Tizhoosh [24]提出的一种反向解，使其比当前解更接近于全局最优解的一种策略。主要原理是对问题的可行解求其反向解，并对原始解和反向解进行排序，从中选出较优解作为新一代个体进行下一次迭代。该策略可以增加种群多样性以及避免早熟现象。

定义 1：反向解[25]：设在区间 $[a,\ b]$ 上存在一个实数 x，将实数 x 的反向数定义为 $x'=a+b-x$。假设在 R 域上存在 N 维点 $p=(x_1,\ x_2,\cdots,\ x_i,\cdots,\ x_N)$，并且 $x_i\in[a_i,\ b_i]$，则定义 $p'=(x_1',\ x_2',\cdots,\ x_i',\cdots,\ x_N')$ 为 p 的反向点。其中，$x_i'=k(a_i+b_i)-x_i$，k 为区间 $[0,\ 1]$ 分布均匀的随机数，称作一般化系数。

定义 2：基于反向解的优化[25]：设待优化问题为最小问题，适应度函数为 f，若存在某个可行解 x，其反向解为 x'，若 $f(x')<f(x)$ 成立，则用 x'替换 x。

定义 3：精英反向解[25]：设在某维空间中，$x_{best}'=(x_1',\ x_2',\cdots,\ x_i',\cdots,\ x_N')$ 为当前群体的精英个体 $x_{best}=(x_1,\ x_2,\cdots,\ x_i,\cdots,\ x_N)$ 的反向解，该方向解定义为 $x_i'=k(a_i+b_i)-x_i$，$x_i\in[a_i,\ b_i]$，$k\in[0,\ 1]$ 为服从均匀分布的随机数，利用该系数可以生成精英个体的多个反向解。

精英反向学习(Elite Opposition-Based Learning, EOBL)是通过当前问题的可行解构造其反向解以此来增加种群多样性，本文将 DBO 算法中融入精英反向学习策略，在 DBO 的每一个迭代过程中，针对精英个体执行反向学习，生成精英个体的反向种群，参与进化竞争。

3.3. 改进的蜣螂优化算法流程

本文提出的基于精英反向学习策略的改进蜣螂优化算法(EoDBO)步骤如下：

1) 对种群进行初始化，并进行相关参数设置，如种群数量，最大迭代次数，当前迭代次数，滚球蜣螂数量、卵球数量、小蜣螂数量和偷窃蜣螂数量。

2) 采用 Sinusoidal 混沌映射随机生成蜣螂个体位置，计算蜣螂个体适应度值，记录当前全局最优和

最劣值，及其对应位置。

3) 根据公式(1)、(2)对滚球蜣螂的位置进行更新，并计算相应的适应度值，记录当前局部最优值和最优位置。

4) 根据公式(3)计算蜣螂产卵区域，并根据公式(4)对卵球位置进行更新。

5) 根据公式(5)计算小蜣螂的觅食区域，并根据公式(6)对小蜣螂的位置进行更新。

6) 根据公式(7)对偷窃蜣螂的位置进行更新。

7) 更新蜣螂种群个体最优位置和全局最优位置。

8) 对蜣螂种群个体实施精英反向学习策略，更新蜣螂位置，得到新的蜣螂个体最优位置和全局最优位置。

9) 判断当前迭代次数是否满足算法最大迭代次数，若满足，则循环结束，输出结果；否则返回步骤(3)。

4. 仿真实验与结果分析

为验证本文所提出的 EoDBO 算法的可行性和寻优能力，本文基于 Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz，16 GB 内存，Windows 10 操作系统和仿真软件 MATLAB R2021a 进行仿真实验。

4.1. 测试函数及对比函数

为验证 EoDBO 算法的寻优能力，本文将它用于求解 12 个国际上通用的标准测试函数，并与基础 DBO 算法、SSA 算法、GWO 算法、WOA 算法、ChOA 算法进行实验比较。测试函数见表 1 所示，其中 F_1~F_7 是单峰测试基准函数，F_8~F_{12} 是多峰测试基准函数。单峰函数用来衡量算法的局部寻优开发能力，多峰函数衡量全局搜索与局部开发的平衡性能。

