Pure Mathematics 理论数学, 2023, 13(7), 1988-2006
Published Online July 2023 in Hans. https://www.hanspub.org/journal/pm
https://doi.org/10.12677/pm.2023.137205

Hans汉斯

鹈鹕初始化的灰狼算法在工程优化上的应用

潘邦勇，刘　敏

贵州大学电气工程学院，贵州　贵阳

收稿日期：2023年6月10日；录用日期：2023年7月14日；发布日期：2023年7月21日

摘　要

为了进一步提高灰狼算法(Grey Wolf Optimization, GWO)的开发能力和勘探能力，提出一种鹈鹕算法(Pelican Optimization Algorithm, POA)初始化灰狼位置的算法(POAGWO)。POAGWO算法利用POA算法来优化GWO算法的初始位置，这有助于增强POAGWO算法的开发能力和勘探能力。然后，通过不同类型的单峰，高维多模态和固定维多模态的6个典型的测试函数对算法的性能进行测试。测试结果表明POAGWO算法在开发能力和勘探能力上都超过了POA算法和GWO算法，并且从6个迭代曲线图中也可以看出改进后的算法可更快地趋向于全局最优解。最后将POAGWO算法应用于4个工程优化设计问题中，应用结果表明POAGWO算法具有很好的求解性能。

关键词

灰狼算法，鹈鹕算法，POAGWO算法，初始化位置，工程优化设计

Application of Grey Wolf Algorithm Initialized by Pelican in Engineering Optimization

Bangyong Pan, Min Liu

School of Electrical Engineering, Guizhou University, Guiyang Guizhou

Received: Jun. 10th, 2023; accepted: Jul. 14th, 2023; published: Jul. 21st, 2023

Abstract

In order to further improve the development and exploration capabilities of Grey Wolf Optimization (GWO), a Pelican Optimization Algorithm (POA) is proposed to initialize the position of Grey Wolf (POAGWO). POAGWO algorithm uses POA algorithm to optimize the initial position of GWO algorithm, which helps to enhance the development ability and exploration ability of POAGWO al-

文章引用: 潘邦勇, 刘敏. 鹈鹕初始化的灰狼算法在工程优化上的应用[J]. 理论数学, 2023, 13(7): 1988-2006.
DOI: 10.12677/pm.2023.137205

gorithm. Then, six typical test functions of different types of single peak, high-dimensional multimodal and fixed dimensional multimodal are used to test the performance of the algorithm. The test results show that POAGWO algorithm outperforms POA algorithm and GWO algorithm in both development and exploration capabilities, and it can also be seen from the six iteration curves that the improved algorithm can quickly approach the global optimal solution. Finally, POAGWO algorithm is applied to four engineering optimization design problems, and the application results show that POAGWO algorithm has good performance.

Keywords

Grey Wolf Optimization, Pelican Optimization Algorithm, POAGWO Algorithm, Initializing the Position, Engineering Optimization Design

1. 引言

2014 年文献[1]提出了一种新型群体智能优化算法：灰狼优化算法。灰狼算法具有结构简单、需要调节的参数少、容易实现等特点，其在对问题的求解精度和收敛速度方面都有良好的性能。因此，GWO 算法被广泛应用于无人作战飞行器路径规划[2]、聚类分析[3]、特征子集选择[4]、多输入输出电力系统[5]等诸多领域。由于 GWO 算法提出的时间不长，因此其理论研究尚未成体系，多数学者都是从特定角度，针对具体问题对 GWO 进行改进和应用研究。这些改进虽然在一定程度提高了算法的性能，但在避免陷入局部最优，以及全局搜索能力上仍不足。因此，更好地提高灰狼算法的收敛速度和寻优能力具有广阔的研究空间[6]。

POA 算法是一种新的元启发式优化算法[7]，灵感来自鹈鹕狩猎行为。它具有调整参数少、收敛速度快、计算简单等优点。由于 POA 算法刚提出来的时间很短，目前对其应用研究尚未有之。

综上所述，根据 GWO 算法存在的一些不足，本文将通过引进鹈鹕算法对灰狼初始种群进行优化，使初始种群的质量大大提高，这有助于加快灰狼迅速搜寻到最优解。然后通过不同类型的单峰，高维多模态和固定维多模态的 6 个典型的测试函数对算法的性能进行测试。最后将 POAGWO 算法应用于压力容器设计、拉伸/压缩弹簧设计、三杆桁架设计和气体输送压缩机设计等 4 个工程优化设计问题。应用结果表明 POAGWO 算法具有很好的求解性能，这也有望将其应用于求解其它的工程优化设计问题。

2. 灰狼算法

2.1. 灰狼算法数学模型

2.1.1. 包围猎物

在捕猎的过程中，将灰狼包围捕捉猎物的群体行为定义为[8]：

$$\vec{\boldsymbol{D}} = \left| \vec{\boldsymbol{C}} \vec{\boldsymbol{X}}_p(t) - \vec{\boldsymbol{X}}(t) \right| \tag{1}$$

$$\vec{\boldsymbol{X}}(t+1) = \vec{\boldsymbol{X}}_p(t) - \vec{\boldsymbol{A}} \vec{\boldsymbol{D}} \tag{2}$$

其中，式(1)则表示了灰狼相对于猎物位置距离；式(2)是灰狼的下一步移动的方位公式；t 则是当前的迭

代次数。

2.1.2. 狩猎

灰狼群体中的个体追捕猎物的数学模型可以用以下公式来描述[8]：

$$\begin{cases} \vec{\boldsymbol{D}}_{\alpha} = \left| \vec{\boldsymbol{C}}_1 \vec{\boldsymbol{X}}_{\alpha} - \vec{\boldsymbol{X}} \right| \\ \vec{\boldsymbol{D}}_{\beta} = \left| \vec{\boldsymbol{C}}_2 \vec{\boldsymbol{X}}_{\beta} - \vec{\boldsymbol{X}} \right| \\ \vec{\boldsymbol{D}}_{\delta} = \left| \vec{\boldsymbol{C}}_3 \vec{\boldsymbol{X}}_{\delta} - \vec{\boldsymbol{X}} \right| \end{cases} \tag{3}$$

