计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(2):224−231 [doi: 10.15888/j.cnki.csa.009397]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

多策略融合改进的蜣螂优化算法①

王乐遥, 顾　磊

(南京邮电大学 计算机学院, 南京 210023)
通信作者: 王乐遥, E-mail: wly39201@163.com

摘　要: 针对标准蜣螂优化算法 (DBO) 存在的全局探索能力欠缺、收敛精度低及易陷入局部最优等不足, 提出了一种融合多策略的改进蜣螂优化算法 (MSDBO). 首先, 引入社会学习策略引导推球蜣螂进行位置更新, 提高了算法全局探索能力, 避免算法陷入局部最优; 其次, 提出一种方向跟随策略, 建立起小偷蜣螂与推球蜣螂个体间的交互, 提高了寻优精度; 最后, 引入环境感知概率, 引导小偷蜣螂合理采用方向跟随策略, 兼顾了性能与时间消耗. 在 12 个基准测试函数上进行求解分析, 并与其他优化算法进行对比, 证明了 MSDBO 的寻优性能明显优于对比算法, 在压力容器设计优化问题上的结果验证了 MSDBO 求解实际工程约束优化问题的有效性.

关键词: 蜣螂优化算法; 社会学习; 方向跟随; 环境感知概率; 基准测试函数; 压力容器设计

引用格式: 王乐遥,顾磊.多策略融合改进的蜣螂优化算法.计算机系统应用,2024,33(2):224–231. http://www.c-s-a.org.cn/1003-3254/9397.html

Improved Dung Beetle Optimization Algorithm with Multi-strategy

WANG Le-Yao, GU Lei

(School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: An improved dung beetle optimization algorithm integrating multiple strategies (MSDBO) is proposed to solve the problems of weak global exploration ability, low convergence accuracy, and easy capture by local optimum solution. Firstly, this study introduces the social learning strategy to guide the dung beetle to update its position, which improves the global exploration ability of the algorithm and avoids the algorithm falling into local optimal. Secondly, the study proposes a direction-following strategy to establish the interaction between the thief and the ball-rolling dung beetle, which improves the accuracy of optimization. Finally, taking into account the performance and time consumption, it introduces environment-aware probability to guide the thief to adopt the direction-following strategy reasonably. Several optimization algorithms are selected and compared with MSDBO. By solving and analyzing 12 benchmark test functions, it is proved that the optimization performance of MSDBO is significantly better than that of the comparison algorithm. The results of pressure vessel design optimization verify the effectiveness of MSDBO in solving practical engineering constraint optimization problems.

Key words: dung beetle optimization (DBO) algorithm; social learning; direction following; environment perception probability; benchmark test function; pressure vessel design

优化是指对特定问题的解决方案进行调整和改进, 使其相关指标达到最优或近似最优的过程, 使用优化技术可以让人类更加合理有效地可持续使用资源. 传统的优化算法通常根据问题的情况使用特定的数学模型或求解方法进行寻优, 但这些算法通常存在一些局限性, 例如依赖于梯度信息、面对非线性复杂问题难

① 基金项目: 国家自然科学基金 (61972210)
收稿时间: 2023-08-10; 修改时间: 2023-09-09; 采用时间: 2023-09-26; csa 在线出版时间: 2023-12-18
CNKI 网络首发时间: 2023-12-19

以处理[1]等, 为此, 启发式算法应运而生并被广泛应用于各领域. 启发式算法通过模拟自然界的演化[2,3]、物理现象[4–6]、人类的行为[7,8]等, 来探索问题空间中的最优解决方案, 往往能够有效地在实际执行时间内解决传统算法无法解决的问题.

群体智能优化算法就是一种典型的启发式算法, 是一类受自然界中各种生物的群体行为启发而创造出的寻优方法. 种群个体在遵循相关规则进行交互的过程中实现对全局最优解的探索. 典型的群体智能优化算法有灰狼优化算法 (grey wolf optimization, GWO)[9]、鲸鱼优化算法 (whale optimization algorithm, WOA)[10]、哈里斯鹰优化算法 (Harris hawks optimization, HHO)[11]等. 根据没有免费午餐定理 (no free lunch)[12], 没有一种优化算法可以解决所有优化问题, 为此, 越来越多的优化算法被提出用于解决新的优化问题, 例如沙丘猫群优化算法 (sand cat swarm optimization, SCSO)[13]、海马优化算法 (sea-horse optimization, SHO)[14]等.

蜣螂优化算法 (dung beetle optimization, DBO)[15]是 Xue 等人于 2022 年提出的一种新颖的优化算法. 该算法模拟了蜣螂推球、跳舞、觅食、偷窃以及繁殖等行为, 将蜣螂种群分为推球蜣螂、育雏球、小蜣螂以及小偷蜣螂 4 个角色, 分别按照不同的搜索策略对问题空间进行搜索. 蜣螂优化算法有着独特的结构, 其关注重点不再是种群的整体任务如觅食或躲避天敌等, 算法的主要思想是种群不同角色通过不同的生存行为及相互之间的经验交互获得有利于自身的位置信息, 蜣螂的位置越好表示其适应环境的生存能力越优秀, 每只蜣螂的位置代表问题的一个解, 最终, 最优蜣螂的位置即为所求问题最优解.

