Computer Science and Application 计算机科学与应用, 2024, 14(8), 134-147
Published Online August 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.148171

Hans汉斯

混合策略改进的蜣螂优化算法

刘松林[1]，高　鹰[2*]

[1]广州大学计算机科学与网络工程学院，广东 广州
[2]广州新华学院信息与智能工程学院，广东 东莞

收稿日期：2024年7月9日；录用日期：2024年8月7日；发布日期：2024年8月15日

摘　要

蜣螂优化算法(Dung Beetle Optimizer, DBO)是Xue等人在2022年提出的一种新的群体智能优化算法，其灵感来源于蜣螂的生物行为过程。针对蜣螂优化算法全局探索和局部开发能力不平衡、容易陷入局部最优等缺点，提出了一种混合策略改进的蜣螂优化算法(MIDBO)。首先，在种群初始化时，引入Tent混沌反向学习策略，使初始种群成员能够均匀分布，增加种群丰富性；其次，引入三角形随机游走策略改进繁殖蜣螂位置更新方式，平衡了全局搜索和局部挖掘能力；然后，加入动态权重系数改进蜣螂偷窃行为，加快算法的收敛速度；最后，引入混合变异算子对最优蜣螂位置进行扰动，提高算法跳出局部最优的能力。将所提算法与其他知名优化算法进行了15个基准测试函数的测试比较，仿真结果表明，MIDBO算法是可行有效的，其寻优精度和收敛速度都有了很大的提高，总体性能更好。

关键词

蜣螂优化算法，混沌反向学习，三角形随机游走，动态权重系数，混合变异算子

Mix-Strategy Improved Dung Beetle Optimizer

Songlin Liu[1], Ying Gao[2*]

[1]School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou Guangdong
[2]School of Information and Intelligent Engineering, Guangzhou Xinhua University, Dongguan Guangdong

Received: Jul. 9th, 2024; accepted: Aug. 7th, 2024; published: Aug. 15th, 2024

Abstract

Dung Beetle Optimizer (DBO) is a new swarm intelligence optimization algorithm proposed by

*通讯作者。

文章引用: 刘松林, 高鹰. 混合策略改进的蜣螂优化算法[J]. 计算机科学与应用, 2024, 14(8): 134-147.
DOI: 10.12677/csa.2024.148171

Xue *et al*. in 2022, inspired by the biological behavior process of dung beetles. A mix-strategy improved dung beetle optimizer (MIDBO) is proposed to address the drawbacks of imbalanced global exploration and local development capabilities, as well as the tendency to fall into local optima. Firstly, during population initialization, a Tent chaotic reverse learning strategy is introduced to enable the initial population members to be evenly distributed and increase population richness; secondly, the introduction of triangle random walk strategy improves the position update method of breeding dung beetles, balancing global search and local mining capabilities; then, a hybrid mutation operator is adopted to improve the theft behavior of dung beetles and accelerate the convergence speed of the algorithm; finally, a mixed mutation operator is introduced to perturb the optimal dung beetle position, improving the algorithm's ability to jump out of local optima. The proposed algorithm was compared with other well-known optimization algorithms through 15 benchmark test functions, and simulation results showed that the MIDBO algorithm is feasible and effective. Its optimization accuracy and convergence speed have been greatly improved, and the overall performance is better.

Keywords

Dung Beetle Optimizer, Chaotic Reverse Learning, Triangular Random Walk, Dynamic Weight Coefficient, Hybrid Mutation Operator

1. 引言

优化问题长期以来一直是研究的热点，存在于各种实际系统中，包括故障诊断系统[1] [2]、能量管理系统[3]、预测系统[4] [5]等[6] [7]。需要注意的是，许多复杂的优化问题(如 NP 完全问题)尤其难以用传统的数学规划方法如共轭梯度法和拟牛顿法求解[8]。在这方面，大量的群体智能优化算法以其易于实现、自学习能力强、框架简单等优点被引入。例如灰狼优化算法(Grey Wolf Optimizer, GWO) [9]、萤火虫算法(Firefly Algorithm, FA) [10]、鲸鱼优化算法(Whale Optimizer Algorithm, WOA) [11]、正余弦优化算法(Sine Cosine Algorithm, SCA) [12]等。这些算法原理简单且容易实现，被人们广泛研究和使用。

蜣螂优化算法(Dung Beetle Optimizer, DBO)是 2022 年由 Xue 等人[13]提出的一种新型群智能优化算法。DBO 具有算法结构简单、参数少、和易实现等特点，目前国内外对其的研究并不多。DBO 算法在单峰和多峰目标函数的求解上都具有优异的表现，其具有稳定性高、寻优精度较高等特点。在求解复杂优化问题的时候，与其他的群智能优化算法一样，DBO 容易陷入局部最优。为提高 DBO 性能，潘劲成[14]等在 DBO 算法加入了混沌映射初始化、改进正弦算法和混合变异算子，以提高算法的全局探索能力，使算法可以跳出局部最优解；李晴[15]将 DBO 用于水质 COD 预测模型优化；Zhu 等人[16]引入量子计算等策略对 DBO 算法进行改进并将其用于多个实际工程约束优化问题的求解。ZHANG 等[17]运用维度学习策略，解决蜣螂算法只能选择一个解的情况，从而获得更优的全局解决方案。