其中，$\vec{\boldsymbol{D}}_{\alpha}$，$\vec{\boldsymbol{D}}_{\beta}$，$\vec{\boldsymbol{D}}_{\delta}$ 分别表示 α 狼，β 狼，δ 狼与其他灰狼间的距离向量。$\vec{\boldsymbol{X}}_{\alpha}$，$\vec{\boldsymbol{X}}_{\delta}$，$\vec{\boldsymbol{X}}_{\delta}$ 分别表示 α 狼，β 狼，δ 狼的当前位置向量；$\vec{\boldsymbol{C}}_1$，$\vec{\boldsymbol{C}}_2$，$\vec{\boldsymbol{C}}_3$ 是随机的矩阵向量；$\boldsymbol{X}$ 则是狼群中当前地位低的狼的方位向量。

$$\begin{cases} \vec{\boldsymbol{X}}_1 = \vec{\boldsymbol{X}}_{\alpha} - \vec{\boldsymbol{A}}_1 \vec{\boldsymbol{D}}_{\alpha} \\ \vec{\boldsymbol{X}}_2 = \vec{\boldsymbol{X}}_{\beta} - \vec{\boldsymbol{A}}_2 \vec{\boldsymbol{D}}_{\beta} \\ \vec{\boldsymbol{X}}_3 = \vec{\boldsymbol{X}}_{\delta} - \vec{\boldsymbol{A}}_3 \vec{\boldsymbol{D}}_{\delta} \end{cases} \tag{4}$$

$$\vec{\boldsymbol{X}}(t+1) = \frac{\vec{\boldsymbol{X}}_1 + \vec{\boldsymbol{X}}_2 + \vec{\boldsymbol{X}}_3}{3} \tag{5}$$

其中，式(14)分别定义了狼群中 ω 狼各自朝 α 狼，β 狼，δ 狼前进的步长和方位，式(15)定义了 ω 狼的下一步移动的方位[9]。

2.2. 灰狼优化算法流程

GWO 算法流程图如图 1 所示。

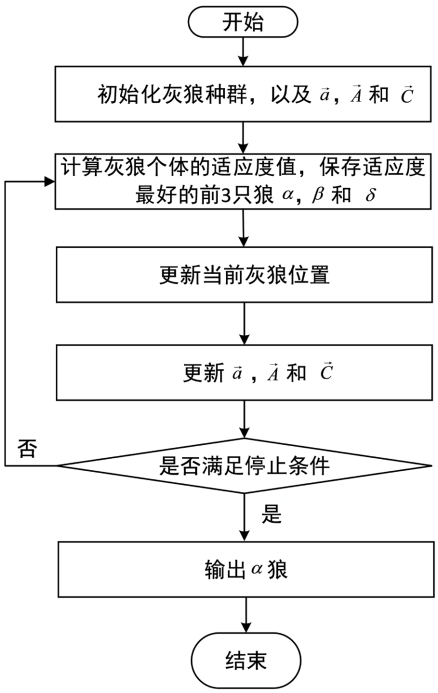

Figure 1. GWO algorithm flow chart
图 1. GWO 算法流程图

灰狼优化算法的基本步骤如下：

步骤 1：初始化灰狼种群，以及 $\vec{\alpha}$ ， $\vec{A}$ 和 $\vec{C}$ 。

步骤 2：计算灰狼个体的适应度值，保存适应度最好的前 3 只狼 α ， β 和 δ 。

步骤 3：根据式(6)~式(12)更新当前灰狼位置。

步骤 4：判断是否达到最大迭代次数，若没有，则重复步骤 2~4；否则输出最优结果。

2.3. 灰狼算法的应用

2.3.1. PID 控制器参数的优化

在工业系统的控制领域中，GWO 算法设计了一种新的最优 PID 控制器，来更好地提高系统的动态性能，同时利用 GWO 算法优化模糊 PID 控制器，应用于互联的水火电力系统的自动控制和用于太阳能热发电站的自动发电控制系统中。

2.3.2. 英支持向量机参数的优化

利用 GWO 算法来优化支持向量机(SVM)的参数，并用优化后的 SVM 来评估水质；利用 GWO 算法调整 SVM 分类器，用于 iEEG 信号分类的案例研究中行。

2.3.3. 神经网络参数的训练

使用 GWO 算法为 BP 神经网络提供初始解决方案，并将 GWO 算法与监督人工神经网络分类器相结合，通过选择人工神经网络的最优参数实现增强的大脑磁共振图像的分类精度。

3. 鹈鹕算法

鹈鹕算法是一种新的随机自然启发优化算法，其主要思想是模拟鹈鹕在狩猎过程中的自然行为。该算法设计的主要思想是对鹈鹕在狩猎过程中的行为和策略进行建模[7]。

3.1. 鹈鹕算法的数学模型

在鹈鹕优化算法中，模拟了鹈鹕在攻击和狩猎时的行为和策略，以此来更新候选解。该狩猎过程分为两个阶段：逼近猎物(勘探阶段)，水面飞行(开发阶段)。

3.1.1. 初始化

鹈鹕种群初始化数学描述如下式：

$$x_{i,j} = l_j + rand \cdot \left(u_j - l_j\right) \tag{6}$$

式中： $x_{i,j}$ 为第 i 个鹈鹕的第 j 维的位置； $rand$ 是[0, 1]范围内的随机数； u_j 和 l_j 分别是求解问题的第 j 维的上下边界。

在鹈鹕优化算法中，鹈鹕种群可以用以下种群矩阵表示：

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \tag{7}$$

式中： $\boldsymbol{X}$ 为鹈鹕的种群矩阵；为第 $\boldsymbol{X}_i$ 个鹈鹕的位置； N 为鹈鹕的种群数量； m 为求解问题的维度。

在鹈鹕优化算法中，求解问题的目标函数可以用来计算鹈鹕的目标函数值；鹈鹕种群的目标函数值

可以用目标函数值向量表示：

$$\boldsymbol{F}=\begin{bmatrix}\boldsymbol{F}_1\\ \vdots\\ \boldsymbol{F}_i\\ \vdots\\ \boldsymbol{F}_N\end{bmatrix}_{N\times 1}=\begin{bmatrix}\boldsymbol{F}\left(x_1\right)\\ \vdots\\ \boldsymbol{F}\left(x_i\right)\\ \vdots\\ \boldsymbol{F}\left(x_N\right)\end{bmatrix}_{N\times 1} \tag{8}$$

3.1.2. 第一阶段：走向猎物(探索阶段)