DBO 独特的结构使其具有不错的寻优性能及实际应用潜力. 李晴[16]将 DBO 用于水质 COD 预测模型优化; 董奕含等人[17]引入 Halton 序列初始化 DBO 并用于瑞雷波频散曲线反演; 潘志远等人[18]将 DBO 用于优化 DV-Hop 定位算法; Zhu 等人[19]引入量子计算等策略对 DBO 进行改进并将其用于多个实际工程约束优化问题的求解. 上述文献证明了 DBO 的实际应用价值, 相关改进也提升了 DBO 的寻优性能, 但是 DBO 在全局探索、寻优精度等方面仍有较大提升空间. 为此, 本文提出了一种多策略融合改进的蜣螂优化算法 (multi-strategy fusion improved DBO, MSDBO), 首先, 使用社会学习策略引导推球蜣螂进行位置更新, 提升算法全局探索能力; 其次, 引入方向跟随策略, 增强算法局部开发能力, 提高了收敛精度; 最后, 引入环境感知概率控制方向跟随策略的调用频率, 平衡算法性能与运行时间消耗. 在 12 个基准测试函数与压力容器设计优化问题上的结果证明了改进算法的有效性与实用性.

1 蜣螂优化算法 (DBO)

DBO 根据蜣螂的不同社会分工, 将种群分为推球蜣螂、育雏球、小蜣螂及小偷蜣螂 4 个角色, 当种群数量$N=30$时文献[15]将这 4 种蜣螂的数量分别设定为 6, 6, 7, 11 只. 蜣螂种群的位置表示为: $X=\{X_i|i=1,2,\cdots,N_{\text{all}}\}$. 为区分起见, 4 种蜣螂子种群的位置使用不同符号表示为$R=\{R_e|e=1,2,\cdots,N_{\text{roll}}\}$、$B=\{B_m|m=1,2,\cdots,N_{\text{ball}}\}$、$L=\{L_h|h=1,2,\cdots,N_{\text{little}}\}$以及$T=\{T_z|z=1,2,\cdots,N_{\text{thief}}\}$. 则有$X=R\cup B\cup L\cup T$且$N_{\text{roll}}+N_{\text{ball}}+N_{\text{little}}+N_{\text{thief}}=N_{\text{all}}$. 若优化问题维度为$D$, 对应目标函数为$f$, 则每只蜣螂的位置即问题的一个解表示为$X_i=\{x_{i,1},x_{i,2},\cdots,x_{i,D}\}$, 个体适应度值即最优值表示为$f(X_i)$, R_e、B_m、L_h、T_z及对应适应度值的表示与X_i类似. 算法按照不同策略更新不同角色蜣螂的位置, 根据适应度评估蜣螂的生存能力, 假设适应度值越小表示优化效果越好, 则适应度值越小表明蜣螂所处位置越佳, 其生存能力越优秀, 则最优及最差位置的数学表示如下:

$$\begin{cases} X^b=\{X_i\in X,i=1,2,\cdots,N|\forall X_j,f(X_i)\leqslant f(X_j)\} \\ X^w=\{X_i\in X,i=1,2,\cdots,N|\forall X_j,f(X_j)\leqslant f(X_i)\} \end{cases}$$

最终得到的最优位置X^b即为所求问题的最优解.

1.1 推球蜣螂位置更新

蜣螂会把动物粪便滚成球状并利用天体线索如太阳月亮等进行导航以实现有效的搬运. 当遇到障碍物时, 蜣螂通常会爬到粪球上跳舞决定新的移动方向.

为模拟推球行为, 蜣螂在整个搜索空间沿着给定方向移动, 推球蜣螂位置更新分为有障碍物与无障碍物两种情况, 当$\lambda<\gamma$时无障碍状态, 这里λ为随机数且$\lambda\in[0,1]$, $\gamma=0.9$, 位置更新公式如下:

$$R_{\text{new},e}^{t+1}=R_e^t+\alpha\times k\times R_e^{t-1}+u\times\Delta x \tag{1}$$

$$\Delta x=\left|R_e^t-X^w\right| \tag{2}$$

其中, t表示当前迭代次数, R_e^t表示第e只推球蜣螂在第t次迭代后的位置. k为常数且$k\in(0,0.2]$, 表示位置偏转系数. α是一个自然系数用于模拟一些自然因素对于移

动方向的影响, 当$\lambda<\eta$时α为1否则α为−1, 其中η为随机数且$\eta\in[0,1]$, u为常数且$u\in[0,1]$. X^w代表全局最差位置, Δx用于模拟光照强度的变化, Δx值越高意味着光源越弱, 蜣螂的路线越弯曲. 当$\lambda\geqslant\gamma$时, 为有障碍物状态, 引入正切函数模拟蜣螂通过跳舞确定偏转角更改方向的行为, 位置更新如下:

$$R_{\text{new},e}^{t+1}=R_e^t+\tan(\theta)\left|R_e^t-R_e^{t-1}\right| \tag{3}$$

其中, θ为弧度表示的偏转角, θ为随机数且$\theta\in[0,\pi]$, 若θ为$0,\pi/2,\pi$, 则不更新位置.

上述位置更新公式中$R_{\text{new},e}^{t+1}$为个体第$t+1$次迭代时得到的候选位置, 最终将与历史最优位置对比, 择优保留, 即: 若$f(R_{\text{new},e}^{t+1})>f(R_e^t)$ 则$R_e^{t+1}=R_e^t$, 否则$R_e^{t+1}=R_{\text{new},e}^{t+1}$. 其他蜣螂位置更新类似, 不再重复说明.

1.2 育雏球位置更新

蜣螂收集的粪球一部分作为食物, 另外一部分则推到安全的地方用于产卵, 作为育雏球繁育下一代, 育雏球所在的区域边界被严格限制如下:

$$\begin{aligned}Lb^*&=\max(X^{b*}\times(1-Q),Lb)\\Ub^*&=\min(X^{b*}\times(1+Q),Ub)\end{aligned} \tag{4}$$

其中, Ub^*与Lb^*分别表示产卵区域的上下界. X^{b*}表示种群所有蜣螂当前的最优位置, $Q=1-t/T$, T表示最大迭代次数, Ub与Lb表示优化问题的上下界.