本文提出了一种混合策略改进的蜣螂优化算法(Mix-Strategy Improved Dung Beetle Optimizer, MIDBO)，以下几个方面对 DBO 算法进行改进：首先，选择 Tent 混沌反向学习初始化种群，从而提高种群多样性；其次，引入三角形随机游走策略来改进繁殖蜣螂位置更新方式，平衡全局搜索和局部挖掘；然后，加入动态权重系数策略以改进蜣螂偷窃行为，加快算法的收敛速度；最后，引入混合变异策略对最优位置进行扰动，进而避免算法陷入局部最优。

2. 蜣螂优化算法

在本章中，我们主要简单介绍一下蜣螂优化算法(DBO)的来源、数学模型以及相应的算法。蜣螂优化算法的灵感主要来源于蜣螂的滚球、跳舞、觅食、偷窃和繁殖等行为。DBO 算法主要包括四个过程：滚球、繁殖、觅食和偷窃，该算法是基于子种群的，每个子种群执行不同的搜索方式。与蛇优化(Snake Optimizer, SO) [18]和蜉蝣算法(Mayfly Algorithm, MA) [19]不同的是，DBO 不是基于双种群的，而是基于多个子种群，作者划分了四个子种群，即滚球蜣螂、繁殖蜣螂、觅食蜣螂和偷窃蜣螂。

1) 滚球蜣螂。蜣螂有一个有趣的习惯，把粪便做成球，然后把它滚到理想位置，在滚动粪球过程中需要通过天体线索导航，以保持粪球在直线路径上滚动。为了模拟滚球行为，要求蜣螂在整个搜索空间中沿着给定的方向移动。蜣螂利用太阳来导航，同时，我们假设光源的强度也会影响蜣螂的路径。在滚动过程中，滚球蜣螂的位置更新公式为：

$$x_i(t+1)=x_i(t)+\alpha\cdot k\cdot x_i(t-1)+b\cdot\Delta x \tag{1}$$

式中，t 为当前迭代次数；$x_i(t)$ 为第 i 只蜣螂在第 t 次迭代时的位置信息；k 为挠度系数，取 0.1；b 取 0.3；α 为自然系数，赋值为−1 或 1 (参考算法 1)。$\Delta x=\left|x_i(t)-X^{\omega}\right|$，$\Delta x$ 用来模拟光强，其中，X^{ω} 为全局最差位置。

算法 1：α 的选择策略
输入：概率值 λ
输出：自然系数 α

1. $\eta=rand(1)$
2. if $\eta<\lambda$ then
3. $\alpha=1$
4. else
5. $\alpha=-1$
6. end if

当蜣螂遇到障碍物无法前进时，就需要通过跳舞来重新定位，以获得新的路线。用切线函数来模拟舞蹈行为，得到新的滚动方向。需要指出的是，只需要考虑定义在区间$[0,\pi]$。一旦蜣螂成功确定了一个新的方向，它就会继续向前滚动球。因此，蜣螂跳舞行为的位置被定义如下：

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

式中，θ 为挠脚度，属于$[0,\pi]$。$\left|x_i(t)-x_i(t-1)\right|$ 为第 i 只蜣螂在第 t 次迭代时位置与其在第第$t-1$次迭代时的位置之差。如果$\theta=0,\pi/2,\pi$时，将不更新蜣螂的位置。

2) 繁殖蜣螂。在自然界中，粪球被蜣螂滚到安全的地方，然后藏起来。为了给它们的后代提供安全的环境，选择合适的产卵地点对蜣螂来说至关重要。提出了一种模拟雌性蜣螂产卵的区域边界选择策略，其公式定义为：

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

式中，X^{*}为当前局部最佳位置，Lb^{*}和Ub^{*}分别为产卵取下限和上限，其中$R=1-t/T_{\max}$，$T_{\max}$为最大迭代次数，Lb 和 Ub 分别表示优化问题的下界和上界。

从式(3)可以明显看出，产卵区域的边界范围是动态变化的，这主要是由 R 值决定的，由此可见，繁

殖蜣螂的位置在迭代过程中也是动态的，表示为：

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

式中，$B_i(t)$为第 t 次迭代时第 i 个繁殖蜣螂的位置信息；b_1 和 b_2 表示大小为$1\times D$ 的两个独立随机变量，D 为优化问题的维数。注意，繁殖蜣螂的位置被限制在一定的范围内，即产卵区。

算法 2：繁殖蜣螂位置更新策略
输入：最大迭代次数 $T_{\max}$ 、当前迭代次数 t、繁殖蜣螂数量 N
输出：第 i 个繁殖蜣螂 B_i 的位置

1. $R=1-t/T_{\max}$
2. for $i=1:N$ do
3. 用公式(4)更新繁殖蜣螂的位置
4. for $j=1:D$ do
5. if $B_{ij}>Ub^*$ then
6. $B_{ij}\leftarrow Ub^*$
7. end if
8. if $B_{ij}>Ub^*$ then
9. $B_{ij}\leftarrow Ub^*$
10. end if
11. end for
12. end for

3) 觅食蜣螂。一些已经长成成虫的蜣螂会从地下钻出来寻找食物，我们称它们为觅食蜣螂，也叫小蜣螂。除此之外，还需要建立最佳受食区域来引导觅食蜣螂受食，这模拟了觅食蜣螂在自然界的觅食过程。最佳受食区域的边界定义如下：

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

式中，X^b 为全局最佳受食位置；Lb^b 和 Ub^b 分别为最佳受食区域的下界和上界，其余参数在式(3)中定义。因此，觅食蜣螂的位置更新公式如下：

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

式中，$x_i(t)$为第 i 只觅食蜣螂在第 t 次迭代时的位置信息；$C1$ 表示一个服从正态分布的随机数；$C2$ 表示属于(0, 1)的随机向量。

4) 偷窃蜣螂。还有一些蜣螂被称为偷窃蜣螂，会从其他蜣螂那里偷粪球，这是自然界中非常常见的现象。从式(5)可以看出，X^b 是最优的食物来源位置。因此，我们可以假设 X^b 周围即表示争夺食物的最佳地点。在迭代过程中，偷窃蜣螂的位置更新公式如下：