在第一阶段，鹈鹕识别猎物的位置，然后向这个确定的区域移动。对鹈鹕逼近猎物策略进行建模，使得 POA 算法可以对搜索空间进行扫描，进而发挥 POA 算法在搜索空间中的不同区域的勘探能力。在 POA 算法中，需要重视的一点是：猎物的位置在搜索空间中是随机生成的，这样增加了 POA 算法在解决精确搜索问题上的勘探能力。对上述概念和逼近猎物策略进行数学建模，如下：

$$x_{i,j}^{p_1}=\begin{cases}x_{i,j}+rand\cdot\left(p_j-I\cdot x_{i,j}\right), & F_p<F_i;\\ x_{i,j}+rand\cdot\left(x_{i,j}-p_j\right), & \text{else},\end{cases} \tag{9}$$

式中，$x_{i,j}^{p_1}$ 为基于第 1 阶段更新后第 i 个鹈鹕的第 j 维的位置；I 为 1 或 2 的随机整数；p_j 为猎物的第 j 维的位置；F_p 为猎物的目标函数值。

在 POA 算法中，如果目标函数值在该位置得到改善，则接受鹈鹕的新位置。在这种类型的更新中，也被称为有效更新，该算法不能移动到非最优区域。这个过程可以用如下公式描述：

$$\boldsymbol{X}_i=\begin{cases}\boldsymbol{X}_i^{P_1}, & F_i^{P_1}<F_i;\\ \boldsymbol{X}_i, & \text{else},\end{cases} \tag{10}$$

式中：$\boldsymbol{X}_i^{P_1}$ 为第 i 个鹈鹕的新位置；$F_i^{P_1}$ 为基于第一阶段更新后的第 i 个鹈鹕的新位置的目标函数值 $\boldsymbol{X}_i^{p_1}$。

3.1.3. 第二阶段：水面上穿起(开采阶段)

在第二阶段，当鹈鹕到达水面后，它们在水面上展开翅膀，将鱼向上移动，然后把猎物放在它们的喉咙袋里。鹈鹕水面飞行的这种策略可以使得它们在被攻击区域内捕获更多的鱼。鹈鹕在狩猎过程中的这种行为在数学建模为：

$$x_{i,j}^{P_2}=x_{i,j}+R\cdot\left(1-\frac{t}{T}\right)\cdot\left(2\cdot rand-1\right)\cdot x_{i,j} \tag{11}$$

式中，$x_{i,j}^{p_2}$ 为基于第 2 阶段更新后第 i 个鹈鹕的第 j 维的位置；R 为 0 或 2 的随机整数；t 为当前迭代次数；T 为最大迭代次数。

在此阶段，还使用了有效的更新来接受或拒绝新的鹈鹕位置，该位置是否被接受的数学建模为：

$$\boldsymbol{X}_i=\begin{cases}\boldsymbol{X}_i^{P_2}, & F_i^{P_2}<F_i;\\ \boldsymbol{X}_i, & \text{else},\end{cases} \tag{12}$$

式中：$\boldsymbol{X}_i^{P_2}$ 为第 i 个鹈鹕的新位置；$F_i^{P_2}$ 为基于第一阶段更新后的第 i 个鹈鹕的新位置的目标函数值 $\boldsymbol{X}_i^{p_2}$。

3.2. 鹈鹕优化算法流程

鹈鹕优化算法的流程图如图 2 所示。

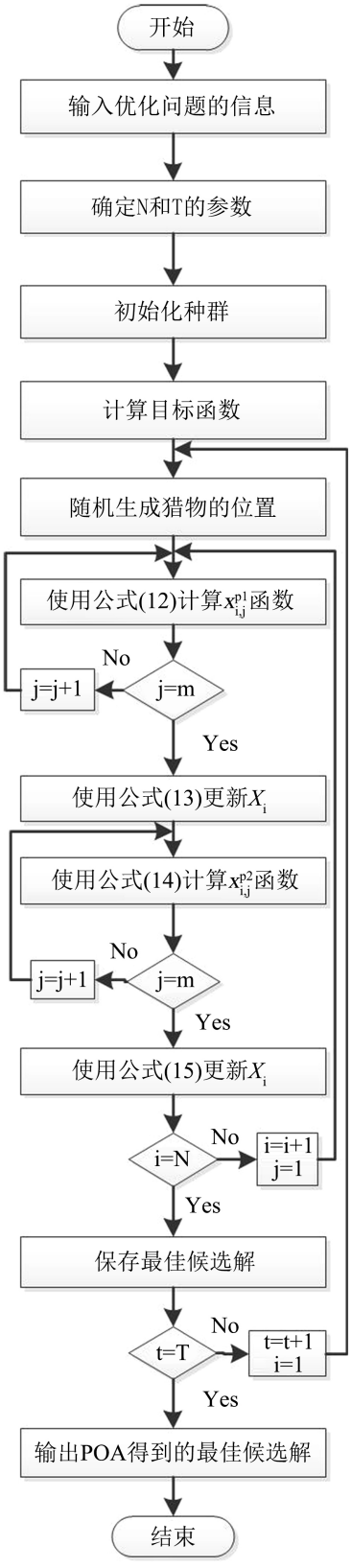

Figure 2. Pelican optimization algorithm flow chart
图 2. 鹈鹕优化算法流程图

POA 算法伪代码如表 1 所示。

Table 1. Pseudo-code of POA
表 1. 鹈鹕优化算法伪代码

开始 POA
1. 输入优化问题信息
2. 确定鹈鹕种群(N)和迭代次数(T)
3. 初始化鹈鹕位置和计算目标函数值
4. For t = 1:T
5. 随机产生鹈鹕位置
6. For I = 1:N
7. 阶段 1：朝猎物移动(探索阶段)
8. For j = 1:m
9. 使用公式(9)计算第 j 维的新位置
10. 结束
11. 使用公式(10)更新第 i 只鹈鹕
12. 阶段 2：第二阶段：水面上穿起(开采阶段)
13. For j = 1:m
14. 使用公式(11)计算第 j 维的新位置
15. 结束
16. 使用公式(12)更新第 i 只鹈鹕
17. 结束
18. 更新最佳候选解
19. 结束
20. 输出 POA 得到的最佳候选解
结束 POA

4. 鹈鹕初始化的灰狼算法

正如我们所知道的那样，启发式算法对初始解的质量是极其依赖的。因为这将直接影响到启发式算法的最终性能。所以，初始种群的选择尤关重要。因此，为了更好地提高灰狼算法初始种群的质量，将用 POA 算法来初始化 GWO 算法的初始位置，故提出了基于鹈鹕初始化灰狼位置的算法，以达到增强灰狼算法的求解性能。

