文章编号：2095-2163(2023)11-0103-09 中图分类号：TQ227.2 文献标志码：A

引入惯性权重与莱维飞行的人工兔优化算法

李姗鸿
(贵州大学 大数据与信息工程学院，贵阳 550025)

摘 要：针对人工兔优化算法(Artificial rabbits optimization，ARO)在种群多样性低、易陷入局部最优和收敛速度慢等问题，本文提出了一种引入惯性权重与莱维飞行的人工兔优化算法(WLARO)。首先，结合 Tent 混沌映射初始化种群，增加种群多样性，提高算法的鲁棒性以及全局寻优能力；其次，在人工兔绕道觅食阶段，引入自适应惯性权重因子增强算法的开发及搜索能力，使算法达到很好的平衡；最后，在人工兔随机躲藏阶段引入莱维飞行策略，避免算法陷入局部最优。将改进后的人工兔算法(WLARO)和其它算法在 10 个基准函数上对比测试，并利用 Wilcoxon 秩和检验验证算法性能。实验结果表明，改进后的人工兔算法在求解精度、收敛速度和寻优能力上都有极大提升。

关键词：人工兔优化算法；惯性权重；莱维飞行；函数优化

Improved artificial rabbits optimization based on Levy flight and adaptive inertial weight strategy

LI Shanhong
(College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China)

Abstract: Aiming at the problems of low population diversity, being prone to fall into local optimum and slow convergence, an artificial rabbit optimization algorithm (WLARO) that introduces inertial weights and Levy flight is proposed in this paper. Firstly, the Tent chaotic map is used to initialize the population, increase the population diversity, and improve the robustness and global optimization ability of the algorithm. Secondly, in the artificial rabbit detour foraging stage, the adaptive inertial weight factor is introduced to enhance the development and search ability of the algorithm, so that the algorithm can achieve a good balance. Finally, the Levy flight strategy is introduced in the random hiding stage of artificial rabbits to avoid the algorithm falling into local optimum. The improved artificial rabbit algorithm (WLARO) is compared with other algorithms on 10 benchmark functions, and Wilcoxon rank sum test is used to verify the performance of the algorithm. The experimental results show that the improved artificial rabbit algorithm has greatly improved the solution accuracy, convergence speed and optimization ability.

Key words: artificial rabbits optimization; inertial weight; Levy flight; function optimization

0 引 言

在过去的几十年中，群智能算法在具有挑战性的工程领域中越来越受欢迎，究其原因则在于这种方法比传统的数值方法更简便和有效。群智能算法的优点体现在许多方面，首先是该方法的随机性，这可以保证算法成功地避免局部极值。其次是黑盒概念，即不需要考虑程序内部，只需要知道输入和预期输出，操作简单。最后，是其算法参数较少、易于实现且数学模型简单。群智能算法有很多种，但都有一个共同的特点，即优化过程分为 2 个步骤：勘探和开发。在探索阶段，算法倾向于在搜索空间中寻找远离当前峰值的最优解，这种搜索过程具有全局性和广泛性。而在开发步骤中，算法倾向于通过搜索解的邻域来改进迄今为止找到的最优解。显然，当开发和探索一起解决问题时，会相互冲突。一个成功的算法应该能够在探索和开发步骤之间保持适当的平衡，这可以减轻局部极端停滞和不成熟收敛的问题。常见的群智能算法有粒子群优化算法(Particle Swarm Optimization，PSO)[1]、蝴蝶优化算法(Butterfly Optimization Algorithm，BOA)[2]、正余弦优化算法(Sine Cosine Algorithm，SCA)[3]、鲸鱼优化算法(Whale Optimization Algorithm，WOA)[4]、野马优化算法(Wild Horse Optimizer，WHO)[5]、麻雀搜索算法(Sparrow

基金项目：国家自然科学基金(62062021，61872034)；贵州省科学技术基金项目(黔科合基础[2020]1Y254)。

作者简介：李姗鸿(1998-)，女，硕士研究生，主要研究方向：认知无线电、智能优化算法。Email：562515752@qq.com。

收稿日期：2022 - 01 - 01

Search Algorithm,SSA)[6]等,但根据无“免费午餐”定理,不可能提出一种能够解决所有工程优化问题的算法,所以国内外学者提出了各种策略来对算法进行优化,以更好地解决各类问题。文献[7]在海马优化算法的初始化中使用 10 种不同的混沌映射产生混沌值、而并非随机值,提高了方法的性能。在海马优化算法中,使用混沌映射生成混沌序列的目的是提高原算法的收敛速度并避免局部最优。文献[8]引入 Tent 混沌映射初始化种群,利用混沌映射具有的随机性、遍历性和有序性等特点,可用于增加种群的多样性,加快算法前期的收敛速度。文献[9]提出了具有惯性权重的蝙蝠优化算法,文章采用简单易行,性能好的随机、线性递减和非线性递减惯量权重。此外,还提出了惯性权重的新变体,其原理是惯性权重呈指数增加,呈指数级增长的惯性权重能避免蝙蝠算法过早收敛。文献[10]提出了一种具有自适应随机惯性权重的粒子群优化算法,其惯性权重是由三角概率密度函数随机生成,并随着算法演变持续更新。在搜索的早期阶段,获得更大权重的概率非常高,这有利于全局搜索。随着迭代次数的增加,获得较小权重的概率增加,这有助于局部优化。文献[11]提出了一种基于 Lévy 飞行和正交学习的新型粒子群算法,利用 Lévy 飞行的跳跃能力来增强探索,具有增强的开发能力和更快的搜索效率。文献[12]提出了一种基于对立学习和 Lévy 飞行分布的飞蛾扑火优化算法,Lévy 飞行是与非高斯随机分布相关的随机游走,其中步长取自 Lévy 飞行,利用 Lévy 飞行分布来防止算法陷入局部最优。

