文章编号:1007－6735(2024)05－0580－09　DOI: 10.13255/j.cnki.jusst.20230404002

混合策略改进的蜣螂优化算法及其工程应用

吉如沁，　秦江涛
（上海理工大学 管理学院，上海 200093）

摘要：针对原始蜣螂优化算法(dung beetle optimizer, DBO)容易陷入局部最优，收敛精度不够等问题，提出一种混合策略改进的蜣螂优化算法(TDBO)。运用 Tent 混沌映射策略初始化种群，使得初始蜣螂的位置分布更加均匀，提高种群的多样性；在蜣螂繁衍阶段使用自适应惯性权重，提升寻优能力；在蜣螂偷窃行为公式中引入莱维飞行，提高算法的搜索能力，使算法跳出局部最优，平横搜索多样性与收敛准确性之间的关系。在 9 个测试函数上分别与基础 DBO 算法、4 种对比算法以及单一策略改进的 DBO 算法进行比较，并通过 Wilcoxon 秩和检验验证 TDBO 算法的性能。结果证明，TDBO 算法在多个函数上速度和精度优于对比算法，并具有显著性差异。通过基准函数的测试、Wilcoxon 秩和检验，以及 3 个工程优化问题的验证，TDBO 算法具有较优的收敛精度和速度。

关键词：蜣螂优化算法；Tent 混沌映射；自适应惯性权重；莱维飞行
中图分类号：TP 301　　**文献标志码：**A

Hybrid strategy improved dung beetle optimizer and its engineering application

JI Ruqin,　QIN Jiangtao
(*Business School, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract: Aiming at the problems that the original dung beetle optimizer (DBO) is easy to prone to local optimum and the low convergence precision, a multi-strategy fusion improved dung beetle optimizer (TDBO) is proposed. Using the Tent chaos initialization population mapping strategy to makes the initial position of dung beetle distribution more uniform, and improve the diversity of population, adaptive inertia weight was applied during the breeding stage to improve the optimization ability. Levy

收稿日期: 2023−04−04
基金项目: 国家自然科学基金资助项目 (72174121，71774111)；上海市 2022 年度“科技创新行动计划”软科学研究项目 (22692112600)；上海市自然科学基金资助项目 (21ZR1444100)
第一作者: 吉如沁(1998−)，男，硕士研究生．研究方向：信息管理与电子商务研究．E-mail：1214059044@qq.com
通信作者: 秦江涛(1966−)，男，副教授．研究方向：制造系统分析研究．E-mail：qinjiangtao_usst@126.com

引文格式: 吉如沁，秦江涛．混合策略改进的蜣螂优化算法及其工程应用[J]．上海理工大学学报，2024，46(5)：580-588.
Citation: JI Ruqin, QIN Jiangtao．Hybrid strategy improved dung beetle optimizer and its engineering application[J]．Journal of University of Shanghai for Science and Technology, 2024, 46(5)：580-588.

flight was introduced into the dung beetle stealing behavior formula to improve the search ability of the algorithm, make the algorithm jump out of the local optimal, equates the relationship between diversity and convergence accuracy. Compared with the basic DBO algorithm, four comparison algorithms and single-strategy improved DBO algorithm on 9 test functions, and Wilcoxon rank sum test is used to verify the performance of the TDBO algorithm. The results show that the speed and accuracy of the TDBO algorithm are better than the comparison algorithm on multiple functions, and the TDBO algorithm has significant difference. Through the test of benchmark functions, the verification of Wilcoxon rank sum test, and validation of three engineering optimization problems, the TDBO algorithm has better convergence accuracy and speed.

Keywords: *dung beetle optimizer*; *Tent chaotic map*; *adaptive inertia weight*; *Levy flight*

蜣螂优化算法（dung beetle optimizer，DBO）[1]作为 2022 年提出的一种新型元启发算法，其灵感来源于蜣螂的滚球、跳舞、觅食、偷窃和繁殖行为。与灰狼优化算法（GWO）[2]、鲸鱼优化算法（WOA）[3]等其他学习算法相比，DBO 算法在基准函数上具有更强的优化能力和更快的效率。但是，DBO 算法和大部分智能算法一样，在后期迭代中，很容易陷入局部最优。为了进一步提升 DBO 算法的性能，本文对该算法进行了研究改进。

对于类似的元启发式算法，已经有很多学者对各种算法的精度、速度和鲁棒性等进行优化改进。王娟等[4]使用 Tent 混沌映射增加了算法种群的多样性，有效提高了海鸥算法的求解精度和速度；Wang 等[5]在萤火虫算法（FA）中引入惯性权重，平衡了 FA 的收敛速度和局部寻优能力，提升了算法的精度和速度；赵青杰等[6]应用动态自适应惯性权重有效提升了算法局部和全局搜索能力；李阳等[7]在算法中引入莱维飞行策略，扩大了种群的搜索范围，一定程度上优化了算法的性能；邬贵昌等[8]在麻雀算法中也使用了莱维飞行策略，有效提升了算法的寻优精度和稳定性。

