文章编号:2096-1472(2024)04-0022-05
DOI:10.19644/j.cnki.issn2096-1472.2024.004.005

一种基于算术运算和透镜成像学习策略的改进灰狼优化算法

王 恒,杨 婷,郭俊亮

(铜仁职业技术学院信息工程学院,贵州 铜仁 554300)

✉ wangheng_trzy@foxmail.com; 731922226@qq.com; 286477281@qq.com

摘 要:针对基本灰狼优化算法收敛速度慢,易陷入局部搜索的情况,提出一种基于算术运算和透镜成像学习策略的改进灰狼优化算法。该算法在基本灰狼优化算法的基础上,引入算术优化算法的乘除算子,利用带透镜成像的反向学习策略增强最优个体的多样性,增强算法的全局探索能力,提高收敛速度。对比实验结果表明,改进的灰狼优化算法具有收敛速度快、易跳出局部寻优状态,在30个基准测试函数的求解中获得了28个测试函数的最优均值,并且求解质量及普适性均优于最新的几种对比算法。

关键词:灰狼优化算法;算术优化算法;透镜成像的反向学习策略

中图分类号:TP391 **文献标志码**:A

An Improved Grey Wolf Optimizer Based on Arithmetic Operations and Lens Imaging Learning Strategy

WANG Heng, YANG Ting, GUO Junliang

(*School of Information Engineering, Tongren Polytechnic College, Tongren 554300, China*)

✉ wangheng_trzy@foxmail.com; 731922226@qq.com; 286477281@qq.com

Abstract: Aiming at the problems that the basic Grey Wolf Optimizer (GWO) converges slowly and is easy to fall into local search, this paper proposes an improved version ALGWO that incorporates arithmetic operations and lens imaging learning strategy. Based on the fundamental principles of GWO, this improved algorithm introduces multiplication and division operators from the Arithmetic Optimization Algorithm (AOA) and utilizes a lens imaging-based opposition learning strategy to enhance the diversity of the optimal individuals. These modifications aim to bolster the algorithm's global exploration capabilities and improve its convergence speed. Comparative experimental results demonstrate that the improved GWO exhibits faster convergence, a stronger ability to escape local optima, and achieves optimal mean values in 28 out of 30 benchmark test functions. Furthermore, it outperforms several state-of-the-art algorithms in terms of solution quality and generality.

Key words: Grey Wolf Optimizer (GWO); Arithmetic Optimization Algorithm (AOA); lens imaging-based opposition learning strategy

0 引言(Introduction)

灰狼优化(Grey Wolf Optimizer,GWO)算法是由学者MIRJALILI[1]于2014年提出的一种新的元启发式算法。由于GWO算法原理简单、编程容易、需要调整的参数少,因此已成功应用于电力系统[2]、自动控制[3]、图像处理[4]、能源市场战略招标[5]及线路规划[6]等领域。然而,与许多元启发式优化算法一样,

收稿日期:2023-08-10
基金项目:铜仁市科学技术局基础科学研究项目[铜市科研(2022)72号]

GWO算法在求解复杂的非线性问题时,容易陷入局部最优且收敛速度慢。

为了解决GWO算法在求解复杂的非线性问题时存在的不足,本文提出了一种基于透镜成像的反向学习策略和乘除算子策略的经改进的灰狼优化算法。其中,引入乘除算子策略,提高了算法的收敛速度,增强了算法的全局探索能力;引入透镜成像的反向学习策略,提高了算法跳出局部最优的能力,提高了最优个体的多样性。对比仿真结果表明,改进的GWO算法具有收敛速度快和易跳出局部寻优状态的优点,并且求解质量和普适性较几种最新算法均表现出一定的优势,获得了较好的计算结果。

1　改进的灰狼优化算法(Improved Grey Wolf optimization algorithm)

1.1　灰狼优化算法

GWO算法是模仿自然界灰狼群体社会等级和捕食行为[1]而衍生的一种元启发式算法。灰狼群体的社会等级为α狼、β狼、δ狼和ω狼。狼的狩猎行为分为跟踪、包围和攻击猎物3个步骤。狼群包围猎物的数学模型定义如下:

$$\boldsymbol{X}=\boldsymbol{X}_{\alpha}(t)-\boldsymbol{A}\cdot|\boldsymbol{C}\cdot\boldsymbol{X}_{\alpha}(t)-\boldsymbol{X}(t)| \tag{1}$$

