

多项式变异和自适应权重优化的阿奎拉鹰算法

李汶娟, 李 广, 聂志刚

(甘肃农业大学 信息科学技术学院, 甘肃 兰州 730070)

摘要:针对基本阿奎拉鹰算法存在收敛精度低、易陷入局部最优的问题,通过在全局搜索阶段引入多项式变异扰动策略,在局部开发阶段引入自适应权重优化策略,改进了阿奎拉鹰算法的局部探索能力,并且引入了 Tent 混沌映射初始化种群,增加种群多样性,引入动态转换概率策略来平衡全局探索和局部开发的比重,故提出多项式变异和自适应权重优化的阿奎拉鹰算法。采用基本阿奎拉鹰算法、哈里斯鹰算法、灰狼算法、鲸鱼算法、海鸥算法做对比,9个基准测试函数和2个工程优化问题对改进后的算法进行寻优性能验证,结果表明:改进后的算法在多数测试函数上取得较好的寻优效果,在工程优化问题中,效果优于多数对比算法。证明了改进后的算法具有更快的收敛速度和精度,并在工程应用中取得较好效果。

关键词:Tent 混沌映射;动态转换概率策略;多项式变异扰动策略;自适应权重;阿奎拉鹰算法

中图分类号:TP301.6

文献标识码:A

文章编号:1673-629X(2024)02-0163-08

doi:10.3969/j.issn.1673-629X.2024.02.024

Polynomial Variance and Adaptive Weight Optimization for Aquila Algorithm

LI Wen-juan, LI Guang, NIE Zhi-gang

(School of Information Science and Technology, Gansu Agricultural University, Lanzhou 730070, China)

Abstract:To address the problem that the basic Aquila algorithm has low convergence accuracy and is prone to fall into local optimum, by introducing a polynomial variance perturbation strategy in the global search phase and an adaptive weight optimization strategy in the local exploitation phase, the local exploration ability of Aquila is improved. A Tent chaos mapping is introduced to initialize the population and increase the population diversity, and a dynamic transformation probability strategy is introduced to balance the weight of global exploration and local exploitation, so the Aquila algorithm with polynomial variance and adaptive weight optimization is proposed. The basic Aquila algorithm, Harris Hawks algorithm, Gray Wolf algorithm, Whale algorithm, and Seagull algorithm are used for comparison, and 9 benchmark test functions and 2 engineering optimization problems are used to verify the improved algorithm's optimization-seeking performance. The results show that the improved algorithm achieves better optimization-seeking results on most of the test functions and outperforms most of the comparison algorithms in engineering optimization problems. It is proved that the improved algorithm has faster convergence speed and accuracy, and achieves good results in engineering applications.

Key words:Tent chaotic mapping; dynamic conversion probability strategy; polynomial variance perturbation strategy; adaptive weight; Aquila optimizer

0 引言

阿奎拉鹰算法 (Aquila Optimizer, AO) 是由 Abualigah, Yousri 等人^[1]于2021年提出的一种模拟阿奎拉鹰捕猎行为的新的元启发式优化算法。阿奎拉鹰通过在四种狩猎方式间巧妙变换来快速捕获猎物。因为种群的全局搜索随机性强、探索广度大,所以该算法具有较强的全局搜索能力和较快的收敛速度;但该算

法在全局与局部探索的条件固定,探索与开发之间不平衡,导致局部搜索能力较弱,且容易陷入局部最优。针对 AO 的不足,众多学者使用多种策略对该算法进行改进。Akyol 等人^[2]提出了一种新的混合阿奎拉鹰-切线搜索算法 (Aquila Optimizer - Tangent Search Algorithm, AO-TSA),即在全局搜索阶段使用 AO,在局部开发阶段使用 TSA 的局部最小逃逸策略,从而避

收稿日期:2023-04-24

修回日期:2023-08-25

基金项目:国家自然科学基金项目(32160416);甘肃省教育厅产业支撑计划项目(2021CYZC-15,2022CYZC-41)