本文针对 DBO 算法的精度和收敛速度进行了优化改进，提出了混合策略改进的蜣螂优化算法（TDBO），从 3 个方面对 DBO 算法作出改进：采用 Tent 混沌映射策略使初始蜣螂种群在搜索空间中分布得更加均匀，以此增加种群多样性；引入自适应惯性权重改进蜣螂繁衍公式，提升算法寻优能力；在蜣螂偷窃行为公式中引入莱维飞行，产生随机步长，为解增加一个扰动量，提高算法的搜索能力，使算法跳出局部最优。通过上述 3 方面的改进，在基于 9 个经典基准测试函数上进行测试，且进行 Wilcoxon 秩和检验，并通过 3 个工程优化问题的验证。实验结果证明了 TDBO 算法在性能上的优越性。

1　蜣螂优化算法介绍

蜣螂优化算法是基于蜣螂的滚球、跳舞、觅食、偷窃和繁殖行为而提出的一种元启发式算法，主要包括蜣螂滚球、觅食、繁衍、偷窃 4 个优化过程。

1.1　蜣螂滚球

a. 无障碍模式。

当蜣螂前行无障碍时，蜣螂在滚动过程中通过太阳来导航，以保持粪球在直线上滚动。滚球蜣螂的位置更新如式 (1) 所示。

$$\begin{cases} x_i^{t+1} = x_i^t + \alpha k x_i^{t-1} + b\Delta x \\ \Delta x = \left| x_i^t - x^{\mathrm{w}} \right| \end{cases} \tag{1}$$

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；$k \in (0, 0.2]$是一个常数量，表示偏转系数，本文取 0.1；b是属于$(0,1)$的常数值，本文取 0.3；α是自然系数，取 1 或−1，当α取 1 时表示无偏差，取−1 时表示偏离原来方向，α的具体取值根据概率方法确定，用来模拟复杂的环境；x^{w}表示全局最差位置；Δx是蜣螂当前位置与全局最差位置的绝对距离，用以模拟光强度变化，越大表示光源越弱，该参数可以提升算法搜索的性能，尽可能地使算法探索整个空间。

b. 跳舞。

当蜣螂遇到障碍物而无法前进时，它会通过跳舞来重新定向自己，以获得新的路线。因此，

蜣螂的位置更新如式(2)所示。

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

式中：θ是属于$[0,\pi]$的角。当$\theta = 0, \dfrac{\pi}{2}$或$\pi$时，$\tan\theta$没有意义，此时蜣螂的位置不会更新。

1.2 繁 殖

为了给后代提供一个安全的环境，作者提出了一个边界选择策略来模拟雌性蜣螂产卵的区域，其定义为

$$\begin{cases} L^* = \max(x^{\mathrm{g}}(1-R), L) \\ U^* = \min(x^{\mathrm{g}}(1+R), U) \end{cases}$$

式中：x^{g}表示当前局部最优位置；$R = 1 - t/T_{\max}$，$T_{\max}$表示最大迭代次数；L和U分别表示优化问题的下界和上界，L^*和U^*分别表示产卵区域的下界和上界。

蜣螂产卵区域是随迭代次数动态调整的，因此，产卵球的位置在迭代过程中也是动态的，如式(3)所示。

$$B_i^{t+1} = x^{\mathrm{g}} + \boldsymbol{b}_1\left(B_i^t - L^*\right) + \boldsymbol{b}_2\left(B_i^t - U^*\right) \tag{3}$$

式中：B_i^t为第i个卵球在第t次迭代的位置信息；$\boldsymbol{b}_1$和$\boldsymbol{b}_2$为两个独立的、大小为$1\times D$的随机向量，D为优化问题的维度。

1.3 觅 食

一些成熟的小蜣螂会从地下出来寻找食物，小蜣螂的觅食区域是动态更新的，表示如下：

$$\begin{cases} L^{\mathrm{l}} = \max\left(x^{\mathrm{b}}(1-R), L\right) \\ U^{\mathrm{l}} = \min\left(x^{\mathrm{b}}(1+R), U\right) \end{cases}$$

式中：x^{b}为当前种群的全局最优位置；L^{l}和U^{l}分别表示小蜣螂觅食区域的下界和上界。小蜣螂的位置更新如式(4)所示。

$$x_i^{t+1} = x_i^t + \boldsymbol{C}_1\left(x_i^t - L^{\mathrm{l}}\right) + \boldsymbol{C}_2\left(x_i^t - U^{\mathrm{l}}\right) \tag{4}$$

式中：$\boldsymbol{C}_1$为服从正态分布的随机数，即$\boldsymbol{C}_1 \sim N(0,1)$；$\boldsymbol{C}_2$为$1\times D$的属于$(0,1)$之间的随机向量。