将育雏球推到确定的产卵区域后, 雌性蜣螂就会在里面产卵, 每只雌性蜣螂在每次迭代过程中只产1颗卵, 育雏球的位置更新如下所示:

$$B_{\text{new},m}^{t+1}=X^{b*}+a_1\times(B_m^t-Lb^*)+a_2\times(B_m^t-Ub^*) \tag{5}$$

其中, B_m^t表示第m只育雏球在第t次迭代后的位置, a_1和a_2是两个大小为$1\times D$的独立随机向量, D为优化问题解的维度.

1.3 小蜣螂位置更新

育雏球中的小蜣螂发育成熟后, 会钻出来寻找食物, 为此需要建立最优觅食区域引导他们觅食以达到对空间进行探索的目的. 最优觅食区域边界定义如下:

$$\begin{aligned}Lb'&=\max(X^b\times(1-Q),Lb)\\Ub'&=\min(X^b\times(1+Q),Ub)\end{aligned} \tag{6}$$

其中, X^b表示全局最优位置, Ub'和Lb'分别表示最优觅食区域的上下界, Q、Lb及Ub的定义与式 (4) 中相同, 小蜣螂位置更新公式如下:

$$L_{\text{new},h}^{t+1}=L_h^t+C_1\times(L_h^t-Lb')+C_2\times(L_h^t-Ub') \tag{7}$$

其中, L_h^t为第h只小蜣螂在第t次迭代后的位置, C_1是一个符合正态分布的随机数且$C_1\in[0,1]$, C_2是各分量介于0到1之间的随机向量.

1.4 小偷蜣螂位置更新

并不是所有蜣螂都会努力地推球, 有些蜣螂会去偷窃推球蜣螂的粪球, 假设全局最优位置是最适合它们偷窃的位置, 那么小偷蜣螂的位置更新公式如下:

$$T_{\text{new},z}^{t+1}=X^b+S\times g\times\left(\left|T_z^t-X^{b*}\right|+\left|T_z^t-X^b\right|\right) \tag{8}$$

其中, X^b表示全局最优位置, X^{b*}的含义与式 (4) 中的描述相同. T_z^t表示第z只小偷蜣螂在第t次迭代后的位置信息, S为常数且设置为0.5, g为各分量介于0到1之间的D维随机行向量.

1.5 标准 DBO 算法步骤

步骤 1. 设置最大迭代次数为T, 种群大小为N, 随机初始化种群并计算个体对应的适应度值.

步骤 2. 更新推球蜣螂位置, 若$\lambda<\gamma$, 为无障碍状态使用式 (1) 更新位置, 反之为有障碍状态按照式 (3) 更新, 其中λ为随机数且$\lambda\in[0,1]$, $\gamma=0.9$.

步骤 3. 根据式 (5) 更新育雏球位置并使用式 (4) 中的上下界对新位置进行边界约束.

步骤 4. 由式 (7) 更新小蜣螂位置.

步骤 5. 由式 (8) 更新小偷蜣螂位置.

步骤 6. 更新全局最优位置X^b及最差位置X^w.

步骤 7. 判断算法是否达到迭代次数, 若是则终止运行, 返回最优位置即问题最优解否则转步骤 2.

2 多策略融合改进的蜣螂优化算法 (MSDBO)

2.1 改进原因

虽然 DBO 有着不错的寻优性能, 但是仍然存在全局探索能力欠缺、易陷入局部最优的问题. 对问题解进行精确寻优要求算法同时具有良好的全局搜索能力与局部开发能力, 全局探索能力不足会让算法的寻优在局部最优区域停滞, 而局部开发能力不足则导致算法不能在全局最优区域取得更高的精度, MSDBO 融合了社会学习策略、方向跟随策略以及环境感知概率对 DBO 进行改进, 兼顾算法的探索与开发, 提高了算法的寻优性能, 接下来具体介绍这 3 种策略.

2.2 社会学习策略

在进行位置更新时, 每只推球蜣螂个体主要依赖自身历史位置信息进行探索, 群体内部缺乏足够的信息交流, 导致对问题空间的探索过于随机、全局探索

效率低, 受到哈里斯鹰优化算法[11]中哈里斯鹰种群在探索阶段共享位置信息合作搜索猎物的策略启发, 提出了一种社会学习策略引导推球蜣螂进行位置更新, 以改善标准 DBO 全局搜索能力欠缺的问题.

与原算法不同, 在该策略中, 蜣螂遇到障碍物的概率被假设为$\gamma=0.5$. 设λ为随机数且$\lambda\in[0,1]$, 当$\lambda\leqslant\gamma$时, 生成一个随机数θ且$\theta\in[0,\pi]$, 当$\theta=0,\pi/2,\pi$时不更新位置, 否则位置更新如下:

$$R_{\text{new},e}^{t+1}=R^{b}-\bar{R}+\tan(\theta)\left|R_{e}^{t}-R_{e}^{t-1}\right| \tag{9}$$

其中, R^{b}表示第t次迭代后最优推球蜣螂此时的候选位置, 定义如式 (10) 所示, $\bar{R}$表示其他推球蜣螂此时的平均候选位置, 定义如式 (11) 所示, 式 (9) 用于替换式 (3).

$$R^{b}=\{R_{\omega}\in R_{\text{new}},\omega=1,2,\cdots,N_{\text{roll}}|\forall\beta,f(R_{\omega}^{t})\leqslant f(R_{\beta}^{t})\} \tag{10}$$

$$\bar{R}=\frac{1}{N_{\text{roll}}-1}\sum_{\varepsilon=1}^{N_{\text{roll}}}R_{\text{new},\varepsilon},\ \varepsilon\neq e \tag{11}$$