其中:$\boldsymbol{X}$和$\boldsymbol{X}_{\alpha}$分别是狼个体和猎物个体的位置向量,t是当前的迭代次数。系数向量$\boldsymbol{A}$和$\boldsymbol{C}$定义如下:

$$\boldsymbol{A}=2a\cdot\boldsymbol{r}_1-a \tag{2}$$

$$\boldsymbol{C}=2\cdot\boldsymbol{r}_2 \tag{3}$$

其中:$\boldsymbol{r}_1$和$\boldsymbol{r}_2$是[0,1]之间的随机向量,a从2线性递减到0,其中$T_{\max}$为最大迭代次数。

其数学模型定义为

$$a=2-\frac{2\cdot t}{T_{\max}} \tag{4}$$

包围猎物后,β狼和δ狼在α狼的带领下追捕猎物。在追捕过程中,狼群的个体位置会随着猎物的逃跑而发生变化。因此,灰狼群可以根据α狼、β狼、δ狼的位置$\boldsymbol{X}_{\alpha}$、$\boldsymbol{X}_{\beta}$、$\boldsymbol{X}_{\delta}$更新灰狼的位置:

$$\boldsymbol{X}_1=\boldsymbol{X}_{\alpha}(t)-\boldsymbol{A}_1\cdot|\boldsymbol{C}_1\cdot\boldsymbol{X}_{\alpha}(t)-\boldsymbol{X}(t)| \tag{5}$$

$$\boldsymbol{X}_2=\boldsymbol{X}_{\beta}(t)-\boldsymbol{A}_2\cdot|\boldsymbol{C}_2\cdot\boldsymbol{X}_{\beta}(t)-\boldsymbol{X}(\mathrm{t})| \tag{6}$$

$$\boldsymbol{X}_3=\boldsymbol{X}_{\delta}(t)-\boldsymbol{A}_3\cdot|\boldsymbol{C}_3\cdot\boldsymbol{X}_{\delta}(t)-\boldsymbol{X}(t)| \tag{7}$$

$$\boldsymbol{X}(t+1)=\frac{\boldsymbol{X}_1+\boldsymbol{X}_2+\boldsymbol{X}_3}{3} \tag{8}$$

其中,$\boldsymbol{X}(t+1)$是当前个体的位置。

1.2　改进GWO算法的思路和策略

1.2.1　算术乘除运算符策略

算术优化算法(Arithmetic Optimization Algorithm, AOA)是学者ABUALIGAH[7]于2021年提出的一种新的元启发式算法,主要利用了数学中的乘、除、加、减4种混合运算。AOA中的乘除算子具有较强的全局探索能力。灰狼种群在更新位置时侧重使用α狼、β狼和δ狼作为精英引导搜索,具有较强的局部开发能力。为此,引入算术乘除算子策略提高GWO算法的全局探索能力。算术乘除算子策略的数学模型定义为

$$\boldsymbol{X}_i^j(t+1)=\begin{cases}\boldsymbol{X}_{\text{best}}^j\div(MOP+\varepsilon)\cdot[(ub_j-lb_j)\cdot\mu+lb_j], & \boldsymbol{r}_3\leqslant 0.5\\ \boldsymbol{X}_{\text{best}}^j\times MOP\cdot[(ub_j-lb_j)\cdot\mu+lb_j], & \boldsymbol{r}_3>0.5\end{cases} \tag{9}$$

其中:$\boldsymbol{X}_{\text{best}}^j$表示当前最优解的第$j$个位置;$\boldsymbol{r}_3$是介于[0,1]的随机向量;$\varepsilon$是一个防止分母为0的整数;$\mu$是调节搜索过程的控制参数,$\mu$的值在基本AOA中为0.5;$ub_j$和$lb_j$分别表示第$j$个位置的上界和下界。$MOP$为概率函数,其数学模型描述如下:

$$MOP=1-\frac{t^{\frac{1}{\tau}}}{T_{\max}^{\frac{1}{\tau}}} \tag{10}$$

其中,$\tau=5$是一个敏感因子,定义了迭代的搜索精度。由公式(10)可知,AOA可以带来高分布,借助乘除算子实现位置更新,可以大大提高算法的全局探索能力。本文设置的阈值为0.3。