人工兔优化算法[13](Artificial rabbits optimization)是 Wang 等学者于 2022 年所提出来的一种群智能优化算法,该算法将自然界中兔子的生存策略进行了数学建模,建立了一个有效的优化框架。采用这种方法,处理了 2 种模拟策略,即绕行觅食和随机隐藏。ARO 优点在于其参数较少、算法复杂度较低、优化的时间较短;而其缺点是具有收敛速度慢、求解的精度不高、无法找到理论值、容易陷入局部最优、不再随着迭代的进行向理论更优的位置搜索等问题,有着较大的改进空间。因此本文提出了一种引入惯性权重与莱维飞行的人工兔优化算法(WLARO)。首先,在种群初始化阶段引入 Tent 惯性权重,增强种群的多样性和动态性;其次,在绕道觅食阶段引入惯性权重,提高算法的收敛精度及寻优速度;最后,在随机隐藏阶段引入 Lévy 飞行策略,对目前的最优位置进行扰动,避免算法陷入局部最优。通过实验仿真分析证明了本文所提出算法的优势,并将算法应用于压力容器工程问题中,取得了较好的结果。

1 人工兔算法

为防止巢穴被捕食者发现,兔子有 2 种生存策略,一种是绕道觅食,另一种是随机躲藏。兔子的视野非常开阔,很轻易就能在大范围内找到食物,这种绕道觅食策略为勘探阶段。兔子善于挖洞筑巢,为了躲避捕食者或猎人的踪迹,兔子会在自己的窝周围挖洞,遇到危险时,会随机选择一个洞穴作为避难所,这种生存方式是随机躲藏,即开发阶段。由于兔子处于食物链的低端,有很多捕食者,为了躲避危险,兔子必须快速奔跑,这样会使兔子的能量缩减,所以通过能量缩减,兔子会在绕路觅食和随机躲藏之间进行切换,在算法中就是全局搜索与局部开发之间的转换。

1.1 绕道觅食(搜索阶段)

俗话说“兔子不吃窝边草”,在 ARO 中,兔子忽略近处的食物,倾向于移动到其他个体的领地寻找食物,所以每个搜索个体倾向于在种群中随机选择另一个搜索个体更新自己的位置,并增加扰动,其数学模型如下:

$$\vec{v}_i(t+1)=\vec{x}_j(t)+R\cdot(\vec{x}_i(t)-\vec{x}_j(t))+round(0.5\cdot(0.05+r_1))\cdot n_1$$
$$i,j=1,\cdots,n \text{ 且 } j\neq i \tag{1}$$
$$R=L\cdot c \tag{2}$$
$$L=(e-e^{(\frac{t-1}{T})^2})\cdot\sin(2\pi r_2) \tag{3}$$
$$c(k)=\begin{cases}1, & \text{if } k==g(l)\\ 0, & \text{else}\end{cases}$$
$$k=1,\cdots,d \text{ 且 } l=1,\cdots\lceil r_3\cdot d\rceil \tag{4}$$
$$g=randperm(d) \tag{5}$$
$$n_1\sim N(0,1) \tag{6}$$

其中,$\vec{v}_i(t+1)$ 是第 $t+1$ 次迭代时第 i 只兔子的候选位置;$\vec{x}_i$ 是第 t 次迭代时第 i 只兔子的当前位置;n 是种群数量;d 是问题维数;T 是最大迭代次数;L 是奔跑长度,即绕道觅食时的速度;r_1、r_2 和 r_3 是[0,1]之间的随机数;n_1 服从标准正态分布的随机数。

1.2 随机躲藏(开发阶段)

为了躲避捕食者,兔子通常会在其巢穴周围挖洞穴藏身。每一次迭代,兔子会沿着搜索空间的维度在自身周围产生 d 个洞,以降低被捕食的概率,第 i 只兔子的第 j 个藏身洞穴的位置见式(7):

$$\vec{b}_{i,j}(t)=\vec{x}_i(t)+H\cdot g\cdot\vec{x}_i(t)$$
$$i=1,\cdots,n \text{ 且 } j=1,\cdots,d \tag{7}$$

$$H = \frac{T - t + 1}{T} \cdot r_4 \tag{8}$$

$$n_2 \sim N(0,1) \tag{9}$$

$$g(k) = \begin{cases} 1, & \text{if } k == j \\ 0, & \text{else} \end{cases} \quad k = 1, \cdots, d \tag{10}$$

其中，$\vec{b}_{i,j}(t)$ 表示随机从 d 个洞穴中选择的藏身洞穴位置（$i = 1, \cdots, N$，$j = 1, \cdots, d$）；r_4 是[0,1]之间的随机数；n_2 是服从标准正态分布的随机数；H 表示隐藏参数，其变化趋势如图 1 所示。在图 1 中，H 的值从 1 线性递减到 $1/T_{\max}$，在整个迭代过程中很好地保持从勘探到开发的平衡和过渡。

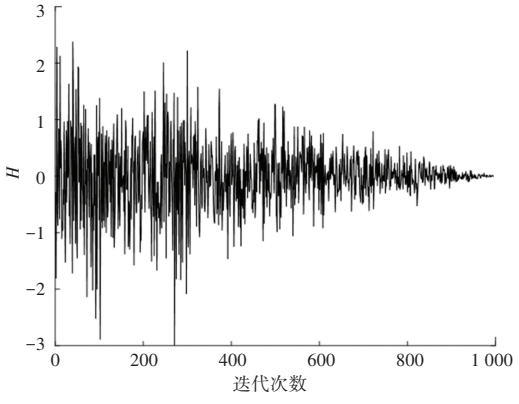


图 1　*H* 变化图

Fig. 1　The change diagram of *H*

随机隐藏的位置更新公式如式(11)所示：

$$\vec{v}_i(t+1) = \vec{x}_j(t) + R \cdot (r_4 \cdot \vec{b}_{i,r}(t) - \vec{x}_j(t)) \quad i = 1, \cdots, n \tag{11}$$

