王乐遥, 顾磊. 带扰动的自适应蜣螂优化算法 [J]. 智能计算机与应用,2026,16(5):40-48. DOI:10. 20169/j. issn. 2095-2163. 24090204

带扰动的自适应蜣螂优化算法

王乐遥, 顾 磊
(南京邮电大学 计算机学院, 南京 210023)

摘 要: 针对蜣螂优化(Dung Beetle Optimizer, DBO)算法易在无效区域盲目开发,求解精度低等不足,提出带扰动的自适应蜣螂优化(Perturbed Adaptive Dung Beetle Optimizer, PADBO)算法。该算法提出一种扰动疏散机制,避免繁育型蜣螂个体聚集,实现算法的多样化局部开发,提升了寻优精度;同时,采用自适应惯性权重策略,动态平衡算法的开发与探索能力,加快了算法的收敛速度。在 8 个标准测试函数上与其他 5 种优化算法的对比实验结果表明,PADBO 具有较高的收敛速度与寻优精度. 通过减速机质量最小化问题的求解结果,验证了 PADBO 求解实际工程设计问题的有效性。

关键词: 蜣螂优化算法; 扰动疏散; 自适应惯性权重; 减速机设计问题
中图分类号: TP393 **文献标志码**: A **文章编号**: 2095-2163(2026)05-0040-09

Perturbed adaptive Dung Beetle Optimizer

WANG Leyao, GU Lei
(School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: Aiming at the limitations of the Dung Beetle Optimizer (DBO), such as blind exploitation in invalid regions and low solution accuracy, a Perturbed Adaptive Dung Beetle Optimizer (PADBO) is proposed. The algorithm introduces a perturbed dispersion mechanism to prevent the aggregation of breeding dung beetle individuals, enabling diversified local exploitation of the algorithm and improving its optimization accuracy. Meanwhile, an adaptive inertia weight strategy is adopted to dynamically balance the exploitation and exploration capabilities, thus accelerating its convergence speed. Comparative experiments on 8 standard test functions with other five optimization algorithms show that PADBO achieves superior convergence speed and optimization accuracy. The results of the reducer weight minimization problem verify the effectiveness of PADBO in solving practical engineering design problems.

Key words: dung beetle optimizer; perturbed dispersion; adaptive inertia weight; reducer design problem

0 引 言

随着科技的迅猛发展,传统的优化算法面对现实各领域复杂生产实践问题时,往往难以胜任,群智能优化算法具有黑盒性质,可以很好地解决传统优化算法无法解决的复杂优化问题,受到了研究人员的广泛关注。按照算法中种群的数量,大致可分为 2 类:(1)基于单一种群的算法;(2)基于子群结构的算法。其中,单一种群的群智能优化算法仅包含一个生物种群,通过模拟种群内个体的觅食、捕猎等特定生物行为实现迭代寻优,此类算法原理简单且易于理解,典型算法有粒子群优化算法 (Particle Swarm Optimization, PSO)[1-2]、海鸥优化算法[3]、蝗虫优化算法[4]、蝙蝠优化算法[5]等。随着群智能理论的逐渐发展,研究人员发现单一种群的群智能优化算法在寻优过程中难以对空间进行充分探索,容易出现种群围绕超级个体搜索而导致早熟收敛的问题,因此提出了一些基于子群结构的群智能优化算法,此类算法具有 2 个及以上的子群,灵感主要是来源于同种生物群体内部不同角色的分工合作、竞争行为,或是不同生物间互利、捕食等关系,不同的子群各自对问题空间进行探索并通过信息交互实现协同寻优,具备更强的全局探索能力及可扩展性,典型算法包括山瞪羚优化算法[6]、几何平均优化算法[7]、美洲狮优化算

基金项目: 江苏省研究生科研与实践创新计划项目(SJCX23_0273,SJCX24_0318);江苏省高校自然科学研究重大项目(22KJA520010)。
作者简介: 王乐遥(1999—),男,硕士研究生。主要研究方向:群智能优化,进化计算。
通信作者: 顾 磊(1978—),男,博士,副教授。主要研究方向:群智能优化,数据挖掘。Email:njgulei@126. com。
收稿日期: 2024-09-02

法[8]、蛇优化算法(Snake Optimizer, SO)[9]、野马优化算法(Wild Horse Optimizer, WHO)[10]等。

蜣螂优化算法(Dung Beetle Optimizer, DBO)[11]是一种典型的基于子群结构的优化算法,通过对蜣螂推球、跳舞、繁育、觅食、偷窃这些行为的模拟及抽象将种群分为 4 个子群,具备优异的寻优性能,在网络入侵检测[12]、模型超参数优化[13]等诸多领域得到了应用。目前对于 DBO 的最新改进也有不少,但不同改进算法针对的蜣螂子群存在差异,例如文献[13]对 DBO 中与小蜣螂和小偷蜣螂的相关公式进行了改进;文献[14]重新设计了推球蜣螂的位置更新方式;文献[15]对小蜣螂觅食过程中的位置更新进行了优化;文献[16]则是改进蜣螂繁衍与偷窃行为的公式,提升了算法的搜索能力。本文针对繁育蜣与小蜣螂子群进行改进,通过对 DBO 中存在的不足进行了具体的分析,提出一种带扰动的自适应蜣螂优化算法(Perturbed Adaptive Dung Beetle Optimizer,PADBO),选取 8 个标准函数进行寻优测试,并将其用于求解减速机设计问题,验证了 PADBO 的有效性及实用性。

1　蜣螂优化算法 DBO

蜣螂优化算法(Dung Beetle Optimizer, DBO)是一种多子群结构的优化算法,DBO 将蜣螂种群 X 分为推球蜣螂 R、繁育蜣螂 B、小蜣螂 L 以及小偷蜣螂 T 四个不同的子群,分别占比 20%、20%、25%、35%。