1.4 偷 窃

在种群中，会有一些蜣螂从其他蜣螂那里偷食物，偷窃蜣螂的位置更新如式(5)所示。

$$x_i^{t+1} = x^{\mathrm{b}} + S\,\boldsymbol{g}\left(\left|x_i^t - x^{\mathrm{g}}\right| + \left|x_i^t - x^{\mathrm{b}}\right|\right) \tag{5}$$

式中：x^{b}为当前种群的全局最优位置，表示最优食物源，即假设该位置附近是竞争食物的最优位置；$\boldsymbol{g}$表示大小为$1\times D$且服从正态分布的随机向量；S表示一个常数值。

1.5 种群划分

对于种群中蜣螂个体的划分，作者按照6∶6∶7∶11的比例。既在30个个体中，滚球蜣螂有6个，繁衍蜣螂有6个，觅食蜣螂有7个，偷窃蜣螂有11个。

2 蜣螂优化算法的改进

2.1 Tent混沌映射

基础DBO算法的初始蜣螂位置是随机分布的，致使蜣螂个体位置在解空间中分布不均匀，从而降低算法的收敛速度和精度。因此，本文在DBO算法的基础上引入混沌映射策略，丰富蜣螂种群多样性。岳龙飞等[9]验证了Tent映射在均匀分布和收敛速度方面相对于其他映射具有更大的优势。

Tent混沌映射的函数表达式如式(6)所示。

$$x_n^{m+1} = \begin{cases} 2x_n^m, & 0 \leqslant x_n^m \leqslant 0.5 \\ 2(1-x_n^m), & 0.5 \leqslant x_n^m \leqslant 1 \end{cases} \tag{6}$$

式中：n表示种群大小；m表示空间维数。

通过式(6)得到混沌序列x_i^k后再通过式(7)得到初始化的种群。

$$y_n^m = L + (U-L)\,x_n^m \tag{7}$$

2.2 自适应惯性权重

惯性权重因子ω在算法优化中是一个很重要的参数，可以有效提高算法的收敛速度和局部寻优能力。繁衍蜣螂在进行局部寻优时是依据上一时刻局部最优位置更新的，没有考虑计算过程中其他位置信息，可能导致算法最终陷入局部最优，达不到整体最优值。参考文献[10-11]，本文使用式(8)的自适应惯性权重。当权重较大时，繁衍蜣螂拥有较强的全局搜索能力，当权值较小时，繁衍蜣螂具有较强的局部寻优能力，能够加快收敛速度。因此，融合自适应惯性权重能够很好地平衡搜索多样性与收敛准确性之间的关系，提高算法的收敛精度和速度。

自适应权重公式如式(8)所示。

$$\omega = \cos\left(\frac{\pi t}{4T_{\max}} + \pi\right) + 1 \tag{8}$$

式中，$T_{\max}$为最大迭代次数。

将式(8)代入式(3)，得到改进的蜣螂繁衍位置更新函数：

$$B_i^{t+1} = x^{\mathrm{g}} + \omega\left(B_i^t - L^*\right) + \omega\left(B_i^t - U^*\right) \quad (9)$$

2.3 莱维飞行

莱维飞行[12]是通过随机交替大、小步长使得算法跳出局部最优的方法。由上述偷窃蜣螂的更新公式可以发现，其搜索步长是固定的，因此算法可能会陷入局部最优的情况。引入莱维飞行后可以避免搜索空间的逐步缩小，使算法跳出局部最优解。改进后的蜣螂偷窃位置更新函数如式(10)所示。

$$x_i^{t+1} = x^{\mathrm{b}} y + S\boldsymbol{g}\left(\left|x_i^t - x^{\mathrm{g}}\right| + \left|x_i^t - x^{\mathrm{b}}\right|\right) \quad (10)$$

式中：y符合莱维分布。

因为莱维飞行过程的复杂性，所以使用Mantegna算法模拟其过程[13]，y步长大小的公式如下:

$$y = 0.001\frac{u}{v^{1/\beta}}$$

u和v均遵循正态分布：

$$u \sim N\left(0, \sigma_u^2\right), v \sim N(0, \sigma_v^2)$$

$$\sigma_u = \left\{\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{2^{\left(\frac{\beta-1}{2}\right)}\Gamma\left(\frac{1+\beta}{2}\right)\beta}\right\}^{\frac{1}{\beta}}, \sigma_v = 1$$

