Advances in Applied Mathematics 应用数学进展, 2026, 15(6), 8-25
Published Online June 2026 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2026.156261

Hans汉斯

基于Sobol序列和随机差分变异的改进人工兔优化算法

闫　佳*，梁昔明

北京建筑大学理学院，北京

收稿日期：2026年4月29日；录用日期：2026年5月23日；发布日期：2026年6月2日

摘　要

针对人工兔优化算法在收敛精度不足和易陷入局部最优的不足，提出了一种基于Sobol序列和随机差分变异的改进人工兔优化算法(SDARO)。算法用Sobol序列替代简单的随机初始化，使初始种群能够更均匀地分布在搜索空间中，从而增强群体的全局搜索能力，以提高算法的收敛速度和精度。在算法的位置更新过程中，引入改进的随机差分变异策略，对部分个体进行变异操作，以帮助群体跳出局部最优。23个基准函数的数值结果表明，与其他优化算法相比，所提出的算法SDARO在收敛精度、收敛速度和伸缩性方面均表现较好。三个工程设计问题的实验结果也进一步验证了算法SDARO能够获得较对比算法更优的设计方案。

关键词

人工兔优化算法，Sobol序列，随机差分变异，数值实验

An Improved Artificial Rabbit Optimization Algorithm Based on Sobol Sequences and Stochastic Difference Variants

Jia Yan*, Ximing Liang

School of Science, Beijing University of Civil Engineering and Architecture, Beijing

Received: April 29, 2026; accepted: May 23, 2026; published: June 2, 2026

Abstract

An improved artificial rabbit optimization algorithm (SDARO) based on Sobol sequences and sto-

*通讯作者。

文章引用: 闫佳, 梁昔明. 基于 Sobol 序列和随机差分变异的改进人工兔优化算法[J]. 应用数学进展, 2026, 15(6): 8-25. DOI: 10.12677/aam.2026.156261

chastic differential variants is proposed to overcome the shortcomings of the artificial rabbit optimization algorithm such as low convergence accuracy and easy falling into local optimum. The algorithm replaces simple random initialization with Sobol sequences to make the initial population distribute more evenly in the search space, which enhances the global search capability of the population and improves the convergence speed and accuracy of the algorithm. In the position updating process of the algorithm, a stochastic differential mutation strategy is improved and used to enable some individuals to perform the mutation operation in order to help the population to escape from the local optimum. Numerical results on 23 benchmark functions show that, compared to other optimization algorithms, the proposed algorithm SDARO performs better in convergence accuracy, convergence speed and scalability. The experimental results on three engineering design problems also show that the proposed algorithm SDARO can obtain better design solutions than the comparison algorithms.

Keywords

Artificial Rabbit Optimization Algorithms, Sobol Sequences, Stochastic Difference Variances, Numerical Experiments

1. 引言

在过去几十年里，元启发式算法在处理各种工程领域的高难度优化问题时越来越受欢迎。大多数元启发式算法来自自然灵感，基本可分为五类：基于生物进化机制的算法(EB)，如遗传算法(Genetic Algorithm, GA) [1]等；基于群体行为的算法(SB)，如粒子群优化算法(Particle Swarm Optimization, PSO) [2]等；基于物理/化学原理的算法(PCB)，如重力搜索优化算法(Gravity Search Algorithm, GSA) [3]等；基于人类社会行为的算法(HB)，如基于教学的优化算法(Teaching-Learning-Based Optimization, TLBO) [4]等以及不属于上述四类的其他算法。

人工兔优化算法(Artificial rabbits optimization, ARO) [5]是 2022 年提出的一种模拟兔子在自然界中生存行为的智能优化算法，兔子行为主要分为迂回觅食和随机躲藏两个阶段，并根据当前能量因子的大小在两个阶段之间切换。迂回觅食阶段，兔子总是在其他兔子巢穴的地方寻找食物并增加干扰源。随机躲藏阶段，兔子会在自己巢穴周围挖许多洞穴，并随机选择一个作为躲避天敌的洞穴。

与其他智能优化算法相同，算法 ARO 也存在收敛精度不高和容易陷入局部最优等不足。针对这些不足，王伟等人[6]基于动态透镜成像学习策略和基于正弦函数的非线性递减能量因子改进人工兔优化算法；尹安琳和张著洪[7]基于 SPM 混沌映射、基于 Levy 飞行机制的精英个体引导策略和改进型能量因子改进人工兔优化算法；张瑞成和孙伟良等人[8]基于 Circle 混沌映射和 Levy 飞行机制改进了人工兔优化算法。以上改进有效地提高了人工兔优化算法的性能，但其中有些改进策略较为复杂，在面对大规模或高维优化问题时算法的计算开销较大，影响算法的实际可行性，且不适用于所有优化问题。根据“没有免费午餐定理”，不存在一种算法可以在所有可能的问题上总是比其他算法表现得更好。

为探讨克服人工兔优化算法陷入局部最优的不足及提升寻优效率的新途径，本文提出了一种新的基于 Sobol 序列[9]和随机差分变异的改进人工兔优化算法(SDARO)，以期拓展人工兔优化算法的适用性和优化能力。算法采用 Sobol 序列初始化，使初始种群能够在搜索空间中分布更均匀，从而加强算法的全

局探索能力。同时引入改进的随机差分变异策略，使个体以固定概率对其侯选位置进行变异，以避免算法过早收敛，从而提升算法的全局搜索能力和优化效率。数值实验表明，改进所得的人工兔优化算法在收敛速度和收敛精度方面均有显著提升。

2. 人工兔优化算法

人工兔优化算法 ARO 源自兔子在自然界中的生存策略，分为迂回觅食阶段和随机躲藏阶段，主要求解如式(2.1)所示的无约束优化问题。

$$\min f(\mathbf{x}) \tag{2.1}$$

其中 $\boldsymbol{x} \in R^n$，若 $f(\boldsymbol{x}^*) \le f(\boldsymbol{x}), \forall \boldsymbol{x} \in R^n$，则称 $\boldsymbol{x}^*$ 为 $f(\boldsymbol{x})$ 在全空间 R^n 上的全局极小点。

2.1. 能量因子

在算法 ARO 中，设计了一个能量因子用来模拟每只兔子从迂回觅食阶段到随机躲藏阶段的转换。能量因子定义如下：

$$A(t) = 4\left(1 - \frac{t}{T}\right)\ln\frac{1}{r} \tag{2.2}$$

其中 r 为(0, 1)中的随机数，t 为当前迭代次数，T 为最大迭代次数。

当能量因子 $A(t) > 1$ 时，兔子倾向于随机探索不同兔子的区域进行觅食，即发生迂回觅食；当能量因子 $A(t) \le 1$ 时，兔子倾向于挖掘自己区域的洞穴并进行躲避，即发生随机躲藏。每只兔子根据能量因子 $A(t)$的值，可以在迂回觅食或随机躲藏之间切换。

2.2. 迂回觅食阶段

假设兔群中的每只兔子都有自己的区域，该区域内有一些草和 n 个洞穴，兔子们总是随机地去对方的区域觅食。同时，会在食物源周围随机活动以保证获得足够的食物。迂回觅食公式如下：

$$\mathbf{v}_i(t+1) = \mathbf{x}_j(t) + \mathbf{R}\cdot\left(\mathbf{x}_i(t) - \mathbf{x}_j(t)\right) + round\left(0.5\cdot(0.05 + r_1)\right)\cdot n_1 \cdot \mathbf{E} \tag{2.3}$$

其中 $i, j = 1,\cdots,N$ 且 $i \ne j$，N 为种群个数，$\mathbf{v}_i(t+1)$ 是第 i 只兔子在 $t+1$ 时间的候选位置，$\mathbf{x}_i(t)$ 是第 i 只兔子在 t 时间的位置，$round$ 表示四舍五入到最接近的整数，r_1 是(0, 1)中的随机数，n_1 服从正态分布 N(0, 1)，$\mathbf{E}$ 为分量全为 1 的 n 维向量。$\mathbf{R} \in R^n$ 为奔跑因子，用于模拟兔子的奔跑特性，公式为：

$$\mathbf{R} = L \cdot \mathbf{c} \tag{2.4}$$

其中 L 是运行长度，公式为：

$$L = \left(\mathrm{e} - \mathrm{e}^{\left(\frac{t-1}{T}\right)^2}\right)\cdot\sin(2\pi r_2) \tag{2.5}$$