$$g_r(k) = \begin{cases} 1, & \text{if } k == \lceil r_5 \cdot d \rceil \\ 0, & \text{else} \end{cases} \quad k = 1, \cdots, d \tag{12}$$

$$\vec{b}_{i,r}(t) = \vec{x}_i(t) + H \cdot f_r \cdot \vec{x}_i(t) \tag{13}$$

其中，$\vec{b}_{i,r}(t)$ 表示从 d 个洞穴中随机选择的洞穴，r_4 为[0,1]之间的随机数。

在实现 2 种策略后，兔子的位置更新如下：

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t), & f(\vec{x}_i(t)) \leqslant f(\vec{v}_i(t+1)) \\ \vec{v}_i(t+1), & f(\vec{x}_i(t)) > f(\vec{v}_i(t+1)) \end{cases} \tag{14}$$

1.3　能量收缩(从探索转向开发)

兔子在迭代的初期阶段经常进行绕道觅食，而在迭代的后期则经常进行随机隐藏。这种搜索机制是由兔子的能量决定的，随着时间的推移，兔子的能量会逐渐减少。因此，用能量因子来模拟从探索到开发的转换过程。ARO 中的能量因子定义如下：

$$A(t) = 4\left(1 - \frac{t}{T}\right) \ln \frac{1}{r} \tag{15}$$

其中，r 是[0,1]之间的随机数，当 $A(t) > 1$ 时，兔子会绕道觅食，进行全局探索阶段；当 $A(t) \leqslant 1$ 时，兔子随机选择一个洞穴躲藏，进行局部开发阶段，能量因子 A 的变化趋势如图 2 所示，其值是整体下降，更好地平衡了探索和开发阶段。

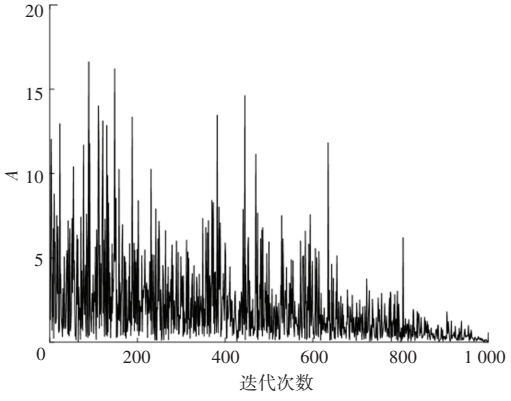


图 2　*A* 变化图

Fig. 2　The change diagram of *A*

2　改进的人工兔优化算法

2.1　Tent 混沌初始化

ARO 采用的是随机初始化种群，种群可能会密集集中在一个区域，不利于算法进行迭代寻优，因此本文引入了 Tent 混沌映射[14]，利用其具有的小周期和不稳定点特点，对种群进行初始化。保证 Tent 映射序列的随机性、遍历性和有效性，其表达式如下：

$$Y_{i+1}^d = \begin{cases} 2Y_i^d + \text{rand}(0,1) \times \dfrac{1}{N}, & 0 \leqslant Y_i^d \leqslant 0.5 \\ 2(1 - Y_i^d) + \text{rand}(0,1) \times \dfrac{1}{N}, & 0.5 < Y_{\frac{d}{t}} \leqslant 1 \end{cases} \tag{16}$$

其中，i 为兔子的序号；N 为兔子的总数；d 为对应的维度。结合改进 Tent 序列进行种群初始化，其表达式为：

$$x_i^d = x_{\min}^d + Y_i^d (x_{\max}^d - x_{\min}^d) \quad i = 1, \cdots, N \text{ 且 } d = 1, \cdots, D \tag{17}$$

其中，$x_{\min}^d$ 表示维度的下界，$x_{\max}^d$ 表示维度的下界。

2.2　自适应惯性权重因子

惯性权重因子[15]对解的搜索精度和收敛速度有着很好的引导作用，较大的惯性权重因子在前期全局搜索时能力较强，较小的惯性权重因子在后期的开发能力较强。因此本文采用一种非线性惯性权重因子，更快达到一定的收敛精度，使之在迭代初期惯性权重缓慢减小，使其有很好的全局搜索能力，更快达到一定的收敛精度，在迭代后期，其解容易陷入局部最优，此时较小惯性权重能够有较好的局部搜索能力使之达到最优解，其公式如下：

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \times (1 - (\frac{t}{T_{\max}})) \quad (18)$$

其中，t 为迭代次数；$T_{\max}$ 为最大迭代次数；$\omega_{\min}$ 的值为 0.4；$\omega_{\max}$ 的值为 0.9；ω 的变化曲线如图 3 所示。随着迭代的增加，惯性权重因子非线性减少、最终达到最小值，将其引入到 ARO 中，可用式(19)表示：

$$\vec{v}_i(t+1) = \omega \cdot \vec{x}_j(t) + R \cdot (\vec{x}_i(t) - \vec{x}_j(t)) + round(0.5 \cdot (0.05 + r_1)) \cdot n_1$$
$$i, j = 1, \cdots, n \text{ 且 } j \neq i \quad (19)$$