1.2.2　基于透镜成像的反向学习策略

根据灰狼的位置更新公式(5)至公式(8),由α狼、β狼和δ狼带领群体中的其他狼进行位置更新。若α狼、β狼和δ狼都处于局部最优,则整个群体会聚集在局部最优区域内,导致种群陷入局部最优的情况。针对这一情况,提出了一种基于透镜成像的反向学习策略,该方法将对立个体与当前最优个体相结合,生成新个体,从而降低了种群陷入局部最优的概率。

假设一维空间中,在轴区间[lb, ub]有一个高度为h的个体P,其在X轴上的投影为x(x为全局最优个体)。将焦距为F的镜头放置在基点位置O上[取基点位置为$(lb+ub)/2$]。个体P通过透镜获得高度为h的倒置图像P^*。在这一点上,第一个倒置的个体x^*通过透镜成像在X轴上产生。基于透镜成像的反向学习策略原理如图1所示。

在图1中,全局最优个体x以O为基点,找到其对应的逆个体x^*。因此,可以根据透镜成像原理推导对应的数学模型。

$$\frac{(ub+lb)/2-x}{x^*-(ub+lb)/2}=\frac{h}{h^*} \tag{11}$$

设$h/h^*=k$,其中k表示拉伸因子。然后通过推导公式(10),可以得到反转点x^*的计算公式。

$$x^*=\frac{ub+lb}{2}+\frac{ub+lb}{2k}-\frac{x}{k} \tag{12}$$

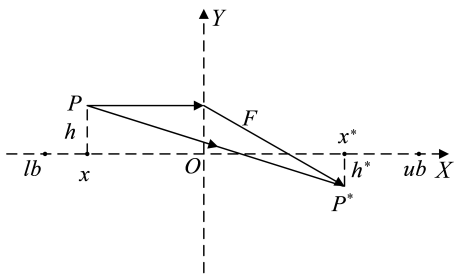

图 1 基于透镜成像的反向学习策略原理

Fig. 1 The principle of opposition learning strategy based on lens imaging

在算法搜索解时，使用拉伸因子 k 作为微观调节因子，增强算法的局部开发能力。然而，在基本的透镜成像的反向学习策略中，拉伸因子一般作为固定值使用，不允许算法探索解空间的全范围。为此，提出一种基于非线性动态递减的伸缩因子策略。算法在迭代初期就可以得到较大的值，这有助于算法在不同维度的区域进行更大范围的搜索，提高种群的多样性。非线性动态拉伸因子定义如下：

$$k=k_{\max}-(k_{\max}-k_{\min})\cdot\left[1-\cos\left(\frac{\pi t}{2T_{\max}}\right)\right] \tag{13}$$

其中：$k_{\max}$ 和 $k_{\min}$ 分别是最大拉伸因子和最小拉伸因子，$T_{\max}$ 是最大迭代次数。可以将公式(12)扩展到 D-维搜索空间，得到数学模型如下：

$$x_j^*=\frac{ub_j+lb_j}{2}+\frac{ub_j+lb_j}{2k}-\frac{x_j}{k} \tag{14}$$

其中：x_j 和 x_j^* 分别是 x 和 x^* 的第 j 维分量，ub_j 和 lb_j 分别是决策变量的第 j 维分量。基于透镜成像的反向学习策略虽然极大地提高了算法的求解精度，但是无法直接判断生成的新反向个体是否优于原始个体，因此引入贪心机制比较新旧个体的适应度值，从而筛选出最优个体。该方法可以不断获得更好的解，提高了算法的寻优能力。贪婪机制的数学模型描述如下：

$$x_{\text{new}}(t)=\begin{cases}x^*, f(x)>f(x^*)\\ x, f(x)\leqslant f(x^*)\end{cases} \tag{15}$$

2 算法描述(Algorithm description)

改进的灰狼优化(ALGWO)算法由算术优化算法的乘除算子和利用基于透镜成像的反向学习策略构成，其中反向学习模块被用于提高最优个体的多样性，增强算法的全局探索能力。ALGWO 算法实现流程图如图 2 所示。