Table 1. Standard test function (dimension n=30)
表 1. 标准测试函数(维度 n = 30)

测试函数	范围	最小值
$F_1(x)=\sum_{i=1}^{n}x_i^2$	[−100, 100]	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[−10, 10]	0
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	[−100, 100]	0
$F_4(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	[−100, 100]	0
$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	[−30, 30]	0
$F_6(x)=\sum_{i=1}^{n}\left(\left[x_i+0.5\right]\right)^2$	[−100, 100]	0
$F_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	[−1.28, 1.28]	0
$F_8(x)=-\sum_{i=1}^{n}\left(x_i\sin\sqrt{\lvert x_i\rvert}\right)$	[−500, 500]	−12569.5
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	[−5.12, 5.12]	0

续表

$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32, 32]$	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600, 600]$	0
$F_{12}(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i, 10, 100, 4)$ $y_i=1+\frac{x_i}{4}$ $u(x_i, a, k, m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<a\end{cases}$	$[-50, 50]$	0

4.2. 实验参数设置

Table 2. Comparison of algorithm optimization results
表 2. 算法优化结果对比

函数	名称	EoDBO	DBO	SSA	GWO	WOA	ChOA
F1	平均值	0	1.23E-109	8.96E-74	1.57E-27	3.85E-74	1.26E-07
	标准差	0	6.75E-109	4.91E-73	3.50E-27	1.91E-73	2.48E-06
F2	平均值	3.84E-153	3.62E-57	1.95E-40	1.12E-16	5.73E-52	3.76E-06
	标准差	2.10E-152	1.67E-56	1.07E-39	1.13E-16	1.19E-51	5.56E-06
F3	平均值	0	1.41E-45	1.32E-70	5.05E-06	47090.4	158.232
	标准差	0	7.72E-45	7.20E-70	8.95E-06	12966.1	479.476
F4	平均值	8.38E-353	2.59E-42	1.36E-54	6.58E-07	54.5600	0.21946
	标准差	0	1.41E-41	7.43E-54	5.71E-07	26.8119	0.36133
F5	平均值	25.01515	25.73096	0.01640	27.2492	27.8623	28.7841
	标准差	3.64473	0.19397	0.02388	0.68983	0.45879	0.31544
F6	平均值	6.84E-05	0.00938	0.00013	0.86759	0.43450	3.85132
	标准差	9.24E-05	0.04345	9.45E-05	0.44661	0.21278	0.44380
F7	平均值	0.00076	0.00131	0.00055	0.00172	0.00372	0.00210
	标准差	0.00068	0.00108	0.00036	0.00072	0.00461	0.00227
F8	平均值	−11231.7	−9831.68	−8009.45	−6101.95	−10247.2	−5708.48
	标准差	1271.27	1932.27	2316.39	690.233	1748.23	61.2739
F9	平均值	0	0.16867	0	2.19325	1.89E-15	4.44135
	标准差	0	0.92383	0	3.48915	1.04E-14	5.57871
F10	平均值	1.01E-15	8.88E-16	8.88E-16	1.03E-13	4.56E-15	19.9624
	标准差	6.48E-16	0	0	1.72E-14	3.43E-15	0.00144
F11	平均值	0	0	0	0.00334	1.08E-02	0.01339
	标准差	0	0	0	0.00636	4.14E-02	0.02158
F12	平均值	5.99E-05	3.86E-03	3.31E-05	0.04694	0.02188	0.58261
	标准差	2.65E-04	1.89E-02	2.80E-05	0.02095	0.01950	0.26338

为确保实验的公平性，将全部算法统一设置迭代次数为 500，种群数量为 30。本文实验中 EoDBO 算法与其余五种算法各运行 30 次，分别计算平均值和标准差并把结果保存。算法的优化性能的好坏采用平均值和标准差去衡量。平均值的大小代表了算法的收敛速度的快慢，算法的鲁棒性的优劣则由标准差的大小衡量，实验结果见表 2。