POAGWO 算法的基本步骤如下：

步骤 1：初始化鹈鹕种群，设置鹈鹕数量为 N，最大迭代次数为 T。

步骤 2：利用式(6)、式(7)和式(8)初始化鹈鹕的位置并计算目标函数。

步骤 3：第一阶段：利用式(9)、式(10)更新鹈鹕位置。

步骤 4：第二阶段：利用式(11)、式(12)更新鹈鹕位置。

步骤 5：判断是否满足终止迭代条件。若满足，则输出最优位置；否则，返回步骤 3；

步骤 6：将鹈鹕算法得到的最优位置作为灰狼的初始位置。

步骤 7：根据式(1)~式(5)更新当前狼位置。

步骤 8：计算灰狼个体的适应度值，保存适应度最好的前 3 只狼作为α，β和δ。

步骤 9：判断是否满足终止迭代条件，若没有，则返回步骤 7；否则输出最优结果。

5. 仿真分析

5.1. 测试函数集及实验参数设置

在测试函数中，GWO 算法、POA 算法和 GWOPOA 算法的种群规模和最大迭代次数分别均设置为 20、50。

为了验证 POAGWO 算法的有效性能，分别将 GWO 算法、POA 算法、POAGWO 算法应用于不同类型的单峰，高维多模态和固定维多模态的 6 个典型的测试函数。这 6 个测试函数如下所示。

Rosenbrock 函数(也称为 Valley 或 Banana 函数)：

$$F_1(\boldsymbol{x})=\sum_{i=1}^{m-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$$

Quartic 函数：

$$F_2(\boldsymbol{x})=\sum_{i=1}^{m}ix_i^4+random(0,1)$$

Schwefel's 函数：

$$F_3(\boldsymbol{x})=\sum_{i=1}^{m}-x_i\sin\left(\sqrt{|x_i|}\right)$$

Penalized 高维函数：

$$F_4(\boldsymbol{x})=\frac{\pi}{m}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{m}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}+\sum_{i=1}^{m}u\left(x_i,10,100,4\right)$$

Where:

$$u\left(x_i,a,i,n\right)=\begin{cases}k\left(x_i-a\right)^n, & x_i>a;\\ 0, & -a\le x_i\le a;\\ k\left(-x_i-a\right)^n, & x_i<-a.\end{cases}$$

Penalized 固定维函数：

$$F_5(\boldsymbol{x})=0.1\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{m}\left(x_i-2\right)^2\left[1+\sin^2\left(3\pi x_i+1\right)\right]\right.$$
$$\left.+\left(x_n-1\right)^2\left[1+\sin^2\left(2\pi x_m\right)\right]\right\}+\sum_{i=1}^{m}u\left(x_i,5,100,4\right)$$

Shekel's Foxholes 函数：

$$F_6(\boldsymbol{x})=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$$

对于选取的测试函数，其具体信息如表 2 所示：

Table 2. Basic information of 6 test functions
表 2. 6 个测试函数基本信息

测试函数	搜索范围	维度	理论最优值
$F_1(\boldsymbol{x})$	[−30, 30]	30	0
$F_2(\boldsymbol{x})$	[−1.28, 1.28]	30	0
$F_3(\boldsymbol{x})$	[−500, 500]	30	−12,569
$F_4(\boldsymbol{x})$	[−50, 50]	30	0
$F_5(\boldsymbol{x})$	[−50, 50]	30	0
$F_6(\boldsymbol{x})$	[−65.53, 65.53]	2	0.998

5.2. 结果数据分析

开发能力证明了该算法在局部搜索并尽可能收敛到全局最优的能力。根据这一概念，一个好的优化算法应该能够准确地扫描所确定的最优区域周围的空间，以提供合适的准最优值(opt)。探索能力展示了算法在问题解决空间中进行全局搜索并跨越局部最优区域以发现主要最优区域的能力，以提供合适的平均值(avg)。因此，在比较几种优化算法时，更准确地扫描搜索空间并能够识别包含全局最优区域的算法具有更高的勘探能力。勘探能力在优化除全局最优外还具有多个局部最优的问题中尤为重要。

对选取的测试函数，分别利用 GWO 算法、POA 算法、POAGWO 算法独立运行 30 次，统计得到的最优值与平均值的结果如表 3 所示。

Table 3. Optimization performance comparison of three algorithms in test function
表 3. 3 种算法在测试函数的寻优性能对比

算法		$F_1(\boldsymbol{x})$	$F_2(\boldsymbol{x})$	$F_3(\boldsymbol{x})$	$F_4(\boldsymbol{x})$	$F_5(\boldsymbol{x})$	$F_6(\boldsymbol{x})$
POA	opt	28.8819	−6705.6837	0.4277	2.5345	0.9980	1.3662E−04
	avg	28.9496	−5314.4642	0.7727	2.9870	1.5600	2.7494E−03
GWO	opt	684.4844	−6137.3029	1.1643	6.9361	0.9980	3.4846E−02
	avg	500.5553	−6137.4419	5.8061	23.5432	9.1621	1.1070E−01
POAGWO	opt	28.1349	−8312.4419	0.0492	1.1273	0.9980	4.0672E−05
	avg	28.7533	−6137.4419	0.1798	1.7891	1.4073	2.5936E−03

从表 3 可以看出，与 GWO 算法、POA 算法相比，POAGWO 算法得到的最优值是最小的。因此，POAGWO 算法具有非常强的竞争力和更高的开发能力。同时，POAGWO 算法得到的平均值也是最小。因此，POAGWO 算法对于问题解决空间中的全局搜索具有很高的探索能力，并且能够识别出最佳的局部区域。