结合图 2，可将 ALGWO 算法描述如下。

步骤 1：随机初始化狼群所有个体的位置信息、最大迭代数和维度。

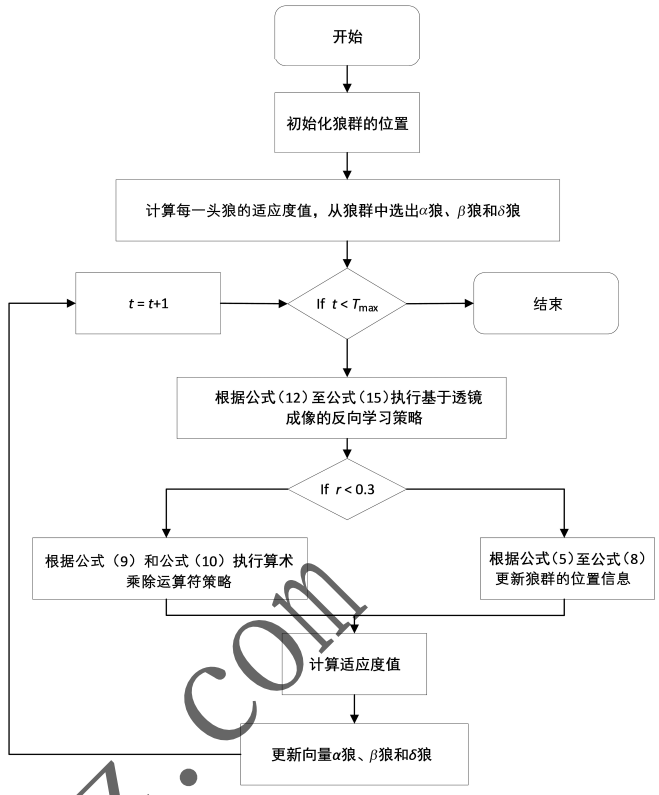

图 2 ALGWO 算法实现流程图

Fig. 2 Flow chart of the ALGWO algorithm

步骤 2：计算每一头狼的适应度值，并对其进行排序，将适应度值排在前 3 的个体分别设定为 α 狼、β 狼、δ 狼，并保存当前最优的位置信息。

步骤 3：执行基于透镜成像的反向学习策略和算术乘除运算符策略依次对种群中每个个体的位置信息进行更新。

步骤 4：针对每个个体更新后的位置信息，重新进行适应度值的计算，根据新的适应度值的大小更新 α 狼、β 狼、δ 狼的位置信息以及历史最优的位置信息，更新狼群的位置信息。

步骤 5：根据迭代的次数重复“步骤 3”至“步骤 5”，当达到最大迭代次数时，停止迭代过程，输出历史最优的位置信息，该位置信息为算法优化后获得的最优解。

3 仿真实验分析(Simulation experiment analysis)

3.1 实验环境及参数设置

所提算法在 Intel (R) Core (TM) i5-9400F CPU、2.50 GHz 频率、16 GB 内存和 Windows 10 (64 bit)操作系统上进行仿真实验。编程软件为 Matlab R2018a。表 1 展示了 CEC2014 标准测试集中 30 个基准测试函数的基本信息，CEC2014 标准测试集共有 30 个单目标测试函数，每个测试函数可选择的维度分别为 10 维、30 维、50 维、100 维。该标准测试集也是应用最广泛的测试集之一，表 2 展示了算法的参数设置。

表1　CEC2014 标准测试集中 30 个基准测试函数

Tab.1 30 benchmark test functions in the CEC2014 standard test sets

测试函数编号	函类型数	函数名称	最优值
CEC01	单峰函数	Rotated High Conditioned Elliptic Function	100
CEC02		Rotated Bent Cigar Function	200
CEC03		Rotated Discus Function	300
CEC04	多峰函数	Shifted and Rotated Rosenbrock′s Function	400
CEC05		Shifted and Rotated Ackley′s Function	500
CEC06		Shifted and Rotated Weierstrass Function	600
CEC07		Shifted and Rotated Griewank′s Function	700
CEC08		Shifted Rastrigin′s Function	800
CEC09		Shifted and Rotated Rastrigin′s Function	900
CEC10		Shifted Schwefel′s Function	1 000
CEC11		Shifted and Rotated Schwefel′s Function	1 100
CEC12		Shifted and Rotated Katsuura Function	1 200
CEC13		Shifted and Rotated HappyCat Function	1 300
CEC14		Shifted and Rotated HGBat Function	1 400
CEC15		Shifted and Rotated Expanded Griewank′s plus Rosenbrock′s Function	1 500
CEC16		Shifted and Rotated Expanded Scaffer′s F6 Function	1 600
CEC17	混合函数	Hybrid Function 1 (N = 3)	1 700
CEC18		Hybrid Function 2 (N = 3)	1 800
CEC19		Hybrid Function 3 (N = 4)	1 900
CEC20		Hybrid Function 4 (N = 4)	2 000
CEC21		Hybrid Function 5 (N = 5)	2 100
CEC22		Hybrid Function 6 (N = 5)	2 200
CEC23	复合函数	Composition Function 1 (N = 5)	2 300
CEC24		Composition Function 2 (N = 3)	2 400
CEC25		Composition Function 3 (N = 3)	2 500
CEC26		Composition Function 4 (N = 5)	2 600
CEC27		Composition Function 5 (N = 5)	2 700
CEC28		Composition Function 6 (N = 5)	2 800
CEC29		Composition Function 7 (N = 3)	2 900
CEC30		Composition Function 8 (N = 3)	3 000