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

式中，$x_i(t)$为第 i 只偷窃蜣螂在第 t 次迭代时的位置信息；g 是一个大小为$1\times D$ 维的随机向量，服从正态分布；S 为常量。

由以上可得，DBO 算法的伪代码如下：

算法 3：DBO 算法伪代码
输入：最大迭代次数 $T_{\max}$ 、种群规模 N
输出：最优位置 X^b 、适应度值 f_b
1. 随机初始化种群和定义算法相关参数
2. while ($t \le T_{\max}$) do
3. for $i = 1:N$ do
4. if $i \in$ 滚球蜣螂 then
5. if $\delta < 0.9$ then
6. 利用公式(1)更新滚球蜣螂位置
7. else
8. 利用公式(2)更新滚球蜣螂位置
9. end if
10. end if
11. if $i \in$ 繁殖蜣螂 then
12. 利用公式(4)更新卵球位置
13. end if
14. if $i \in$ 觅食蜣螂 then
15. 利用公式(6)更新觅食蜣螂位置
16. end if
17. if $i \in$ 偷窃蜣螂 then
18. 利用公式(7)更新偷窃蜣螂位置
19. end if
20. end for
21. 如果存在比当前全局最优值更优的值，则更新最优解和最优值
22. $t = t + 1$
23. end while
24. 输出适应度值 f_b

3. 基于混合策略改进的蜣螂优化算法

本章针对蜣螂优化算法容易陷入局部最优、全局搜索性较差等不足，提出了一种混合策略改进的蜣螂优化算法(MIDBO)，具体的改进策略如下。

3.1. Tent 混沌反向学习

在解决复杂的优化问题时，由于 DBO 算法的种群初始化是随机的，并不能保证初始种群能够均匀分布，种群位置随机分布，使得部分个体远离最优解，从而会影响算法的收敛速度。为了解决原算法中存在的这些问题，引入了 Tent 混沌映射。

混沌序列因其随机性，遍历性等特点，近年来已经成为实现对优化算法改进的一种常用手段，改进方法基于混沌远离，通过混沌奇数产生随机数代替常规伪随机数生成群体的初始位置，且在大量的文献中已得到验证。如刘志强等利用 Tent 混沌映射丰富了狼群算法的初始化，提出了 TGWO，并通过实验验证了该算法在路径寻优上的优越性[20]。基于混沌序列的思想，本文采用了一种 Tent 混沌映射，它具有良好的分布性和随机性，并且映射呈现的结果分布密度比较均匀，其表达式如下：

$$x(t+1) = \begin{cases} x(t)/\alpha, & x(t) < \alpha \\ (1-x(t))/(1-a), & x(t) \ge \alpha \end{cases} \tag{8}$$

本文对 α 取 0.5，t 是迭代次数。

然后，对混沌初始化得到的种群进行反向学习，反向学习的公式如(9)所示：

$$X_{obl} = w \cdot (Lb + Ub) - X \tag{9}$$

其中，w 是一个大小为$1\times D$维的随机向量，服从正态分布；Lb 和 Ub 分别表示解空间的下界和上界，X 表示初始化种群。

最后，将初始种群和反向种群合并，计算其适应度值，选择最优 N 个个体作为新的初始化种群。

使用 Tent 混沌反向学习来初始化种群，提升了种群多样性，让种群分布更加均匀，从而使算法可以快速收敛并接近最优解。

3.2. 三角形随机游走

三角形随机游走通常指的是在一个三角形内部进行随机步行的过程，在数学和计算机科学中都有一些应用和研究，其应用主要包括：① 在随机游走模型中，用于研究分子在空间中的扩散或者在材料科学中的颗粒运动模拟；② 在算法设计中，随机游走可以用来模拟搜索算法或者优化问题的解空间；③ 在概率分析中，用于研究在给定约束下的随机行为的概率性质。

将三角形随机游走策略运用在本文中，蜣螂不再直接向食物源移动，而是采用三角形的路径在食物周围游走，这样可以大大增加蜣螂在搜索过程的随机性，避免算法陷入局部最优，同时使蜣螂有更大的机会探索搜索空间，从而找到更好的解。

首先，得到种群和食物源之间的距 L_1，种群的游走步长范围为 L_2；然后，根据公式(12)定义行走的方向 ρ；最后用公式(14)求出获得种群游走后得到的位置 X_{new}^{*}：

$$L_1 = X^{*} - x_i(t) \tag{10}$$

$$L_2 = \varphi_1 \cdot L_1 \tag{11}$$

$$\rho = 2 \cdot \pi \cdot \varphi_1 \tag{12}$$

$$P = L_1^2 + L_2^2 - 2 \cdot L_1 \cdot L_2 \cdot \cos(\rho) \tag{13}$$

$$X_{new}^{*} = X^{*} + \varphi_1 \cdot \left(2 - \left(2 \cdot t / T_{\max}\right)\right) \cdot P \tag{14}$$