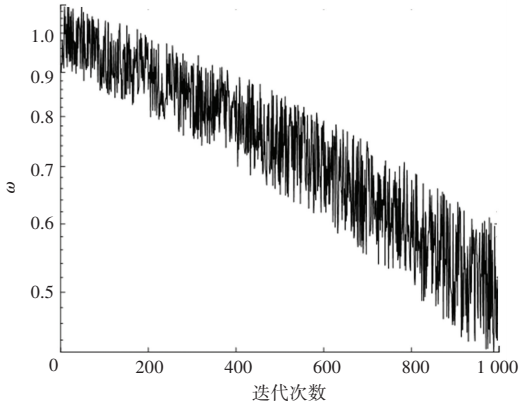


图 3 ω 变化图

Fig. 3 The change diagram of ω

2.3 Lévy 飞行策略

Lévy 飞行策略[16]是非高斯随机步态，其步长服从概率分布，在寻找最优解过程中，主要是生成一个正则随机数，给算法更新提供动态变化，Lévy 飞行不仅可以在短距离中进行局部搜索，还可以在长距离全局搜索。因此在搜索到最优值附近时，Lévy 飞行能达到增强局部搜索能力的作用，有效解决 ARO 陷入局部最优的问题，本文将 Lévy 飞行策略引入式(11)中的人工兔的最优位置，因为 ARO 会根据当前位置与人工兔最优位置的距离来进行位置更新，改进后的 ARO 大大降低了人工兔陷入局部最优的风险，而且仍然能充分执行局部探索，改进公式如下所示：

$$\vec{v}_i(t+1) = Levy(D) \cdot \vec{x}_j(t) + R \cdot (r_4 \cdot \vec{b}_{i,r}(t) - \vec{x}_j(t)), \quad i = 1, \cdots, N \quad (20)$$

其中，D 为问题的维度，Lévy 飞行的计算公式如下：

$$Levy(D) = 0.01 \cdot \frac{r_6 \cdot \sigma}{|r_7|^{\frac{1}{\beta}}} \quad (21)$$

$$\sigma = \left\{ \frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \beta/2)}{\Gamma[(1+\beta)/2] \cdot \beta \cdot 2^{(\beta-1)/2}} \right\}^{1/\beta} \quad (22)$$

其中，Γ 表示标准的 Gamma 函数，β 是相关参数，本文设置其值为 1.5。

2.4 WLARL 算法步骤

由 2.1~2.3 节可得引入惯性权重与莱维飞行的人工兔优化算法（ARO）的步骤如下：

Step 1 初始化算法参数，建立搜索空间的矩阵。

Step 2 使用改进的 **Tent** 混沌映射初始化种群，计算其个体适应度值并进行排序。

Step 3 通过式(15)计算 A 的值，若 $A > 1$ 则随机选择一只兔子，根据式(2)~(5)计算 R 的值，通过式(19)执行绕道觅食，然后计算更新兔子的适应度值，并通过式(14)更新兔子位置。

Step 4 若 $A \leqslant 1$，则随机生成洞穴并根据式(13)选择一个洞穴隐藏，通过式(20)执行随机躲藏，然后计算更新兔子的适应度值，并通过式(14)更新兔子位置。

Step 5 判断迭代次数是否达到上限，若是则停止迭代，得到最优位置以及其适应度值，否则重复执行 Step 3~Step 5，直到满足终止迭代条件，算法流程图如图 4 所示。

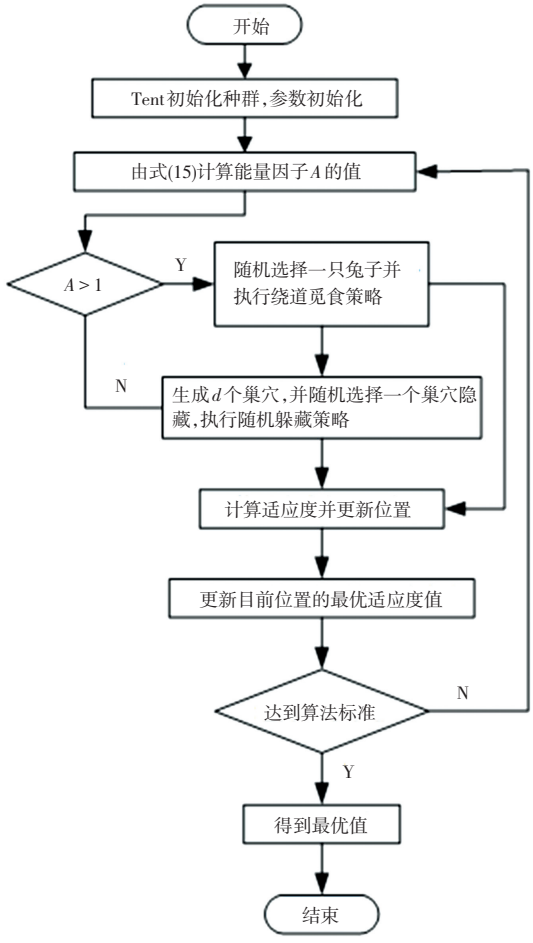


图 4 算法流程图

Fig. 4 Implementation flow chart of WLARO algorithm

3　仿真实验和结果分析

3.1　函数选取与参数设置

本文采用 Matlab R2020b 进行实验仿真，运行环境为 64 位 Windows 10 操作系统，处理器类型为 AMD Ryzen 7 5800H。

为了验证 WLARO 算法的有效性和改进策略的优异性，将优化后的人工兔算法（WLARO）与传统人工兔算法（ARO）、蝴蝶优化算法（BOA）、粒子群算法（PSO）、正弦余弦算法（SCA）、樽海鞘算法（SSA）、鲸鱼优化算法（WOA）、野马优化算法（WHO）进行了对比实验。本文引入 10 个标准测试函数（见表 1）。F1～F6 是单峰函数，局部最优即全局最优，用来检验本文提出的 WLARO 算法的收敛速度和收敛精度；F_7 ～ F_{10} 是多峰函数，具有多个局部极值，尤其 F_9 和 F_{10} 的变量之间相互关联，使算法很难搜索到全局最优，用来测试算法的跳出局部最优的能力。在实验中，将使用 30 次独立运行实验以测试算法性能，设置种群个数为 50，最大迭代次数为1 000。另外，测试函数的维度也是影响算法寻优的一个关键因素，因此表 1 所列出的测试函数的维度从 2 到 200 不等，可以更加全面地验证算法性能。各算法的主要参数设置见表 2。不同算法的结果对比见表 3。