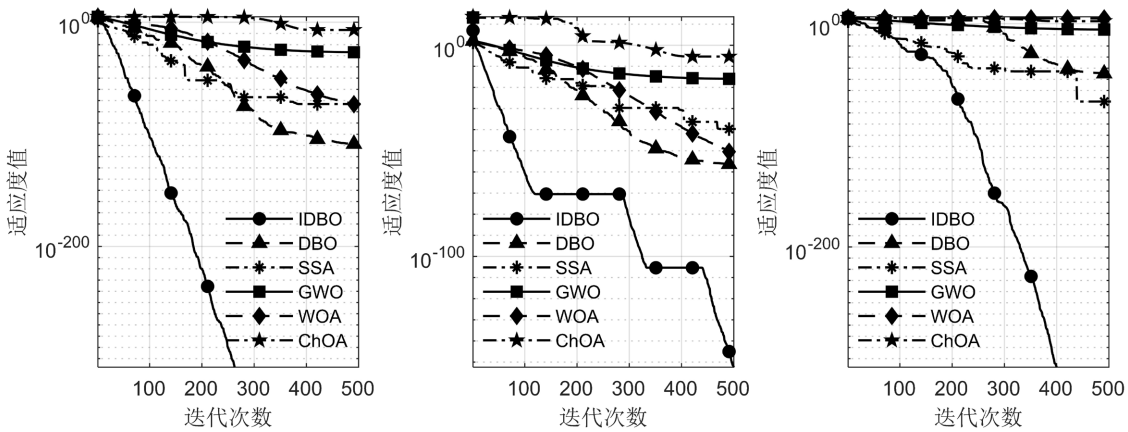
4.3. 实验分析

由表 2 可知，在测试单峰函数 F1~F4 时，EoDBO 算法相较于其余五种算法明显具有更好的收敛精度和更强的鲁棒性。在测试单峰函数 F5、F6 时，EoDBO 的收敛精度和稳定性要弱于 SSA 算法，且与其余四种算法相比，EoDBO 算法优势也并不明显。在测试单峰函数 F7，EoDBO 和 SSA 算法在收敛精度和稳定性上基本相当，且优于其余四种算法。根据上述分析可得，EoDBO 算法能有效增强 DBO 算法的寻优能力与局部开发能力，并有效提高了 DBO 算法的局部收敛精度。

多峰基准测试函数含有多个局部高峰，优化算法求解时极易陷入局部最优，因此多峰基准测试函数常用于测试算法逃离局部最优的能力。在求解多峰函数 F8 时，ChOA 算法和 GWO 算法在求解精度上明显弱于其余算法，且 EoDBO 算法、SSA 算法的求解精度最高。在求解函数 F9、F11 时，EoDBO 算法、SSA 算法和 WOA 算法均具有良好的求解精度。在求解函数 F10 时，EoDBO 算法、SSA 算法和 DBO 算法具有良好的求解精度。在求解函数 F12 时，EoDBO 算法、SSA 算法具有较好的求解精度。通过对比可以发现，在求解多峰函数时，EoDBO 算法和 SSA 算法相较于其他四种算法表现更优，能更好地降低陷入局部最优的风险从而提高整体搜索速度。

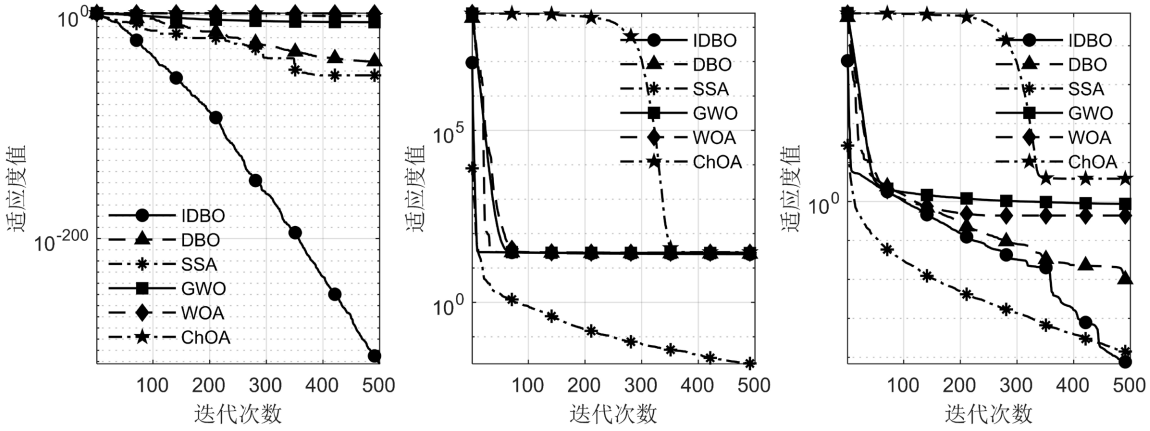
为了更直观显示实验对比分析结果，本文给出 EoDBO 算法与其余五种算法在相同实验条件下得到的函数仿真曲线，见图 1~4。