当随机数$\lambda>\gamma$时为无障碍状态, 位置更新如下:

$$R_{\text{new},e}^{t+1}=R^{\text{rand}}+u\times\Delta x+\alpha\times k\times R_{e}^{t-1} \tag{12}$$

$$\Delta x=Distance_{\text{chebychev}}(R_{e}^{t},R^{w}) \tag{13}$$

其中, R^{rand}表示随机选择的一只推球蜣螂此时的候选位置, R^{w}表示第t次迭代后最差推球蜣螂此时的候选位置, 定义与上述R^{b}类似, Δx表示第e只推球蜣螂第t次迭代后的位置R_{e}^{t}与R^{w}之间的切比雪夫距离, u为常数设置为 2, 参数α,k取值与标准 DBO 相同, 式 (12) 用于替换式 (1).

引入上述策略改进后, 推球蜣螂内部具有了社会性质, 个体不仅可以利用自身信息, 还能充分利用其他推球蜣螂个体共享的信息进行位置迭代, 这不仅提高了全局探索效率, 还有助于算法跳出局部最优, 为其他类别的子种群在局部空间的精确寻优奠定基础.

2.3 方向跟随策略

标准 DBO 中, 虽提及了小偷蜣螂的偷窃行为, 但是并未直接建立其与推球蜣螂的交互, 两个群体各自的位置更新相对独立且缺乏必要的信息交流, 导致信息孤岛的形成并陷入局部最优. 本节将使用公式建立起小偷蜣螂与推球蜣螂之间的交互并作为改进方法避免算法陷入局部最优, 提高算法的寻优精度.

为提高算法的局部开发能力, 所有小偷蜣螂更新完位置之后采取方向跟随策略再次更新位置. 在偷窃前小偷蜣螂需要对推球蜣螂进行观察, 根据各自的观察随机选择一只推球蜣螂, 对其进行跟随以便于进行下一步的偷窃行为, 假设全局最优位置是最适合观察的位置, 使用方向跟随策略进行位置更新公式如下:

$$T_{\text{new},z}^{t+1}=X^{b}+Direction(R^{\text{target}}-X^{b})\times\delta \tag{14}$$

$$\delta=\frac{(r_{21}\times(1-t/T))^{2}}{t}\times\left(\left\|X^{b}+r_{22}\times(Ub-Lb)\right\|\right) \tag{15}$$

其中, X^{b}为全局最优位置, R^{target}为小偷蜣螂随机选择的目标推球蜣螂所在的位置, $Direction(R^{\text{target}}-X^{b})$表示从最优位置移动到随机选择的推球蜣螂所在位置的直线方向的单位向量, r_{21}与r_{22}为随机数且$r_{21},r_{22}\in[0,1]$, δ是移动步长.

二维情况下所提出的方向跟随策略如图 1 所示. 其中, 实心点表示推球蜣螂可能的位置, A表示全局最优位置向量, B表示随机选择的一只推球蜣螂的位置向量, 将$B-A$记作C. 由运动的合成与分解可知, 设$\delta\geqslant 0$, 则$A+\delta\times C$表示从A沿着方向C移动了δ个单位距离后一个新的位置向量. 由概率学理论可知, 推球蜣螂分布在最优位置周围各方向的概率相等, 因此, 小偷蜣螂群体采用该策略以一定步长从全局最优位置A处沿着方向C移动可以实现对最有希望找到最优解的区域 (即图中虚线圆形区域) 的充分开发, $A+\delta\times C$是采用方向跟随策略得到的新位置向量. 得到的新位置将与之前取得的位置对比, 择优保留.

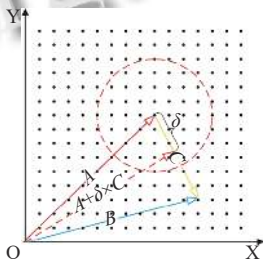

图 1 方向跟随策略示意图 (二维)

2.4 环境感知概率

虽然上述提出的方向跟随策略可以提高算法的局部开发能力, 但是频繁地执行会消耗额外的运算时间, 且不一定会取得更好的结果, 应该考虑性能与时间消耗的平衡. 因此, 只有当小偷蜣螂采用原先的寻优策略陷入停滞的时候, 才会以一定的概率采用方向跟随策略建立与推球蜣螂之间的交互, 综上所述, 需要设定一个阈值去控制小偷蜣螂采用跟随策略的概率.

假定食物充足时, 小偷蜣螂不需要偷窃, 当食物不充足时, 小偷蜣螂对于环境缺乏食物的情况有所感知并转换搜索策略, 判定食物是否充足的依据是小偷蜣螂在新一轮迭代中是否取得了比上一轮迭代更好的解. 为此, 在小偷蜣螂更新位置前, 设定计数器$count$记录在新一轮迭代中未取得比上一轮更优解的小偷个体数量, 则环境感知概率计算公式如下:

$$prob = \sqrt{count/N_{\text{thief}}} \tag{16}$$

其中, N_{thief}是小偷蜣螂数量, 得到环境感知概率后, 生成随机数$p \in [0,1]$, 当$p < prob$时, 小偷蜣螂采用方向跟随策略准备偷窃行为.

2.5 改进蜣螂优化算法的步骤及流程描述

MSDBO 的流程图如图 2 所示, 执行步骤如下.