表 1　测试函数

Tab. 1　Test functions

函数名称	函数公式	搜索区间	理论值	类型
Sphere	$F_1(X)=\sum_{i=1}^{n}x_i^2$	[-100,100]	0	单峰
Schwefel′s 2.22	$F_2(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[-10,10]	0	单峰
Schwefel′s 1.2	$F_3(X)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	[-100,100]	0	单峰
Schwefel′s 2.21	$F_4(X)=\max\{\lvert x_j\rvert\}$	[-100,100]	0	单峰
Step	$F_5(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	[-100,100]	0	单峰
Quartic	$F_6(X)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	[-1.28,1.28]	0	单峰
Rastrigin	$F_7(X)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	[-5.12,5.12]	0	多峰
Ackley	$F_8(X)=-20\exp(-0.2\sqrt{\sum_{i=1}^{n}x_i^2/n})-\exp(\sum_{i=1}^{n}\cos(2\pi x_i/n))+20+e$	[-32,32]	0	多峰
Griewank	$F_9(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	[-600,600]	0	多峰
Shekel′s Foxholes	$F_{10}=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	[-65.65,65.65]	1	多峰

表 2　算法参数

Tab. 2　Parameters of functions

算法	主要参数设置	算法	主要参数设置	算法	主要参数设置	算法	主要参数设置
WLARO	$\omega_{max}=0.9$，$\omega_{min}=0.2$，$\beta=1.5$	WHO	$PC=0.1$，$PS=0.2$	SCA	$a=2$	ARO	-
BOA	$p=0.8$，$c=0.01$，$a=0.1$	PSO	$a_1=a_2=1.5$	WOA	$b=1$	SSA	-

表 3 不同算法的结果比较

Tab. 3 Comparison of results of different algorithms

函数	算法	平均值	最优值	最差值	标准差	函数	算法	平均值	最优值	最差值	标准差
F_1	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	F_2	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ARO	1.54E−122	6.52E−139	5.63E−94	1.91E−93		ARO	6.49E−68	1.70E−75	8.61E−67	2.02E−67
	BOA	1.69E−14	1.43E−14	1.99E−14	1.05E−15		BOA	9.73E−12	2.75E−12	1.16E−11	2.35E−12
	PSO	1.48E+01	7.72E+00	2.33E+01	3.64E+00		PSO	1.31E+01	8.62E+00	1.81E+01	2.23E+00
	SCA	3.59E−03	1.66E−07	6.71E−02	1.21E−02		SCA	5.07E−06	3.67E−09	3.70E−05	7.79E−06
	SSA	9.64E−09	6.54E−08	1.35E−08	2.10E−09		SSA	3.23E−01	6.53E−04	1.51E+00	3.97E−01
	WOA	3.50E−167	2.80E−188	1.00E−165	**0.00E+00**		WOA	4.30E−105	4.90E−117	1.30E−103	2.30E−104
	WHO	1.20E−106	9.70E−118	2.60E−105	4.80E−106		WHO	1.03E−58	6.47E−68	1.16E−57	2.88E−58
F_3	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	F_4	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ARO	1.96E−97	2.40E−118	5.86E−96	1.07E−96		ARO	2.84E−51	9.12E−59	8.27E−50	1.51E−50
	BOA	1.69E−14	1.52E−14	1.87E−14	9.06E−16		BOA	1.19E−11	1.08E−11	1.30E−11	5.75E−13
	PSO	1.24E+02	5.72E+01	1.94E+02	2.93E+01		PSO	2.31E+00	1.82E+00	3.09E+01	8.28E+00
	SCA	2.23E+03	1.61E+01	8.15E+03	2.25E+03		SCA	1.37E+01	3.95E+01	1.91E+01	1.05E+01
	SSA	5.34E+01	1.67E+00	1.82E+02	5.16E+01		SSA	4.05E+00	4.56E−01	8.52E+00	1.96E+00
	WOA	1.37E+04	1.36E+03	3.39E+04	7.98E+03		WOA	3.25E+01	2.27E−02	8.51E+01	3.28E+01
	WHO	1.37E−62	1.01E−79	3.45E−61	6.35E−62		WHO	1.62E−40	1.09E−45	4.04E−39	7.35E−40
F_5	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	F_6	WLARO	**3.28E−05**	**7.03E−07**	**1.05E−05**	**2.79E−05**
	ARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		ARO	2.04E−04	1.78E−05	5.92E−04	1.55E−04
	BOA	5.38E+00	3.69E+00	6.74E+00	7.61E−01		BOA	6.04E−03	2.01E−03	1.38E−02	2.61E−03
	PSO	1.49E+01	6.04E+00	2.32E+01	4.45E+00		PSO	4.89E+01	1.34E+01	9.59E+01	2.07E+01
	SCA	4.42E+00	3.61E+00	6.31E+00	5.21E−01		SCA	2.93E−02	3.76E−03	1.69E−01	3.37E−02
	SSA	9.07E−09	5.96E−09	1.24E−08	1.32E−09		SSA	6.27E−02	1.98E−02	1.07E−01	2.24E−02
	WOA	4.61E−03	1.18E−03	2.41E−02	4.02E−03		WOA	8.91E−04	7.49E−06	4.89E−03	1.11E−03
	WHO	3.85E−15	1.33E−18	5.76E−14	1.26E−14		WHO	3.90E−04	1.68E−05	1.36E−03	3.04E−04
F_7	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	F_8	WLARO	**8.88E−16**	**8.88E−16**	**8.88E−16**	**0.00E+00**
	ARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		ARO	**8.88E−16**	**8.88E−16**	**8.88E−16**	**0.00E+00**
	BOA	1.88E+01	**0.00E+00**	1.94E+02	5.74E+01		BOA	1.13E−11	4.44E−12	1.35E−11	1.67E−12
	PSO	2.06E+02	1.63E+02	2.43E+02	2.23E+01		PSO	4.32E+00	3.34E+00	5.10E+00	3.81E−01
	SCA	9.01E+00	1.91E−05	9.79E+01	2.00E+01		SCA	1.39E+01	1.57E−04	2.02E+01	9.31E+00
	SSA	5.12E+01	2.98E+01	8.06E+01	1.32E+01		SSA	1.82E+00	2.31E−05	3.02E+00	5.86E−01
	WOA	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		WOA	4.56E−15	8.88E−16	7.99E−15	2.55E−15
	WHO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		WHO	2.31E−15	8.88E−16	4.44E−15	1.77E−15
F_9	WLARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	F_{10}	WLARO	9.98E−01	9.98E−01	9.98E−01	**0.00E+00**
	ARO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		ARO	9.98E−01	9.98E−01	9.98E−01	2.93E−15
	BOA	9.84E−16	0.00E+00	3.44E−15	1.03E−15		BOA	**1.00E+00**	9.98E−01	1.02E+00	6.91E−03
	PSO	5.27E−01	3.64E−01	6.91E−01	9.41E−02		PSO	1.55E+00	9.98E−01	6.90E+00	1.15E+00
	SCA	1.95E−01	1.28E−05	1.02E+00	2.70E−01		SCA	1.26E+00	9.98E−01	2.98E+00	6.86E−01
	SSA	8.70E−03	2.41E−08	3.20E−02	9.18E−03		SSA	9.98E−01	9.98E−01	9.98E−01	2.00E−16
	WOA	3.30E−03	**0.00E+00**	5.48E−02	1.23E−02		WOA	2.04E+00	9.98E−01	1.08E+01	2.97E+00
	WHO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**		WHO	1.15E−76	1.72E−86	3.36E−75	6.13E−76