1.1　DBO 中不同子群的位置更新方式

推球蜣螂会利用天体的偏振光导航,以此实现搬运路径尽可能为直线,当遇到障碍物阻拦前进时,会通过跳舞行为确定一个偏转角度,进而调整方向继续完成搬运行为。推球蜣螂位置更新公式如下:

$$R_e^{t+1}=\begin{cases}\tilde{R}_e^t+\alpha\times k\times\tilde{R}_e^{t-1}+u\times|\tilde{R}_e^t-\mathrm{worst}(\tilde{X}^t)|, & \lambda<\gamma\\ \tilde{R}_e^t+\tan(\theta)|\tilde{R}_e^t-\tilde{R}_e^{t-1}|, & \lambda\geqslant\gamma\end{cases}\tag{1}$$

其中,$\tilde{R}_e^t$ 表示第 e 只推球蜣螂第 $t-1$ 次迭代的位置;$t+1$ 表示当前迭代次数;λ 为随机数,且 $\lambda\in[0,1]$;$\gamma=0.9$ 表示推球蜣螂没有遇到障碍物并正常行进的概率,当 $\lambda<\gamma$ 时,蜣螂搬运没有障碍阻拦;$\tilde{R}_e^t$ 表示第 e 只推球蜣螂第 t 次迭代后自身的历史最优位置;k 为位置偏转系数的常数,且 $k\in(0,0.2]$;α 为用于模拟自然因素对搬运方向影响的系数,当 $\alpha<\eta$ 时,α 为 1,否则 α 为-1。其中 η 为随机数,且 $\eta\in[0,1]$,u 为常数,取值为 0.3。$\tilde{X}^t$ 代表第 t 次迭代后蜣螂种群所有个体历史最优位置的集合,$\mathrm{worst}(\tilde{X}^t)$ 表示 $\tilde{X}^t$ 中的最差位置,$|\tilde{R}_e^t-\mathrm{worst}(\tilde{X}^t)|$ 表示 $\tilde{R}_e^t$ 与 $\mathrm{worst}(\tilde{X}^t)$ 之间各维度的距离向量。当 $\lambda\geqslant\gamma$ 时,蜣螂搬运遇到障碍阻拦,θ 为随机数且 $\theta\in[0,1]$,表示通过跳舞决定的方向偏转角弧度,正切函数用于模拟蜣螂通过跳舞确定偏转角更改方向的行为,$|\tilde{R}_e^t-\tilde{R}_e^{t-1}|$ 表示 $\tilde{R}_e^t$ 与 $\tilde{R}_e^{t-1}$ 之间各维度的距离向量,当 $\theta=0、\pi/2、\pi$ 时,推球蜣螂保持历史最优位置不变。

繁育蜣螂在每次迭代时会在育雏球里产下一颗卵,对应位置更新公式如下:

$$B_m^{t+1}=X^*+b_1\times(\tilde{B}_m^t-\mathrm{Lb}^*)+b_2\times(\tilde{B}_m^t-\mathrm{Ub}^*)\tag{2}$$

其中,$X^*=\mathrm{best}(R^{t+1}\cup B^t\cup L^t\cup T^t)$ 表示推球蜣螂更新完位置后整个蜣螂种群当前最优位置,即当前最优位置;$\tilde{B}_m^t$ 表示第 m 只繁育蜣螂第 t 次迭代后自身的历史最优位置;b_1 与 b_2 是两个规模为 $1\times D$ 的独立随机向量;D 为问题解的维度;Lb^* 和 Ub^* 分别表示用于安置育雏球的安全区域对应的下界和上界,公式如下:

$$\begin{cases}\mathrm{Lb}^*=\max(X^*\times(1-Q),\mathrm{Lb})\\ \mathrm{Ub}^*=\min(X^*\times(1+Q),\mathrm{Ub})\end{cases}\tag{3}$$

其中,X^* 与公式(2)中相同,$Q=1-t/M$,M 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,当产下卵之后,育雏球会被严格放置在上述安全区域内。

蜣螂的卵在育雏球中会经历若虫、蛹阶段,发育成小蜣螂并钻出地面寻找食物,对应位置更新公式如下:

$$L_h^{t+1}=\tilde{L}_h^t+c_1\times(\tilde{L}_h^t-\mathrm{Lb}')+c_2\times(\tilde{L}_h^t-\mathrm{Ub}')\tag{4}$$

其中,$\tilde{L}_h^t$ 为第 h 只小蜣螂在第 t 次迭代后的历史最优位置,c_1 是一个符合正态分布的随机数,是各分量介于 0~1 之间的随机向量。Lb' 和 Ub' 计算公式如下:

$$\begin{cases}\mathrm{Lb}'=\max(X^b\times(1-Q),\mathrm{Lb})\\ \mathrm{Ub}'=\min(X^b\times(1+Q),\mathrm{Ub})\end{cases}\tag{5}$$

其中,$X^b=\mathrm{best}(\tilde{X}^t)$ 表示全局最优位置,Lb、Ub 及 Q 的定义与公式(3)中所述相同。

小偷蜣螂会抢夺推球蜣螂搬运的粪球,假设全

局最优位置是最有利于抢夺粪球的地方,则小偷蜣螂的位置更新公式如下:

$$T_z^{t+1} = X^b + S \times \boldsymbol{g} \times (|\tilde{T}_z^t - X^*| + |\tilde{T}_z^t - X^b|) \quad (6)$$

其中,$\tilde{T}_z^t$ 表示第 z 只小偷蜣螂在第 t 次迭代后自身的历史最优位置;S 是一个常数,取值 0.5;$\boldsymbol{g}$ 为符合正态分布的 D 维随机向量。