GWO 算法、POA 算法和 POAGWO 算法在 6 个测试函数上的迭代曲线如图 3~8 所示。

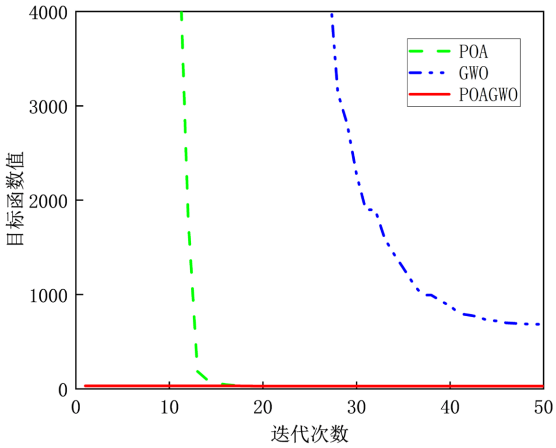

Figure 3. The convergence curve of $F_1(x)$

图 3. $F_1(x)$ 的收敛曲线

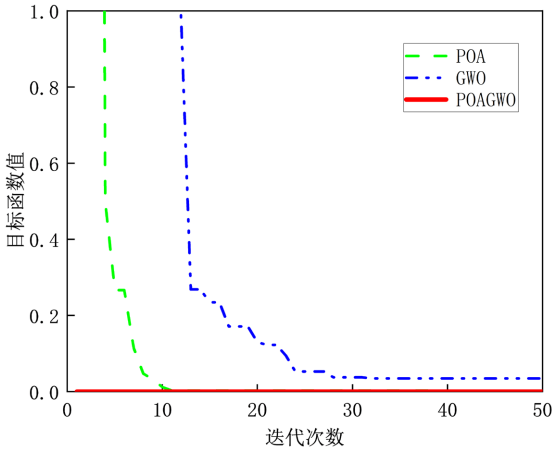

Figure 4. The convergence curve of $F_2(x)$

图 4. $F_2(x)$ 的收敛曲线

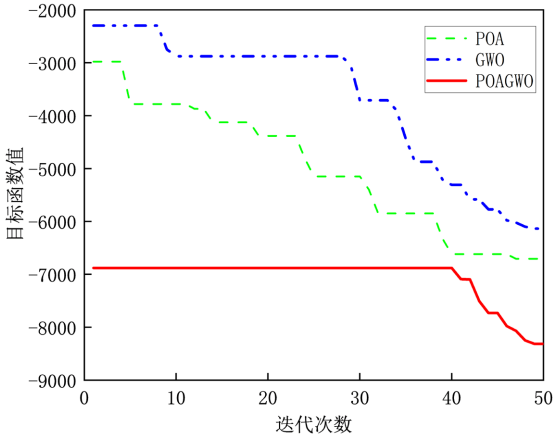

Figure 5. The convergence curve of $F_3(x)$

图 5. $F_3(x)$ 的收敛曲线

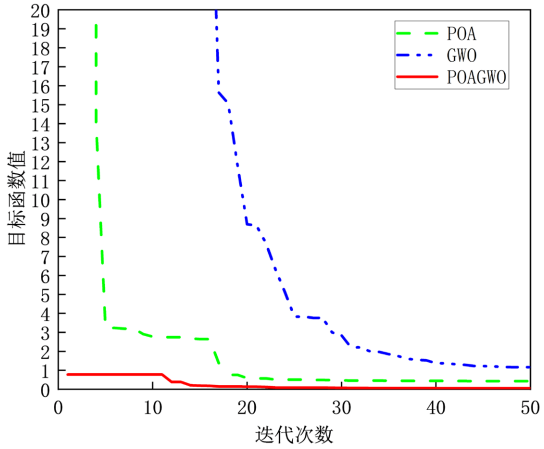

Figure 6. The convergence curve of $F_4(x)$

图 6. $F_4(x)$的收敛曲线

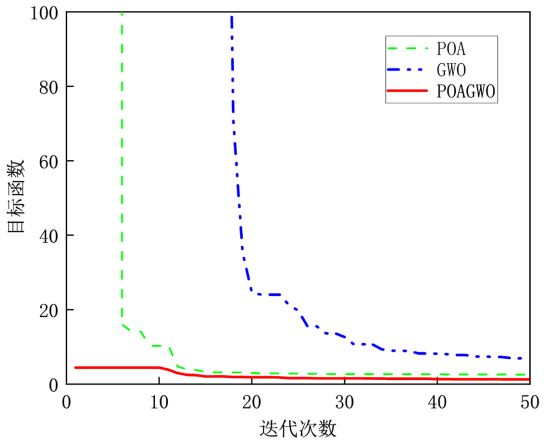

Figure 7. The convergence curve of $F_5(x)$

图 7. $F_5(x)$的收敛曲线

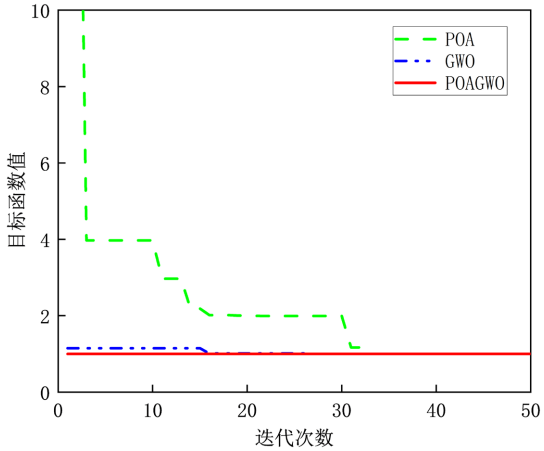

Figure 8. The convergence curve of $F_6(x)$

图 8. $F_6(x)$的收敛曲线

从图 3~8 可以很明显看出，在收敛速度和收敛精度上，POAGWO 算法都要优于 POA 算法和 GWO 算法。所以总体来说，与 POA 算法和 GWO 算法相比，POAGWO 算法表现较为优异。

6. 工程优化应用

在本节中，为了进一步验证本文提出的 POAGWO 算法在解决实际应用中的问题方面的表现。为此，将该方法应用于压力容器设计优化、拉伸/压缩弹簧设计优化、三杆桁架设计优化和气体输送压缩机设计优化等 4 个优化问题，并与其他优化算法做比较。

6.1. 压力容器设计优化

压力容器设计优化问题是一个经典的工程最小化设计问题，其示意图如图 9 所示[10]。该问题的数学模型如下。

Consider:

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

Minimize:

$$F=0.6224x_1x_3x_4+1.778x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

Subject to:

$$g_1\left(\boldsymbol{x}\right)=-x_1+0.0193x_3\le 0$$

$$g_2\left(\boldsymbol{x}\right)=-x_2+0.0954x_3\le 0$$

$$g_3\left(\boldsymbol{x}\right)=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3+1296000\le 0$$

$$g_4\left(\boldsymbol{x}\right)=x_4-240\le 0$$

With bounds:

$$0\le x_1,x_2\le 100,\ \ 10\le x_3,x_4\le 200$$

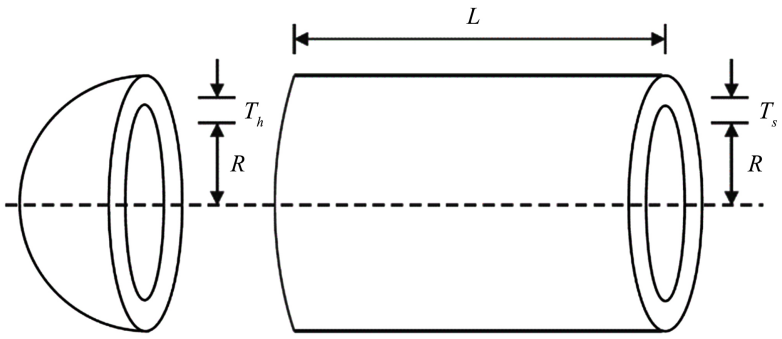

Figure 9. Pressure vessel design diagram
图 9. 压力容器设计示意图

对压力容器设计优化问题采用 POAGWO 算法求解，并与利用文献[11]中的 SBSM 算法、文献[12]中的 CPSO 算法、文献[13]中的 HPSO 算法、文献[14]中的 ES 算法和文献[11]中的 CS 算法求解压力容器设计问题的结果进行比较。比较结果如表 4、表 5 所示。