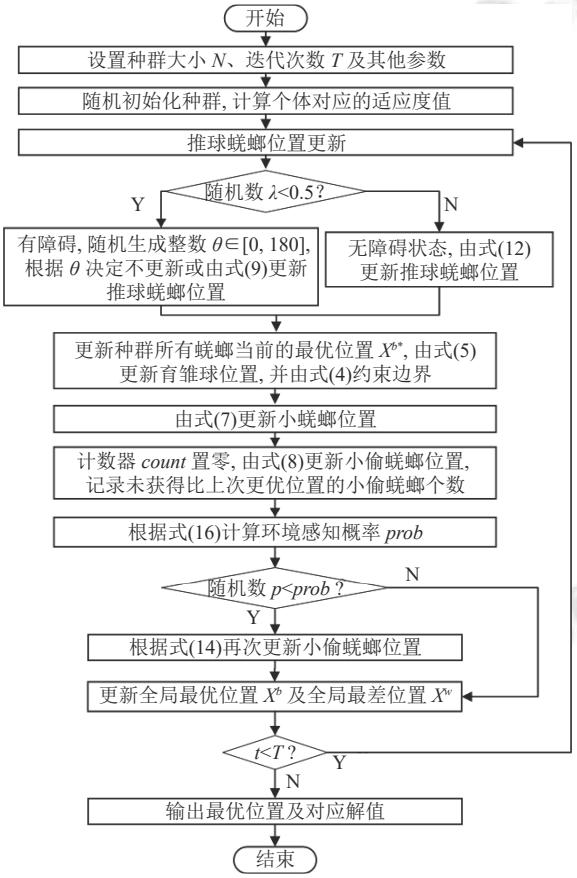

图 2 改进算法流程图

步骤 1. 设置最大迭代次数T, 设置种群大小N, 随机初始化种群并计算个体对应的适应度值.

步骤 2. 更新推球蜣螂位置, 若$\lambda < \gamma$, 为有障碍状态, 由式 (9) 更新位置, 反之为无障碍状态由式 (12) 更新, 其中λ为随机数且$\lambda \in [0,1]$, $\gamma = 0.5$.

步骤 3. 根据式 (5) 更新育雏球位置, 使用式 (4) 中的上下界对新位置进行边界约束.

步骤 4. 由式 (7) 更新小蜣螂位置.

步骤 5. 设置计数器$count$, 由式 (8) 更新小偷蜣螂位置, 记录未取得更优位置的小偷个体数.

步骤 6. 根据式 (16) 计算环境感知概率$prob$, 生成随机数$p \in [0,1]$, 若$p < prob$, 小偷蜣螂子种群根据式 (14) 再次更新位置.

步骤 7. 更新全局最优位置X^b及最差位置X^w.

步骤 8. 判断算法是否达到迭代次数, 若是则终止运行, 返回最优信息, 否则跳转到步骤 2.

2.6 时间复杂度分析

本文采用大O法分析算法的时间复杂度. 假设种群规模为N, 最大迭代次数为T, 问题规模为D, 种群初始化的时间复杂度为O(N); 个体适应度计算及位置更新的时间复杂度为O($N \times T + N \times T \times D$). 则 DBO 的时间复杂度为:

$$\text{O}(N + N \times (T + T \times D)) \tag{17}$$

对于改进的 DBO, 改进点 1 对于复杂度没有太大的影响, 假设改进点 2 的平均调用概率是p, 则改进点 2 的复杂度为O($p \times N_{\text{thief}} \times (T + T \times D)$), 所以, MSDBO 的时间复杂度为:

$$\text{O}(N + (N + p \times N_{\text{thief}}) \times (T + T \times D)) \tag{18}$$

综上所述, MSDBO 与标准 DBO 相比, 时间复杂度是一个数量级的, 并没有增加太多时间复杂度.

3 仿真实验与结果分析

3.1 实验设计

3.1.1 实验环境

本文仿真实验操作系统为 64 位 Windows 10, CPU 为 Intel(R) Core(TM) i7-7700HQ 2.80 GHz, 编程平台为 Matlab R2022a.

3.1.2 测试函数

为有效验证本文所提出改进算法的寻优性能, 使用 12 个基准测试函数对算法进行测试. 其中, F_1–F_6为单峰测试函数, 用于评估算法的局部开发能力及求解精度, F_7–F_{12}为多峰测试函数, 具有多个局部最优解, 用于评估算法的全局探索能力与跳出局部最优的能力. 测试函数的信息如表 1 所示.

3.1.3 对比算法参数及实验设置

除了标准 DBO 以外, 本文选取 3 种被广泛应用的优化算法以及 2 种最近提出的优化算法作为对比, 分

别是: GWO[9]、WOA[10]、HHO[11]、SCSO[13]、SHO[14], 各算法参数均按对应原文设置, 如表 2 所示.

表 1 测试函数表

函数	名称	维度	搜索空间	最优值	类型
F_1	Sphere Function	30	[−100, 100]	0	单峰
F_2	Schwefel's Problem 2.22	30	[−10, 10]	0	单峰
F_3	Schwefel's Problem 1.2	30	[−100, 100]	0	单峰
F_4	Schwefel's Problem 2.21	30	[−100, 100]	0	单峰
F_5	Rosenbrock's Function	30	[−30, 30]	0	单峰
F_6	Step Function	30	[−100, 100]	0	单峰
F_7	Schwefel's Problem 2.26	30	[−500, 500]	−12 569.486 6	多峰
F_8	Penalized Function 1	30	[−50, 50]	0	多峰
F_9	Foxholes Function	2	[−65.536, 65.536]	0.998	多峰
F_{10}	Shekel's Family 1	4	[0, 10]	−10.153 2	多峰
F_{11}	Shekel's Family 2	4	[0, 10]	−10.402 9	多峰
F_{12}	Shekel's Family 3	4	[0, 10]	−10.536 4	多峰

在实验设置上, 所有算法迭代次数均为 500 次, 种群规模均为 30, 各独立运行 30 次. 采用平均值、标准差作为算法性能评价标准, 其中, 均值越小表示算法收敛精度越高, 标准差越小表示算法寻优越稳定. 为避免单次运行的偶然性, 采用 30 次独立运行的平均适应度变化曲线反映各算法的收敛速度与寻优性能.