作者简介:李汶娟(1998-),女,硕士研究生,研究方向为智能计算;通信作者:李 广(1971-),男,教授,博士,研究方向为作物模拟模型与农业信息化。

免陷入局部最优。张玉军等人^[3]提出一种将算术优化算法(Arithmetic Optimization Algorithm, AOA)与阿奎拉鹰优化算法混合的新方法(Arithmetic Optimization Algorithm-Aquila Optimizer, AOAOA),通过改进逃逸能量参数,让更多个体在迭代后期进行全局搜索,并在减速器设计等工程问题中取得不错的效果。贾鹤鸣等人^[4]将阿奎拉鹰算法与哈里斯鹰算法结合提出新的混合算法,新算法的收敛速度和优化精度明显改善。Akyol、张玉军、贾鹤鸣等人都是将 AO 算法与另一种算法混合起来使用,改变了 AO 算法原有的结构,同时可能造成新算法在寻优过程中的时间复杂度增加。通过对 AO 算法求解过程的分析,该文提出了一种多项式变异和自适应权重优化的阿奎拉鹰算法(Polynomial Variance and Adaptive Weight Optimization for Aquila Algorithm, MAO),改进策略均是在 AO 算法原始结构上进行改进优化。通过 Tent 混沌映射^[5]初始化种群,增加种群的多样性;引入动态转换概率策略用于更好地平衡全局搜索和局部开发;在全局搜索期间引入多项式变异扰动策略对全局最优值进行扰动,增加了解的多样性;在局部开发阶段引入自适应权重策略可以动态调整最优值权重,加快算法收敛速度。最后采用 CEC2017 中的 9 个基准测试函数和 2 个工程优化问题对改进后的算法进行寻优性能验证。拟解决 AO 算法在局部搜索方面能力较弱,容易陷入局部最优的问题,为智能优化算法的应用提出新的方向。

1 算法基础理论与改进方案

1.1 阿奎拉鹰优化算法

1.1.1 探索阶段

阿奎拉鹰使用全局飞行探索和轮廓飞行与短滑行攻击策略全局搜索猎物,设置随机数 r_1 使两种策略随机切换,其取值为 $[0,1]$ 。

(1) 全局飞行探索。

阿奎拉鹰通常先高空飞行,便于搜寻猎物位置,其位置更新遵循下述表达:

$$X(t+1) = X_{\text{best}}(t) \times (1 - \frac{t}{T}) + (X_M(t) - X_{\text{best}}(t) \times r_2) \quad (1)$$

$$X_M(t) = \frac{1}{N} \sum_{i=1}^N X_i(t) \quad (2)$$

其中, $X(t+1)$ 是下一次迭代的位置; $X_{\text{best}}(t)$ 是鹰群最优位置,反映了猎物的大致位置; $(1 - \frac{t}{T})$ 表示通过迭代次数来控制探索; $X_M(t)$ 是鹰群平均位置; N 是鹰群个体数; r_2 为 $[0,1]$ 的随机数; T 是算法最大迭代次数; t 是当前迭代次数。

(2) 轮廓飞行与短滑行攻击。

当在高空发现猎物后,阿奎拉鹰会在猎物上空盘旋,准备着陆攻击。

$$X(t+1) = X_{\text{best}}(t) \times \text{LF}(D) + X_R(t) + (y - x) \times r_3 \quad (3)$$

其中, $X_R(t)$ 代表种群的随机位置, D 代表问题的维度, r_3 代表 $[0,1]$ 的随机数, $\text{LF}(D)$ 代表莱维飞行函数,公式如下:

$$\text{LF}(D) = s \times \frac{\mu \times \sigma}{|v|^{1/\beta}} \quad (4)$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{1/\beta} \quad (5)$$

其中, s 为固定常数 0.01, μ 与 v 为 $[0,1]$ 的随机数, β 为固定常数 1.5。

$$y = R \times \cos\theta \quad (6)$$

$$x = R \times \sin\theta \quad (7)$$

$$R = R_1 + U \times D_1 \quad (8)$$

$$\theta = -\omega \times D_1 + \frac{3\pi}{2} \quad (9)$$

其中, y 与 x 表示以螺旋形式搜索的过程, R_1 表示固定的搜索周期,一般在 $[1,20]$ 之间取值。 U 为固定常数 0.005 65, D_1 是 1 到搜索空间维度之间的整数, ω 为固定常数 0.005。

1.1.2 开发阶段

(1) 低空俯冲狩猎。

阿奎拉鹰已完成全局搜索,确定猎物大致位置,开始进行低空狩猎,数学模型如下:

$$X(t+1) = (X_{\text{best}}(t) - X_M(t)) \times \alpha - r_5 + ((\text{UB} - \text{LB}) \times r_6 + \text{LB}) \times \delta \quad (10)$$

其中, UB 为种群搜索上界, LB 为种群搜索下界, α 和 δ 均为值为 0.1 的适应参数, r_5 和 r_6 均为 $[0,1]$ 的随机数。

(2) 地面近距离攻击。

阿奎拉鹰进行低空游走,准备着陆攻击猎物。数学模型如下:

$$X(t+1) = \text{QF} \times X_{\text{best}}(t) - (G_1 \times X(t) \times r_7) - G_2 \times \text{LF}(D) + r_8 \times G_1 \quad (11)$$

$$\text{QF}(t) = \frac{2 \times r_9 - 1}{t(1-t)^2} \quad (12)$$

$$G_1 = 2 \times r_{10} - 1 \quad (13)$$

$$G_2 = 2 \times (1 - \frac{t}{T}) \quad (14)$$

其中, $\text{QF}(t)$ 是用来平衡算法搜索策略的质量函数, G_1 代表猎物逃逸轨迹, G_2 代表鹰群空中的飞行斜率, $r_7 \sim r_{10}$ 均为 $[0,1]$ 的随机数。