1.2 DBO 的拓扑结构

从 2.1 节可知,在蜣螂种群寻优过程中,4 个子群之间主要围绕全局最优、当前最优以及全局最差这 3 个位置进行信息交互,并通过不断迭代以达到对问题空间进行探索的目的,算法对应的拓扑结构[11]如图 1 所示。

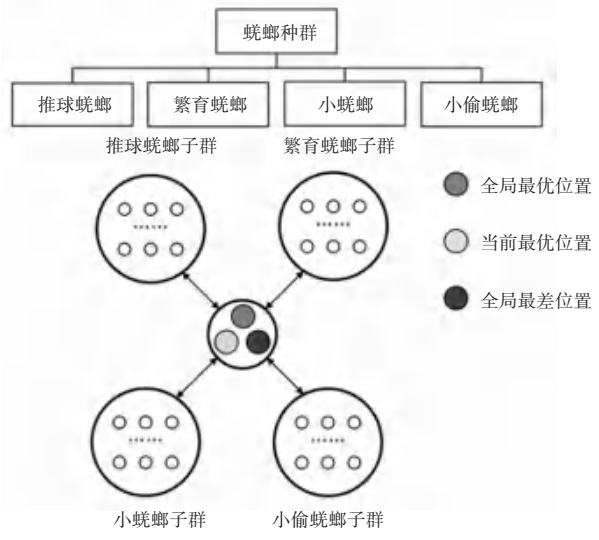


图 1 DBO 的拓扑结构图

Fig. 1 Topological map of DBO

1.3 DBO 中的探索与开发

DBO 没有明确划分探索与开发,文献[17]指出,开发是指在寻优过程中基于一些当前最优信息来寻找更优的解决方案,通常是在种群最优方案附近进行搜索,探索则旨在对问题空间进行彻底搜索并生成丰富且多样化的解决方案,以小蜣螂为例,其在更新位置时主要是在自身历史最优位置 $\tilde{L}$ 的基础上进行寻优,而不是对 X^* 或 X^b 加以利用,很明显,与 X^*、X^b 相比,$\tilde{L}$ 更多样化,因此,小蜣螂主要负责探索任务,同理可知推球蜣螂也负责探索任务,而繁育蜣螂和小偷蜣螂负责开发任务。

1.4 DBO 存在的不足

DBO 具有多个不同的子群,具有较强的可扩展性及寻优性能,但仍然存在一些问题,主要表现为:

(1)繁育蜣螂基于当前最优位置进行局部开发,以生成新的位置,但是会被严格限制在以当前最优位置 X^* 为中心的安全区域内,若整个蜣螂种群陷入局部最优,则无法找到更具潜力的全局最优开发区域,最终会被束缚在以 X^* 为中心的安全区域内开展无效开发。

(2)通过上述分析可知,推球蜣螂和小蜣螂基于自身历史最优位置对问题空间进行全局探索,但当蜣螂种群寻优长时间停滞时,自身历史最优位置也无法得到更新,在这种情况下,全局探索就丧失多样性,最终不可避免地出现早熟收敛问题。为此,在第 3 节提出了一种带扰动的自适应蜣螂优化算法,引入扰动疏散机制避免繁育蜣螂子群进行盲目开发,以获得更高的寻优精度,将自适应惯性权重与小蜣螂的位置更新公式(5)结合,使小蜣螂子群可以更好地进行全局探索,为蜣螂种群提供丰富多样化的位置信息,避免算法早熟收敛。

2 带扰动的自适应蜣螂优化算法 PADBO

2.1 扰动疏散机制

繁育蜣螂基于当前最优位置进行局部开发,得到的新位置会被限制在以当前最优位置 X^* 为中心的安全区域内。与全局历史最优位置 X^b 不同,X^* 是每次迭代中的最优位置,会随种群迭代而不断发生动态更新,这种寻优设计从本质上来看是有利于算法跳出局部最优的,但如果以 X^* 为中心的安全区域不包含精度更高的可行解,那么显然繁育蜣螂的寻优只是在做无用功,为此,本文提出一种扰动疏散机制,首先,对每只繁育蜣螂,在 X^* 基础上加上扰动因子,并使用柯西变异策略[18]得到不同的安全区域中心,将繁育蜣螂个体疏散在不同的区域进行开发,一维柯西分布曲线如图 2 所示,概率密度函数公式如下:

$$g(x) = \frac{1}{\pi} \times \frac{1}{1 + x^2}, \quad -\infty < x < \infty \quad (7)$$

每只繁育蜣螂对应的安全区域中心公式如下:

$$\text{Center}_m^{t+1} = (X^* + \text{ones}(1,D) \odot B_{m,\xi}^t) \times (1 + \text{cauchy}) \quad (8)$$

其中,Center_m^{t+1} 表示第 m 只繁育蜣螂在第 $t+1$ 次迭代时选择的安全区域中心;$\text{ones}(1,D)$ 表示一个全 1 的 D 维向量;$\odot$ 表示点乘运算;$B_{m,\xi}^t$ 为扰动因子,表示第 m 只繁育蜣螂第 t 次迭代中随机某一维度的位置($1 \leqslant \xi \leqslant D$);cauchy 为符合公式(7)中柯西分布概率密度的随机数,实际编程中通过调用编程平台的概率分布对象生成。

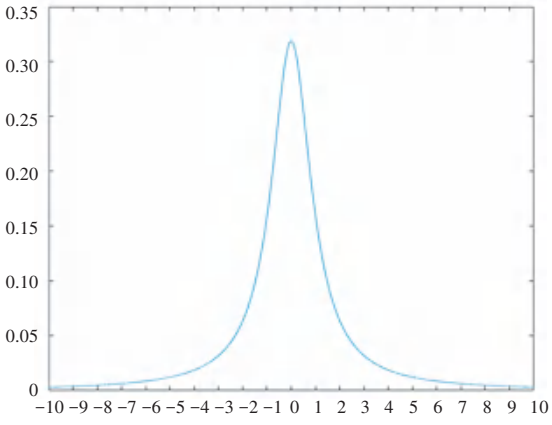


图 2　柯西分布曲线

Fig. 2　Cauchy distribution curve

在得到安全区域中心位置信息后，繁育蜣螂位置更新公式如下：

$$B_m^{t+1} = \text{Center}_m^{t+1} + 2 \times q \times (\tilde{B}_m^t - \text{Center}_m^{t+1}) \quad (9)$$