表 2 算法参数设置

算法	参数	参数设置
GWO[9]	收敛因子a	[2, 0]
WOA[10]	对数螺旋常数b	1
HHO[11]	跳跃强度J	[0, 2]
SCSO[13]	灵敏度范围r	[2, 0]
SHO[14]	捕食成功率r_1	0.9
DBO[15]	遇障概率γ	0.1
	偏转系数k	0.1
	常数u与s	0.3, 0.5
MSDBO	遇障概率γ	0.5
	偏转系数k	0.1
	常数u与s	2, 0.5

3.2 实验分析

实验结果如表 3 所示, 平均适应度变化曲线如图 3 所示, 可以看出改进的蜣螂优化算法在 12 个选定的标准测试函数上的表现均优于其他对比算法.

表 3 与其他优化算法的对比结果

函数	指标	MSDBO	DBO	SHO	SCSO	HHO	WOA	GWO
F_1	平均值	**0**	3.2588E−107	5.2929E−141	2.7096E−114	2.6295E−96	9.9211E−76	2.2211E−27
	标准差	**0**	1.7823E−106	2.0303E−140	8.8291E−114	1.3739E−95	2.5879E−75	5.5254E−27
F_2	平均值	**0**	1.0381E−60	4.0791E−78	2.3351E−60	7.0731E−49	1.3432E−51	7.4798E−17
	标准差	**0**	4.8898E−60	1.1336E−77	7.6738E−60	3.8102E−48	3.1318E−51	4.5327E−17
F_3	平均值	**0**	1.3859E−56	1.3707E−96	4.9692E−98	3.7786E−69	42 263.608 1	1.5933E−05
	标准差	**0**	7.5907E−56	7.5058E−96	2.7162E−97	2.0696E−68	14 425.831 1	2.6779E−05
F_4	平均值	**0**	5.777E−57	2.0365E−56	4.7602E−50	5.1024E−50	43.584 4	9.3607E−07
	标准差	**0**	3.0126E−56	6.4732E−56	2.5625E−49	1.7071E−49	27.394 6	8.0325E−07
F_5	平均值	**2.5385E−05**	25.752 3	28.161 8	28.085	1.0869E−02	27.922 5	26.978 8
	标准差	**1.1172E−04**	1.9981E−01	4.0786E−01	9.2274E−01	1.1126E−02	4.7762E−01	7.1021E−01
F_6	平均值	**3.8142E−10**	5.2634E−04	3.197 3	2.004 1	1.2965E−04	4.2478E−01	9.5069E−01
	标准差	**6.8838E−10**	1.0661E−03	5.6947E−01	5.6575E−01	1.8468E−04	2.5401E−01	3.9586E−01
F_7	平均值	**−12 569.486 6**	−8 343.187 9	−6 192.69	−6 667.888 5	−12 568.72	−10 628.561	−6 212.821 5
	标准差	**2.3661E−08**	1 824.828 1	559.632 7	629.697 2	1.254 3	1 644.063 7	599.117 2
F_8	平均值	**2.9575E−11**	1.8116E−04	2.7014E−01	9.8043E−02	7.9339E−06	2.166E−02	5.2687E−02
	标准差	**8.4855E−11**	5.2449E−04	1.3728E−01	4.5424E−02	1.0323E−05	1.3299E−02	2.6221E−02
F_9	平均值	**0.998**	1.295 7	4.652 5	4.136 3	1.328	2.574 3	5.110 3
	标准差	**1.7001E−16**	6.967E−01	4.306 5	3.820 2	9.4661E−01	2.497 3	4.285 5
F_{10}	平均值	**−9.983 3**	−7.272 7	−5.468 5	−5.519 5	−5.052 8	−8.531 5	−9.476 3
	标准差	9.3076E−01	2.559 3	2.246 7	2.040 9	**2.89E−03**	2.530 3	1.750 4
F_{11}	平均值	**−10.225 8**	−8.051 6	−5.732 9	−6.677 4	−5.228 5	−8.225 6	−10.146 7
	标准差	9.7043E−01	2.769 5	2.414 2	2.791 6	**7.9504E−01**	2.901 7	1.394
F_{12}	平均值	**−10.356 1**	−7.873 9	−5.667 1	−6.123 7	−5.291 6	−6.736 6	−10.264 3
	标准差	9.8735E−01	3.207 7	2.074 6	2.579 8	**9.162E−01**	3.004 6	1.481 2

对于单峰测试函数而言, 在 F_1–F_4 上, MSDBO 可以稳定地寻找到最优值 0, 从对应的迭代曲线上来看, MSDBO 刚开始与其他算法性能相近, 当迭代进行到 100 次左右, MSDBO 能迅速地收敛到最优值 0, 而其他算法往往一直到迭代结束仍未收敛到最优值 0, 这说明 MSDBO 不易陷入局部最优并且具备较强的局部开

发能力, 而其他算法因为局部开发能力较弱从而在局部最优区域周围盲目开发陷入停滞; 在 F_5 与 F_6 上, MSDBO 与其他算法相比, 取得了更高的精度, 从变化曲线上可以看出, 在 F_5 上, 绝大多数算法在进行到第 100 次迭代左右即陷入停滞, 而 MSDBO 在则迭代到第 400 次左右才陷入停滞, 在 F_6 上, 其他优化算法在迭代结束前或早或晚都陷入了停滞或者以缓慢的速度收敛, 而 MSDBO 直到 500 次迭代结束仍有较大的寻优提升空间.