1.2 算法改进方案

第一,采用 Tent 混沌映射来初始化种群,增加了种群的多样性。混沌运动可以根据自身遍历所有状态,而不会在一定范围内重复这些状态。所以,采用 Tent 混沌映射策略使种群分布更均匀。

第二,引入动态转换概率策略用于更好地平衡全局搜索和局部开发,动态转换概率的引入打破了原有根据固定迭代次数划分全局探索与局部开发。

第三,在全局搜索期间引入多项式变异扰动策略对全局最优值进行扰动,增加了解的多样性。

最后,在局部开发阶段引入自适应权重策略可以动态调整最优值权重,加快算法收敛速度。

2 阿奎拉鹰算法的改进(MAO)

2.1 基于 Tent 混沌映射的阿奎拉鹰种群初始化

种群的多样性是影响算法效率的关键因素之一。混沌搜索的使用比无序随机搜索更有优势。基本的 AO 算法在搜索空间中随机生成初始化种群,导致种群多样性不足,影响算法寻优效率。改进后的算法采用了 Tent 混沌映射初始化种群,增加了种群多样性。Tent 混沌映射的公式如下:

$$Z_{k+1} = \begin{cases} 2Z_k, & 0 \leq Z_k < 0.5 \\ 2(1 - Z_k), & 0.5 \leq Z_k \leq 1 \end{cases} \quad (15)$$

其中, k 代表映射次数, Z_k 代表第 k 次映射的函数值。

2.2 动态转换概率策略

在基础 AO 中,依赖迭代次数切换全局搜索和局部开发,当 $t \leq \frac{2}{3}T$ 进行全局探索, $t > \frac{2}{3}T$ 进行局部开发,阶段划分条件固定,使探索与开发之间不平衡,导致在算法后期不能快速收敛。因此,该文采用转换概率来随机切换全局探索 and 局部开发,以提高算法在后期开发过程中的收敛速度。其公式如下:

$$P = 0.6 - 0.1 \times \frac{T-t}{T} \quad (16)$$

其中, t 为当前迭代次数, T 为总迭代次数。当 $P > r$ 时(r 为 $[0,1]$ 的随机数),采用全局飞行探索和轮廓飞行与短滑翔攻击策略进行全局搜索,反之,采用低空俯冲狩猎和地面近距离攻击策略进行局部开发。

2.3 多项式变异扰动策略

采用多项式变异算子对全局最优位置进行扰动,扩大搜索广度,增加了种群多样性,有利于算法跳出局部最优,加快收敛速度。其公式如下:

$$c_1 = \frac{X_{\text{best}}(t) - \text{LB}}{\text{UB} - \text{LB}} \quad (17)$$

$$c_2 = \frac{\text{UB} - X_{\text{best}}(t)}{\text{UB} - \text{LB}} \quad (18)$$

$$\lambda =$$

$$\begin{cases} [2 \times \mu + (1 - 2 \times \mu) \times (1 - c_1) \times \sin(2\pi\omega)] \times e^{(1-\frac{1}{T})}, & \text{rand} < 0.5 \\ 1 - [2 \times (1 - \mu) + 2 \times (\mu - 0.5)^2(1 - c_2) \times \cos(2\pi\omega)] \times e^{(1-\frac{1}{T})}, & \text{rand} \geq 0.5 \end{cases} \quad (19)$$

其中, λ 为多项式变异算子,可根据 $[0,1]$ 的随机数 rand 随机切换策略,使变异方式多样, μ, ω 为 $[0,1]$ 的随机数。将公式 3 改为公式 20:

$$X(t+1) = X_{\text{best}}(t) \times \text{LF}(D) \times \lambda + X_R(t) - \text{rand} \times (y - x) \quad (20)$$

2.4 自适应权重策略

惯性权重因子是很重要的一个参数,惯性权重较大时,算法的全局搜索能力较强,惯性权重较小时,局部搜索能力较强^[6]。在基础 AO 的局部开发过程中,采用平衡搜索策略的函数 QF 平衡最优位置在寻优过程中所占比重,引入 ω 后,随着迭代次数的增加, ω 逐步趋近于 1,即 $X_{\text{best}}(t)$ 在局部开发过程中所占比重增大,有利于算法后期快速收敛。

$$\omega = \sin\left(\frac{\pi t}{2T} + \pi\right) + 1 \quad (21)$$

其中, t 为当前迭代次数, T 为总迭代次数,将公式 11 改为公式 22:

$$X(t+1) = \omega \times X_{\text{best}}(t) - (G_1 \times X(t) \times r_7) - G_2 \times \text{LF}(D) + r_8 \times G_1 \quad (22)$$