式中，Γ为Gamma函数，参数$\beta = 1.5$。

2.4 算法流程

a. 算法参数初始化，如：种群数量，空间维度和最大迭代次数等；

b. 种群初始化，通过式(6)的 tent 混沌映射，得到初始化蜣螂位置；

c. 计算种群的适应度值，获得最优个体；

d. 通过式(1)或(2)更新滚球蜣螂的位置；通过式(9)更新繁衍蜣螂的位置；通过式(4)更新觅食蜣螂的位置；通过式(10)更新偷窃蜣螂的位置；

e. 更新蜣螂种群信息，找出并保存适应度最优的蜣螂位置；

f. 当满足迭代条件时，输出算法最优结果，否则继续进行从 d ~ f 的流程。

2.5 时间复杂度分析

假设 DBO 算法的种群大小为N，空间维度为M，那么该算法的初始化时间复杂度为$O(1)$，计算适应度为$O(N)$，总的迭代过程的复杂度为$O(NM)$，DBO 算法总的时间复杂度为

$$O(1) + O(N) + O(NM) = O(NM)$$

在 TDBO 算法中，Tent 混沌映射的初始化时间复杂度为$O(NM)$，计算适应度为$O(N)$，莱维飞行和自适应惯性权重对位置更新的时间复杂度分别为$O(NM)$，总的迭代过程的复杂度为$O(NM)$，TDBO 算法总的时间复杂度为

$$O(NM) + O(N) + O(NM) + O(NM) + O(NM) = O(NM)$$

综上所述，TDBO 算法的时间复杂度并没有提升。

3 实验结果分析

本文将从以下 3 个部分进行实验验证：a. 将 TDBO 算法与 GWO 算法、麻雀搜索算法(SSA)[14]、WOA 算法、北方苍鹰优化算法(NGO)[15]和 DBO 算法这 5 个基本元启发式算法进行对比，验证 TDBO 算法的寻优能力和鲁棒性; b. 将 TDBO 算法与单独策略改进型的 DBO 算法进行比较，验证不同改进策略的有效性; c. 采用 Wilcoxon 秩和检验的方法验证 TDBO 算法与对比算法的差异性。

如表 1 所示，本文选取了多个常用的基准测试函数。其中，F_1～F_5为多维单峰值函数，F_6和F_7为多峰函数，F_8和F_9为固定维度多峰函数。为了避免实验的随机性，算法的最大迭代次数设置为 500，种群规模大小均设为 30，每个算法依据相应的参考文献设置其他的参数。

3.1 与其他基准算法进行对比

为了验证改进后 DBO 算法的收敛性以及稳定性，将本文提出的 TDBO 算法与 WOA，GWO，SSA，NGO，DBO 算法在 9 个基准测试函数上进行对比实验。

各个算法在上述函数上都独立运行 30 次，避免一次运行结果的偶然性。每个算法所得数据的平均值和标准差如表 2 所示。

从表 2 可以看出，在多维单峰值函数F_1～F_5中，TDBO 函数的性能远优于其他对比函数，并且除了F_4函数，其他均达到了理论最优值。在多峰函数F_6～F_7中，TDBO 算法也展现了不错的性能，均达到了理论最优值。其中的F_6和F_7函数，SSA 和 NGO 算法也都达到理论最优，无法比较 TDBO 算法与这两者的性能，但是相对于未改进的初始 DBO 算法，TDBO 算法的性能均有较大提升。在固定多峰值函数F_8和F_9中，TDBO

表 1 基本测试函数

Tab.1 Basic test functions

函数表达式	维度	搜索区间	最优值
$F_1(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[–10,10]	0
$F_2(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{i}x_j\right)^2$	30	[–100,100]	0
$F_3(x)=\mathbf{max}\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	30	[–100,100]	0
$F_4(x)=\sum_{i=1}^{n-1}ix_i^4+\mathbf{random}[0,1)$	30	[–1.28,1.28]	0
$F_5(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	30	[–1,1]	0
$F_6(x)=\sum_{i=1}^{n}[x_i^2-10\mathbf{cos}(2\pi x_i)+10]$	30	[–5.12,5.12]	0
$F_7(x)=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\mathbf{cos}\frac{x_i}{\sqrt{i}}+1$	30	[–600,600]	0
$F_8(x)=0.5+\frac{\mathbf{sin}^2\left(\sqrt{x_1^2+x_2^2}-0.5\right)}{(1.0+0.001(x_1^2+x_2^2))^2}$	2	[–100,100]	0
$F_9(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[–5,5]	0.0003