其中，q 表示一个随机数，且 $q \in [0,1]$。

更新完位置后，对应安全区域边界上下界约束公式如下式：

$$\begin{cases} \text{Lb}_m^{t+1} = \max(\text{Center}_m^{t+1} + (1-Q),\ \text{Lb}) \\ \text{Ub}_m^{t+1} = \min(\text{Center}_m^{t+1} \times (1+Q),\ \text{Ub}) \end{cases} \quad (10)$$

其中，Lb_m^{t+1}、Ub_m^{t+1} 分别表示第 m 只繁育蜣螂第 $t+1$ 次迭代时的安全区域下边界和上边界，Q 和 Lb、Ub 的定义与公式(3)中相同。

从图 2 可知，符合柯西分布的随机数较多集中在[−2,2]范围内，这能让繁育蜣螂分散在安全点周围区域避免盲目聚集，提高开发效率，当随机数处于[−2,2]范围之外的时候，能够起到一定的全局探索作用，所以上述所提的扰动疏散策略可以对算法的局部开发与全局探索进行适度的平衡。

2.2　惯性权重调整的小蜣螂位置更新

DBO 中，小蜣螂子群基于自身历史最优位置，以单一的位置更新方式对空间进行探索，为种群提供寻优信息，当算法寻优长时间停滞时，小蜣螂子群将很难为蜣螂种群提供有用的寻优信息。受到一些改进粒子群算法中的启发，在小蜣螂位置更新公式(4)中引入自适应惯性权重策略，权重计算方式如下：

$$w_h^{t+1} = \begin{cases} w_{\min} + (w_{\max} - w_{\min}) \times \dfrac{f(L_h^t) - f'_{\min}}{f'_{\text{avg}} - f'_{\min}}, & f(L_h^t) \leqslant f'_{\text{avg}} \\ w_{\max}, & f(L_h^t) > f'_{\text{avg}} \end{cases} \quad (11)$$

其中，w_h^{t+1} 表示第 h 只小蜣螂进入第 $t+1$ 次迭代时的惯性权重；$w_{\min}$ 和 $w_{\max}$ 分别为最小和最大权重参数；$f(L_h^t)$ 表示第 h 只小蜣螂第 t 次迭代位置对应的适应度；f'_{avg} 和 $f'_{\min}$ 分别表示第 t 次迭代所有蜣螂位置的平均适应度和最小适应度。文献[19]在对粒子群算法进行改进时指出，$X = X' + V$ 的位置更新方式有利于局部开发，$X = w \times X' + (1-w) \times V$ 的位置更新方式有利于全局探索，这里 X' 表示个体对空间进行探索时的基准点，V 在粒子群算法中表示速度算子，在其他算法中可以是移动步长或其他位置更新算子，受到该文献启发，小蜣螂的位置更新改进公式如下：

$$L_h^{t+1} = w_h^{t+1} \times \tilde{L}_h^t + (1 - w_h^{t+1}) \times (c_1 \times (w_h^{t+1} \times \tilde{L}_h^t - \text{Lb}') + \boldsymbol{c}_2 \times (w_h^{t+1} \times \tilde{L}_h^t - \text{Ub}')) \quad (12)$$

其中，w_h^{t+1} 由公式(11)计算得到，表示第 $t+1$ 次迭代时第 h 只小蜣螂的惯性权重；c_1 为正态分布的随机数；$\boldsymbol{c}_2$ 为各分量介于 0~1 之间的 D 维随机向量。其余变量均与公式(5)中相同。结合公式(11)与公式(12)可知，当小蜣螂个体适应度比种群平均适应度差的时候，会得到一个较大的惯性权重，而当个体适应度比种群平均适应度好的时候，与最小适应度的差值越小，得到的惯性权重越小，这有利于小蜣螂个体根据实际情况动态地调整全局探索的方向，为种群带来有希望位置信息，避免早熟收敛。