2.5 MAO 算法流程

步骤 1:初始化参数。设置种群数量规模 N 、搜索维度 D 、最大迭代次数 T 、上界(UB)、下界(LB)、动态转换概率 P 。

步骤 2:采用 Tent 混沌映射对初始化种群重新更新,生成 $N \times D$ 大小的矩阵。

步骤 3:计算每只阿奎拉鹰的自适应度,保存最优位置 X_{best} 。

步骤 4:生成随机数 r_1 ,当 $P > r_1$,进行全局飞行探索和轮廓飞行与短滑翔攻击策略,生成随机数 r_2 ,若 $r_2 < 0.5$,采用全局飞行探索策略,反之,采用轮廓飞行与短滑翔攻击策略,并在该策略中添加多项式变异算子对最优位置进行扰动。

步骤 5:当 $P \leq r_1$,进行低空俯冲狩猎和地面近距离攻击策略,生成随机数 r_3 ,若 $r_3 < 0.5$,采用低空俯冲狩猎策略,反之,采用地面近距离攻击策略,并在该策略中添加自适应权重因子,加快算法的收敛速度。

步骤 6:判断算法是否达到最大迭代次数,若达到,结束循环,输出全局最优解,若未达到,返回步骤 3 继续迭代。

MAO 算法流程如图 1 所示。

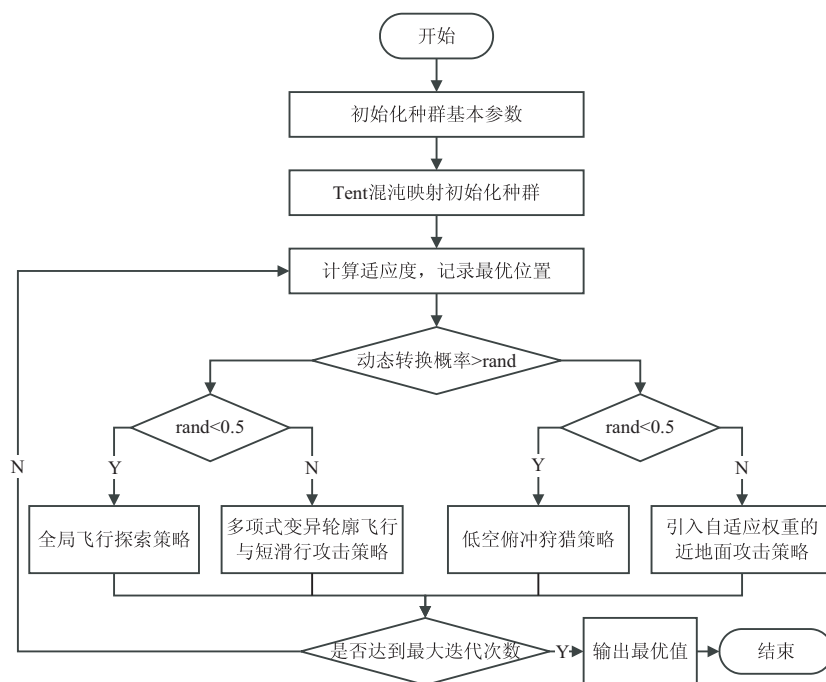


图 1 MAO 算法流程

2.6 MAO 算法时间复杂度分析

设阿奎拉鹰群体规模为 N , 最大迭代次数为 T , 维度为 D , 标准 AO 算法的时间复杂度为 $O(N \cdot (T \cdot D + 1))$ [7], 根据 MAO 算法流程的描述和时间复杂度 O 的运算规则, 初始化计算复杂度为 $O(N)$, 引入多项式变异扰动策略对全局位置进行更新的时间复杂度为 $O(T \cdot D \cdot N)$, 以及引入惯性权重因子对局部位置更新的时间复杂度为 $O(T \cdot D \cdot N)$ 。综上所述, MAO 的时间复杂度为 $O(N \cdot (T \cdot D + 1))$, 与基本 AO 相比没有增加计算量。

3 MAO 算法性能验证实验

3.1 实验设计

实验环境为: 操作系统 Windows10, CPU 为 Intel (R) Core (TM) i7-8550U CPU@1.80 GHz 1.99 GHz, 内存为 4 GB, 仿真实验软件为 Matlab2016a。通过 AO 基本算法及四种较新颖的优化算法 (哈里斯鹰算法 (Harris Hawks Optimization, HHO) [8]、灰狼优化算法 (Grey Wolf Optimization, GWO) [9]、鲸鱼优化算法 (Whale Optimization Algorithm, WOA) [10]、海鸥算法 (Seagull Optimization Algorithm, SOA) [11]) 与改进后的 MAO 算法进行实验对比。为了实验的公平性、客观性, 所有算法的初始种群规模设置为 30, 迭代次数设为 500。选用 CEC2017 中的 9 个经典测试函数 [12] 进行仿真对比实验来分析改进 MAO 算法的有效性。