表 2 不同算法测试结果

Tab.2 Test results of different algorithms

函数	GWO	SSA	WOA	NGO	DBO	TDBO
F_1	1.16E-16	6.02E-29	5.70E-50	1.34E-45	9.95E-52	0.00E+00
	8.24E-17	2.45E-28	2.31E-49	1.86E-45	5.45E-51	0.00E+00
F_2	4.83E-06	1.65E-29	3.85E+04	1.79E-22	2.35E-59	0.00E+00
	1.20E-05	8.85E-29	1.17E+04	5.74E-22	1.29E-58	0.00E+00
F_3	8.70E-07	2.47E-30	4.92E+01	2.51E-37	1.21E-53	0.00E+00
	7.82E-07	8.72E-30	2.69E+01	3.31E-37	4.61E-52	0.00E+00
F_4	2.49E-03	2.22E-03	3.00E-03	5.60E-04	1.37E-03	1.21E-04
	1.49E-03	2.58E-03	3.62E-03	2.28E-04	1.10E-03	1.06E-04
F_5	6.01E-93	3.14E-53	4.23E-99	3.61E-180	3.93E-105	0.00E+00
	3.27E-92	1.72E-52	2.29E-14	0.00E+00	2.15E-104	0.00E+00
F_6	2.65E+00	0.00E+00	7.60E-15	0.00E+00	0.44E+00	0.00E+00
	3.55E+00	0.00E+00	2.47E-14	0.00E+00	1.44E+00	0.00E+00
F_7	3.97E-03	0.00E+00	3.70E-18	0.00E+00	5.15E-04	0.00E+00
	5.27E-03	0.00E+00	2.03E-17	0.00E+00	2.82E-03	0.00E+00
F_8	3.45E-02	0.00E+00	2.24E-02	9.72E-03	9.72E-03	0.00E+00
	8.39E-03	0.00E+00	1.78E-02	6.17E-14	1.10E-07	0.00E+00
F_9	4.42E-03	3.47E-04	8.12E-04	3.15E-04	7.49E-04	6.45E-04
	8.28E-03	1.05E-05	5.64E-04	3.85E-06	3.32E-04	1.60E-04

算法相较于基础 DBO 算法均有所改善，但是在 F_8 函数中，无法比较与 SSA 算法的性能；在 F_9 函数中，性能不及 NGO 算法和 SSA 算法。

3.2 算法收敛曲线对比分析

本文给出了算法在上述测试函数上的曲线收敛图，用来直观地比较 TDBO 算法和对比算法的收敛速度。

由图 1 可知，在多维单峰值函数 F_1～F_5 和多峰值函数 F_6～F_7 中，TDBO 算法的收敛速度和精度明显高于其他对比算法。说明 TDBO 算法具有较好的性能并且可以跳出局部极值点，收敛到最优值。在固定多峰函数 F_8 上，TDBO 算法相对于