其中，φ_1表示属于(0, 1)之间的随机向量。

通过采用三角形随机游走策略，可以增加蜣螂在搜索过程中的随机性，避免算法陷入局部最优，提高算法的收敛速度和全局搜索能力。

3.3. 动态权重系数

针对收敛速度慢的问题，将基于迭代次数的动态权重系数引入偷窃蜣螂的位置更新公式(7)，偷窃蜣螂改进后的位置更新公式如下：

$$x_i(t+1) = k_1 \cdot X^{b} + k_2 \cdot S \cdot g \cdot \left(\left|x_i(t) - X^{*}\right| + \left|x_i(t) - X^{b}\right|\right) \tag{15}$$

其中，$k_1 = 1 - t^3 / T_{\max}^3$，$k_2 = t^3 / T_{\max}^3$。在上式中，$X^{b}$ 的权重系数 k_1 在迭代前期较大，使蜣螂在搜索空间内探索更优区域；而在后期，蜣螂靠近最优区域，其权重系数 k_2 逐渐增大，使蜣螂在最优区域邻域开发，以此提高全局探索和局部开发的平衡能力。

3.4. 柯西－高斯变异

在原 DBO 算法的迭代后期，容易出现局部最优停滞的情况，为解决这一问题，引入了柯西－高斯变异策略，选择当前适应度最高的个体进行变异，然后比较其变异前后的位置，选择较优的位置代入下一次迭代。具体的公式如下所示：

$$u_{best}(t)=x_{best}(t)\cdot\left(1+\beta_1 cauchy\left(0,\sigma^2\right)+\beta_2 Gauss\left(0,\sigma^2\right)\right) \tag{16}$$

式中，$u_{best}(t)$ 为最优个体变异后的位置；σ^2 为柯西－高斯变异的标准差；$cauchy\left(0,\sigma^2\right)$ 是满足柯西分布的随机变量；$Gauss\left(0,\sigma^2\right)$ 是满足高斯分布的随机变量；$\beta_1=1-t/T_{\max}$ 和 $\beta_2=t/T_{\max}$ 是随迭代次数自适应调整的动态参数。在迭代过程中，β_1 缓慢较小，β_2 缓慢增大，是算法可以跳出当前停滞，同时平衡其全局探索和局部开发能力。

3.5. 基于混合策略改进的蜣螂优化算法

对于 DBO 算法中存在的收敛速度慢，易陷入局部最优等问题，本文提出了一种多策略改进的蜣螂优化算法，在种群初始化时，加入 Tent 混沌反向学习策略，使初始种群成员均匀分布；其次，引入了三角形随机游走策略改进繁殖蜣螂位置更新方式，平衡了全局搜索和局部挖掘能力；又加入动态权重系数改进了偷窃蜣螂位置更新方式；最后，引入柯西－高斯变异策略对最优蜣螂位置进行扰动，提高算法跳出局部最优的能力。MIDBO 算法的步骤如下：

1) 初始化种群大小 N，最大迭代次数 $T_{\max}$ 等参数，根据公式(8)来初始化种群；
2) 计算种群个体的目标函数值，确定种群的最优位置和最优值；
3) 根据 λ 的值来判断使用公式(1)还是公式(2)来更新滚球蜣螂位置；
4) 用公式(3)更新雌蜣螂产卵的区域边界，并用公式(6)来更新繁殖蜣螂位置；
5) 用公式(5)更新觅食蜣螂的最佳受食区域，然后用公式(14)更新繁殖蜣螂位置；
6) 根据公式(15)更新偷窃蜣螂位置；
7) 利用公式(16)进行柯西－高斯变异扰动寻优；
8) 更新全局最优解和最优值；
9) 判断是否达到最大迭代次数，如果是，输出最优解和最优值；否则，跳转到(3)继续迭代。

4. 仿真实验结果与分析

4.1. 基准测试函数

Table 1. Test functions
表 1. 测试函数

基准函数	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
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	[−100, 100]	0
$F_4(x)=\max_i\left\{\lvert x_i\rvert,1\le i\le n\right\}$	30	[−100, 100]	0
$F_5(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	[−128, 128]	0
$F_6(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[−500, 500]	418.9829×Dim
$F_7(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i+10)\right]$	30	[−5.12, 5.12]	0

续表

$F_8(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+e$	30	[−32, 32]	0
$F_9(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[−600, 600]	0
$F_{10}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	2	[−65, 65]	1
$F_{11}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_1x_3+x_4}\right]^2$	4	[−5, 5]	0.0003075
$F_{12}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[−5, 5]	−1.0316285
$F_{13}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$	4	[0, 10]	−10.1523
$F_{14}(x)=-\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$	4	[0, 10]	−10.4028
$F_{15}(x)=-\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^{\mathrm{T}}+c_i\right]^{-1}$	4	[0, 10]	−10.5363

为了测试本文所提改进策略的性能，选用了 15 个经典基准测试函数进行仿真测试。其中，F_1~F_5 是单峰函数，可以测试勘探能力；F_6~F_9 是多峰函数，F_{10}~F_{15} 是固定维度的多峰函数，包含多个局部最优值，用于评估算法的全局搜索和挖掘能力。基准测试函数的函数表达式、搜索范围和理论最优解如表 1 所示。

4.2. 实验参数设置

在本章中，将验证每种改进策略的有效性，同时验证文章改进算法性能。本次选取了近年来比较新颖的五种基本元启发式算法与 MIDBO 进行对比实验，分别是蜣螂优化算法(DBO)、减法平均优化算法(Subtraction-Average-Based Optimizer, SABO) [21]、灰狼优化算法(GWO)、鲸鱼优化算法(WOA)、麻雀搜索算法(Sparrow Search Algorithm, SSA) [22]。本文所选取的对比算法与蜣螂算法的应用场景和运行过程大抵相同，故选取这四种算法与蜣螂优化算法进行对比，使得对比实验更具说服力。