3.2 MAO 与其他五种算法实验结果分析

表 1 为 MAO 算法与其余五种算法的仿真实验数

据, 在实验中所有算法均固定参数 (维度 $D = 30$, 种群数量 $N = 30$, 最大迭代次数 $T = 500$)。从表 1 中的最小值可清晰看出每种算法的收敛精度, 提出的 MAO 算法在测试函数 $f_1 \sim f_3, f_5, f_6$ 均取得最好的寻优精度 (以上 5 个测试函数最小值分别为 $4.68\text{E}-182, 1.27\text{E}-170, 1.64\text{E}-105, 8.50\text{E}-08, 1.01\text{E}-07$, 其值均小于其余算法的最小值), 在 4 个测试函数 $f_4, f_7 \sim f_9$ 上与其余较优的算法达到相同寻优精度。其中, $f_1 \sim f_3$ 为单峰函数, $f_4 \sim f_6$ 为多峰函数, $f_7 \sim f_9$ 为固定维多峰函数, MAO 算法均能取得最好的寻优精度。综上分析, 改进的 MAO 算法相比于基础的 AO 算法以及 HHO, GWO, WOA, SOA, 可以达到更好的收敛精度与寻优性能。

将表 1 中的仿真实验数据绘制为图 2 中的收敛曲线。从图 2 中可直观看出, 每种算法的收敛性能 (除 3 个测试函数 $f_7 \sim f_9$ 外, 其余函数值均对表 1 中的数据取自然对数)。在 5 个测试函数 $f_1 \sim f_3, f_5, f_6$ 中 MAO 算法收敛性能明显优于基础 AO 算法。通过表 1 与图 2 对比, 在 f_1 上, MAO 与 AO 相比提高了 38 个数量级; 在 f_2 上, MAO 与 AO 相比提高了 21 个数量级; 在 f_3 上, MAO 与 AO 相比提高了 26 个数量级, 并且有更强的局部搜索能力, 由此得出结论, MAO 在单峰函数上表现优秀, 收敛效果提升明显。在 f_5 上, MAO 与 AO 相比提高了 3 个数量级; 在 f_6 上, MAO 与 AO 相比提高了 2 个数量级; 在 f_6 的收敛曲线中, MAO 在迭代过程中出现多个拐点, 证明 MAO 在收敛过程中易跳出局部最优。综上分析, MAO 算法在收敛速度和收敛精度上都明显优于基础 AO 算法。

表1 MAO与五种算法的性能分析

函数	算法	平均最优值	标准差	最大值	最小值
f_1 Sphere	MAO	2.04E+01	4.49E+02	1.00E+04	4.68E-182
	AO	2.81E+01	5.50E+02	1.22E+04	5.50E-144
	HHO	2.29E+01	4.22E+02	9.27E+03	2.71E-102
	GWO	8.65E+01	8.62E+02	1.28E+04	1.71E-57
	WOA	2.00E+02	1.52E+03	1.77E+04	9.71E-82
	SOA	1.12E+04	7.67E+03	1.65E+04	1.63E-29
f_2 Schwefel 1.2	MAO	4.23E+01	9.27E+02	2.07E+04	1.27E-170
	AO	5.82E+01	8.73E+02	1.73E+04	5.58E-149
	HHO	1.12E+02	1.39E+03	2.39E+04	1.43E-106
	GWO	1.50E+02	1.62E+03	2.49E+04	1.22E-25
	WOA	4.53E+03	5.96E+03	2.46E+04	6.63E+01
	SOA	1.21E+04	7.67E+03	1.73E+04	3.46E-18
f_3 Schwefel 2.21	MAO	1.73E-01	3.44E+00	1.58E+00	1.64E-105
	AO	1.60E-01	3.31E+00	1.10E+00	2.70E-79
	HHO	3.56E-01	4.19E+00	7.73E+01	5.11E-51
	GWO	6.55E-01	4.80E+00	6.90E+01	2.79E-19
	WOA	1.86E+01	1.35E+01	5.79E+01	1.17E+01
	SOA	5.29E+01	3.18E+01	7.29E+01	1.06E-11
f_4 Ackley	MAO	6.76E-02	9.68E-01	2.02E+01	8.88E-16
	AO	7.06E-02	1.04E+00	2.03E+01	8.88E-16
	HHO	1.10E-01	1.28E+00	1.81E+01	8.88E-16
	GWO	3.12E-01	1.98E+00	2.01E+01	7.99E-15
	WOA	6.85E-01	2.88E+00	2.05E+01	4.44E-15
	SOA	1.61E+01	7.72E+00	2.01E+01	4.11E-11
f_5 Penalized1	MAO	1.19E+05	2.66E+06	5.95E+07	8.50E-08
	AO	9.15E+04	2.04E+06	4.58E+07	5.81E-05
	HHO	1.27E+05	2.84E+06	6.35E+07	1.33E-05
	GWO	7.59E+04	8.99E+05	1.27E+07	1.10E-06
	WOA	3.07E+04	2.39E+05	2.88E+06	6.83E-03
	SOA	5.09E+07	3.43E+07	2.33E+08	3.94E-02
f_6 Penalized2	MAO	4.66E+05	1.04E+07	1.43E+08	1.01E-07
	AO	2.86E+05	6.39E+06	2.11E+08	1.75E-05
	HHO	4.14E+05	9.24E+06	2.07E+08	2.20E-05
	GWO	7.16E+05	1.19E+07	2.59E+08	7.41E-06
	WOA	1.00E+05	1.29E+06	2.65E+07	3.31E-03
	SOA	7.78E+07	5.83E+07	1.44E+08	3.19E-01
f_7 Six_Hump Camel_Back	MAO	-1.02E+00	8.52E-02	6.56E-01	-1.03E+00
	AO	-9.82E-01	9.30E-02	-2.29E-01	-1.03E+00
	HHO	-1.03E+00	2.19E-02	-6.44E-01	-1.03E+00
	GWO	-1.03E+00	4.56E-02	-1.72E-02	-1.03E+00
	WOA	-1.02E+00	1.49E-01	2.28E+00	-1.03E+00
	SOA	-1.00E+00	2.98E-02	-8.94E-01	-1.03E+00
f_8 Sheke_7	MAO	-1.04E+01	4.59E-01	-3.61E-01	-1.04E+01
	AO	-1.04E+01	5.13E-01	-1.07E+00	-1.04E+01
	HHO	-5.06E+00	2.41E-01	-5.49E-01	-5.09E+00
	GWO	-7.68E+00	2.80E+00	-8.05E-01	-1.04E+01
	WOA	-5.02E+00	3.88E-01	-6.04E-01	-5.09E+00
	SOA	-5.02E-01	3.76E-02	-4.27E-01	-5.22E-01
f_9 Sheke_10	MAO	-1.04E+01	8.06E-01	-7.82E-01	-1.05E+01
	AO	-1.05E+01	5.26E-01	-7.82E-01	-1.05E+01
	HHO	-8.79E+00	1.51E+00	-5.32E-01	-1.00E+01
	GWO	-8.15E+00	2.78E+00	-5.83E-01	-1.05E+01
	WOA	-8.99E+00	2.82E+00	-7.82E-01	-1.05E+01
	SOA	-1.76E+00	1.55E+00	-6.17E-01	-5.13E+00