2.3　PADBO 的流程图和步骤

PADBO 的流程图如图 3 所示，具体步骤如下：

Step 1　设置算法参数，随机初始化具有 N 只个体的蜣螂种群 X，依次划分指定数量的个体到 4 个不同的子群中；

Step 2　使用公式(1)更新推球蜣螂子群中每只个体的位置并计算适应度值；

Step 3　对于繁育蜣螂子群中每只个体，利用公式(9)更新位置，并使用公式(10)约束上下边界，计算适应度值；

Step 4　对于小蜣螂子群中的每只个体，使用公式(12)更新位置，计算适应度值；

Step 5　对于小偷蜣螂子群中的每只个体，使用公式(6)更新位置，计算适应度值；

Step 6　更新所有蜣螂个体历史最优位置，若有位置比全局最优 X^b 更优，则更新 X^b；

Step 7　若算法达到迭代次数则终止运行，返回全局最优位置 X^b，否则转到 Step 2。

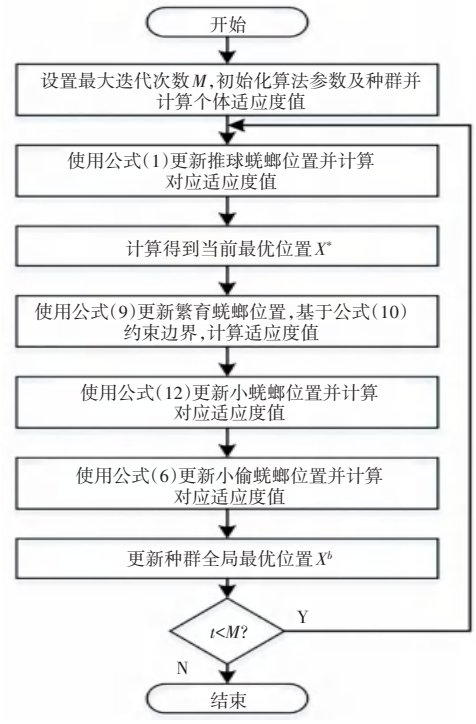


图 3 PADBO 的流程图

Fig. 3 Flow chat of PADBO

3 实验分析

3.1 测试函数及实验环境

为了测试第 2 节中提出改进算法的寻优性能，本文选取了 8 个经典的测试函数进行测试，函数的信息见表 1，其中 $F_1 \sim F_4$ 为单峰测试函数，用于测试算法在局部精细化搜索的能力，$F_5 \sim F_8$ 为多峰测试函数，用于测试算法在问题空间中进行全局探索的能力. 本文所有实验都是在 64 位 Windows11 操作系统上进行的，主要配置为 2.60 GHz 主频 13th Gen Intel(R) Core(TM) i7-13650 HX CPU，内存 16 GB，编程软件为 MATLAB R2022a。

表 1 测试函数信息

Table 1 Information about testing functions

函数	名称	搜索空间	理论最优值
F_1	Sphere Function	[-100,100]	0
F_2	Schwefel Pro. 2.22	[-10,10]	0
F_3	Schwefel Pro. 1.2	[-100,100]	0
F_4	Schwefel 2.21	[-100,100]	0
F_5	Alpine Function	[-30,30]	0
F_6	Schwefel Pro. 2.26	[-100,100]	-12 569.486 6
F_7	Penalized Func. 1	[-500,500]	0
F_8	Penalized Func. 2	[-50,50]	0

3.2 对比算法与实验设置

为了验证本文所提 PADBO 的优越性，本文选取 TCSPSO[20]、SO[9]、WHO[10]、DBO[11] 以及 IDBO[12] 与之进行对比，这些对比算法的参数均按照原文设置，具体见表 2。

表 2 算法参数设置

Table 2 Parameter settings of algorithms

算法	参数	参数设置
PADBO	偏转系数 k	0.1
	没有遇到障碍的概率 γ	0.9
	常数 u 与 s	0.3, 0.5
	惯性权重 $w_{\min}$、$w_{\max}$	0.4, 1.4
IDBO[12]	策略选择概率 P_s	0.5
	偏转系数 k	0.1
	没有遇到障碍的概率 γ	0.9
	常数 u 与 s	0.3, 0.5
DBO[11]	偏转系数 k	0.1
	没有遇到障碍的概率 γ	0.9
	常数 u 与 s	0.3,0.5
WHO[10]	种马比例 ps	0.2
	交叉概率 pc	0.13
SO[9]	系数 c_1、c_2、c_3	0.5, 2, 0.05
	threshold 的 2 个阈值	0.25, 0.6
TCSPSO[20]	惯性权重范围	[0.4,0.9]
	系数 c_1	2
	系数 c_2	2

所有参与对比的算法迭代次数都为 500 次，种群规模设为 30，8 个测试函数维度取 30，每个算法都独立重复运行 30 次，记录 30 次运行结果的平均值、标准差，以及单独运行 1 次消耗的时间，绘制 30 次运行的平均迭代曲线反映算法的收敛速度及趋势。

3.3 结果分析

实验结果见表 3，平均迭代曲线如图 4 所示。

$F_1 \sim F_4$ 是单峰函数，只有一个全局最优解，从表 3 可知，对于测试函数 $F_1 \sim F_4$，PADBO 可以稳定地收敛到最优值，结合平均迭代曲线图 4(a)~(d)可知，与其他算法相比，PADBO 在 $F_1 \sim F_4$ 上不仅具有最高的求解精度，同时还具有较快且平稳的收敛速度.

$F_5 \sim F_8$ 是多峰函数，具有多个局部极小值，对于算法多样化探索及跳出局部最优的能力是不小的考验. 从表 3 可知，对于函数 F_5，WHO、SO 以及 DBO 表现欠佳，与其他一些算法相比，平均值指标与最优值指标具有明显差异，其中，SO 和 DBO 可以找到一定精度的解，说明还是具有局部逃逸能力的，但是与平均值指标之间较大的数量级差异说明这种

局部逃逸能力并不稳定，而 PADBO 在各项指标上都优于其他算法并且可以稳定地达到最优，从图 4(e)可以看出，其收敛过程没有出现较为明显的停滞，说明其跳出局部最优的能力及稳定性也要强于其他对比算法，这也证明 PADBO 可以较为有效地避免标准 DBO 易早熟、收敛于局部最优的不足。对于函数 F_6，结合曲线与表中数据可知，PADBO 可以较快地找到与理论最优最接近的解，说明其全局探索与局部开发能力兼备，而其他对比算法中，只有 SO 在最优指标上可以接近理论最优，但是仍有一定偏差，其他指标则表明 SO 存在局部开发不足的问题，从而导致整体上寻优结果不如 PADBO。在函数 F_9 上，PADBO 在各维度全方位领先其他算法。在函数 F_{10} 上，PADBO 表现优异，各项评价指标都要优于参与对比的其他算法，值得称道的是其在各维度所取得的最优值都要领先其他算法超过 20 个数量级，说明改进策略可以很好地帮助算法从局部最优位置逃离并获得更高的求解精度。此外，对比算法中同样作为 DBO 改进算法的 IDBO，在单峰和多峰函数上相较于 DBO 都取得了更高的精度，说明其改进方案一定程度上缓解了 DBO 存在的早熟收敛、全局探索能力不佳等问题，