由图 1~3(a)可以看到，在求解单峰函数 F1~F4 时，EoDBO 算法的求解速度明显优于其余五种算法。在求解函数 F5 时，EoDBO 算法和 SSA 算法的求解速度相当，但是 SSA 算法的精度要优于 EoDBO 算法。在求解 F6、F7 时，EoDBO 算法和 SSA 算法在求解速度和精度上基本相当，且优于其他四种算法。由图 3(b)、图 3(c)和图 4 可以看到，在求解多峰函数 F8 时，EoDBO 算法和 SSA 算法在求解速度和求解精度上基本相当，且优于其余四种算法。在求解函数 F9~F11，EoDBO 算法在求解速度上相较于其余五种算法具有明显优势。在求解函数 F12 时，EoDBO 算法、SSA 算法在求解速度和求解精度上基本相当，且优于其余四种算法。



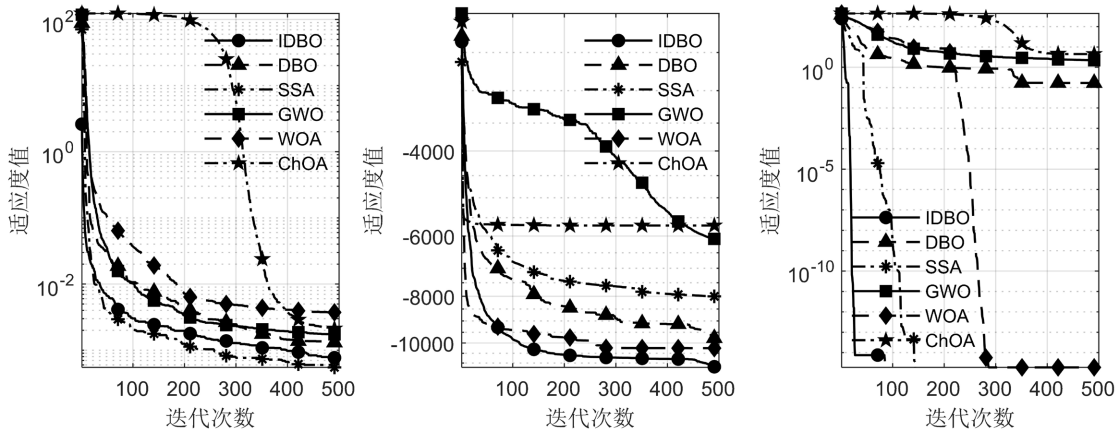
(a) 函数 F1 收敛曲线　(b) 函数 F2 收敛曲线　(c) 函数 F3 收敛曲线