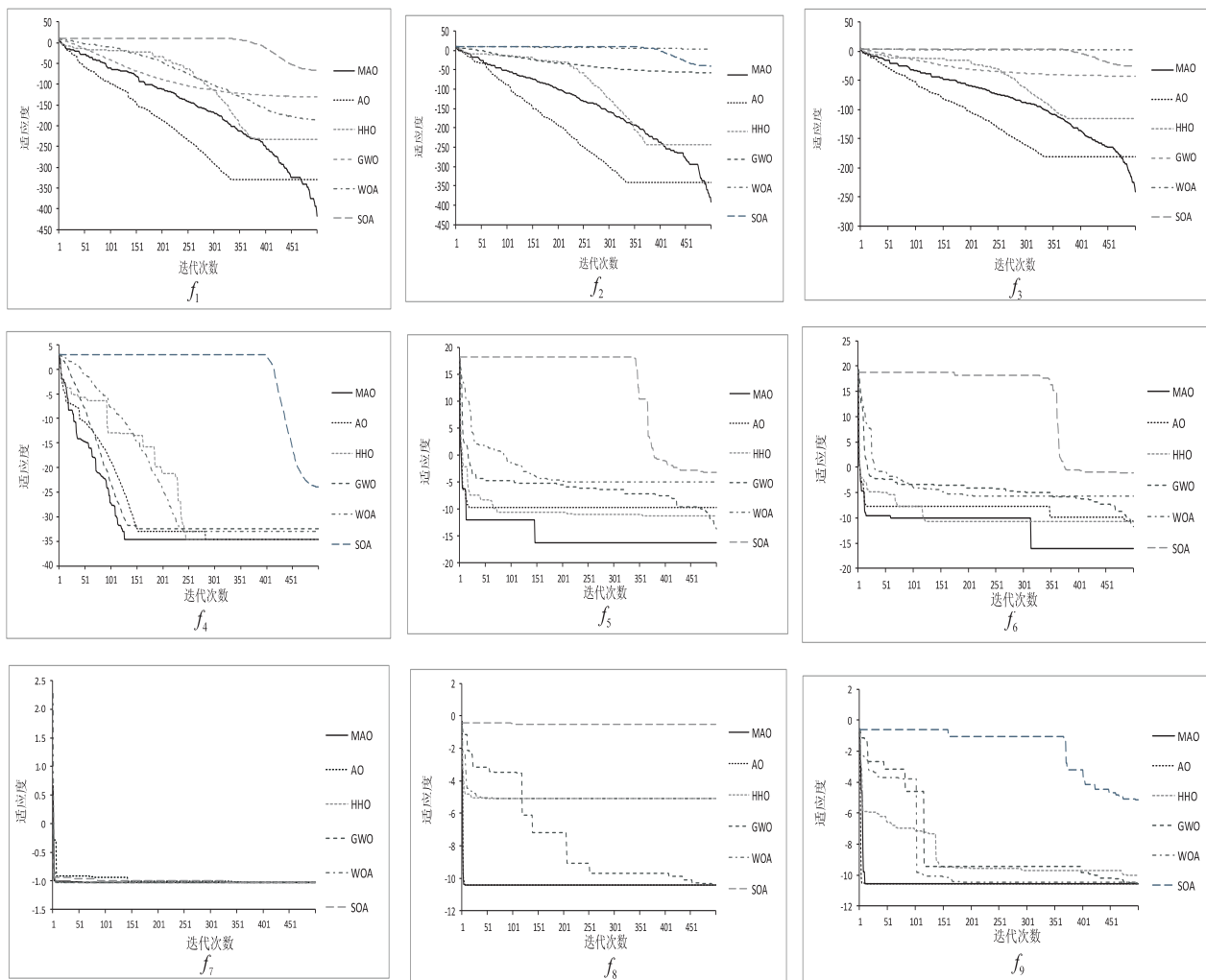


图2 CEC2017 测试函数收敛曲线

4 改进 MAO 算法在机械优化问题中的应用

为了验证改进的 MAO 算法的有效性,将其应用在两个工程优化问题中(包含三杆桁架设计优化问题^[13]、悬臂梁设计优化问题^[14])。机械优化设计问题^[15]主要通过选取设计变量、目标函数、约束条件建成数学模型,对于这一类问题的数学模型,一般可表达为如下约束优化问题:

$$\begin{aligned}
 & \min \text{size}; f(x) \\
 & \text{subject to: } g_p(x) \leq 0, p = 1, 2, \dots, j \\
 & h_m(x) = 0, m = 1, 2, \dots, y \\
 & x_{\text{ub}} \leq x_i \leq x_{\text{lb}}, i = 1, 2, \dots, n
 \end{aligned} \quad (23)$$

式中, x 为设计变量, $x = (x_1, x_2, \dots, x_n) \in R_n$, $f(x)$ 为目标函数; g_p 为第 p 个不等式约束; h_m 为第 m 个等式约束; x_{ub} 和 x_{lb} 分别为设计变量的上下界。

目前,对于求解机械优化中的无约束优化问题,较为主流的处理方案是罚函数法^[16]。罚函数方法是在

目标函数中加入一个惩罚函数将约束问题转换成一个无约束问题^[17]。罚函数的表达式为:

$$F(x) = f(x) + \lambda [h^2(x) + \min \{0, g(x)\}^2] \quad (24)$$

式中, $F(x)$ 为惩罚函数, $f(x)$ 为优化问题的原始目标函数, λ 为惩罚因子, $h^2(x)$ 和 $\min \{0, g(x)\}^2$ 分别为与等式相关的惩罚项和与不等式相关的惩罚项^[18]。惩罚因子 λ 的取值对算法具有重要影响,当 λ 取值过大时,容易导致算法早熟收敛,难以搜索到最优解;当 λ 取值过小时,无法达到预期的惩罚效果, λ 取值是通过大量实验确定的^[19]。