其中 r_2 是(0, 1)中的随机数，式(2.5)反映了个体在进行迂回觅食时的移动速度。$\mathbf{c} = \left[c(1), c(2), \cdots, c(n)\right]$ 为映射向量，$c(k)\ (k = 1,\cdots,n)$ 取值为：

$$c(k) = \begin{cases} 1, & k = g(l) \\ 0, & k \ne g(l) \end{cases} \tag{2.6}$$

其中 $l = 1,\cdots,\lceil r_3 \cdot n \rceil$，$\lceil\cdot\rceil$ 是上限函数，$\mathbf{g} = randperm(n)$ 返回从 1 到 n 的整数的随机排列的向量，r_3 是(0, 1)中的随机数。

2.3. 随机躲藏阶段

每次迭代时，兔子在自己区域内随机选取搜索空间一个维度挖掘洞穴，受到捕食者的追逐和攻击时，个体将选择该洞穴躲避，以避免被捕获。随机躲藏过程公式如下：

$$\mathbf{v}_i(t+1)=\mathbf{x}_i(t)+\mathbf{R}\cdot\left(r_4\cdot\mathbf{b}_{i,r}(t)-\mathbf{x}_i(t)\right) \tag{2.7}$$

其中 $i=1,\cdots,N$，r_4 是(0, 1)中的随机数，$\mathbf{b}_{i,r}$ 代表随机生成的一个用于躲避的洞穴，用公式表示为：

$$\mathbf{b}_{i,r}(t)=\mathbf{x}_i(t)+H\cdot\mathbf{w}_r\cdot\mathbf{x}_i(t) \tag{2.8}$$

其中 $H=(T-t+1)\cdot n_2/T$ 是隐藏参数，在迭代过程中以随机扰动的方式从 1 到 $1/T$ 线性递减，n_2 服从正态分布 $N(0,1)$，$\mathbf{w}_r=\left[w_r(1),w_r(2),\cdots,w_r(n)\right]$ 为映射向量，$w_r(k)\ (k=1,\cdots,n)$ 取值如下：

$$w_r(k)=\begin{cases}1, & k=\lceil r_5\cdot n\rceil\\ 0, & k\neq\lceil r_5\cdot n\rceil\end{cases} \tag{2.9}$$

其中 r_5 是(0, 1)中的随机数。

2.4. 位置更新

第 i 只兔子在执行迂回觅食或随机躲藏后，将根据侯选位置和当前位置的适应度值进行选择，其位置更新公式为：

$$\mathbf{x}_i(t+1)=\begin{cases}\mathbf{x}_i(t), & f(\mathbf{x}_i(t))\le f(\mathbf{v}_i(t+1))\\ \mathbf{v}_i(t+1), & f(\mathbf{x}_i(t))\ge f(\mathbf{v}_i(t+1))\end{cases} \tag{2.10}$$

即当第 i 只兔子的候选位置优于当前位置时，它就会放弃当前位置，停留在由式(2.3)或式(2.7)生成的候选位置上，否则它保持原位置不变。

2.5. 算法 ARO 步骤

第一步设置种群数量 N 和最大迭代次数 T，令当前迭代次数 $t=0$；
第二步随机初始化种群 $\mathbf{x}_i(t)$ $(1\le i\le N)$，根据适应度值找到最佳个体 $\mathbf{x}_{best}$；
第三步令 $i=1$；
第四步由式(2.2)计算能量因子 $A(t)$；
第五步若 $A(t)>1$，根据式(2.3)执行迂回觅食阶段并转第七步；
第六步根据式(2.7)执行随机躲藏阶段；
第七步根据式(2.10)进行位置更新；
第八步若 $i<N$，则令 $i=i+1$ 并返回第四步；
第九步根据适应度值更新最佳个体 $\mathbf{x}_{best}$；
第十步若 $t\ge T$，则输出 $\mathbf{x}_{best}$；否则，令 $t=t+1$ 并返回第三步。

3. 改进的人工兔优化算法

3.1. Sobol 序列初始化

在算法 ARO 中采用随机的方式初始化种群，这种方式产生的种群个体在搜索空间中通常分布不均匀，如图 1 所示，使得算法寻优时难以对整个搜索空间进行全面搜索，从而影响算法收敛速度和搜索精度。

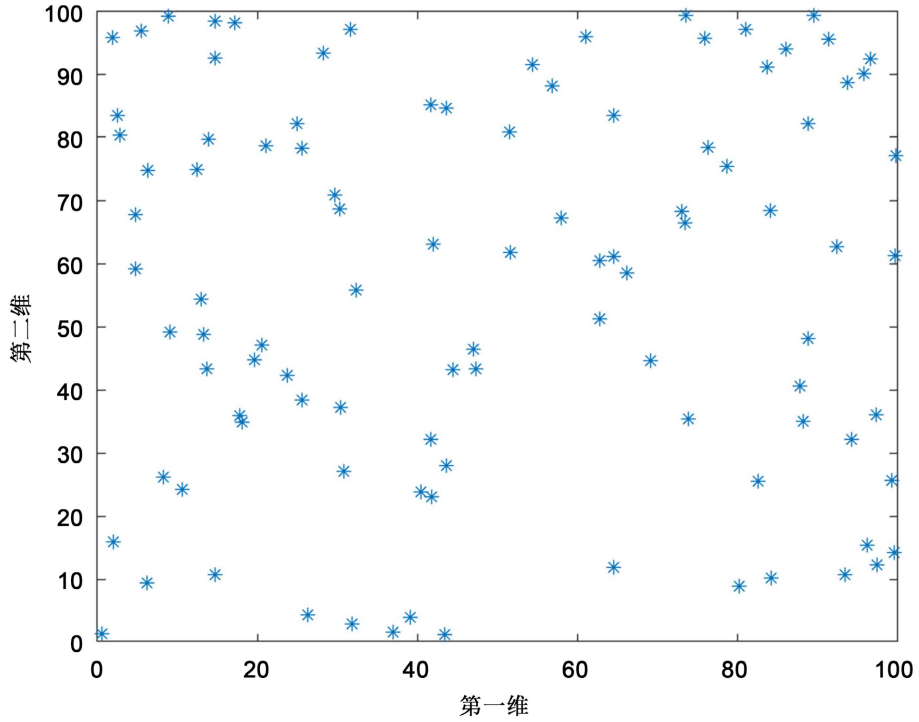

Figure 1. Schematic of random sampling method (2D)
图 1. 随机取样方式示意图(二维)

Sobol 序列是一种低差异序列，可使用基于二进制位的方法生成高效均匀分布的数值序列。Sobol 序列记为：$\mathbf{S}=\left[S_1,S_2,\cdots,S_n\right]$，其中 S_i 按下面方式产生：先将正整数 i 表示为二进制：$i=b_s(i)b_{s-1}(i)\cdots b_1(i)$，其中 $b_m(i)$ 为 0 或 1 ($m=1,\cdots,s$)，则 S_i 为：

$$S_i=v_1b_1(i)\oplus v_2b_2(i)\oplus\cdots\oplus v_sb_s(i) \tag{3.1}$$

其中 $\oplus$ 为二进制异或运算符号，v_j 通过以下过程递归生成：先随机生成一个形如式(3.2)的系数为 0 或 1 的 p 次多项式：

$$P=x^p+a_1x^{p-1}+\cdots+a_{p-1}x+1 \tag{3.2}$$

其中 a_r 为 0 或 1 $(r=1,\cdots,p-1)$。再随机生成正奇数 m_j $(j=1,\cdots,p)$，使得 $m_j<2^j$，并根据式(3.3)得到 v_j：

$$v_j=\frac{m_j}{2^j} \tag{3.3}$$

当 $j>p$ 时，根据式(3.4)递归生成 v_j：

$$v_j=a_1v_{j-1}\oplus a_2v_{j-2}\oplus\cdots\oplus a_{p-1}v_{j-p+1}\oplus v_{j-p}\oplus\frac{v_{j-p}}{2^p} \tag{3.4}$$