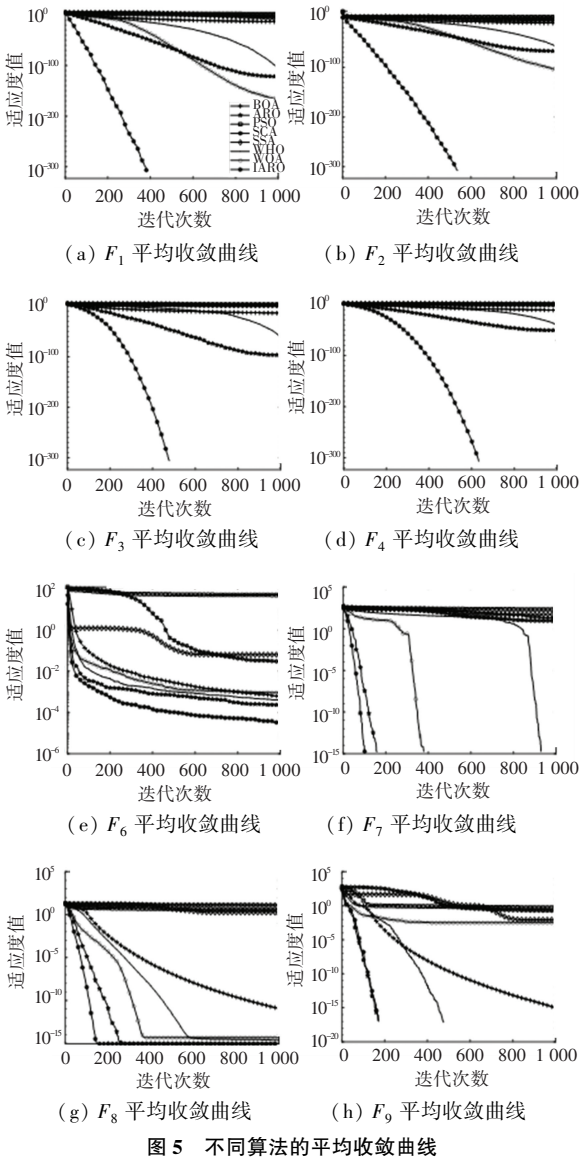
由表 3 可以看出，在与其他的传统群智能优化算法相比时，F_1 到 F_5 这 5 个单峰函数上 WLARO 算法的表现都是远超过其他的传统群智能算法，平均值、最优值、最差值和标准差都到达了 0，证明了其不仅寻优精度高、且具有极强的鲁棒性，在 30 次实验中均未出现个别寻优值偏离理论值的现象，证明其稳定性极好，说明本文提出的 3 个策略对算法进行改进是有效的；在 F_6 这个单峰函数上，虽然 WLARO 算法没有找到测试函数的理论值，但相比于其他算法，无论是最优值、还是标准差仍然是最优；在 F_7 与 F_9 这 2 个多峰函数上，WLARO 算法同样能够找到测试函数的理论值，30 次数据的标准差都为 0，即其寻优过程非常稳定；在 F_8 函数上，虽然 WLARO 算法没有搜索到算法的理论值，但相比于其他的几个传统算法有着较大的改进，在 30 次实验中每一次都在迭代后期陷入了局部最优值 8.88E-16，其寻优精度相比于其他传统算法提高了 16 个数量级；在 F_{10} 多峰函数上，WLARO 算法没有找到理论值，但是非常接近理论值，且标准差为 0，算法十分稳定。综上，虽然相比于其他的各种传统群智能算法，WLARO 算法对各个测试函数精度以及稳定性的提升不尽相同，但总地来说，WLARO 算法在求解各种基准函数上都具有一定的优势。