Figure 1. Convergence curves of benchmark functions F1~F3
图 1. 基准函数 F1~F3 收敛曲线



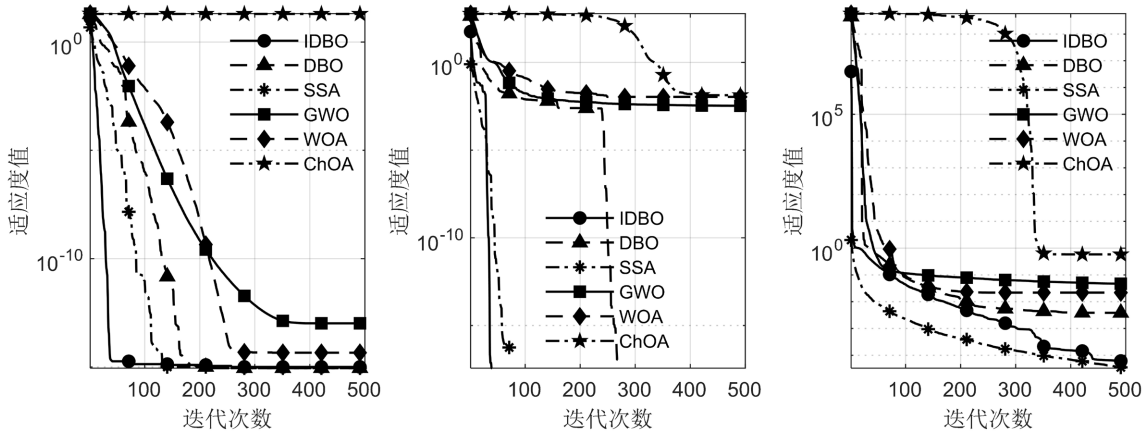
(a) 函数 F4 收敛曲线　(b) 函数 F5 收敛曲线　(c) 函数 F6 收敛曲线

Figure 2. Convergence curves of benchmark functions F4~F6
图 2. 基准函数 F4~F6 收敛曲线



(a) 函数 F7 收敛曲线　(b) 函数 F8 收敛曲线　(c) 函数 F9 收敛曲线

Figure 3. Convergence curves of benchmark functions F7~F9
图 3. 基准函数 F7~F9 收敛曲线



(a) 函数 F10 收敛曲线　(b) 函数 F11 收敛曲线　(c) 函数 F12 收敛曲线

Figure 4. Convergence curves of benchmark functions F10~F12
图 4. 基准函数 F10~F12 收敛曲线

综上所述，改进后的蜣螂算法在搜索中能有效改善全局寻优精度，提高了算法稳定性，并使算法具有较强的逃离局部最优的能力。

5. 结束语

蜣螂优化算法是模拟蜣螂自然行为的一种仿生智能算法，本文针对在蜣螂优化算法的基础上提出了一种改进的蜣螂优化算法。改进的蜣螂优化算法前期使用 Sinusoidal map 混沌思想以增加初始种群的多样性，从而增加全局搜索速度；在后期引入精英反向学习策略以增强其逃离局部最优的能力。通过求解 12 个基本测试函数并与其余五种智能算法进行比较，改进的蜣螂优化算法能有效提高算法的收敛精度，且提高整个搜索过程的搜索速度与稳定性。

参考文献

[1] 朱思峰, 赵明阳, 柴争义. 边缘计算场景中基于粒子群优化算法的计算卸载[J]. 吉林大学学报(工学报), 2022, 52(11): 2698-2705.

[2] 赵希梅, 陈广国, 金鸿雁. 基于改进灰狼优化算法的 PMSM 滑膜自抗扰控制[J]. 电机与控制学报, 2022, 26(11): 132-140.

[3] 高琴, 郭玉霞. 基于鲸鱼优化算法的车载成像雷达超分辨设计[J]. 信息技术与信息化, 2022(10): 188-191.

[4] 高大唤, 梁宏涛, 杜军威, 等. 改进黑猩猩优化算法的测试数据生成研究[J]. 计算机工程与应用, 2022, 58(23): 83-93.

[5] 张泽鹏, 茅云生, 傅何琪, 等. 基于混沌麻雀算法的船用焊接机器人轨迹优化[J]. 船舶工程, 2022, 44(5): 134-140.

[6] Kennedy, J. and Eberhart, R. (1995) Particle Swarm Optimization. *Proceedings of ICNN'95—International Conference on Neural Networks*, Perth, 27 November-1 December 1995, 1942-1948. https://doi.org/10.1109/icnn.1995.488968

[7] Krishnanand, K.N. and Ghose, D. (2009) Glowworm Swarm Optimisation: A New Method for Optimising Multi-Modal Functions. *International Journal of Computational Intelligence Studies*, **1**, 93-119. https://doi.org/10.1504/ijcistudies.2009.025340