但是，从整体上而言，IDBO 的表现与 PADBO 相比仍显不足，这也证明了 PADBO 的优越性。

运行时间上，PADBO 与 DBO 相比增加了一些运行时间，主要是 2 个部分带来的：(1)每次迭代时扰动疏散策略都需要使用 Matlab 提供的概率分布对象产生符合柯西分布的随机数；(2)对于每只小蜣螂个体，在每次迭代时都需要计算一个新的惯性权重. 整体上而言，PADBO 取得了更高精度的解，消耗的时间也不算太多，而且与部分算法相比，运行时间其实是相近的，所以 PADBO 兼顾了寻优精度与运行时间消耗。

表 3　测试函数上的实验结果

Table 3　Experimental results on testing functions

函数	指标	PADBO	IDBO	DBO	WHO	SO	TCSPSO
F_1	平均值	**0**	7.792 9e−307	1.384 9e−105	7.346 5e−42	1.023 5e−94	2.258 4e−43
	标准差	**0**	**0**	7.585 4e−105	3.877 0e−41	2.753 4e−94	1.109 8e−42
	运行时间/s	0.117 5	0.048 7	0.039 1	0.110 3	0.055 6	0.045 5
F_2	平均值	**0**	4.510 5e−162	8.434 8e−60	4.182 6e−25	4.732 0e−43	2.989 5e−27
	标准差	**0**	2.465 2e−161	4.575 0e−59	1.168 2e−24	7.164 8e−43	8.232 2e−27
	运行时间/s	0.127 0	0.050 8	0.038 8	0.112 3	0.030 9	0.047 9
F_3	平均值	**0**	6.094 7e−261	1.819 4e−57	1.642 9e−24	6.435 1e−56	9.849 3e−36
	标准差	**0**	**0**	9.965 1e−57	6.214 9e−24	2.021 9e−55	2.490 4e−35
	运行时间/s	0.186 6	0.125 3	0.099 7	0.176 8	0.085 2	0.166 7
F_4	平均值	**0**	2.477 0e−144	8.045 4e−51	6.023 9e−17	1.850 8e−40	1.965 9e−21
	标准差	**0**	1.356 7e−143	4.387 2e−50	1.469 1e−16	4.498 7e−40	6.712 7e−21
	运行时间/s	0.129 3	0.056 3	0.032 9	0.112 5	0.029 7	0.043 2
F_5	平均值	**0**	2.083 4e−158	4.874 1e−04	0.042 7	0.068 6	2.065 4e−28
	标准差	**0**	1.087 0e−157	0.001 4	0.228 8	0.138 9	5.265 5e−28
	运行时间/s	0.131 8	0.057 9	0.037 2	0.110 6	0.033 7	0.049 6
F_6	平均值	**−1.256 9e+04**	−1.115 0e+04	−8.143 0e+03	−8.821 5e+03	−1.252 1e+04	−7.167 2e+03
	标准差	**0.066 4**	1.136 1e+03	1.536 5e+03	580.025 8	63.374 5	695.816 5
	运行时间/s	0.139 0	0.067 5	0.046 0	0.130 1	0.037 3	0.053 3
F_7	平均值	**3.428 5e−07**	1.051 7e−05	5.676 8e−04	0.034 4	0.063 5	9.025 9e−04
	标准差	**1.031 8e−06**	4.272 5e−05	0.002 0	0.118 6	0.121 0	8.350 3e−04
	运行时间/s	0.239 5	0.200 9	0.142 5	0.227 6	0.141 7	0.225 8
F_8	平均值	**3.422 8e−07**	3.723 7e−04	0.616 5	0.036 3	0.383 6	0.186 6
	标准差	**1.443 1e−06**	0.002 0	0.369 6	0.046 0	0.746 1	0.188 3
	运行时间/s	0.233 8	0.205 0	0.165 5	0.236 0	0.137 0	0.237 6