Sobol 序列具有分布均匀性好的优点，相较于种群随机初始化，采取 Sobol 序列初始化种群可使其个体更均匀地分布在搜索空间中，如图 2 所示，这有利于对整个搜索空间的全面搜索，从而提高算法的收敛速度和搜索精度。采用 Sobol 序列初始化所得个体位置如式(3.5)所示：

$$\mathbf{x}_i(0)=lb*\mathbf{E}+(ub-lb)\mathbf{S} \tag{3.5}$$

其中 $\mathbf{x}_i(0)$ 为第 i 个个体的位置($1\le i\le N$)，lb 与 ub 分别为优化问题中变量的下限与上限。

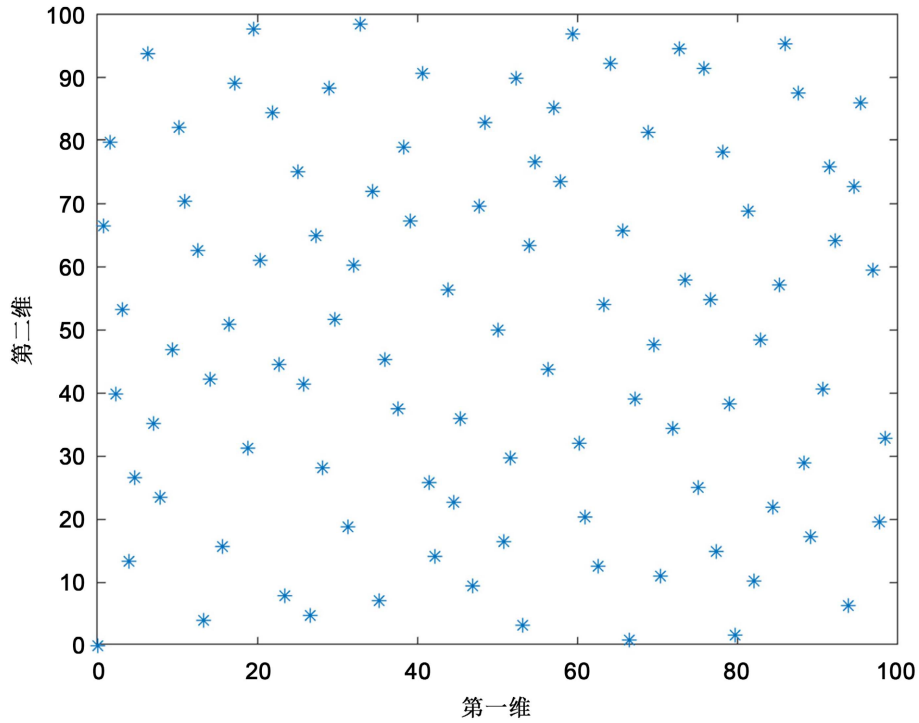

Figure 2. Schematic of Sobol sequence sampling method (2D)
图 2. Sobol 序列取样方式示意图(二维)

3.2. 改进的随机差分变异

在人工兔优化算法 ARO 中，其迭代前期多偏向进行迂回觅食阶段，个体倾向于向其他个体所在区域移动，很难探索新区域；迭代后期，所有个体均向当前群体中最优个体所在区域靠拢，致使群体多样性降低，从而导致由式(2.3)和式(2.7)产生的侯选位置 $\mathbf{v}_i(t+1)$ 容易陷入局部最优。本文考虑对侯选位置进行适当变异以促使其跳出局部最优。

随机差分变异是一种常见的变异策略，它将当前个体的侯选位置与群体中随机选取的个体位置和当前最优个体位置分别进行随机差分生成新侯选位置，其表达式为：

$$\mathbf{v}_i^*(t+1)=rand\left(\mathbf{x}_{best}-\mathbf{v}_i(t+1)\right)+rand\left(\mathbf{x}_j(t)-\mathbf{v}_i(t+1)\right) \tag{3.6}$$

其中 $i=1,\cdots,N$，t 为当前迭代次数，$\mathbf{v}_i^*(t+1)$ 为变异后的侯选位置，$\mathbf{x}_{best}$ 为当前最优个体位置，$\mathbf{x}_j(t)$ 为群体中随机选取的个体位置($j\neq i$)，*rand* 为[0, 1]之间的随机数，该随机数提高了变异后侯选位置的随机性，可有效防止兔群早熟收敛而陷入局部最优。

人工兔优化算法 ARO 的迭代初期主要进行全局搜索，随机差分变异中随机项 *rand* 的叠加，可能导致变异步长较大，致使兔群在某些无效区域进行过多的搜索，或者错过潜在最优解所在的局部区域。随着迭代进行，算法进入局部探索，此时随机差分变异的高随机性容易过度扰动当前解的结构，导致变异后的侯选位置偏离目标区域，从而降低寻优效率。鉴于此，本文对随机差分变异进行如下改进：

$$\mathbf{v}_i^*(t+1)=\mathbf{v}_i(t+1)+rand\left(\mathbf{x}_{best}-\mathbf{v}_i(t+1)\right)-rand\left(\mathbf{x}_j(t)-\mathbf{v}_i(t+1)\right) \tag{3.7}$$

式(3.7)引入侯选位置 $\mathbf{v}_i(t+1)$ 作为基准，通过两项扰动进行调整。$\mathbf{v}_i(t+1)$ 的引入确保变异过程不会让个体完全脱离其当前侯选位置，使变异幅度被控制在一定范围内，避免了新侯选位置过度偏离，保障了算法的稳定性；第一个差分项通过最优个体 $\mathbf{x}_{best}$ 产生指向全局最优解的分量，引导个体向全局最优靠

拢，加速算法收敛；第二个差分项通过随机个体 $\mathbf{x}_j(t)$ 产生随机分量，引导个体向随机方向探索，增强物种多样性。两扰动项均由随机数与差分项共同影响，使得变异步长能够随算法迭代而自适应缩减，进一步将扰动幅度控制在一定范围内，避免了新候选位置过度偏离，保障了算法的稳定性。

综上所述，改进的随机差分变异以当前候选解为基准避免变异过度偏离，同时通过双差分项将扰动幅度控制在稳定范围内，有效平衡了算法收敛和随机探索。

若所有个体均进行变异，将产生过高的计算量。因此对每个个体先生成一个随机数 $rand \in (0,1)$，只对满足 $rand > (t/T)^2$ 的个体的侯选位置进行变异操作。变异后的个体根据式(3.8)选择其候选位置。

$$\boldsymbol{v}_i(t+1) = \begin{cases} \boldsymbol{v}_i(t+1), & f(\boldsymbol{v}_i(t+1)) \le f(\boldsymbol{v}_i^*(t+1)) \\ \boldsymbol{v}_i^*(t+1), & f(\boldsymbol{v}_i(t+1)) > f(\boldsymbol{v}_i^*(t+1)) \end{cases} \tag{3.8}$$

3.3. 算法步骤

在人工兔优化算法 ARO 中，用 Sobol 序列替代种群的随机初始化，并使个体侯选位置以一定概率进行改进的随机差分变异操作，从而得到一种改进的人工兔优化算法，记为 SDARO，其迭代步骤如下：

第一步设置种群数量 N 和最大迭代次数 T，令当前迭代次数 $t = 0$；

第二步按式(3.5)初始化种群 $\mathbf{x}_i(t)$ $(1 \le i \le N)$，根据适应度值找到最佳个体 $\mathbf{x}_{best}$；