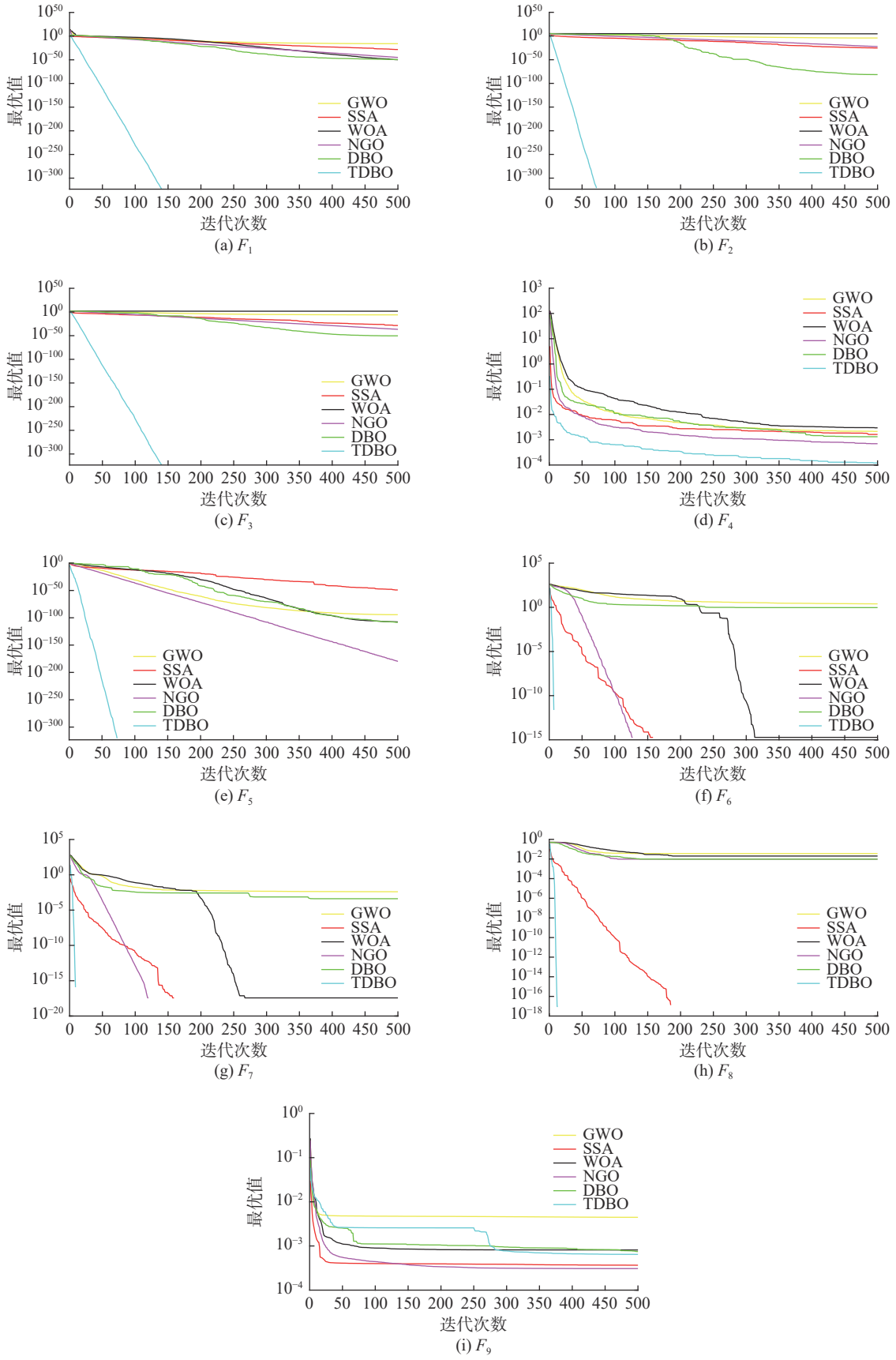

图 1　各测试函数下的收敛曲线

Fig.1　Convergence curve under each test function

对比算法也展现出了极快的收敛速度。虽然在F_9上精度和速度略低于NGO算法和SSA算法，但是相对于未改进的DBO算法，TDBO算法的精度和速度均有所提升。

3.3 消融实验

通过将TDBO算法与基于Tent映射改进的DBO1算法、融合莱维飞行的DBO2算法和融合自适应惯性权重的DBO3算法在9个测试函数中作性能比较，进一步研究单个改进策略的有效性，实验结果见表3。

由表3可知，在函数F_1～F_4和F_8中TDBO算法的性能均优于单一策略改进的DBO算法和基本DBO算法。在函数F_5中，TDBO算法寻优精度和DBO2算法相同，在F_6中，TDBO算法的精度和DBO2，DBO3算法相同，都达到了理论最优值。在F_7中，TDBO，DBO1和DBO2算法均到达最优值。在F_9中，DBO2算法展现出了最优的性能。在所有的函数中，单一策略改进的DBO算法和多策略改进的TDBO算法均优于基本DBO算法。由此可知，经过3种改进策略的TDBO算法，寻优精度和鲁棒性明显优于单一策略改进的算法，能够有效弥补单一策略的不足，从而最大限度地提升基本DBO算法的性能。

3.4 Wilcoxon 秩和检验

为了验证TDBO算法相较于对比算法的优越性，运用Wilcoxon秩和检验[16]的方法来进行验证，显著性水平p设置为5%。将算法对立运行的30次结果进行统计检验，当p值小于5%的情况下，说明两种算法显著性明显，否则两种算法之间的性能相差不大。当TDBO算法与对比算法得到的实验数据相差不大时，说明两个算法之间性能相当，则Wilcoxon秩和检验的结果表示为NaN。

由表4可知，TDBO算法和GWO算法在F_9上性能显著性差异不明显；和SSA算法在测试函数F_6～F_8上性能相当；和WOA算法在F_6和F_9上性能没有显著性差异；和NGO算法在函数F_6～F_7上性能相当，和DBO算法在F_6，F_7和F_9上性能没有显著性差异。除此之外，其余p值均小于5%。总体看来，本文提出的TDBO算法相较于其他对比算法具有显著性差异。

表3 消融实验测试结果

Tab.3 Results of ablation experiment testing

函数	DBO1	DBO2	DBO3	DBO	TDBO
F_1	4.40E-57	0.00E+00	2.56E-84	9.95E-52	0.00E+00
	2.08E-56	0.00E+00	1.22E-83	5.45E-51	0.00E+00
F_2	3.15E-62	2.72E-89	2.30E-66	2.35E-59	0.00E+00
	1.72E-61	1.49E-88	1.26E-65	1.29E-58	0.00E+00
F_3	6.64E-57	2.56E-73	7.07E-73	1.21E-53	0.00E+00
	3.46E-56	1.40E-72	4.82E-72	4.61E-52	0.00E+00
F_4	1.10E-03	8.27E-04	1.34E-03	1.37E-03	1.21E-04
	8.19E-04	4.93E-04	1.09E-03	1.10E-03	1.06E-04
F_5	5.16E-111	0.00E+00	3.05E-123	3.93E-105	0.00E+00
	2.83E-110	0.00E+00	1.67E-122	2.15E-104	0.00E+00
F_6	1.33E-01	0.00E+00	0.00E+00	0.44E+00	0.00E+00
	7.27E-01	0.00E+00	0.00E+00	1.44E+00	0.00E+00
F_7	0.00E+00	0.00E+00	1.15E-03	5.15E-03	0.00E+00
	0.00E+00	0.00E+00	6.29E-03	2.82E-02	0.00E+00
F_8	9.39E-03	3.24E-04	9.72E-03	9.72E-03	0.00E+00
	1.77E-08	1.78E-03	5.86E-08	1.88E-07	0.00E+00
F_9	6.46E-04	5.19E-04	6.88E-04	7.09E-04	6.45E-04
	2.84E-04	2.12E-04	3.04E-04	3.32E-04	1.60E-04