4.1 三杆桁架优化问题

三杆桁架设计优化的目标是通过调整三杆桁架横截面积的大小来寻优的三杆桁架体积。三杆桁架优化问题具有 1 个非线性的适应度函数,3 个不等式约束条件,2 个决策优化变量。其优化目标函数表达式如公式 25 所示:

$$\begin{aligned}
 & \min \text{size}; f(x) = (2\sqrt{2x_1} + x_2) \cdot l \\
 & \text{subject to: } g_1(x) = \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1^2} + 2x_1x_2} p - \sigma \leq 0
 \end{aligned}$$

$$g_2(x) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2}}p - \sigma \leq 0$$

$$g_3(x) = \frac{1}{\sqrt{2x_2} + x_1}p - \sigma \leq 0$$

$$0 \leq x_i \leq 1, i = 1, 2$$

$$l = 100 \text{ cm}, p = 2\text{KN/cm}^2, \sigma = 2\text{KN/cm}^2$$

(25)

式中, $f(x)$ 代表三杆桁架最优体积, 变量 l 为挠度, p

为屈曲, σ 为桁架构件的应力约束, x_1 和 x_2 为评估最佳横截面积的两侧桁杆架长度。

表2中将 MAO 等六种算法应用于该问题的优化中, $f(x)$ 的值越小说明三杆桁架体积越小, 优化效果越好。如表2所示, MAO 算法求解三杆桁架最优体积为 263.913 5, 在所有对比算法求解三杆桁架最优体积中最小。故 MAO 算法在三杆桁架工程优化问题中能优于基础的 AO 算法及其余四种算法。