第三步令 $I = 1$；

第四步由式(2.2)计算能量因子 $A(t)$；

第五步若 $A(t) > 1$，根据式(2.3)执行迂回觅食阶段，并转第七步；

第六步根据式(2.7)执行随机躲藏阶段；

第七步生成随机数 $rand$，若 $rand > (t/T)^2$，则按式(3.7)进行随机差分变异并根据式(3.8)选择其候选位置；

第八步根据式(2.10)进行位置更新；

第九步若 $i < N$，则令 $i = i + 1$ 并返回第四步；

第十步根据适应度值更新最佳个体 $\mathbf{x}_{best}$；

第十一步若 $t \ge T$，则输出 $\mathbf{x}_{best}$；否则，令 $t = t + 1$ 并返回第三步。

4. 实验结果与分析

为评估所得改进人工兔优化算法(SDARO)的性能，使用 CEC2005 中的 23 个经典函数进行数值实验，该 23 个函数包括两种类型：单峰函数(UFs, F1~F7)和多峰函数(MFs, F8~F23)，其中 F1~F13 的维数是可变的，F14~F23 的维数是固定的；UFs 具有唯一的全局最优值，可以评估优化算法的开发性能，MFs 有多个局部极值，可以评估优化算法的探索性能。

4.1. 固定迭代次数对比目标函数值精度

选取五种对比算法：人工兔优化算法(ARO) [5]、粒子群算法(PSO) [2]、人工蜂群算法(Artificial Bee Colony Algorithm, ABC) [10]、差分进化算法(Differential Evolution Algorithm, DE) [11]和布谷鸟优化算法(Chirp Scaling Algorithm, CS) [12]。算法种群规模均设置为 50，问题 F1~F13 的目标函数维数设为 30。在固定最大迭代次数为 1000 次的情况下，将算法 SDARO 与五种对比算法对 23 个目标函数分别独立极小化 30 次，并记录各算法对每个目标函数 30 次所得目标函数值的平均值(mean)、标准差(std)和最好值(best)，结果如表 1~3 所示，其中五种对比算法上的相应结果均来自文献[5]，加粗数值表示一行中最小的数值。

Table 1. Objective function values after 1000 iterations for 30-dimensional UFs
表 1. 对 30 维 UFs 迭代 1000 次的目标函数值结果

函数		SDARO	ARO	PSO	ABC	DE	CS
F1	mean	**0**	1.82E−124	2.15E−04	2.36E−03	3.64E−14	9.68E−03
	std	**0**	6.63E−124	2.25E−04	1.53E−03	6.06E−14	4.52E−03
	best	**0**	2.29E−142	3.00E−06	7.09E−04	1.26E−15	3.27E−03
F2	mean	**0**	2.68E−69	2.96E−04	2.32E−04	4.38E−08	1.40E+00
	std	**0**	1.21E−68	2.31E−04	1.53E−04	2.53E−88	5.61E−01
	best	**0**	1.44E−77	4.20E−05	8.10E−05	1.49E−08	6.31E−01
F3	mean	**0**	1.24E−95	2.84E+03	9.56E+03	5.69E+00	4.73E+02
	std	**0**	6.78E−94	1.34E+03	1.75E+03	3.91E+00	1.10E+02
	best	**0**	7.00E−115	1.14E+03	5.93E+03	9.26E−01	2.90E+02
F4	mean	**0**	9.92E−52	1.74E+01	2.45E+01	9.17E+00	3.25E+00
	std	**0**	2.96E−52	3.62E+00	2.28E+00	4.00E+00	8.55E−01
	best	**0**	1.84E−59	1.09E+01	1.98E+01	2.18E+00	1.60E+00
F5	mean	**6.75E−05**	4.55E−03	9.47E+01	5.47E+02	3.00E+01	3.86E+01
	std	**1.45E−04**	5.12E−03	7.90E+01	2.10E+02	1.76E+01	1.03E+01
	best	**2.28E−07**	2.41E−04	7.62E+00	2.40E+02	4.01E+00	2.87E+01
F6	mean	**0**	**0**	1.33E−01	**0**	1.33E−01	**0**
	std	**0**	**0**	4.34E−01	**0**	4.34E−01	**0**
	best	**0**	**0**	**0**	**0**	**0**	**0**
F7	mean	**2.39E−04**	2.51E−04	5.64E−02	9.53E−02	2.15E−01	3.09E−02
	std	2.95E−04	**1.07E−04**	2.03E−02	2.39E−02	7.24E−02	7.93E−03
	best	**2.81E−05**	6.30E−05	1.92E−02	5.29E−02	1.16E−01	1.38E−02

Table 2. Objective function values after 1000 iterations for 30-dimensional MFs
表 2. 对 30 维 MFs 迭代 1000 次的目标函数值结果

函数		SDARO	ARO	PSO	ABC	DE	CS
F8	mean	**−11746.1615**	−11209.6764	−5139.3732	−5123.7760	−5310.2672	−8691.8135
	std	663.3832	462.3670	577.6833775	460.6752	661.6603	**235.2491**
	best	**−12569.4885**	−11996.76693	−6951.1321	−6.53E+03	−7387.3949	−9032.4037
F9	mean	**0**	**0**	30.8756	157.8279	164.9567	83.2324
	std	**0**	**0**	8.8839	21.3024	17.5115	13.0994
	best	**0**	**0**	1.59E+01	1.11E+02	1.28E+02	5.89E+01
F10	mean	**4.44E−16**	8.88E−16	7.60E−03	5.33E−02	5.41E−08	4.1525
	std	**0**	**0**	8.23E−03	4.05E−02	2.62E−08	1.4865
	best	**4.44E−16**	8.88E−16	9.45E−04	1.24E−02	1.81E−08	2.02E+00

续表

	mean	**0**	**0**	1.48E−02	1.63E−01	2.05E−03	9.63E−02
F11	std	**0**	**0**	1.36E−02	1.15E−01	3.89E−03	4.18E−02
	best	**0**	**0**	0.0001	8.28E−03	7.99E−15	0.0261
	mean	**5.55E−13**	4.84E−08	0.3758	15.0140	6.91E−03	1.01E+00
F12	std	**2.14E−12**	4.75E−08	7.26E−01	4.6712	2.63E−02	3.17E−01
	best	4.49E−15	5.53E−09	1.44E−04	7.76E+00	**1.33E−16**	5.65E−01
	mean	**9.67E−07**	3.70E−05	0.1909	36.2589	5.31E−02	1.34E−05
F13	std	**4.87E−07**	4.09E−04	3.88E−01	18.1859	2.89E−01	6.31E−02
	best	**1.76E−14**	1.04E−08	2.75E−03	1.29E+01	3.90E−15	6.29E−02

从表 1 可以看出，算法 SDARO 对函数 F1~F4 和 F6 在 30 次独立极小化中均能得到它们的理论最优目标函数值；对函数 F5 和 F7，算法 SDARO 也能得到相较于对比算法更小的目标函数值。除函数 F7 外，算法 SDARO 均能得到比对比算法更小的目标函数值标准差；对函数 F7，算法 SDARO 得到与算法 ARO 相同精度的目标函数值标准差，且小于其他四种对比算法相应的目标函数值标准差。表明算法 SDARO 在极小化单峰函数 F1~F7 时具有非常高效的寻优性能和较强的稳定性。

从表 2 可以看出，对函数 F8~F13，算法 SDARO 都能得到优于对比优化算法的目标函数值平均值。除函数 F8 外，算法 SDARO 均能得到优于对比算法的目标函数值标准差。除函数 F12 外，SDARO 算法在 30 次独立极小化中获得的最好目标函数值均优于五种对比算法。特别地，对于函数 F8，算法 SDARO 获得的标准差虽稍逊于对比算法，但最好目标函数值达到了理论最优值。表明算法 SDARO 在处理高维多峰函数时能有效探索收敛区域，增强了算法寻优能力，同时算法在大多函数上的稳定性均有明显提高。

从表 3 中可以看出，除函数 F20 外，算法 SDARO 均可得到优于或等于其他对比算法的目标函数值平均值。对函数 F14~F23，SDARO 算法在 30 次独立极小化中获得的最好目标函数值均为目标函数对应的理论最优目标函数值。值得注意的是，针对结构复杂的函数 F20~F23，算法 SDARO 虽有效探索收敛区域，但所得标准差均会高于部分对比算法，表明算法全局探索与局部开发之间的平衡仍需进一步协调以加强稳定性。整体而言算法 SDARO 在求解固定维多峰函数方面具有很强的竞争力。

总而言之，在固定最大迭代次数的情况下，算法 SDARO 大多能得到较对比算法更高精度的目标函数值。

Table 3. Objective function values after 1000 iterations for fixed dimensional MFs
表 3. 对固定维 MFs 迭代 1000 次的目标函数值结果