表 4　Wilcoxon 秩和检验 p 值

Tab.4　Wilcoxon rank and test p-value

测试函数	TDBO/GWO	TDBO/SSA	TDBO/WOA	TDBO/NGO	TDBO/DBO
F_1	1.73E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F_2	1.73E-06	2.56E-06	1.73E-06	1.73E-06	1.73E-06
F_3	1.73E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F_4	1.73E-06	2.35E-06	1.73E-06	2.13E-06	1.73E-06
F_5	1.73E-06	2.56E-06	1.73E-06	1.73E-06	1.73E-06
F_6	3.60E-06	NaN	1.61E-01	NaN	2.50E-01
F_7	1.56E-02	NaN	1.25E-02	NaN	2.50E-01
F_8	1.73E-06	NaN	5.54E-06	1.73E-06	1.73E-06
F_9	2.37E-01	2.88E-06	1.53E-01	1.73E-06	2.99E-01

4　工程优化问题应用

4.1　压力容器设计问题描述

压力容器设计问题是在 4 个约束条件下使得总成本最小。它包括 4 个变量，分别为壳体厚度 x_1、封头厚度 x_2、壳体半径 x_3以及圆柱形截面长度 x_4，其具体数学模型如下所示。

目标函数：

$$\min f(x) = 0.622\,4x_1x_3x_4 + 1.778\,1x_2x_3^2 + 3.166\,1x_1^2x_4 + 19.84x_1^2x_3$$

约束条件：

$$\begin{cases} g_1(x) = -x_1 + 0.019\,3x_3 \leqslant 0 \\ g_2(x) = -x_2 + 0.009\,54x_3 \leqslant 0 \\ g_3(x) = -\pi x_3^2x_4 - 4\pi x_3^3/3 + 1\,296\,000 \leqslant 0 \\ g_4(x) = x_4 - 240 \leqslant 0 \\ x_1 \geqslant 0, x_2 \leqslant 99, x_3 \geqslant 10, x_4 \leqslant 200 \end{cases}$$

如表 5 所示，在压力容器设计问题优化中，TDBO 算法求解出的最优值是 6 种算法之中最小的，说明 TDBO 算法在求解该工程设计问题时具有较好的性能。

4.2　悬臂梁设计问题描述

悬臂梁设计问题是在满足开口端垂直位移上限的条件下使得悬臂梁的重量最小化。x_1，x_2，x_3，x_4，x_5 分别表示 5 个空心方块的长度，具体数学模型如下所示。

目标函数：

$$\min f(x) = 0.06224(x_1 + x_2 + x_3 + x_4 + x_5)$$

约束条件：

$$\begin{cases} g_1(x) = \dfrac{61}{x_1^3} + \dfrac{37}{x_2^3} + \dfrac{19}{x_3^3} + \dfrac{7}{x_4^3} + \dfrac{1}{x_5^3} \leqslant 1 \\ 0.01 \leqslant x_i \leqslant 100,\ i = 1,2,3,4,5 \end{cases}$$

表 5　压力容器设计问题测试结果比较

Tab.5　Comparison of test results for pressure vessel design problem

算法	x_1	x_2	x_3	x_4	最优值(排序)
GWO	0.8125	0.4345	42.0891	176.7587	6051.5639(5)
SSA	0.8544	0.4223	44.2742	151.4194	6029.0111(4)
WOA	1.0057	0.4982	52.0119	83.1434	6413.7919(6)
NGO	0.7934	0.4130	41.1021	189.4275	5976.5747(3)
DBO	0.8125	0.4023	42.0984	176.6367	5949.1354(2)
TDBO	0.8019	0.3976	41.5449	183.6372	5931.8711(1)

如表 6 所示，TDBO 算法的最优值以较小的优势达到最小。相较于初始 DBO 算法，寻优结果得到了较好的改善。

表 6　悬臂梁设计问题测试结果比较

Tab.6　Comparison of test results for cantilever beam design problem

算法	x_1	x_2	x_3	x_4	x_5	最优值(排序)
GWO	5.9806	5.3460	4.4886	3.4912	2.1690	1.33662 (4)
SSA	6.0288	5.3116	4.5059	3.4750	2.1529	1.33655 (3)
WOA	9.6875	4.0717	5.5529	3.1009	2.9477	1.57844 (6)
NGO	6.0122	5.3194	4.4981	3.4917	2.1525	1.33653 (2)
DBO	5.9311	5.2286	4.5970	3.5156	2.2143	1.33732 (5)
TDBO	6.0107	5.3108	4.5060	3.4933	2.1528	1.33651 (1)

4.3　工字钢设计问题描述

工字钢结构设计问题是通过调整其长和高以

及两个厚度来达到最小的垂直挠度。其具体数学模型如下所示。

目标函数：

$$\min f(x)=\frac{5\,000}{\dfrac{x_3(x_1-2x_4)^3}{12}+\dfrac{x_2x_4^3}{6}+2x_2x_4\left(\dfrac{x_1-x_4}{2}\right)^2}$$