[8] Yang, X. and Suash Deb, (2009) Cuckoo Search via Lévy Flights. 2009 *World Congress on Nature & Biologically Inspired Computing* (*NaBIC*), Coimbatore, 9-11 December 2009, 210-214. https://doi.org/10.1109/nabic.2009.5393690

[9] Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) Grey Wolf Optimizer. *Advances in Engineering Software*, **69**, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[10] Mirjalili, S. and Lewis, A. (2016) The Whale Optimization Algorithm. *Advances in Engineering Software*, **95**, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[11] Khishe, M. and Mosavi, M.R. (2020) Chimp Optimization Algorithm. *Expert Systems with Applications*, **149**, Article 113338. https://doi.org/10.1016/j.eswa.2020.113338

[12] Xue, J. and Shen, B. (2020) A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm. *Systems Science & Control Engineering*, **8**, 22-34. https://doi.org/10.1080/21642583.2019.1708830

[13] 王乐洋, 靳锡波, 许光煜. 震源参数反演的动态惯性殷子的粒子群算法[J]. 武汉大学学报(信息科学版), 2021, 46(4): 510-519.

[14] 张文胜, 郝孜奇, 朱冀军, 等. 基于改进灰狼算法优化 BP 神经网络的短时交通流预测模型[J]. 交通运输系统工程与信息, 2020, 20(2): 196-203.

[15] 吴泽忠, 宋菲. 基于改进螺旋更新位置模型的鲸鱼优化算法[J]. 系统工程理论与实践, 2019, 39(11): 2928-2944.

[16] 兰州新, 何庆. 一种新型的柯西扰动黑猩猩优化算法[J]. 小型微型计算机系统, 2023, 44(4): 715-723. https://kns.cnki.net/kcms/detail/21.1106.TP.20220215.1320.034.html, 2022-02-15.

[17] 钱敏, 黄海松. 范青松. 基于反向策略的混沌麻雀搜索算法[J]. 计算机仿真, 2022, 39(8): 333-339.

[18] Xue, J. and Shen, B. (2022) Dung Beetle Optimizer: A New Meta-Heuristic Algorithm for Global Optimization. *The Journal of Supercomputing*, **79**, 7305-7336. https://doi.org/10.1007/s11227-022-04959-6

[19] Shen, Y. (2018) Improved Chaos Genetic Algorithm Based State of Charge Determination for Lithium Batteries in Electric Vehicles. *Energy*, **152**, 576-585. https://doi.org/10.1016/j.energy.2018.03.174

[20] Saxena, A. (2019) A Comprehensive Study of Chaos Embedded Bridging Mechanisms and Crossover Operators for Grasshopper Optimisation Algorithm. *Expert Systems with Applications*, **132**, 166-188. https://doi.org/10.1016/j.eswa.2019.04.043

[21] Chen, H., Li, W. and Yang, X. (2020) A Whale Optimization Algorithm with Chaos Mechanism Based on Quasi-Opposition for Global Optimization Problems. *Expert Systems with Applications*, **158**, Article 113612. https://doi.org/10.1016/j.eswa.2020.113612

[22] Qu, C. (2020) Virtual Reconstruction of Random Moving Image Capturing Points Based on Chaos Embedded Particle Swarm Optimization Algorithm. *Microprocessors and Microsystems*, **75**, Article 103069. https://doi.org/10.1016/j.micpro.2020.103069

[23] Saremi, S., Mirjalili, S. and Lewis, A. (2014) Biogeography-Based Optimisation with Chaos. *Neural Computing and Applications*, **25**, 1077-1097. https://doi.org/10.1007/s00521-014-1597-x

[24] Tizhoosh, H.R. (2005) Opposition-Based Learning: A New Scheme for Machine Intelligence. *International Conference on Computational Intelligence for Modelling*, *Control and Automation and International Conference on Intelligent Agents*, *Web Technologies and Internet Commerce* (*CIMCA-IAWTIC*'06), Vienna, 28-30 November 2005, 695-701. https://doi.org/10.1109/cimca.2005.1631345

[25] Seif, Z. and Ahmadi, M.B. (2015) An Opposition-Based Algorithm for Function Optimization. *Engineering Applications of Artificial Intelligence*, **37**, 293-306. https://doi.org/10.1016/j.engappai.2014.09.009