函数		SDARO	ARO	PSO	ABC	DE	CS
	mean	**0.998**	0.998004	0.998004	0.9980054	0.998004	0.998004
F14	std	**8.25E−17**	2.93E−15	2.51E−13	7.00E−06	4.26E−14	5.12E−13
	best	**0.998**	0.9980038	0.998003838	0.99800384	0.998004	0.998004
	mean	**0.0003075**	**0.0003075**	0.00036665	0.00050967	**0.0003075**	**0.0003075**
F15	std	4.84E−08	9.22E−18	0.000190754	5.00E−05	**1.42E−19**	3.92E−08
	best	**0.0003075**	**0.0003075**	**0.0003075**	0.00040355	**0.0003075**	**0.0003075**

续表

	mean	**−1.03162845**	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628
F16	std	6.71E−16	6.71E−16	6.78E−16	**6.65E−16**	6.78E−16	6.78E−16
	best	**−1.03162845**	−1.031628	**−1.03162845**	−1.0316284	−1.031628	−1.031628
	mean	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**
F17	std	**0**	**0**	**0**	**0**	**0**	**0**
	best	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**	**0.3978874**
	mean	**3**	**3**	**3**	**3**	**3**	**3**
F18	std	**1.30E−15**	2.19E−15	1.99E−15	2.86E−11	2.03E−15	1.88E−15
	best	**3**	**3**	**3**	**3**	**3**	**3**
	mean	**−3.86278**	**−3.86278**	**−3.86278**	**−3.86278**	**−3.86278**	**−3.86278**
F19	std	2.63E−15	2.71E−15	2.71E−15	**1.67E−15**	2.71E−15	2.71E−15
	best	**−3.86278215**	−3.862782	**−3.86278215**	−3.8627821	−3.86278	−3.86278
	mean	−3.3101	−3.301995	−3.25407213	−3.321995	−3.28633	**−3.322**
F20	std	3.75E−02	4.34E−03	0.060415416	**1.53E−15**	0.055415	1.92E−13
	best	**−3.322**	−3.321995	−3.32199517	−3.3219952	**−3.322**	**−3.322**
	mean	**−10.1532**	**−10.1532**	−6.13117176	−10.110816	−9.98479	**−10.1532**
F21	std	4.11E−04	**7.12E−15**	2.835886928	0.16828352	0.922443	3.80E−14
	best	**−10.1532**	**−10.1532**	−10.1531997	**−10.1532**	**−10.1532**	**−10.1532**
	mean	**−10.402941**	−10.4029	−8.21793419	−10.4029	−10.4029	−10.4029
F22	std	1.50E−04	**9.90E−16**	2.978914237	3.53E−14	1.75E−15	2.80E−14
	best	**−10.402941**	−10.40294	−10.4029406	**−10.402941**	−10.4029	−10.4029
	mean	**−10.53641**	**−10.53641**	−7.71692326	−10.5364	−10.5364	−10.5364
F23	std	7.70E−07	1.81E−15	3.250563289	**8.09E−14**	1.81E−15	4.20E−12
	best	**−10.53641**	**−10.53641**	−10.5363098	**−10.53641**	−10.5364	−10.5364

4.2. 给定目标函数值精度对比迭代次数

为进一步探索算法 SDARO 的数值效果，设置其最大迭代次数为 1000，种群规模为 50，问题 F1~F13 的目标函数维数为 30。算法 SDARO 和算法 ARO 分别对目标函数独立极小化 30 次，统计 30 次独立极小化中所得目标函数值达到相应理论最优目标函数值 10^{-6} 精度所需要的最少迭代次数(min)、平均迭代次数(mean)以及最多迭代次数(max)，相应结果如表 4 所示，其中加粗数值表示一行中最小的数值。

从表 4 中可以看出，除对函数 F7、F8 和 F20 外，算法 SDARO 达到目标函数值精度 10^{-6} 所需的最少、平均及最多迭代次数均小于算法 ARO 的相应次数；对函数 F7 和 F8，算法 SDARO 和算法 ARO 均无法在 1000 次迭代内达到指定的目标函数值精度 10^{-6}。因为 Sobol 序列的默认起始点为 0，因此对函数 F1~F4、F6 和 F9~F11，算法 SDARO 初始化种群中的最佳个体位置即为相应问题的最优解，从而达到目标函数值精度。因此，算法 SDARO 达到指定目标函数值精度 10^{-6} 所需的迭代次数普遍比算法 ARO 少，即算法 SDARO 比算法 ARO 更快收敛到指定精度的目标函数值，其计算量更小。

Table 4. The number of iterations required to achieve 10^{-6} accuracy of objective function value
表 4. 目标函数值达到 10^{-6} 精度所需迭代次数统计结果

	函数	SDARO	ARO	函数	SDARO	ARO	函数	SDARO	ARO
min		**1**	96		**1**	85		**51**	55
mean	F1	**1**	122	F9	**1**	121	F17	**81**	94
max		**1**	151		**1**	141		**140**	142
min		**1**	149		**1**	146		**52**	60
mean	F2	**1**	167	F10	**1**	168	F18	**68**	71
max		**1**	199		**1**	192		**78**	83
min		**1**	146		**1**	100		**59**	70
mean	F3	**1**	191	F11	**1**	116	F19	**75**	80
max		**1**	254		**1**	141		**84**	89
min		**1**	203		**340**	529		**43**	103
mean	F4	**1**	243	F12	**516**	621	F20	343	**216**
max		**1**	279		**687**	780		**1000**	**1000**
min		**689**	1000		**220**	634		**112**	138
mean	F5	**990**	1000	F13	**410**	928	F21	**185**	208
max		**1000**	**1000**		**1000**	**1000**		**311**	365
min		**1**	35		**15**	35		**124**	139
mean	F6	**1**	53	F14	**34**	53	F22	**171**	228
max		**1**	68		**59**	87		**234**	283
min		**1000**	**1000**		**56**	165		**119**	137
mean	F7	**1000**	**1000**	F15	**72**	264	F23	**182**	233
max		**1000**	**1000**		**117**	394		**269**	332
min		**1000**	**1000**		**15**	35			
mean	F8	**1000**	**1000**	F16	**28**	51			
max		**1000**	**1000**		**41**	75			

4.3. 不同规模的数值实验

为了验证算法 SDARO 对较大规模问题的数值效果，将函数 F1~F13 的维数设置为 500，其余参数的设置与 3.1 相同。将算法 SDARO 与五种对比算法 ARO [5]、PSO [2]、ABC [10]、DE [11]和 CS [12]分别对函数 F1~F13 独立极小化 30 次，统计各算法对每个函数极小化 30 次所得目标函数值的平均值(mean)和标准差(std)，结果如表 5 所示，其中五种对比算法的相应结果均来自文献[5]，加粗数值表示一行中最小的数值。

从表 5 可以看出，算法 SDARO 对单峰函数 F1~F4 和 F6 在 30 次独立极小化中均能得到它们的理论最优目标函数值；对单峰函数 F5 和 F7，算法 SDARO 相较于其他算法也能得到更小的目标函数值。对函数 F1~F6，算法 SDARO 在 30 次独立极小化中所得目标函数值的标准差均小于其他对比函数。表明算法 SDARO 在极小化高维单峰函数时具有非常高效的优化性能和较强的稳定性。

Table 5. Objective function values obtained by six algorithms on 500-dimensional functions
表 5. 六种算法对 500 维函数的目标函数值结果