为保证实验的有效性和公平性，测试在同一环境下进行，实验环境为：Windows11 操作系统，CPU 为 Intel Core i7-12700H，主频 2.3 GHz，内存 16 GB，算法用 MatlabR2022a 编写。

Table 2. Algorithm parameter settings
表 2. 算法参数设置

算法	参数值
MIDBO	$K=0.1$, $b=0.3$, $S=0.5$
DBO	$K=0.1$, $b=0.3$, $S=0.5$
PSO	$c1=c2=2$, $\omega_{\max}=0.9$, $\omega_{\min}=0.2$
SABO	$\omega_{\max}=0.9$, $\omega_{\min}=0.2$, $k=2$
WOA	α 由 2 线性下降到 0
SSA	$PD=70\%$, $R_2=0.8$, $SD=20\%$

所要测试算法的种群规模统一设定为 30，最大迭代次数为 500 次，各算法其余的参数设置如表 2 所示。

4.3. 实验结果对比分析

为减小实验过程中偶然性的影响，设置最大迭代次数为 500 次，对每个测试函数独立运行 30 次，计算其最优值(Best)、平均值(Mean)、标准差(Std)，这三者作为评价指标。实验结果如表 3 所示(粗体为最优值)，平均适应度曲线如图 1 所示，对于每个测试函数，在固定的迭代次数下，从最优值可以看出算法的寻优能力和收敛精度，而均值和标准差反映了算法的鲁棒性和稳定性。

Table 3. Experimental results of seven algorithms
表 3. 六种算法实验结果

		MIDBO	DBO	SSA	GWO	WOA	SABO
F_1	Best	0	1.76E−162	0	2.07E−29	2.35E−86	1.86E−115
	Mean	0	1.49E−108	1.23E−57	3.12E−27	3.81E−72	4.83E−113
	Std	0	8.13E−108	6.67E−57	1.57E−27	8.33E−72	7.56E−113
F_2	Best	0	1.77E−84	6.73E−104	1.42E−17	5.93E−57	2.97E− 59
	Mean	0	1.56E−57	1.48E−25	9.18E−17	1.21E−51	1.11E−58
	Std	0	8.28E−57	7.87E−25	5.70E−17	6.53E−51	7.56E−59
F_3	Best	0	1.63E−158	4.08E−163	1.39E−09	1.15E+24	1.67E−66
	Mean	0	1.41E−90	3.90E−25	4.62E−05	4.65E+04	6.31E−49
	Std	0	7.79E−90	2.13E−24	1.49E−05	1.71E+04	3.45E−48
F_4	Best	0	5.84E−72	3.99E−105	4.69E−01	5.80E+01	3.68E− 36
	Mean	0	1.33E−44	5.96E−29	4.60E+00	8.58E+01	2.65E−35
	Std	0	7.29E−43	3.27E−28	3.43E+00	1.09E+01	1.46E−35
F_5	Best	1.82E−06	1.09E−05	1.04E−04	6.08E−04	5.68E−04	1.32E−05
	Mean	8.13E−05	8.96E−04	1.89E−03	2.20E−03	2.99E−03	3.85E−04
	Std	8.86E−05	6.52E−04	1.36E−03	1.21E−03	3.23E−03	2.48E−04
F_6	Best	−1.26E+04	−1.13E+04	−9.35E+03	−7.76E+03	−1.25E+04	−4.83E+03
	Mean	−1.13E+04	−7.52E+03	−8.32E+03	−5.93E+03	−1.06E+04	−3.16E+03
	Std	2.70E−01	1.56E+03	4.93E+02	8.30E+02	1.89E+03	4.43E+02
F_7	Best	0	0	0	0	0	0
	Mean	0	0	0	3.58E+00	1.89E−15	2.08E−14
	Std	0	0	0	4.35E+00	1.03E−14	2.79E−14
F_8	Best	4.44E−16	4.44E−16	4.44E−16	7.51E−14	4.44E−16	7.53E−15
	Mean	4.44E−16	4.44E−16	5.62E−16	1.00E−13	3.87E−15	8.02E−15
	Std	0	6.49E−16	0	1.29E−14	2.37E−15	1.54E−15
F_9	Best	0	0	0	0	0	0
	Mean	0	0	0	3.37E−03	1.16E−02	0
	Std	0	0	0	7.79E−03	2.03E−17	0