3.2　测试函数收敛曲线

根据实验所得数据，图 5 给出了 8 个基准函数的平均收敛曲线，由图 5(a)～图 5(d)可知，在迭代前期，WLARO 算法在一段时间内收敛曲线下降速度很快，表明引入的 Tent 惯性权重起到了作用，增加了种群的多样性，使得算法一开始收敛速度就较快，随着更迭次数的增加算法持续寻优，未出现停止搜索的状况，并一直寻优到了理论最优值。图 5(e)和图 5(g)的曲线有多处拐点，说明本文提出的 Lévy 飞行策略能够使得算法跳出局部最优，算法的收敛精度得到提升。图 5(f)和图 5(h)的 WLARO 算法的曲线虽然和其他 2 种群智能算法的收敛精度相差不大，但从曲线可明显看出 WLARO 算法的收敛速度较其他算法有很大的优势，在迭代前期就迅速收敛，说明本文提出的引入惯性权重与莱维飞行的人工兔优化算法能够提高算法的收敛速度。

3.3　Wilcoxon 秩和检验

虽然在 30 次独立实验所得到的平均值与标准差虽然有一定的参考意义，但在多次实验中，可能会出现某一次实验的效果极优或极差的情况，这是实验数据的平均值与标准差无法体现的，因此在评估改进的算法的性能时，仅仅依据平均值与标准差是不够的，需要进行统计检验来证明所提出的改进算法的优越性。



(a) F_1 平均收敛曲线　(b) F_2 平均收敛曲线

(c) F_3 平均收敛曲线　(d) F_4 平均收敛曲线

(e) F_6 平均收敛曲线　(f) F_7 平均收敛曲线

(g) F_8 平均收敛曲线　(h) F_9 平均收敛曲线

图 5　不同算法的平均收敛曲线

Fig. 5　Average convergence curves of different algorithms

本文在 5%的显著性水平下进行 Wilcoxon 秩和检验[17]，判断 WLARO 算法在某些特定问题上是否有显著的性能提升。Wilcoxon 秩和检验是非参数统计的一种检验方法，本文使用该检验方法与 BOA、PSO、SCA、SSA、ARO、WOA 和 DA 算法进行比较，判断 WLARO 与其他算法之间是否存在显著性的差异。表 4 列出了所有测试函数中 WLARO 算法与其他算法秩和检验的 P 值。表 4 中，每个数据表示 WLARO 算法与该数据对应的算法在对应的测试函数中相比的 P 值，当 P 小于 5%时，可认为是拒绝零假设的有利证据[18]，即“改进算法与其对比的算法是有显著区别”这一说法是错误的。因此，表 4 中 P 的数值越小，说明 WLARO 与其对比的算法区别越

大，S 为“+”表明 WLARO 算法显著性高于其他算法，WLARO 的结果有明显改善。结合表 4 的数据，即可综合判断改进算法的效果，WLARO 算法与其他算法在各个测试函数的 Wilcoxon 秩和检验的 P 值均远远小于 0.05，所以很明显本文提出的改进算法在统计上具有优越性。

表 4 Wilcoxon 秩和检验的值

Tab. 4 Values of Wilcoxon rank sum test of test function

函数	ARO		BOA		PSO		SCA		SSA		WHO		WOA	
	P	*S*	*P*	*S*	*P*	*S*	*P*	*S*	*P*	*S*	*P*	*S*	*P*	*S*
F_1	1.07E-12	+	1.21E-12	+	1.21E-12	+	1.17E-13	+	3.16E-12	+	1.18E-12	+	1.21E-12	+
F_2	1.01E-12	+	1.21E-12	+	9.28E-13	+	2.07E-12	+	2.64E-12	+	1.08E-12	+	1.21E-12	+
F_3	1.07E-12	+	1.21E-12	+	1.18E-12	+	4.90E-13	+	3.61E-13	+	1.18E-12	+	1.18E-12	+
F_4	1.04E-12	+	1.21E-12	+	1.18E-12	+	1.17E-13	+	1.46E-12	+	1.17E-12	+	1.11E-12	+
F_6	1.09E-12	+	1.21E-12	+	1.21E-12	+	1.17E-13	+	3.64E-12	+	1.19E-12	+	1.21E-12	+
F_7	2.67E-11	+	3.02E-11	+	2.67E-11	+	4.56E-11	+	3.51E-10	+	2.99E-11	+	2.99E-11	+
F_9	1.06E-12	+	8.90E-13	+	4.96E-13	+	1.17E-13	+	6.46E-13	+	1.19E-12	+	1.20E-12	+
F_{10}	8.04E-13	+	1.15E-12	+	1.14E-12	+	1.22E-12	+	1.44E-11	+	1.17E-12	+	1.21E-12	+
F_{11}	1.07E-12	+	1.21E-12	+	1.20E-12	+	1.17E-13	+	3.43E-12	+	1.17E-12	+	1.21E-12	+
F_{14}	3.07E-13	+	1.04E-12	+	9.91E-13	+	1.99E-11	+	1.73E-11	+	1.16E-12	+	1.11E-12	+

3.4 实际工程案例及分析

为进一步将 WLARO 算法在具体的工程中得到实际优化应用，本文选用压力容器设计问题。这个问题的目标是使压力容器的制造成本最小化，压力容器设计问题的目标是使压力容器制作（配对、成型和焊接）成本最小。压力容器的设计如图 6 所示，压力容器的两端都有盖子封顶，头部一端的封盖为半球状。L 是不考虑头部的圆柱体部分的截面长度，R 是圆柱体部分的内壁直径，T_S 和 T_h 分别表示圆柱体部分壁厚和头部的壁厚，L、R、T_S 和 T_h 为压力容器设计问题的 4 个优化变量，问题的目标函数和 4 个优化约束表示如下：

$$x = [x_1, x_2, x_3, x_4] = [T_S, T_h, R, L] \tag{23}$$

$$\min f(x) = 0.622\,4x_1x_3x_4 + 1.778\,1x_2x_3^2 + 3.166\,1x_1^2x_4 + 19.84x_1^2x_3 \tag{24}$$