表2　算法的参数设置

Tab.2 Algorithm parameter settings

算法	参数设置
正余弦优化算法(SCA)[8]	M=2
黑猩猩优化算法(ChOA)[9]	f_{max}=2.5, f_{min}=0
鲸鱼优化算法(WOA)[10]	a_{max}=2, a_{min}=0, b=1
灰狼优化算法(GWO)[1]	a_{max}=2, a_{min}=0
改进的灰狼优化算法(ALGWO)	a_{max}=2, a_{min}=0

3.2　算法性能对比分析

为进一步验证改进的灰狼优化算法计算复杂特征问题的有效性和稳定性,使用 30 个具有复杂特征的 CEC2014 基准测试函数进行优化求解。函数类型包括单峰函数、多峰函数、混合函数和复合函数。将改进的灰狼优化算法与灰狼优化算法、正余弦优化算法、黑猩猩优化算法、鲸鱼优化算法 4 种算法进行比较。为保证实验的公平性,将空间维度设置为 30,最大迭代次数为 1 000 次。每个算法独立运行 30 次,CEC2014 基准测试函数的优化结果比较如表 3 所示。

表3　CEC2014 基准测试函数的优化结果比较

Tab.3 Comparison of optimization results for CEC2014 benchmark functions

测试函数编号	改进的灰狼优化算法		灰狼优化算法		正余弦优化算法		黑猩猩优化算法		鲸鱼优化算法	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
CEC01	**1.91×10^7**	**4.58×10^7**	815×10^7	5.86×10^7	4.19×10^8	1.09×10^8	5.99×10^8	1.08×10^8	8.55×10^7	5.42×10^7
CEC02	9.94×10^9	7.24×10^9	**5.62×10^9**	**1.38×10^8**	2.65×10^10	4.30×10^9	4.38×10^10	6.87×10^9	3.424×10^9	3.13×10^9
CEC03	**2.84×10^4**	**2.29×10^2**	5.62×10^4	1.10×10^3	5.88×10^4	1.19×10^4	8.26×10^4	8.02×10^3	4.58×10^4	1.05×10^4
CEC04	**5.85×10^2**	**4.21×10^1**	6.67×10^2	2.63×10^2	2.54×10^3	7.60×10^2	3.33×10^3	1.43×10^3	6.61×10^2	5.31×10^1
CEC05	**5.20×10^2**	**2.46×10^-2**	5.21×10^2	5.38×10^-2	5.21×10^2	5.83×10^-2	5.21×10^2	5.33×10^-2	5.21×10^2	5.33×10^-2
CEC06	**6.09×10^2**	**3.44×10^-1**	6.15×10^2	7.12×10^-1	6.37×10^2	2.71	6.36×10^2	2.02	6.15×10^2	2.64
CEC07	**7.04×10^2**	**1.97×10^1**	7.39×10^2	2.17×10^1	9.17×10^2	3.11×10^1	1.17×10^3	8.54×10^1	7.24×10^2	2.95×10^1
CEC08	**8.55×10^2**	**1.14×10^1**	8.94×10^2	3.07×10^1	1.07×10^3	2.54×10^1	1.07×10^3	2.21×10^1	8.89×10^2	2.09×10^1
CEC09	**9.95×10^2**	**3.45×10^1**	1.01×10^3	4.02×10^1	1.21×10^3	6.35×10^1	1.19×10^3	5.15×10^1	1.02×10^3	4.81×10^1
CEC10	**2.88×10^3**	**4.65×10^2**	4.06×10^3	6.92×10^2	7.67×10^3	5.63×10^2	7.80×10^3	9.38×10^2	3.56×10^3	5.14×10^2
CEC11	**3.97×10^3**	3.28×10^2	4.32×10^3	4.24×10^2	8.68×10^3	3.15×10^2	8.93×10^3	**2.90×10^2**	4.48×10^3	1.21×10^3
CEC12	**1.20×10^3**	2.06	1.20×10^3	**1.89**	1.20×10^3	3.93×10^-1	1.20×10^3	4.05×10^-1	1.20×10^3	5.10
CEC13	**1.30×10^3**	**6.64×10^-2**	1.30×10^3	5.72×10^-1	1.30×10^3	2.69×10^-1	1.30×10^3	5.68×10^-1	1.30×10^3	5.59×10^-1
CEC14	**1.40×10^3**	**3.72**	1.41×10^3	7.91	1.47×10^3	1.12×10^1	1.56×10^3	3.48×10^1	1.41×10^3	9.24