续表

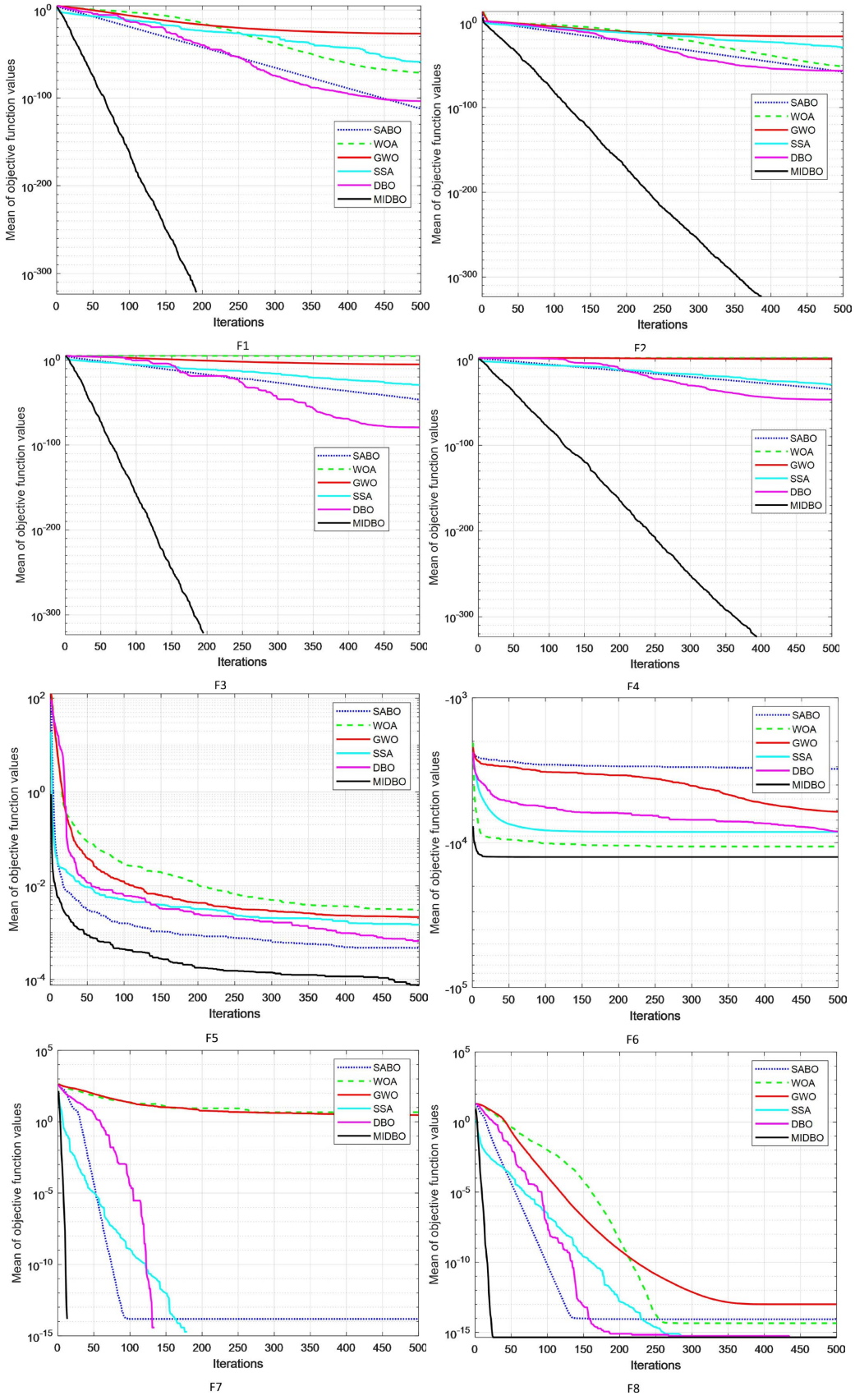
	Best	9.98E−01	9.98E−01	9.98E−01	9.98E−01	9.98E−01	9.98E−01
F_{10}	Mean	1.13E+00	5.60E+00	5.35E+00	4.20E+0	4.32E+00	3.49E+00
	Std	5.03E−01	4.39E+00	5.45E+00	4.11E+00	4.33E+00	2.60E+00
	Best	3.08E−04	3.10E−04	3.07E−04	3.08E−04	3.18E−04	3.08E−04
F_{11}	Mean	4.06E−04	7.73E−04	3.63E−04	5.15E−03	8.23E−04	7.57E−04
	Std	8.35E−05	4.51E−04	1.81E−04	8.54E−03	5.63E−04	1.31E−03
	Best	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01
F_{12}	Mean	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	4.70E−01
	Std	5.41E−04	1.11E−03	0	6.65E−05	3.89E−05	1.18E−01
	Best	−1.02E+01	−1.01E+01	−1.02E+01	−1.02E+01	−1.02E+01	−1.01E+01
F_{13}	Mean	−9.82E+00	−4.43E+00	−8.45E+00	−9.51E+00	−8.62E+00	−9.50E+00
	Std	4.51E−01	2.36E+00	2.44E+00	1.68E+00	2.62E+00	1.96E+00
	Best	−1.04E+01	−1.04E+01	−1.04E+01	−1.04E+01	−1.04E+01	−1.04E+01
F_{14}	Mean	−1.01E+01	−6.23E+00	−8.10E+00	−1.04E+01	−7.68E+00	−9.53E+00
	Std	2.03E−01	2.76E+00	2.68E+00	4.43E+00	3.22E+00	1.65E+00
	Best	−1.05E+01	−1.05E+01	−1.05E+01	−1.05E+01	−1.05E+01	−1.05E+01
F_{15}	Mean	−1.05E+00	−8.67E+00	−9.09E+00	−9.99E+00	−6.46E+00	−9.07E+00
	Std	1.35E−01	2.67E+00	2.43E+00	2.06E+00	3.22E+00	1.76E+00

从表 3 中的数据可以看出，对于 15 个测试函数，本文提出的 MIDBO 算法的最优值、平均值、标准差基本上要优于 DBO、SSA、WOA、GWO 以及 SABO 算法，说明 MIDBO 算法具有更高的寻优精度、稳定性和鲁棒性。

表 3 中给出了 5 个单峰测试函数(F_1~F_5)。在 F_1~F_4 上，MIDBO 表现稳定，能够迅速收敛至最优值 0，与其他算法相比，尤其是在迭代初期性能相近。然而，随着迭代次数的增加，MIDBO 展现出强大的局部开发能力，使其能够有效避免陷入局部最优。相反，其他算法由于局部开发能力的限制，往往在迭代结束时仍未能达到最优值 0，可能在局部最优区域陷入停滞；在 F_5 上，MIDBO 也能找到最优解。在整体优化结果的均值、最优值和标准差都排在了第一，体现了 MIDBO 的优秀的局部开发性能。