约束条件为：

$$g_1(x) = -x_1 + 0.019\,3x_3 \leqslant 0 \tag{23}$$

$$g_2(x) = -x_2 + 0.009\,54x_3 \leqslant 0 \tag{24}$$

$$g_3(x) = -\pi x_3^2 + 04\pi x_3^3/3 + 129\,600 \leqslant 0 \tag{25}$$

$$g_4(x) = x_4 - 240 \leqslant 0 \tag{26}$$

$$0 \leqslant x_1 \leqslant 100, 0 \leqslant x_2 \leqslant 100, 10 \leqslant x_3 \leqslant 100, 10 \leqslant x_4 \leqslant 100 \tag{27}$$

利用文献[19]中罚函数的方法进行约束组合处理建立约束目标函数，同时应用 ARO、MFO、GA 和 DE 与 WLARO 算法进行求解结果的比较，其对比结果见表 5。从表 5 可看出，WLARO 在实际应用中也是有效的，具有较好的寻优能力，进一步验证了该算法在工程应用中的可行性。

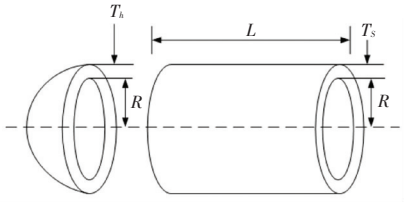


图 6 压力容器设计模型

Fig. 6 Design model of pressure vessel

表 5 不同优化算法对压力容器问题求解的结果对比

Tab. 5 Comparision of pressure vessel solved by different optimization algorithms

算法	T_S	T_h	R	L	$f(x)$
WLARO	0.778 1	0.384 6	40.319 6	199.999 6	5 885.337 5
ARO	0.778 2	0.384 7	40.323 3	199.947 9	5 885.667 9
MFO	0.812 5	0.437 5	42.098 4	176.636 5	6 059.714 3
GA	0.812 5	0.437 5	42.097 3	176.654 0	6 059.946 3
DE	0.812 5	0.437 5	42.098 4	176.637 6	6 059.734 0

4 结束语

针对传统人工兔优化算法的缺点，本文提出了一种引入惯性权重与莱维飞行的人工兔优化算法（WLARO）。首先，在种群初始化阶段引入 Tent 惯性权重，增强种群的多样性和动态性；其次，在绕道觅食阶段引入惯性权重，提高算法的收敛精度及寻

优速度；最后，在随机隐藏阶段引入 Lévy 飞行策略，对目前的最优位置进行扰动，避免算法陷入局部最优。通过实验仿真分析证明了本文所提出算法的优势，并将算法应用于压力容器工程问题中，取得了较好的结果。今后的工作会将 WLARO 应用于更加复杂的应用场景中，优化实际复杂工程的难题，如应用到车联网资源管理中是下一步研究的重点。

参考文献

[1] KENNEDY J. Particle swarm optimization[J]. Proc. of 1995 IEEE Int. Conf. Neural Networks, (Perth, Australia), 2011, 4(8): 1942-1948.

[2] ARORA S, SINGH S. Butterfly optimization algorithm: a novel approach for global optimization[J]. Soft Computing, 2019, 23(3): 715-734.

[3] MIRJALILIS. SCA: A sine cosine algorithm for solving optimization problems[J]. Knowledge-Based Systems, 2016, 96: 120-133.

[4] MIRJALILIS, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[5] 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.

[6] XUE Jiankai, SHEN Bo. A novel swarm intelligence optimization approach: sparrow search algorithm [J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.

[7] ALTUNBEYÖ F. A modified seahorse optimization algorithm based on chaotic maps for solving global optimization and engineering problems[J]. Engineering Science and Technology, an International Journal, 2023, 41: 101408.

[8] 周鹏，董朝轶，陈晓艳，等. 基于 Tent 混沌和透镜成像学习策略的平衡优化器算法[J]. 控制与决策, 2023, 38(6): 1569-1576.

[9] MALEK M R A, AZIZ N A A, ALELYANI S, et al. Comfort and energy consumption optimization in smart homes using bat algorithm with inertia weight[J]. Journal of Building Engineering, 2022, 47: 103848.

[10] CHEN Haitao, WANG Wenchuan, WANG Xiaonan, et al. Multi-objective reservoir operation using particle swarm optimization with adaptive random inertia weights [J]. Water Science and Engineering, 2020, 13(2): 136-144.

[11] WANG Zhenyu, CHEN Yicun, DING Sheng, et al. A novel particle swarm optimization algorithm with Lévy flight and orthogonal learning[J]. Swarm and Evolutionary Computation, 2022, 75: 101207.

[12] SHARMA A, SHARMA A, AVERBUKH M, et al. Improved moth flame optimization algorithm based on opposition-based learning and Lévy flight distribution for parameter estimation of solar module[J]. Energy Reports, 2022, 8: 6576-6592.

[13] WANG Liying, CAO Qingjiao, ZHANG Zhenxing, et al. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems [J]. Engineering Applications of Artificial Intelligence, 2022, 114: 105082.

[14] 刘园园，贺兴时. 基于 Tent 混沌映射的改进的萤火虫算法[J]. 纺织高校基础科学学报, 2018, 31(4): 511-518.

[15] HENDROP P, RISANURI H, IGI A. Enhancing sentiment classification performance using hybrid query expansion ranking and binary particle swarm optimization with adaptive inertia weights[J]. ICT Express, 2022, 8(2): 189-197.

[16] 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.

[17] PEROLAT J, COUSO B I, LOQUIN K. Generalizing the Wilcoxon rank-sum test for interval data[J]. International Journal of Approximate Reasoning, 2015, 56(1): 108-121.

[18] DERRAC J, GARCíA S, MOLINA D, et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms [J]. Swarm & Evolutionary Computation, 2011, 1(1): 3-18.

[19] YANG Xinshe. Optimization techniques and applications with examples[M]. USA: John Wiley & Sons, 2018.