Table 4. Performance test results of six algorithms for pressure vessel design optimization
表 4. 6 种算法对压力容器设计优化的性能测试结果

算法	最优值	平均值	最差值	标准值
SBSM	6171.0000	6335.0500	6453.6500	---
CPSO	6061.0777	6147.1332	6363.8041	86.4500
HPSO	6288.6770	6099.9393	6288.6700	86.2000
ES	6059.7143	6355.3431	6846.6284	2.600E+2
CS	6059.7143	6447.7360	6495.3470	505.6930
POAGWO	5885.3328	5929.1756	6100.0740	68.8217

从表 4 可以看出，在对压力容器设计优化问题的求解结果中的最优值、最差值、平均值和标准差的比较中，POAGWO 算法都是要优于 SBSM 算法、CPSO 算法、HPSO 算法、ES 算法和 CS 算法，这足以说明了 POAGWO 算法相比于与 SBSM 算法、CPSO 算法、HPSO 算法、ES 算法和 CS 算法在对拉伸/压缩弹簧设计优化问题的求解中拥有更高的寻优性能。

Table 5. Optimization results of six algorithms for pressure vessel design optimization
表 5. 6 种算法对压力容器设计优化的优化结果

算法	SBSM	CPSO	HPSO	ES	CS	POAGWO
T_s	0.8125	0.8125	0.8125	0.8125	0.8125	0.7811
T_h	0.4375	0.4375	0.4375	0.4375	0.4375	0.3862
R	42.0984	42.0913	42.0984	42.0984	42.0984	40.4706
L	176.6371	176.7465	176.6366	176.6371	176.6366	198.0507
g_1	−0.0023	−1.400E−03	−8.800E−07	−1.000E−6	---	−6.350E−05
g_2	−0.0359	−3.600E−03	−3.600E−02	−0.0359	−0.0359	−1.539E−04
g_3	−23420.5966	−118.77	3.1226	−0.8358	---	−727.6033
g_4	−57.71550	−63.2540	−463.3634	−63.3629	---	−41.9493
F	6171.000	6061.0780	6059.7140	6059.7144	6059.7143	5885.3328

从表 5 中可以看出，POAGWO 算法计算后的最小值为 5885.3328，而 SBSM 算法、CPSO 算法、HPSO 算法、ES 算法和 CS 算法计算后的结果分别为 6171.000，6061.0780，6059.7140，6059.7144，6059.7143，均差于 POAGWO 算法。这表明 POAGWO 算法能够搜寻到更好的最优值，是解决压力容器设计优化问题的一种更有效的优化算法。

6.2. 拉伸/压缩弹簧设计优化

拉伸/压缩弹簧设计优化问题是一个重量最小化问题，其示意图如图 10 所示[15]。该问题的数学模型

如下：

Consider:

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

Minimize:

$$F=\left(x_3+2\right)x_1^2x_2$$

Subject to:

$$g_1\left(\boldsymbol{x}\right)=1-\frac{x_2^3x_3}{71785x_1^4}\le 0$$

$$g_2\left(\boldsymbol{x}\right)=-1+\frac{4x_2^2-x_1x_2}{12566x_2x_1^3}+\frac{1}{5108x_1^2}\le 0$$

$$g_3\left(\boldsymbol{x}\right)=1-\frac{140.45x_1}{x_3x_2^2}\le 0$$

$$g_4\left(\boldsymbol{x}\right)=\frac{x_1-x_2}{1.5}-1\le 0$$

With bounds:

Figure 10. Stretch/compression spring design schematic
图 10. 拉伸/压缩弹簧设计示意图

利用 POAGWO 算法对拉伸/压缩弹簧设计优化问题进行求解，并与利用文献[16]中的 CGFDE 算法、CPSO 算法求解拉伸/压缩弹簧设计优化问题的结果进行比较。比较结果如表 6、表 7 所示。

Table 6. Performance test results of three algorithms for tension/compression spring design optimization
表 6. 3 种算法对拉伸/压缩弹簧设计优化的性能测试结果

算法	Best	Mean	Worst	Std
CGFDE	0.0377	0.0457	0.0484	0.0156
CPSO	0.0591	0.0978	0.104	0.0224
POAGWO	0.0099	0.0101	0.0118	0.0004

从表 6 可以看出，在对拉伸/压缩弹簧设计优化问题的求解结果中的最优值、最差值、平均值和标准差的比较中，POAGWO 算法都是要优于 CGFDE 算法、CPSO 算法，这足以说明了 POAGWO 算法相比于与 CGFDE 算法、CPSO 算法在对拉伸/压缩弹簧设计优化问题的求解中拥有更高的寻优性能。

Table 7. Optimization results of three algorithms for tension/compression spring design
表 7. 3 种算法对拉伸/压缩弹簧设计的优化结果

变量	CGFDE	CPSO	POAGWO
d	0.0670	0.0760	0.005
D	0.4940	0.6020	0.0372
P	14.990	14.990	8.8278
g_1	−0.2493	−0.3655	−0.0107
g_2	−0.3720	−0.4823	−0.0069
g_3	−1.5724	−0.9649	−4.7568
g_4	−0.6260	−0.5480	−0.7188
F	0.0377	0.0591	0.0099

从表 7 中可以看出，POAGWO 算法计算后的最小值为 0.0099，而 CGFDE 算法和 CPSO 算法计算后的结果分别为 0.0377、0.0591，均差于 POAGWO 算法。这表明 POAGWO 算法能够搜寻到更好的最优值，是解决拉/压弹簧设计优化问题的一种更有效的优化算法。

6.3. 三杆桁架设计优化

三杆桁架设计优化问题是土木工程中的经典设计优化问题[17]，常用于评估不同算法的性能。其优化目标是最小化三杆桁架的重量。两个变量(x_1，x_2)需要优化以调整杆的横截面积。这个问题的数学模型如下：