表 3 中给出了 10 个多峰测试函数(F_6~F_{15})。与单峰测试函数不同，多峰测试函数有多个极值，并且随维度的增加而增加，因此，多峰测试函数可以用来评估算法的探索能力。在 F_6~F_9 上，MIDBO 展现出优越的性能，其收敛速度明显快于其他算法，并且没有出现明显的停滞现象，因此可以看出，MIDBO 在寻找多峰测试函数最优解的过程中具有更好的探索能力；在 F_{10}~F_{15} 上，他们的维度是固定的，对于 F_{10}、F_{12}~F_{15} 函数 MSDBO 都取得了更优值，能在全局探索和局部开发中自适应转换，虽然函数 F_{11} 未取得最优值，但也是排在几种算法的前列。

通过上述可知，改进的 MIDBO 算法能有效提高全局搜索能力，并在算法后期，有效提高局部开发能力。综上，MIDBO 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。


F1
F2
F3
F4
F5
F6
F7
F8
Mean of objective function values
Iterations
SABO
WOA
GWO
SSA
DBO
MIDBO

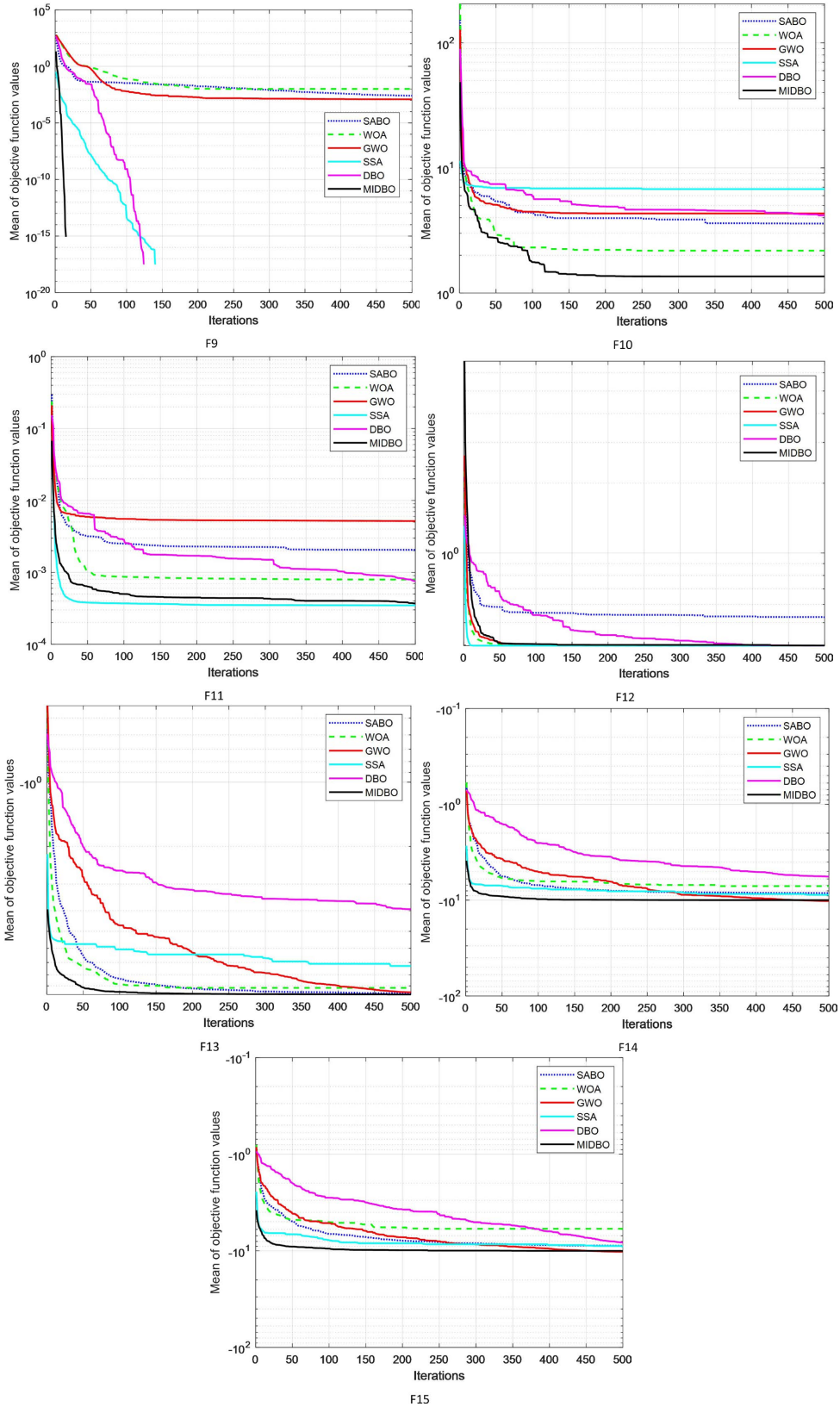


Figure 1. Test function convergence curve
图 1. 测试函数收敛曲线图

5. 结语

本文在蜣螂优化算法的基础上提出了基于混合策略改进的蜣螂优化算法(MIDBO)。通过引入 Tent 混沌反向学习、三角形随机游走和动态权重系数、柯西－高斯变异策略，能够在很大程度上改善基本 DBO 算法寻优精度低和易陷入局部最优的不足。并且，本文还进行了 15 个基准函数的仿真测试，实验结果表明，MIDBO 算法是可行有效的，在收敛精度和收敛速度上都优于原 DBO 算法、减法平均优化算法、灰狼优化算法、鲸鱼优化算法、麻雀搜索算法。其寻优性能有了很大的提高。下一步考虑继续改进算法，提高算法在有多约束条件优化问题中的寻优性能，或将重点转移到本文算法在实际问题中的拓展应用。