CEC15	**1.59×10^3**	**8.96×10^1**	1.81×10^3	6.92×10^2	1.85×10^4	1.14×10^4	1.26×10^5	9.52×10^4	1.80×10^3	7.87×10^2
CEC16	**1.61×10^3**	**3.90×10^-1**	1.61×10^3	4.55×10^1	1.61×10^3	5.37×10^-1	1.61×10^3	2.24×10^-1	1.61×10^3	7.96×10^-1
CEC17	**2.62×10^6**	**2.05×10^6**	4.25×10^6	2.38×10^6	1.38×10^7	7.71×10^6	3.29×10^7	1.94×10^7	2.72×10^6	2.54×10^6
CEC18	**6.84×10^3**	**5.53×10^3**	3.55×10^4	5.11×10^4	3.13×10^8	1.33×10^8	9.12×10^8	9.99×10^8	1.69×10^7	2.72×10^7
CEC19	**1.93×10^3**	**2.35×10^1**	1.95×10^3	2.88×10^1	2.03×10^3	3.65×10^1	2.16×10^3	3.07×10^2	1.97×10^3	4.45×10^1
CEC20	**1.54×10^4**	**2.04×10^4**	2.45×10^4	2.24×10^4	4.45×10^4	2.32×10^4	1.08×10^5	3.92×10^4	2.48×10^4	6.62×10^4
CEC21	**3.65×10^4**	**6.30×10^4**	4.30×10^5	4.77×10^5	4.06×10^6	2.34×10^6	1.13×10^7	3.24×10^6	1.33×10^6	2.12×10^6
CEC22	**2.50×10^3**	**1.86×10^2**	2.61×10^3	2.15×10^2	3.26×10^3	2.16×10^2	3.18×10^3	2.64×10^2	2.61×10^3	1.76×10^2
CEC23	**2.50×10^3**	**0**	2.63×10^3	1.26×10^1	2.71×10^3	1.95×10^1	2.74×10^3	4.44×10^1	2.64×10^3	1.04×10^1
CEC24	**2.60×10^3**	**0**	2.60×10^3	4.96×10^-3	2.61×10^3	1.95×10^1	2.60×10^3	4.63×10^-2	2.60×10^3	1.09×10^-2

续表

测试函数编号	改进的灰狼优化算法		灰狼优化算法		正余弦优化算法		黑猩猩优化算法		鲸鱼优化算法	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
CEC25	$\mathbf{2.70\times10^{3}}$	**0**	2.71×10^{3}	2.36	2.74×10^{3}	1.19×10^{1}	2.71×10^{3}	1.37×10^{1}	2.71×10^{3}	5.36
CEC26	2.78×10^{3}	4.19×10^{1}	$\mathbf{2.71\times10^{3}}$	$\mathbf{3.14\times10^{-1}}$	2.70×10^{3}	5.08×10^{-1}	2.80×10^{3}	4.84×10^{1}	2.74×10^{3}	4.96×10^{1}
CEC27	$\mathbf{3.28\times10^{3}}$	$\mathbf{1.13\times10^{2}}$	3.47×10^{3}	2.20×10^{2}	3.86×10^{3}	2.73×10^{2}	3.92×10^{3}	2.00×10^{2}	3.38×10^{3}	1.50×10^{2}
CEC28	$\mathbf{3.72\times10^{3}}$	$\mathbf{4.01\times10^{1}}$	4.46×10^{3}	3.54×10^{2}	5.59×10^{3}	4.55×10^{2}	5.75×10^{3}	2.48×10^{2}	4.20×10^{3}	3.74×10^{2}
CEC29	$\mathbf{4.93\times10^{3}}$	$\mathbf{2.59\times10^{3}}$	2.82×10^{6}	2.50×10^{6}	3.02×10^{7}	1.49×10^{7}	5.15×10^{7}	3.31×10^{7}	1.15×10^{6}	1.94×10^{6}
CEC30	$\mathbf{6.72\times10^{4}}$	$\mathbf{2.14\times10^{4}}$	8.44×10^{4}	3.07×10^{4}	4.65×10^{5}	1.77×10^{5}	8.10×10^{5}	2.14×10^{5}	8.63×10^{4}	5.85×10^{4}