Consider:

$$\boldsymbol{x}=\left[x_1,x_2\right]$$

Minimize:

$$F=l\left(x_2+2\sqrt{2}x_1\right)$$

Subject to:

$$g_1\left(\boldsymbol{x}\right)=\frac{x_2}{2x_1x_2+\sqrt{2}x_1^2}P-\sigma\le 0$$

$$g_2\left(\boldsymbol{x}\right)=\frac{x_2+\sqrt{2}x_1}{2x_1x_2+\sqrt{2}x_1^2}P-\sigma\le 0$$

$$g_3\left(\boldsymbol{x}\right)=\frac{1}{x_1+\sqrt{2}x_2}P-\sigma\le 0$$

Where:

$$l=100,\ P=2,\ \sigma=2$$

With bound:

$$0\le x_1,x_2\le 1$$

利用 POAGWO 算法对三杆桁架设计优化问题进行求解，并与利用文献[18]中的 AOA 算法，文献[19]中的 CJAYA 算法，文献[20]中的 mGWO 算法、JADE 算法、CMA-ES 算法、modGWO 算法和 GABC 算法，文献[21]中的 m-SCA 算法、SCA 算法和 CS 算法，文献[22]中的 MFO 算法，文献[23]中的 HHO-SCA 算法，文献[24]中的 CCSA 算法和 Best-so-far 算法求解三杆桁架设计优化问题的结果进行比较。比较结果如表 8 所示。

Table 8. Optimization results of 15 algorithms for three-bar truss design optimization
表 8. 15 种算法对三杆桁架设计优化的优化结果

变量	x_1	x_2	F
AOA	0.79369	0.39426	263.9154
CJAYA	0.7886925585	0.4081990117	263.895844
mGWO	0.7885845	0.4085071	263.8961
JADE	0.78753	0.41150	263.8968
CMA-ES	0.78623	0.41608	263.9867
modGWO	0.7878452	0.410618	263.8974
GABC	9.78784	0.41062	263.8966
m-SCA	0.81915	0.36956	263.8972
SCA	0.78669	0.41426	263.9348
CS	0.78867	0.40902	263.9716
MFO	0.7882447709	0.4094669058	263.8959797
HHO-SCA	0.788498	0.40875	263.8958665
CCSA	0.78865625	0.40830170	263.895844
Best-so-far	0.7886711	0.40912463	263.895976
POAGWO	0.78867513	0.4082482952	263.895843

从表 8 可以看出，POAGWO 算法在三杆桁架设计三杆桁架设计问题中获得的最小重量为 263.895843。AOA 算法、CJAYA 算法、mGWO 算法、JADE 算法、CMA-ES 算法、modGWO 算法、GABC 算法、m-SCA 算法、SCA 算法、CS 算法、MFO 算法、HHO-SCA 算法、CCSA 算法、Best-so-far 算法得到的结果分别为 263.9154，263.895844，263.8961，263.8968，263.9867，263.8974，263.8966，263.8972，263.9348，263.9716，263.895979682，263.8958665，263.895844 和 263.895976，均比 POAGWO 算法得出的结果大。因此，POAGWO 算法得出的结果更符合实际工程中效益最大化的制造需求。可以看出，POAGWO 算法比其他算法具有更好的可搜索性。

6.4. 气体输送压缩机设计优化

气体输送压缩机设计优化问题是机械工程领域流行的优化设计问题。它有四个优化变量和一个不等式约束[25]。气体输送压缩机设计问题的数学模型描述如下：

Consider:

$$\boldsymbol{x}=\left[x_1,x_2,x_3,x_4\right]=\left[h,l,t,b\right]$$

Minimize:

$$F=8.61\times10^5x_1^{1/2}x_2x_3^{-2/3}x_4^{-1/2}+3.69\times10^4x_3+7.72\times10^8x_1^{-1}x_2^{0.219}-765.43\times10^6x_1^{-1}$$

Subject to:

$$g\left(\boldsymbol{x}\right)=x_4x_2^{-2}+x_2^{-2}-1\le0$$

With bounds:

$$20\le x_1\le50,\ 1\le x_2\le2,\ 20\le x_3\le50\ \text{ and }\ 0.1\le x_4\le60$$

利用 POAGWO 算法对气体输送压缩机设计问题进行求解，并与利用文献[26]中的 WOAmM 算法、WOA 算法、SOS 算法、BOA 算法、SSA 算法、PSO 算法和 DE 算法求解气体输送压缩机设计优化问题的结果进行比较。比较结果如表 9 所示：

Table 9. Optimization results of 8 algorithms for gas transport compressor design optimization
表 9. 8 种算法对气体输送压缩机设计优化的优化结果

变量	*h*	*l*	*t*	*b*	*F*
WOAmM	50	1.18	24.83	0.3831	2.9649E+06
WOA	50	1.18	24.58	0.3883	2.9658E+06
SOS	50	1.18	24.58	0.3883	2.9649E+06
BOA	33.19	1.10	26.48	0.2162	3.0070E+06
SSA	26.19	1.10	21.47	0.2119	3.0341E+06
PSO	31.79	1.10	31.57	0.2224	3.0509E+06
DE	50	1.18	24.59	0.3884	2.9649E+06
POAGWO	49.96	1.1778	24.314	0.3874	2.9649E+06

从表 9 中可以看出，POAGWO 算法计算后的最小值为 2.964895E+06，而 WOAmM 算法、WOA 算法、SOS 算法、BOA 算法、SSA 算法、PSO 算法、DE 算法计算后的结果分别为 2.9649E+06、2.9658E+06、2.9649E+06、3.007E+06、3.0341E+06、3.0509E+06、2.9649E+06，均差于 POAGWO 算法。这表明 POAGWO 算法在求解气体输送压缩机设计优化问题的上性能优越，可以找到更好的最优值。

7. 总结

1) 本文中提出一种基于鹈鹕初始化的灰狼算法，通过求解单峰、高维多模态和固定维多模态的 6 个目标函数对算法进行了测试。此外，为了进一步分析所提算法的性能，将 POAGWO 算法的优化结果与 POA 算法和 GWO 算法的优化结果进行了比较。结果表明，POAGWO 算法具有很强的全局寻优能力，并且 POAGWO 算法在有效检查搜索空间和寻找最优区域方面具有很强的开发能力和勘探能力。