基金项目

北航北斗技术成果转化及产业化资金资助项目(项目编号：BARI202104)。

参考文献

[1] Li, M., Yan, C., Liu, W., Liu, X., Zhang, M. and Xue, J. (2022) Fault Diagnosis Model of Rolling Bearing Based on Parameter Adaptive AVMD Algorithm. *Applied Intelligence*, **53**, 3150-3165. https://doi.org/10.1007/s10489-022-03562-9

[2] Qin, Y., Jin, L., Zhang, A. and He, B. (2021) Rolling Bearing Fault Diagnosis with Adaptive Harmonic Kurtosis and Improved Bat Algorithm. *IEEE Transactions on Instrumentation and Measurement*, **70**, 1-12. https://doi.org/10.1109/tim.2020.3046913

[3] Karami, H., Ehteram, M., Mousavi, S., Farzin, S., Kisi, O. and El-Shafie, A. (2018) Optimization of Energy Management and Conversion in the Water Systems Based on Evolutionary Algorithms. *Neural Computing and Applications*, **31**, 5951-5964. https://doi.org/10.1007/s00521-018-3412-6

[4] Li, J., Lei, Y. and Yang, S. (2022) Mid-Long Term Load Forecasting Model Based on Support Vector Machine Optimized by Improved Sparrow Search Algorithm. *Energy Reports*, **8**, 491-497. https://doi.org/10.1016/j.egyr.2022.02.188

[5] Wei, D., Wang, J., Li, Z. and Wang, R. (2022) Wind Power Curve Modeling with Hybrid Copula and Grey Wolf Optimization. *IEEE Transactions on Sustainable Energy*, **13**, 265-276. https://doi.org/10.1109/tste.2021.3109044

[6] Abdulhammed, O.Y. (2021) Load Balancing of Iot Tasks in the Cloud Computing by Using Sparrow Search Algorithm. *The Journal of Supercomputing*, **78**, 3266-3287. https://doi.org/10.1007/s11227-021-03989-w

[7] Zhang, Y. and Mo, Y. (2022) Chaotic Adaptive Sailfish Optimizer with Genetic Characteristics for Global Optimization. *The Journal of Supercomputing*, **78**, 10950-10996. https://doi.org/10.1007/s11227-021-04255-9

[8] Wu, G. (2016) Across Neighborhood Search for Numerical Optimization. *Information Sciences*, **329**, 597-618. https://doi.org/10.1016/j.ins.2015.09.051

[9] Mee Song, H., Sulaiman, M.H. and Mohamed, M.R. (2014) An Application of Grey Wolf Optimizer for Solving Combined Economic Emission Dispatch Problems. *International Review on Modelling and Simulations* (*IREMOS*), **7**, 838-844. https://doi.org/10.15866/iremos.v7i5.2799

[10] Hackl, A., Magele, C. and Renhart, W. (2016). Extended Firefly Algorithm for Multimodal Optimization. 2016 19*th International Symposium on Electrical Apparatus and Technologies* (*SIELA*), Bourgas, 29 May-1 June 2016, 1-4. https://doi.org/10.1109/siela.2016.7543010

[11] Aljarah, I., Faris, H. and Mirjalili, S. (2016) Optimizing Connection Weights in Neural Networks Using the Whale Optimization Algorithm. *Soft Computing*, **22**, 1-15. https://doi.org/10.1007/s00500-016-2442-1

[12] Mirjalili, S. (2016) SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowledge-Based Systems*, **96**, 120-133. https://doi.org/10.1016/j.knosys.2015.12.022

[13] 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

[14] 潘劲成, 李少波, 周鹏, 杨贵林, 吕东超. 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.

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

[16] Zhu, F., Li, G., Tang, H., Li, Y., Lv, X. and Wang, X. (2024) Dung Beetle Optimization Algorithm Based on Quantum Computing and Multi-Strategy Fusion for Solving Engineering Problems. *Expert Systems with Applications*, **236**, Article ID: 121219. https://doi.org/10.1016/j.eswa.2023.121219

[17] Zhang, R. and Zhu, Y. (2023) Predicting the Mechanical Properties of Heat-Treated Woods Using Optimization-Algorithm-Based BPNN. *Forests*, **14**, Article 935. https://doi.org/10.3390/f14050935

[18] Hashim, F.A. and Hussien, A.G. (2022) Snake Optimizer: A Novel Meta-Heuristic Optimization Algorithm. *Knowledge-Based Systems*, **242**, Article ID: 108320. https://doi.org/10.1016/j.knosys.2022.108320

[19] Zervoudakis, K. and Tsafarakis, S. (2020) A Mayfly Optimization Algorithm. *Computers & Industrial Engineering*, **145**, Article ID: 106559. https://doi.org/10.1016/j.cie.2020.106559

[20] 刘志强, 何丽, 袁亮, 等. 采用改进灰狼算法的移动机器人路径规划[J]. 西安交通大学学报, 2022, 56(10): 49-60.

[21] Trojovský, P. and Dehghani, M. (2023) Subtraction-Average-Based Optimizer: A New Swarm-Inspired Metaheuristic Algorithm for Solving Optimization Problems. *Biomimetics*, **8**, Article 149. https://doi.org/10.3390/biomimetics8020149

[22] 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