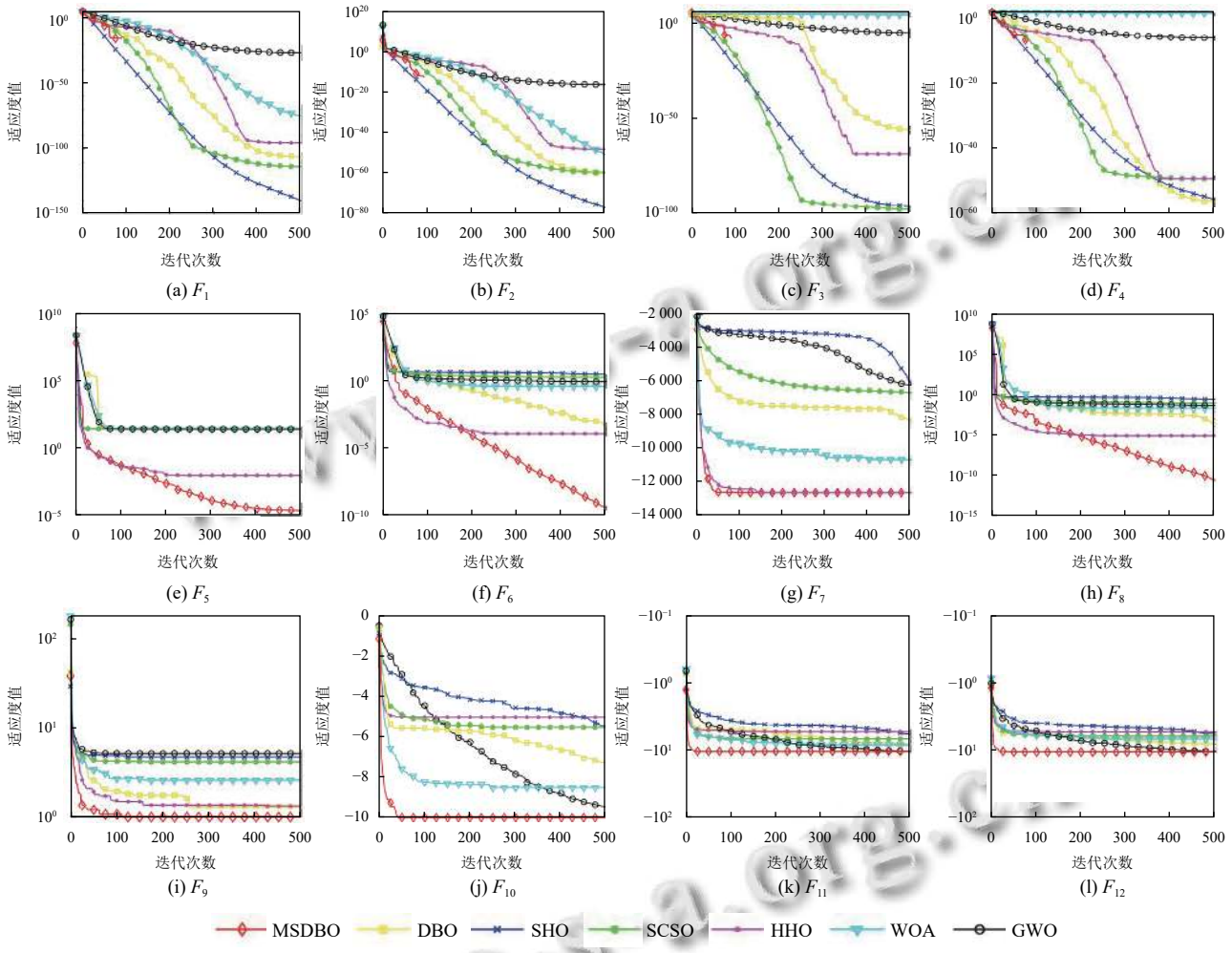

图 3 平均适应度变化曲线

对于多峰测试函数而言, 在 F_7 及 F_9–F_{12} 上, MSDBO 都取得了更优结果, 从变化曲线可以看出, 其中, MSDBO 在 F_7 及 F_{10}–F_{12} 上收敛较其他算法快且没有出现明显的停滞情况, 在 F_9 上, 虽然出现了多次停滞, 但在每次短暂的停滞之后总是能跳出局部最优, 并可以稳定地寻找到最优解, 在函数 F_8 上则取得了比其他算法更高的精度且仍有继续寻优的空间.

综上, MSDBO 在上述不同类型的基准测试函数上的表现均优于对比算法.

3.3 压力容器设计问题

为了验证 MSDBO 对现实中工程设计问题的优化性能, 本文引入压力容器设计优化对算法进行测试, 各参数不变, 记录各算法独立运行 30 次所求得的最优解及最优解对应的解决方案进行对比, 如表 4 所示.