函数		SDARO	ARO	PSO	ABC	DE	CS
F1	mean	**0**	9.12E−111	2.61E+05	5.43E+05	5.44E+04	4.69E+04
	std	**0**	2.80E−111	1.73E+04	1.44E+04	2.17E+03	4.74E+03
F2	mean	**0**	1.45E−61	7.12E+02	3.89E+93	2.93E+02	1.00E+10
	std	**0**	4.11E−61	2.84E+01	8.69E+93	2.23E+01	3.42E+09
F3	mean	**0**	1.89E−82	4.55E+06	5.44E+06	3.97E+06	2.93E+06
	std	**0**	1.08E−81	1.43E+06	5.42E+05	7.59E+05	2.25E+05
F4	mean	**0**	3.35E−44	8.16E+01	9.27E+01	5.09E+01	3.75E+01
	std	**0**	7.35E−44	3.85	0.98	5.03	2.07
F5	mean	**2.1401**	3.38	5.38E+08	1.75E+09	3.00E+07	1.40E+07
	std	**2.5401**	3.12	7.57E+07	1.18E+08	4.32E+06	2.68E+06
F6	mean	**0**	**0**	2.66E+05	5.62E+05	5.37E+04	5.52E+04
	std	**0**	0	2.57E+04	2.33E+04	5.22E+03	2.67E+03
F7	mean	**2.82E−04**	3.12E−04	3.86E+03	1.25E+04	2.03E+02	8.81E+01
	std	1.78E−04	**1.08E−04**	4.20E+02	1.12E+03	3.71E+01	8.97531897
F8	mean	**−165980.9899**	−8.58E+04	−2.03E+04	−1.83E+04	−2.00E+04	−5.00E+04
	std	20354.6302	1.57E+03	1.70E+03	1.79E+03	**1.47E+03**	1.66E+03
F9	mean	**0**	**0**	3.81E+03	6.32E+03	2.61E+03	4.43E+03
	std	**0**	0	1.41E+02	8.48E+01	7.50E+02	7.00E+01
F10	mean	**4.44E−16**	8.88E−16	1.74E+01	1.93E+01	1.24E+01	1.67E+01
	std	**0**	**0**	3.86E−02	1.58E−01	4.47E−01	8.32E−01
F11	mean	**0**	**0**	2.26E+03	5.00E+03	4.69E+02	4.54E+02
	std	**0**	**0**	1.00E+02	2.34E+02	2.96E+01	3.13E+01
F12	mean	**3.06E−05**	5.45E−03	7.67E+08	3.45E+09	4.62E+06	6.29E+05
	std	**3.67E−05**	1.82E−03	1.05E+08	2.58E+08	5.69E+05	3.32E+05
F13	mean	**8.51E−03**	1.46	1.65E+09	6.85E+09	5.13E+07	1.00E+10
	std	**0.011442**	0.98	3.90E+08	7.19E+08	1.68E+07	2.13E+09

对多峰函数 F9 和 F11，算法 SDARO 与算法 ARO 均可得到它们的理论最优目标函数值；对多峰函数 F8、F10、F12 和 F13，算法 SDARO 均能得到优于其他算法的目标函数值。同时对函数 F9~F13，算法 SDARO 在 30 次独立极小化中所得目标函数值的标准差均小于其他对比函数。表明算法 SDARO 处理高维多峰函数时具有很强的寻优能力和稳定性。

综上所述，算法 SDARO 在极小化高维问题时具有较对比算法更明显的数值稳定性。

5. 应用

为了验证算法 SDARO 求解实际问题的数值性能，将算法 SDARO 应用到压力容器设计、张拉/压缩

弹簧设计和悬梁臂设计三个经典工程问题中，该三个工程问题均为约束优化问题。由于算法 SDARO 主要求解无约束优化问题，因此在求解三个工程问题过程中，均先采用罚函数法将它们转化为一系列无约束优化问题，再利用算法 SDARO 求解转化后的无约束优化问题。

5.1. 压力容器设计

压力容器由空心圆柱体和空心半球体组成。压力容器设计的目标是在满足四个约束条件的情况下最小化压力容器制造成本，该问题共有四个变量：半球体厚度(T_s)、圆柱体厚度(T_h)、圆柱体内半径(R)和圆柱长度(L)。令

$$\boldsymbol{x}=\left[x_1,x_2,x_3,x_4\right]=\left[T_s,T_h,R,L\right]$$

要求各变量满足：

$$0\le x_1\le 99,0\le x_2\le 99,$$
$$10\le x_3\le 200,10\le x_4\le 200$$

压力容器设计的数学模型如下：

$$\begin{aligned}\min\quad & f_1(\boldsymbol{x})=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3\\ \text{s.t.}\quad & g_1(\boldsymbol{x})=-x_1+0.0193x_3\le 0\\ & g_2(\boldsymbol{x})=-x_2+0.00954x_3\le 0\\ & g_3(\boldsymbol{x})=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3+1296000\le 0\\ & g_4(\boldsymbol{x})=x_4-240\le 0\end{aligned}$$

对该压力容器设计问题，设置算法种群规模为 30，最大迭代次数为 1000，用算法 SDARO 和十种对比算法 ARO [5]、飞蛾火焰优化算法(Moth-flame Optimization, MFO) [13]、CS [12]、协同进化粒子群算法(Co-evolutionary PSO, CPSO) [14]、混合粒子群算法 (Hybrid PSO, HPSO) [15]、协同进化差分进化算法(Co-evolutionary DE, CDE) [16]、蜣螂优化算法(Dung Beetle Optimizer, DBO) [17]、哈里斯鹰优化算法(Harris Hawks Optimization, HHO) [18]、北方苍鹰优化算法(Northern Goshawk Optimization, NGO) [19]及改进的黑寡妇优化算法(IBWOA) [20]分别独立求解 30 次，各算法 30 次所得目标函数值的最小值及其对应变量的取值如表 6 所示，其中算法 ARO、MFO、CS、CPSO、HPSO 和 CDE 的相应结果均来自文献[5]，算法 DBO、HHO、NGO 和 IBWOA 的相应结果均来自文献[21]。

从表 6 可以看出，算法 SDARO 所得设计向量 $\boldsymbol{x}=(0.7781686,0.3846492,40.31962,200)$ 对应的最优目标函数值 $f_1=5885.3328$ 明显小于其它十种优化算法的相应结果，表明了算法 SDARO 所得设计方案可以更有效地降低压力容器的制作成本。

Table 6. Results of the pressure vessel design problem
表 6. 压力容器设计问题的数值结果

算法	T_s	T_h	R	L	f_1
SDARO	0.7781686	0.3846492	40.31962	200	5885.3328
ARO	0.778243	0.384751	40.323389	199.947942	5885.6679
MFO	0.8125	0.4375	42.0984	176.6356	6059.7143
CS	0.8125	0.4375	42.0984	176.6366	6059.7143
CPSO	0.8125	0.4375	42.091266	176.7465	6061.0777

续表

HPSO	0.8125	0.4375	42.0984	176.6366	6059.7143
CDE	0.8125	0.4375	42.098411	176.6377	6059.7340
DBO	0.790507	0.400050	40.778197	193.771680	5959.4951
HHO	0.832124	0.431377	42.905333	168.397260	6112.6071
NGO	0.849501	0.413433	42.637891	170.161217	6171.8100
IBWOA	0.790291	0.390621	40.945189	191.471146	5906.6529

5.2. 张拉/压缩弹簧设计

弹簧由金属丝绕成多个线圈构成。张拉/压缩弹簧设计的目标是在满足四个约束条件的情况下最小化弹簧重量，该问题共有三个变量：金属丝直径(d)、线圈直径(D)和线圈数量(N)。令

$$\boldsymbol{x}=\left[x_1,x_2,x_3\right]=\left[d,D,N\right]$$

要求各变量满足：

$$0.05\le x_1\le 2,0.25\le x_2\le 1.3,2\le x_3\le 15$$

张拉/压缩弹簧设计地数学模型如下：

$$\begin{aligned}
\min\quad & f_2\left(\boldsymbol{x}\right)=\left(x_3+2\right)x_2x_1^2\\
\text{s.t.}\quad & g_1\left(\boldsymbol{x}\right)=1-\frac{x_3x_2^3}{71785x_1^4}\le 0\\
& g_2\left(\boldsymbol{x}\right)=\frac{4x_2^2-x_1x_2}{12566\left(x_2x_1^3-x_1^4\right)}+\frac{1}{5108x_1^4}\le 0\\
& g_3\left(\boldsymbol{x}\right)=1-\frac{140.45x_1}{x_2^2x_3}\le 0\\
& g_4\left(\boldsymbol{x}\right)=\frac{x_1+x_2}{1.5}-1\le 0
\end{aligned}$$