约束条件：

$$\begin{cases} g_1(x)=2x_2x_4+x_3(x_1-2x_4)-300\leqslant 0 \\ g_2(x)=\dfrac{18x_1\times 10^4}{x_3(x_1-2x_4)^3+2x_2x_4\left(4x_4^2+3x_1(x_1-2x_4)\right)}+ \\ \dfrac{15x_2\times 10^3}{(x_1-2x_4)x_3^3+2x_4x_2^3}-6\leqslant 0 \\ 10\leqslant x_1\leqslant 80,10\leqslant x_2\leqslant 50,x_3\geqslant 0.9,x_4\leqslant 5 \end{cases}$$

如表7所示，除了GWO算法和WOA算法的最优值稍差之外，其余4个算法达到了相同的最优值。以此得出，改进后的TDBO算法并未降低初始DBO算法的性能。

表7 工字钢设计问题测试结果比较

Tab.7 Comparison of test results for I-beam design problem

算法	x_1	x_2	x_3	x_4	最优值(排序)
GWO	80	50	0.9	2.321 4	0.013 076(2)
SSA	80	50	0.9	2.321 7	0.013 074(1)
WOA	80	30.876 7	0.901 2	3.801 4	0.013 530(3)
NGO	80	50	0.9	2.321 7	0.013 074(1)
DBO	80	50	0.9	2.321 7	0.013 074(1)
TDBO	80	50	0.9	2.321 7	0.013 074(1)

5 结 论

根据DBO算法易陷入局部最优，可能存在收敛精度不够的问题。本文依据Tent映射策略、自适应惯性群权重和莱维飞行策略，提出一种混合策略改进的蜣螂优化算法(TDBO)，从而有效提高了算法的搜索能力，使算法较容易跳出局部最优，更好地平衡搜索多样性与收敛准确性之间的关系。在9个基准测试函数上进行实验，并与GWO，SSA，WOA，NGO和DBO算法进行对比得出，TDBO算法的收敛速度、精度和鲁棒性都表现较好，且通过秩和检验证明了TDBO算法与其他算法具有差异性。最后，在3个工程优化问题中，TDBO算法也取得了不错的效果。当然，算法的改进策略不局限于本文所提出的几种方法。未来的研究，一方面会尝试更多的改进策略，另一方面也会把TDBO算法用于解决实际问题，如深度神经网络的超参数优化、路径优化等。

参考文献：

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

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

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

[4] 王娟, 秦江涛. 混沌映射与 *t*-分布变异策略改进的海鸥优化算法 [J]. 计算机应用研究, 2022, 39(1): 170−176,182.

[5] WANG W C, XU L, CHAU K W, et al. Yin-Yang firefly algorithm based on dimensionally Cauchy mutation[J]. Expert Systems with Applications, 2020, 150: 113216.

[6] 赵青杰, 李捷, 于俊洋, 等. 基于动态自适应权重和柯西变异的蝙蝠优化算法 [J]. 计算机科学, 2019, 46(S1): 89−92.

[7] 李阳, 李维刚, 赵云涛, 等. 基于莱维飞行和随机游动策略的灰狼算法 [J]. 计算机科学, 2020, 47(8): 291−296.

[8] 邬贵昌, 韦文山, 李尚平, 等. 基于混沌的多策略优化麻雀算法及应用 [J]. 微电子学与计算机, 2022, 39(12): 21−30.

[9] 岳龙飞, 杨任农, 张一杰, 等. Tent混沌和模拟退火改进的飞蛾扑火优化算法 [J]. 哈尔滨工业大学学报, 2019, 51(5): 146−154.

[10] 丁容, 高建瓴, 张倩. 融合自适应惯性权重和柯西变异的秃鹰搜索算法 [J]. 小型微型计算机系统, 2023, 44(5): 910−915.

[11] 党婷婷, 林丹. 含有动态自适应惯性权重的蜘蛛猴优化算法 [J]. 计算机工程与应用, 2019, 55(14): 40−47.

[12] NASRI D, MOKEDDEM D. Optimisation of multi-objective problems using an efficient Levy flight grasshopper algorithm[J]. International Journal of High Performance Systems Architecture, 2022, 11(1): 26−35.

[13] 张严, 秦亮曦. 基于Levy飞行策略的改进樽海鞘群算法 [J]. 计算机科学, 2020, 47(7): 154−160.

[14] 薛建凯. 一种新型的群智能优化技术的研究与应用——麻雀搜索算法 [D]. 上海: 东华大学, 2020.

[15] DEHGHANI M, HUBÁLOVSKÝ Š, TROJOVSKÝ P. Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems[J]. IEEE Access, 2021, 9: 162059−162080.

[16] WILCOXON F. Individual comparisons by ranking methods[J]. Biometrics Bulletin, 1945, 1(6): 80−83.

（编辑：丁红艺）