表 4 各算法求解压力容器设计问题的结果对比

算法	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	$f(x)$
MSDBO	**0.77817**	**0.38465**	**40.31962**	**200.0000**	**5885.3328**
DBO	0.78278	0.38969	40.36694	200.0000	5941.2042
SHO	0.78076	0.38577	40.43034	198.4649	5892.5161
SCSO	0.77853	0.38609	40.32700	199.8975	5891.0951
HHO	0.82491	0.40661	42.52920	171.3710	5993.0669
WOA	0.85976	0.44319	44.41161	149.9366	6119.8287
GWO	0.78060	0.38649	40.42647	198.7264	5898.4004

压力容器设计是一个经典的工程优化问题, 该问题的目标是在圆柱形容器壁厚度 (T_s)、封头厚度 (T_h)、容器内径 (R)、容器长度 (L) 这 4 个变量在满足一定

约束条件下总成本最小, 目标函数f、约束条件g及变量范围如下:

$$\begin{cases} x=[x_1,x_2,x_3,x_4]=[T_s,T_h,R,L] \\ f(x)=1.7781x_2x_3^2+0.6224x_1x_3x_4 \\ \qquad +3.1661x_1^2x_4+19.84x_1^2x_3 \\ g_1(x)=0.00954x_3\leqslant x_2 \\ g_2(x)=0.0193x_3\leqslant x_1 \\ g_3(x)=x_4\leqslant 240 \\ g_4(x)=-3\pi x_3^2x_4-4\pi x_3^3\leqslant -3888000 \\ 0\leqslant x_1,x_2\leqslant 99\ 10\leqslant x_3,x_4\leqslant 200 \end{cases}$$

通过对上述的压力容器优化设计问题的测试, 可以看出本文算法在处理实际工程约束优化问题方面具有一定的应用潜力.

4 结论与展望

为克服标准 DBO 算法全局探索能力欠缺、收敛精度低等问题, 提出了一种多策略改进的蜣螂优化算法 (MSDBO). 首先提出并使用了一种社会学习策略引导推球蜣螂进行位置更新, 提高了算法全局探索及跳出局部最优的能力; 其次, 提出方向跟随策略建立起小偷蜣螂与推球蜣螂之间的交互, 提高了算法的寻优精度. 最后, 提出环境感知概率引导小偷蜣螂个体合理采取方向跟随策略, 平衡了算法的性能与时间消耗. 通过对 12 个不同类型的基准测试函数的求解分析, 验证了改进算法具有良好的寻优性能, 在压力容器设计问题上的测试结果表明 MSDBO 在求解实际工程约束优化问题上也有着良好的应用潜力. 未来工作是研究 MSDBO 在聚类等问题中的应用.

参考文献

1 Yang XS. Nature-inspired optimization algorithms: Challenges and open problems. Journal of Computational Science, 2020, 46: 101104. [doi: 10.1016/j.jocs.2020.101104]

2 Holland JH. Genetic algorithms. Scientific American, 1992, 267(1): 66–73. [doi: 10.1038/scientificamerican0792-66]

3 Storn R, Price K. Differential evolution —A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341–359. [doi: 10.1023/A:1008202821328]

4 Abedinpourshotorban H, Shamsuddin SM, Beheshti Z, *et al*. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. Swarm and Evolutionary Computation, 2016, 26: 8–22. [doi: 10.1016/j.swevo.2015.07.002]

5 Hashim FA, Hussain K, Houssein EH, *et al*. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. Applied Intelligence, 2021, 51(3): 1531–1551. [doi: 10.1007/s10489-020-01893-z]

6 Neggaz N, Houssein EH, Hussain K. An efficient henry gas solubility optimization for feature selection. Expert Systems with Applications, 2020, 152: 113364. [doi: 10.1016/j.eswa.2020.113364]

7 Mousavirad SJ, Ebrahimpour-Komleh H. Human mental search: A new population-based metaheuristic optimization algorithm. Applied Intelligence, 2017, 47: 850–887. [doi: 10.1007/s10489-017-0903-6]

8 Emami H. Stock exchange trading optimization algorithm: A human-inspired method for global optimization. The Journal of Supercomputing, 2022, 78(2): 2125–2174. [doi: 10.1007/s11227-021-03943-w]

9 Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software, 2014, 69: 46–61. [doi: 10.1016/j.advengsoft.2013.12.007]

10 Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software, 2016, 95: 51–67. [doi: 10.1016/j.advengsoft.2016.01.008]

11 Heidari AA, Mirjalili S, Faris H, *et al*. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 2019, 97: 849–872. [doi: 10.1016/j.future.2019.02.028]

12 Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67–82. [doi: 10.1109/4235.585893]

13 Seyyedabbasi A, Kiani F. Sand cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. Engineering with Computers, 2023, 39(4): 2627–2651. [doi: 10.1007/s00366-022-01604-x]

14 Zhao SJ, Zhang TR, Ma SL, *et al*. Sea-horse optimizer: A novel nature-inspired meta-heuristic for global optimization problems. Applied Intelligence, 2023, 53(10): 11833–11860. [doi: 10.1007/s10489-022-03994-3]

15 Xue JK, Shen B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. The Journal of Supercomputing, 2023, 79(7): 7305–7336. [doi: 10.1007/s11227-022-04959-6]

16 李晴. 基于紫外-可见光谱法的水质 COD 在线监测系统设计[硕士学位论文]. 绵阳: 西南科技大学, 2023.

17 董奕含, 喻志超, 胡天跃, 等. 基于改进蜣螂优化算法的瑞雷波频散曲线反演方法. 油气地质与采收率, 2023, 30(4): 86–97. [doi: 10.13673/j.pgre.202304038]

18 潘志远, 卜凡亮. 基于蜣螂算法优化的 DV-Hop 定位算法. 电子测量与仪器学报, 2023, 37(7): 33–41. [doi: 10.13382/j.jemi.B2306338]

19 Zhu F, Li GS, Tang H, *et al*. Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems. Expert Systems with Applications, 2024, 236: 121219. [doi: 10.1016/j.eswa.2023.121219]

(校对责编: 孙君艳)