对该张拉/压缩弹簧设计问题，设置算法种群规模为 30，最大迭代次数为 1000，用算法 SDARO 和十种对比算法 ARO [5]、CPSO [14]、HPSO [15]、MFO [13]、和声搜索算法(Harmony Search, HS) [22]、CDE [16]、改进的黑寡妇优化算法(QIWBWO) [23]、蝠鲼觅食优化算法(Manta Ray Foraging Optimization Algorithm, MRFO) [24]、算术优化算法(Arithmetic Optimization Algorithm, AOA) [25]及 HHO [18]分别独立求解 30 次，各算法 30 次所得目标函数值的最小值及其对应变量地取值如表 7 所示，其中算法 ARO、CPSO、HPSO、MFO、HS、CDE 的相应结果均来自文献[5]，算法 QIWBWO、MRFO、AOA、HHO 的相应结果均来自文献[26]。

Table 7. Results of the tension/compression spring problem
表 7. 张拉/压缩弹簧问题的数值结果

算法	T_s	T_h	R	f_2
SDARO	0.051897	0.361749	11	0.01266602
ARO	0.051897	0.361749	11	0.01266602
CPSO	0.051728	0.357644	11.244543	0.0126747

续表

HPSO	0.051706	0.357126	11.265083	0.0126652
MFO	0.051994	0.364109	10.868422	0.0126669
HS	0.051154	0.349871	12.076432	0.0126706
CDE	0.051609	0.354714	11.410831	0.0126702
QIWBWO	0.0546000	0.4314000	7.9840000	0.0129000
MRFO	0.0523734	0.3733461	10.3831265	0.0126813
AOA	0.0508000	0.3348000	11.7020000	0.0126810
HHO	0.0562000	0.4754000	6.6670000	0.0130160

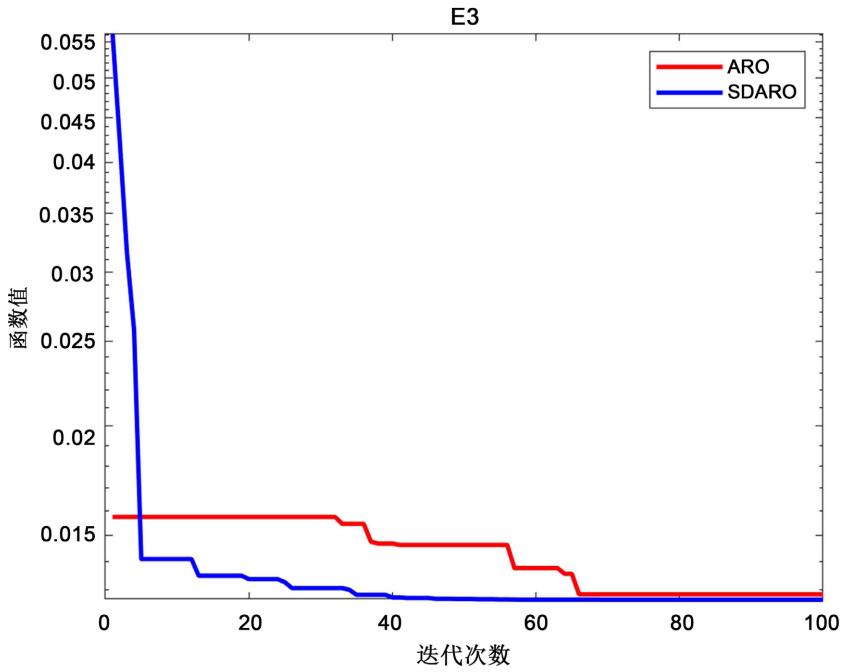

Figure 3. Convergence curves of algorithms SDARO and ARO for solving the tension/compression spring design problem
图 3. 算法 SDARO 和算法 ARO 求解张拉/压缩弹簧设计问题的收敛曲线

从表 7 可以看出，算法 SDARO 所得设计向量 $\boldsymbol{x}=(0.051897,0.361749,11)$ 对应的最优目标函数值 $f_2=0.01266602$ 与算法 ARO 的相应值相同，且小于另外九种算法所得目标函数值。从图 3 给出的算法 SDARO 和算法 ARO 求解该问题的目标函数值下降曲线可以看出，算法 SDARO 比算法 ARO 更快地搜索到问题的最优解，进而可以有效地减少计算时间成本。

5.3. 悬臂梁设计

悬臂梁由五个长度和厚度均相等的部分组成，但五部分的横截面边长不同。悬臂梁设计的目标是在满足一个约束条件的情况下最小化悬臂梁重量，该问题共有五个变量 $x_i\,(i=1,\cdots,5)$，分别代表五个部分的横截面边长。令

$$\mathbf{x}=\left[x_1,x_2,x_3,x_4,x_5\right]$$

要求各横截面边长满足：

$$0.01 \le x_i \le 100, i = 1, \cdots, 5$$

悬臂梁设计问题的数学模型如下：

$$\begin{aligned} \min \quad & f_2(\mathbf{x}) = 0.0624\left(x_1 + x_2 + x_3 + x_4 + x_5\right) \\ \text{s.t.} \quad & g_1(\mathbf{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0 \end{aligned}$$

对此悬臂梁设计问题，设置算法种群规模为 30，最大迭代次数为 1000，用算法 SDARO 和十种对比算法 ARO [5]、共生生物搜索算法(Symbiotic Organ- isms Search, SOS) [27]、CS [12]、蚱蜢优化算法(Grasshopper Optimization Algorithm, GOA) [28]、移动渐进线的方法(Method of Moving Asymptotes, MMA) [29]、MFO [13]、天鹰优化算法(Aquila Optimizer, AO) [30]、AOA [25]、人工大猩猩部队优化器(Artificial Gorilla Troops Optimizer, GTO) [31]及饥饿游戏搜索算法(Hunger Games Search, HGS) [32]分别独立求解 30 次，各算法 30 次所得目标函数值的最小值及其对应变量的取值如表 8 所示，其中算法 ARO、SOS、CS、GOA、MMA、MFO 的相应结果均来自文献[5]，算法 AO、AOA、GTO、HGS 的相应结果均来自文献[33]。

Table 8. Numerical results for cantilever beam design problem
表 8. 悬臂梁设计问题的数值结果

算法	x_1	x_2	x_3	x_4	x_5	f_3
SDARO	6.01598	5.30928	4.49431	3.50146	2.15263	1.339956
ARO	6.00683	5.31144	4.49352	3.50290	2.15905	1.339956
SOS	6.01878	5.30344	4.49587	3.49896	2.15564	1.3996
CS	6.00890	5.30490	4.50230	3.50770	2.15040	1.33999
GOA	6.01167	5.31297	4.48307	3.50279	2.16333	1.33996
MMA	6.01000	5.30000	4.49000	3.49000	2.15000	1.34000
MFO	5.98487	5.31673	4.49733	3.51362	2.16162	1.39988
AO	6.02384	5.24827	4.51331	3.51537	2.17753	1.340248
AOA	5.51940	6.18295	4.19171	4.06200	2.79905	1.419919
GTO	6.01626	5.30758	4.49427	3.50171	2.15381	1.339957
HGS	6.01201	5.30493	4.50546	3.51320	2.13837	1.339977

从表 8 可以看出，算法 SDARO 所得设计向量 $\bar{x}$ = (6.01597908, 5.30928341, 4.49430544, 3.50146085, 2.15263085)对应的最优目标函数值 f_3 = 1.339956 与算法 ARO 的相应值相同，且小于另外九种算法所得目标函数值。另外算法 SDARO 不仅得到了与算法 ARO 相同的悬臂梁最小设计重量，而且得到了与算法 ARO 不同的悬臂梁设计变量值，即算法 SDARO 可以为悬臂梁设计问题提供不同的最优设计方案。

6. 结束语

针对算法 ARO 收敛精度低和易早熟收敛的不足，提出了基于 Sobol 序列和随机差分变异的改进人工兔优化算法(SDARO)。利用 Sobol 序列替代了种群的随机初始化，确保初始种群在搜索空间内分布更加均匀，有利于算法的全局搜索。同时改进了随机差分变异策略，对部分个体的侯选位置进行变异操作，以防止种群陷入局部最优。对 23 个基准函数的数值结果表明，算法 SDARO 对单峰函数和多峰函数的寻

优均有较高的搜索精度和稳定性。不同规模的数值实验结果表明，在求解高维无约束优化问题时，算法SDARO展现出更强的数值稳定性。三个工程设计问题的实验结果进一步验证了算法SDARO的有效性，它能在实际应用中为工程设计提供更有效的设计方案。

基金项目

国家自然科学基金(12361106)，北京建筑大学2021年校级教育科学研究项目(Y2113)。

参考文献

[1] Holland, J.H. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press.