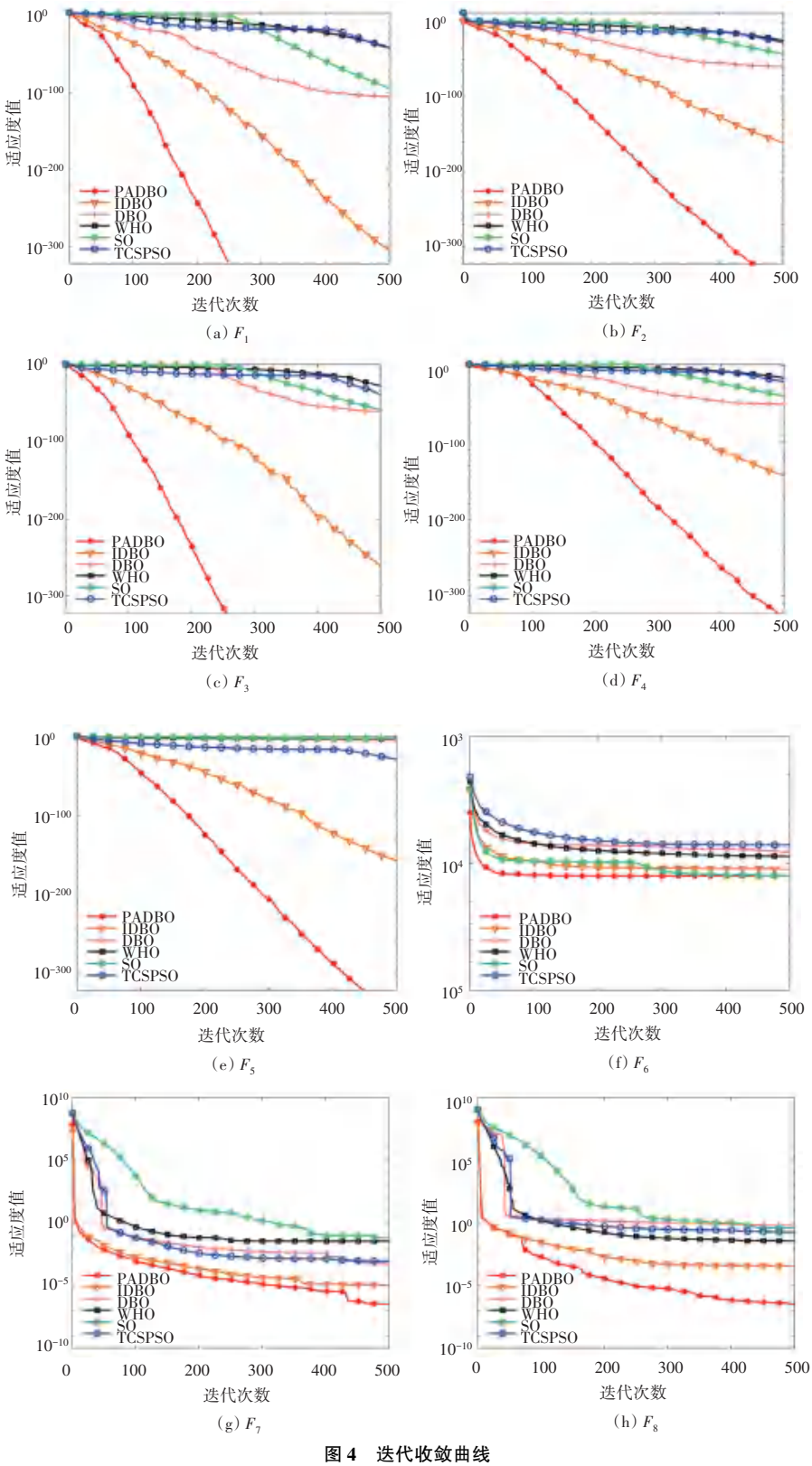


图 4 迭代收敛曲线

Fig. 4 Iterative convergence curve

3.4 减速机质量最小化问题

减速机质量最小化问题是机械系统中一个经典的约束优化问题. 减速机质量最小化问题[21]有七个决策变量,分别是齿面宽度 b、齿轮模数 m、小齿轮的齿数 z、轴承之间第一根轴与第二根轴的长度 l_1 和 l_2、轴承之间第一根轴和第二根轴的直径 d_1 及 d_2, 其优化目标是在 11 个约束条件下使得设计出来的减速机质量最小,该问题的设计模型如图 5 所示,对应目标

函数 f、决策变量范围及约束条件 g 如下所示：

$$x=[x_1,x_2,x_3,x_4,x_5,x_6,x_7]=[b,m,z,l_1,l_2,d_1,d_2]$$

$$2.6\leqslant x_1\leqslant 3.6,\ 0.7\leqslant x_2\leqslant 0.8,\ 17\leqslant x_3\leqslant 28;$$

$$7.3\leqslant x_4,x_5\leqslant 8.3,2.9\leqslant x_6\leqslant 3.9,5\leqslant x_7\leqslant 5.5$$

$$f(\boldsymbol{x})=0.785\,4x_2^2x_1(14.933\,4x_3-43.093\,4+3.333\,3x_3^2)+0.785\,4(x_5x_7^2+x_4x_6^2)-1.508x_1(x_7^2+x_6^2)+7.477(x_7^3+x_6^3)$$

s.t.

$$\begin{cases}g_1(\boldsymbol{x})=-x_1x_2^2x_3+27\leqslant 0\\ g_2(\boldsymbol{x})=-x_1x_2^2x_3^2+397.5\leqslant 0\\ g_3(\boldsymbol{x})=-x_2x_6^4x_3x_4^{-3}+1.93\leqslant 0\\ g_4(\boldsymbol{x})=-x_2x_7^4x_3x_5^{-3}+1.93\leqslant 0\\ g_5(\boldsymbol{x})=x_6^{-3}\sqrt{16.91\times10^6+(745x_4x_2^{-1}x_3^{-1})^2}\leqslant 110\\ g_6(\boldsymbol{x})=x_7^{-3}\sqrt{157.5\times10^6+(745x_5x_2^{-1}x_3^{-1})^2}\leqslant 85\\ g_7(\boldsymbol{x})=x_2x_3-40\leqslant 0\\ g_8(\boldsymbol{x})=-x_1x_2^{-1}+5\leqslant 0\\ g_9(\boldsymbol{x})=x_1x_2^{-1}-12\leqslant 0\\ g_{10}(\boldsymbol{x})=1.5x_6-x_4+1.9\leqslant 0\\ g_{11}(\boldsymbol{x})=1.1x_7-x_5+1.9\leqslant 0\end{cases}$$

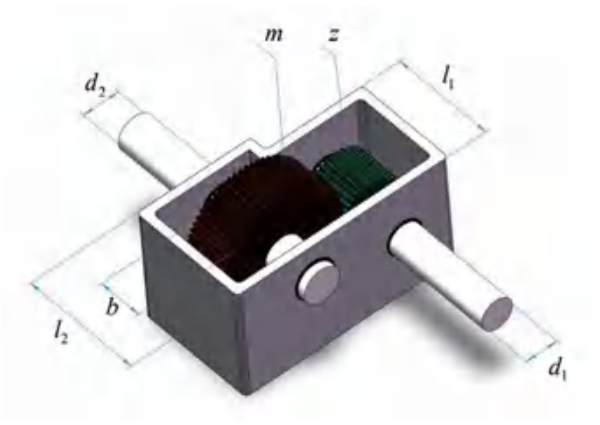


图 5　减速机设计模型

Fig. 5　Reduction gear design model

选取 TCSPSO、SO、WHO、DBO 与本文所提出的 PADBO 用于求解该问题，所得的最优方案见表 4。从表 4 中可知，部分算法如 DBO、WHO、SO，仅因部分决策变量精度不够，导致所求方案不如 PADBO，而 PADBO 在求解该问题时，变量取值更精确，因此得到了质量最小的设计方案，说明 PADBO 可以精确有效地求解实际问题。