表3中加粗显示的数值是表中每一行中最小的值，表明相关方法在对应索引的情况下表现良好。从优化质量和搜索稳定性两个方面比较了灰狼优化算法和改进的灰狼优化算法的性能特点，特别是在CEC06、CEC07、CEC12、CEC13和CEC14基准测试函数上，ALGWO算法基本收敛到理论最优值。在CEC02和CEC26基准测试函数上，ALGWO的表现略逊于灰狼优化算法。由于融合两种策略需要较多的计算量，导致收敛精度降低。由表3可知，相比灰狼优化算法、正余弦优化算法、黑猩猩优化算法及鲸鱼优化算法，改进的灰狼优化算法依次获得28、28、28、28个最小均值，改进的灰狼优化算法与以上4种算法相比占有绝对优势，其求解质量优势也较为明显。在CEC2014标准测试集中的验证结果表明，改进的灰狼优化算法在求解质量以及普适性方面占有一定的优势。

4 结论(Conclusion)

本文提出了一种改进的灰狼优化算法，该算法针对原始灰狼优化算法基于算术运算和透镜成像学习策略，将修正反向学习策略与透镜成像学习策略和乘除算子策略相结合，在增强最优个体的多样性的同时，提高了算法的全局探索能力，并提高了收敛速度。在CEC2014标准测试集中30个基准测试函数上进行测试，并与其他元启发式算法进行性能比较分析。实验结果表明，改进的灰狼优化算法比其他算法具有更好的稳定性和寻优性能。未来，将使用改进的灰狼优化算法解决更多复杂的实际应用问题。

参考文献(References)

[1] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in engineering software, 2014, 69: 46-61.

[2] NUAEKAEW K, ARTRIT P, PHOLDEE N, et al. Optimal reactive power dispatch problem using a two-archive multi-objective grey wolf optimizer[J]. Expert systems with applications, 2017, 87(C): 79-89.

[3] PRECUP R E, DAVID R C, PETRIU E M. Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity[J]. IEEE transactions on industrial electronics, 2017, 64(1): 527-534.

[4] MARTIN B, MAROT J, BOURENNANE S. Mixed grey wolf optimizer for the joint denoising and unmixing of multispectral images[J]. Applied soft computing, 2019, 74: 385-410.

[5] SAXENA A, KUMAR R, DAS S. β-Chaotic map enabled Grey Wolf Optimizer[J]. Applied soft computing, 2019, 75: 84-105.

[6] 姚鹏，王宏伦. 基于改进流体扰动算法与灰狼优化的无人机三维航路规划[J]. 控制与决策，2016，31(4)：701-708.

[7] ABUALIGAH L, DIABAT A, MIRJALILI S, et al. The arithmetic optimization algorithm[J]. Computer methods in applied mechanics and engineering, 2021, 376: 113609.

[8] MIRJALILI S. SCA: a Sine Cosine Algorithm for solving optimization problems[J]. Knowledge-based systems, 2016, 96: 120-133.

[9] KHISHE M, MOSAVI M R. Chimp optimization algorithm [J]. Expert systems with applications, 2020, 149: 113338.

[10] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in engineering software, 2016, 95(C): 51-67.

作者简介：

王 恒(1985-)，男，硕士，讲师。研究领域：智能计算与混合系统，人工智能。本文通信作者。

杨 婷(1995-)，女，本科，讲师。研究领域：人工智能，深度学习。

郭俊亮(1987-)，男，硕士，副教授。研究领域：机器学习与人工智能，深度学习。