[2] Eberhart, R. and Kennedy, J. (1995) A New Optimizer Using Particle Swarm Theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, 4-6 October 1995, 39-43. https://doi.org/10.1109/mhs.1995.494215

[3] Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. (2009) GSA: A Gravitational Search Algorithm. *Information Sciences*, **179**, 2232-2248. https://doi.org/10.1016/j.ins.2009.03.004

[4] Rao, R.V., Savsani, V.J. and Vakharia, D.P. (2012) Teaching-Learning-Based Optimization: An Optimization Method for Continuous Non-Linear Large Scale Problems. *Information Sciences*, **183**, 1-15. https://doi.org/10.1016/j.ins.2011.08.006

[5] Wang, L., Cao, Q., Zhang, Z., Mirjalili, S. and Zhao, W. (2022) Artificial Rabbits Optimization: A New Bio-Inspired Meta-Heuristic Algorithm for Solving Engineering Optimization Problems. *Engineering Applications of Artificial Intelligence*, **114**, Article ID: 105082. https://doi.org/10.1016/j.engappai.2022.105082

[6] 王伟, 龙文. 动态透镜成像学习人工兔优化算法及应用[J]. 广西科学, 2023, 30(4): 735-744.

[7] 尹安琳, 张著洪. 复杂环境下无人机路径规划及其改进型人工兔优化[J]. 系统仿真学报, 2025, 37(1): 79-94.

[8] 张瑞成, 孙伟良, 梁卫征. 基于SSAE-IARO-BiLSTM的工业过程故障诊断研究[J]. 振动与冲击, 2024, 43(15): 244-250, 260.

[9] Joe, S. and Kuo, F.Y. (2003) Remark on Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM Transactions on Mathematical Software*, **29**, 49-57. https://doi.org/10.1145/641876.641879

[10] Karaboga, D. and Akay, B. (2009) A Comparative Study of Artificial Bee Colony Algorithm. *Applied Mathematics and Computation*, **214**, 108-132. https://doi.org/10.1016/j.amc.2009.03.090

[11] Storn, R. and Price, K. (1997) Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, 341-359. https://doi.org/10.1023/a:1008202821328

[12] Yang, X. and Deb, S. (2009) Cuckoo Search via Lévy Flights. 2009 *World Congress on Nature & Biologically Inspired Computing* (*NaBIC*), Coimbatore, 9-11 December 2009, 210-214. https://doi.org/10.1109/nabic.2009.5393690

[13] Mirjalili, S. (2015) Moth-flame Optimization Algorithm: A Novel Nature-Inspired Heuristic Paradigm. *Knowledge-Based Systems*, **89**, 228-249. https://doi.org/10.1016/j.knosys.2015.07.006

[14] He, Q. and Wang, L. (2007) An Effective Co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems. *Engineering Applications of Artificial Intelligence*, **20**, 89-99. https://doi.org/10.1016/j.engappai.2006.03.003

[15] He, Q. and Wang, L. (2007) A Hybrid Particle Swarm Optimization with a Feasibility-Based Rule for Constrained Optimization. *Applied Mathematics and Computation*, **186**, 1407-1422. https://doi.org/10.1016/j.amc.2006.07.134

[16] Huang, F., Wang, L. and He, Q. (2007) An Effective Co-Evolutionary Differential Evolution for Constrained Optimization. *Applied Mathematics and Computation*, **186**, 340-356. https://doi.org/10.1016/j.amc.2006.07.105

[17] Xue, J. and Shen, B. (2023) 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

[18] Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. and Chen, H. (2019) Harris Hawks Optimization: Algorithm and Applications. *Future Generation Computer Systems*, **97**, 849-872. https://doi.org/10.1016/j.future.2019.02.028

[19] Dehghani, M., Hubalovsky, S. and Trojovsky, P. (2021) Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems. *IEEE Access*, **9**, 162059-162080. https://doi.org/10.1109/access.2021.3133286

[20] Wang, Y., He, Q., Zhang, D., Lu, S. and Yuan, C. (2023) Improving Li-Ion Battery Health: Predicting Remaining Useful Life Using IWBOA-ELM Algorithm. *Journal of Energy Storage*, **72**, Article ID: 108547.

https://doi.org/10.1016/j.est.2023.108547

[21] 刘微, 任腾腾, 韩广雨, 等. 多策略改进蜣螂优化算法及其应用[J]. 电子测量技术, 2024, 47(12): 109-121.

[22] Mahdavi, M., Fesanghary, M. and Damangir, E. (2007) An Improved Harmony Search Algorithm for Solving Optimization Problems. *Applied Mathematics and Computation*, **188**, 1567-1579. https://doi.org/10.1016/j.amc.2006.11.033

[23] Hu, G., Du, B., Li, H. and Wang, X. (2022) Quadratic Interpolation Boosted Black Widow Spider-Inspired Optimization Algorithm with Wavelet Mutation. *Mathematics and Computers in Simulation*, **200**, 428-467. https://doi.org/10.1016/j.matcom.2022.04.031

[24] Zhao, W., Zhang, Z. and Wang, L. (2020) Manta Ray Foraging Optimization: An Effective Bio-Inspired Optimizer for Engineering Applications. *Engineering Applications of Artificial Intelligence*, **87**, Article ID: 103300. https://doi.org/10.1016/j.engappai.2019.103300

[25] Abdollahzadeh, B., Gharehchopogh, F.S. and Mirjalili, S. (2021) African Vultures Optimization Algorithm: A New Nature-Inspired Metaheuristic Algorithm for Global Optimization Problems. *Computers & Industrial Engineering*, **158**, Article ID: 107408. https://doi.org/10.1016/j.cie.2021.107408

[26] 张庭溢, 汪弘健. 幼儿园小朋友优化算法[J]. 计算机工程与应用, 2024, 60(23): 109-125.

[27] Cheng, M. and Prayogo, D. (2014) Symbiotic Organisms Search: A New Metaheuristic Optimization Algorithm. *Computers & Structures*, **139**, 98-112. https://doi.org/10.1016/j.compstruc.2014.03.007

[28] Saremi, S., Mirjalili, S. and Lewis, A. (2017) Grasshopper Optimisation Algorithm: Theory and Application. *Advances in Engineering Software*, **105**, 30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004

[29] Chickermane, H. and Gea, H.C. (1996) Structural Optimization Using a New Local Approximation Method. *International Journal for Numerical Methods in Engineering*, **39**, 829-846. https://doi.org/10.1002/(sici)1097-0207(19960315)39:5<829::aid-nme884>3.0.co;2-u

[30] Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M. and Gandomi, A.H. (2021) The Arithmetic Optimization Algorithm. *Computer Methods in Applied Mechanics and Engineering*, **376**, Article ID: 113609. https://doi.org/10.1016/j.cma.2020.113609

[31] Yang, Y., Chen, H., Heidari, A.A. and Gandomi, A.H. (2021) Hunger Games Search: Visions, Conception, Implementation, Deep Analysis, Perspectives, and Towards Performance Shifts. *Expert Systems with Applications*, **177**, Article ID: 114864. https://doi.org/10.1016/j.eswa.2021.114864

[32] Naruei, I. and Keynia, F. (2021) Wild Horse Optimizer: A New Meta-Heuristic Algorithm for Solving Engineering Optimization Problems. *Engineering with Computers*, **38**, 3025-3056. https://doi.org/10.1007/s00366-021-01438-z

[33] 张金钱, 王先鹏, 孔凡康等. 求解工程优化问题的多种智能优化算法仿真[J]. 计算机仿真, 2024, 41(5): 372-377, 454.