表 4　不同算法在减速机质量最小化问题上的寻优结果

Table 4　Optimization results of different algorithms for the mass minimization problem of gearboxes

算法	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(x)$
PADBO	3.500 0	0.700 0	17.000 0	7.300 0	7.715 3	3.350 5	5.286 7	**2 994.424 5**
DBO	3.500 0	0.700 0	17.000 0	7.300 0	7.715 3	3.352 5	5.286 7	3 003.759 8
WHO	3.500 6	0.700 0	17.000 0	7.300 0	7.715 3	3.350 5	5.286 7	2 994.680 4
SO	3.500 0	0.700 0	17.000 0	7.556 8	7.716 9	3.351 0	5.286 7	2 996.851 9
TCSPSO	3.565 5	0.701 3	17.019 8	7.703 8	8.040 2	3.395 5	5.349 0	3 092.277 4

4　结束语

针对 DBO 存在无效开发、全局探索多样性缺失的问题，提出了一种带扰动的自适应蜣螂优化算法，该方法引入扰动疏散机制，缓解了繁育蜣螂子群盲目集中、无效寻优导致的开发效率低的问题，提高了算法寻优的精度；通过自适应惯性权重对小蜣螂子群的位置更新公式进行了改进，使小蜣螂个体可以动态地调整方向，实现探索与开发之间的平衡。在 8 个测试函数上的寻优结果验证了改进方案的有效性，减速机质量最小化问题的求解结果进一步验证了算法的实用性。此外，在设计 DBO 改进方案时，本文曾试图同时设计新的推球蜣螂子群和小偷蜣螂子群的位置更新方法，但实验结果表明，4 个子群中蜣螂的位置同时更新未必能取得更优的实验结果，因此，未来工作将围绕交替更新或按权重更新各子群蜣螂位置的方法展开深入研究。

参考文献

[1] KENNEDY J, EBERHART R. Particle swarm optimization[C]// Proceedings of Internation Conference on Neural Networks. Piscataway, NJ: IEEE, 1995, 4: 1942-1948.

[2] CHAUHAN D, SHIVANI C, CHENG R. Competitive swarm optimizer: A decade survey [J]. Swarm and Evolutionary Computation, 2024, 87: 101543.

[3] DHIMAN G, KUMAR V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems [J]. Knowledge- based Systems, 2019, 165: 169-196.

[4] 曾庆丰，蔡延光，胡城，等. 蝗虫优化算法综述[J]. 自动化与信息工程，2024，45(1)：1-11.

[5] 罗育林，胡长江，邓敦杰. 基于蝙蝠优化算法的智能机器人路径规划方法[J]. 沈阳工业大学学报，2024，46(2)：191-196.

[6] BENYAMIN A, FARHAD S G, NIMA K, et al. Mountain gazelle optimizer: A new nature-inspried metaheuristic algorithm

for global optimization problems[J]. Advances in Engineering Software, 2022, 174:103282.
[7] FARSHAD R, HAMID R S, MOHAMED A E, et al. GMO: Geometric mean optimizer for solving engineering problems[J]. Soft Computing, 2023, 27(15):10571-10606.
[8] BENYAMIN A, NIMA K, SAEID B, et al. Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning[J]. Cluster Computing, 2024, 27(5):5235-5283.
[9] HASHMIN F A, HUSSIEN A G. Snake optimizer: A novel meta-heuristic optimization algorithm[J]. Knowledge-based Systems, 2022, 242:108320.
[10] NARUEI I, KEYNIA F. Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems[J]. Engineering with Computers, 2022, 38(S4): 3025-3056.
[11] 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.
[12] LI Y, ZHANG J L, YAN Y T, et al. Enhancing network intrusion detection through the application of the dung beetle optimized fusion model[J]. IEEE Access, 2024, 12:9483-9496.
[13] LI Y H, SUN K X, YAO Q, et al. A dual-optimization wind speed forecasting model based on deep learning and improved dung beetle optimization algorithm[J]. Energy, 2024, 286:129604.
[14] 秦喜文, 冷春晓, 董小刚. 基于混合策略的蜣螂优化算法研究[J]. 吉林大学学报(信息科学版), 2024, 42(5): 829-839. DOI:10.19292/j.cnki.jdxxp.20240328.001.
[15] 孙仟硕, 王英博. 融合多策略的改进蜣螂优化算法及其应用[J]. 信息与控制, 2024, 53(5): 631-641. DOI:10.13976/j.cnki.xk.2024.3194.
[16] 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]. 上海大学学报, 2024, 46(5): 580-588. DOI:10.13255/j.cnki.jusst.20230404002.
[17] SAKA M P, HASANCEBI O, GEEM Z W. Metaheuristics in structural optimization and discussions on harmony search algorithm[J]. Swarm and Evolutionary Computation, 2016, 28: 88-97.
[18] ZHU Y B, LI W S, LI T. A hybrid artificial immune optimization for high-dimensional feature selection[J]. Knowledge-based Systems, 2023, 260: 110111.
[19] LIU H, ZHANG X W, TU L P. A modified particle swarm optimization using adaptive strategy[J]. Expert Systems with Applications, 2020, 152: 113353.
[20] ZHANG X W, LIU H, ZHANG T, et al. Terminal crossover and steering-based particle swarm optimization algorithm with disturbance[J]. Applied Soft Computing, 2019, 85:105841.
[21] XIAO J H, LOU X J, SHI B Y, et al. An improved hybrid membrane algorithm based on hormone modulation mechanism for engineering design problems[J]. Expert Systems with Applications, 2023, 227: 120240.