2) 通过应用 POAGWO 算法求解压力容器设计优化、拉伸/压缩弹簧设计优化、三杆桁架设计优化和气体输送压缩机设计优化等 4 个工程设计问题，并与其他智能算法进行比较。结果表明，POAGWO 算法

解决实际应用中的设计优化问题方面具有令人满意的性能。另外，也证明了该算法具有很强的求解实际问题的能力。

3) 在实际工程中，POAGWO 算法可以用来解决各种科学领域的优化问题，例如信号去噪、机器学习、电力系统优化等。除此之外，POAGWO 算法中的某些参数是否可以优化，使算法的搜索精度更高，这是有待进一步研究的问题。

参考文献

[1] Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) Grey Wolf Optimizer. *Advances in Engineering Software*, **69**, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[2] Zhang, S., Zhou, Y.Q., Li, Z.M. and Pan, W. (2016) Grey Wolf Optimizer for Unmanned Combat Aerial Vehicle Path Planning. *Advances in Engineering Software*, **99**, 121-136. https://doi.org/10.1016/j.advengsoft.2016.05.015

[3] Zhang, S. and Zhou, Y.Q. (2015) Grey Wolf Optimizer Based on Powell Local Optimization Method for Clustering Analysis. *Discrete Dynamics in Nature and Society*, **2015**, Article ID: 481360. https://doi.org/10.1155/2015/481360

[4] Emary, E., Zawbaa, H.M., Grosan, C. and Hassenian, A.E. (2015) Feature Subset Selection Approach by Gray-Wolf Optimization. In: Abraham, A., Krömer, P. and Snasel, V., Eds., *Afro-European Conference for Industrial Advancement. Advances in Intelligent Systems and Computing*, Vol. 334, Springer, Cham, 1-13. https://doi.org/10.1007/978-3-319-13572-4_1

[5] El-Gaafary, A.A.M., Mohamed, Y.S., Hemeida, A.A. and Mohamed, A.A. (2015) Grey Wolf Optimization for Multi Input Multi Output System. *Universal Journal of Communications and Networks*, **3**, 1-6. https://doi.org/10.13189/ujcn.2015.030101

[6] 张晓凤, 王秀英. 灰狼优化算法研究综述[J]. 计算机科学, 2019, 46(3): 30-38.

[7] Trojovský, P. and Dehghani, M. (2022) Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. *Sensors*, **22**, Article No. 855. https://doi.org/10.3390/s22030855

[8] 聂启颖, 朱振才, 张永合, 王亚敏. 面向深空探测图像分割的群智能混合优化算法[J]. 激光与光电子学进展, 2021, 58(2): 55-62.

[9] 宋宣毅, 刘月田, 马晶, 王俊强, 孔祥明, 任兴南. 基于灰狼算法优化的支持向量机产能预测[J]. 岩性油气藏, 2020, 32(2): 134-140.

[10] Kannan, B.K. and Kramer, S.N. (1994) An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. *Journal of Mechanical Design*, **116**, 405-411. https://doi.org/10.1115/1.2919393

[11] Gandomi, A.H., Yang, X.-S. and Alavi, A.H. (2013) Erratum to: Cuckoo Search Algorithm: A Metaheuristic Approach to Solve Structural Optimization Problems. *Engineering with Computers*, **29**, 245. https://doi.org/10.1007/s00366-012-0308-4

[12] Runarsson, T.P. and Yao, X. (2000) Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, **4**, 284-294. https://doi.org/10.1109/4235.873238

[13] 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

[14] Mezura-Montes, E., Coello, C. and Landa-Becerra, R. (2003) Engineering Optimization Using a Simple Evolutionary Algorithm. Proceedings. 15*th IEEE International Conference on Tools with Artificial Intelligence*, Sacramento, 5 November 2003, 149-156. https://doi.org/10.1109/TAI.2003.1250183

[15] Mirjalili, S. and Lewis, A. (2016) The Whale Optimization Algorithm. *Advances in Engineering Software*, **95**, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[16] 黄辉先, 胡鹏飞. 基于共轭梯度法的反馈差分进化混合算法及其在弹簧设计中的应用[J]. 计算机工程与科学, 2018, 40(7): 1316-1322.

[17] Ray, T. and Saini, P. (2001) Engineering Design Optimization Using a Swarm with an Intelligent Information Sharing among Individuals. *Engineering Optimization*, **33**, 735-748. https://doi.org/10.1080/03052150108940941

[18] Abualigah, L., Diabat, A., Mirjalili, S., Elaziz, M.A. 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

[19] Migallón, H., Jimeno-Morenilla, A., Rico, H., Sánchez-Romero, J.L. and Belazi, A. (2021) Multi-Level Parallel Chao-

tic Jaya Optimization Algorithms for Solving Constrained Engineering Design Problems. *The Journal of Supercomputing*, **77**, 12280-12319. https://doi.org/10.1007/s11227-021-03737-0

[20] Gupta, S. and Deep, K. (2020) A Memory-Based Grey Wolf Optimizer for Global Optimization Tasks. *Applied Soft Computing*, **93**, Article ID: 106367. https://doi.org/10.1016/j.asoc.2020.106367

[21] Gupta, S. and Deep, K. (2019) A Hybrid Self-Adaptive Sine Cosine Algorithm with Opposition Based Learning. *Expert Systems with Applications*, **119**, 210-230. https://doi.org/10.1016/j.eswa.2018.10.050

[22] 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

[23] Kamboj, V.K., Nandi, A., Bhadoria, A. and Sehgal, S. (2020) An Intensify Harris Hawks Optimizer for Numerical and Engineering Optimization Problems. *Applied Soft Computing*, **89**, Article ID: 106018. https://doi.org/10.1016/j.asoc.2019.106018

[24] Zamani, H., Nadimi-Shahraki, M.H. and Gandomi, A.H. (2019) CCSA: Conscious Neighborhood-Based Crow Search Algorithm for Solving Global Optimization Problems. *Applied Soft Computing*, **85**, Article ID: 105583. https://doi.org/10.1016/j.asoc.2019.105583

[25] Glass, M. and Mitsos, A. (2018) Parameter Estimation in Reactive Systems Subject to Sufficient Criteria for Thermodynamic Stability. *Chemical Engineering Science*, **197**, 420-431. https://doi.org/10.1016/j.ces.2018.08.035

[26] Pelusi, D., Mascella, R., Tallini, L., Nayak, J., Naik, B. and Deng, Y. (2020) An Improved Moth-Flame Optimization Algorithm with Hybrid Search Phase. *Knowledge-Based Systems*, **191**, Article ID: 105277. https://doi.org/10.1016/j.knosys.2019.105277