表2 三杆桁架优化问题中不同算法性能比较

	MAO	AO	HHO	GWO	WOA	SOA
$f(x)$	263.913 5	264.507	264.14	263.915 9	264.329 3	263.953 5
x_1	0.789 45	0.789 01	0.771 01	0.793 29	0.814	0.797 28
x_2	0.406 22	0.413 42	0.460 64	0.395 41	0.340 94	0.384 49
最优排序	1	6	4	2	5	3

4.2 悬臂梁优化问题

悬臂梁包括5个截面为方形的空心单元。每个单元由一个变量定义, 而厚度是恒定的, 因此共有5个结构参数, 即5个决策变量。悬臂梁设计优化的目标是使其矩形截面的质量越小越好, 约束条件是满足一个垂直位移约束。其优化目标函数表达式如公式26所示:

$$\min \text{ size}; f(x) = 0.062 4 \sum_{i=1}^5 x_i$$

$$\text{subject to: } \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

表3 悬臂梁优化问题中不同算法性能比较

	MAO	AO	HHO	GWO	WOA	SOA
$f(x)$	1.341 2	1.346 4	1.346 7	1.340 3	1.399 5	1.345 3
x_1	5.877 1	5.607 4	5.639 7	6.085 3	5.475 7	6.188 1
x_2	5.351 7	5.549 1	5.621 5	5.263 3	5.159 3	5.446
x_3	4.557 9	4.631 7	4.666 9	4.473 5	4.529	4.592 3
x_4	3.483 4	3.644 4	3.579 4	3.464 6	5.127 2	3.332 9
x_5	2.223 1	2.144 5	2.073 9	2.193 2	2.136 2	1.999 3
最优排序	2	5	4	1	6	3

4.3 改进 MAO 算法应用总结

改进的 MAO 算法在悬臂梁应用问题上的优化效果均优于 AO, HHO, WOA, SOA 四种算法, 略差于 GWO 算法; 但在三杆桁架设计工程应用问题上效果优于该文选用的五种对比算法。综上所述, 改进的 MAO 算法在寻优性能上优于基础 AO 算法, 验证了 MAO 算法改进的有效性与实用性。

5 结束语

针对阿奎拉鹰优化算法局部搜索能力较弱的问题, 提出一种多项式变异和自适应权重优化的阿奎拉

$$0.01 \leq x_i \leq 100, i = 1, 2, \dots, 5$$

(26)

式中, $f(x)$ 代表悬臂梁矩形截面的最优质量, x_i 代表不同单元的悬臂梁高度或宽度。

将 MAO 等六种算法应用于该问题的优化中, 如表3所示, MAO 算法求解悬臂梁矩形截面最优质量为 1.341 2, 优于 AO, WAO, HHO, SOA 算法求取的最优解, 略差于 GWO 算法求取的最优解。

鹰算法, 在初始化种群时使用 Tent 混沌映射策略, 均匀种群位置; 在全局搜索过程中增加多项式变异扰动, 降低算法陷入局部最优的可能性; 在局部开发过程中, 采用自适应权重因子, 加快局部收敛速度寻得最优解; 在平衡全局搜索和局部开发时采用动态切换概率, 增加寻优灵活性。

基于以上改进策略提升了算法的寻优性能, 选用九个测试函数和两类工程优化问题, 通过与基础 AO 算法和四种较新的算法对比, 验证了改进的 MAO 算法的有效性。未来将继续深入研究该算法的优化改进方案, 将改进算法应用到更多的领域中。

参考文献:

- [1] ABUALIGAH L, YOUSRI D, ELAZIZ M E A, et al. Aquila optimizer; a novel meta-heuristic optimization algorithm [J]. Computers & Industrial Engineering, 2021, 157: 1-63.
- [2] AKYOL S. A new hybrid method based on Aquila optimizer and tangent search algorithm for global optimization [J]. Journal of Ambient Intelligence and Humanized Computing, 2023, 14: 8045-8065.
- [3] ZHANG Y J, YAN Y X, ZHAO J, et al. AOAAO: the hybrid algorithm of arithmetic optimization algorithm with aquila optimizer [J]. IEEE Access, 2022, 10: 10907-10933.
- [4] 贾鹤鸣, 刘庆鑫, 刘宇翔, 等. 融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法 [J]. 智能系统学报, 2023, 18 (1): 104-116.
- [5] DONG S, BU C, WANG Y. Improved whale optimization algorithm based on the tent chaotic mapping and nonlinear convergence factor [J]. Journal of Physics: Conference Series, 2020, 1682: 012055.
- [6] 文 斌, 朱 哈. 基于自适应权重的立体匹配优化算法 [J]. 计算机工程, 2021, 47 (4): 268-276.
- [7] 付小鹏, 王 勇, 冯爱武. 采用混合搜索策略的阿奎拉优化算法 [J]. 计算机应用研究, 2022, 39 (10): 3026-3032.
- [8] HUSSAIN K, ZHU W, SALLEH M N. Long-term memory Harris' hawk optimization for high dimensional and optimal power flow problems [J]. IEEE Access, 2019, 7: 147596-147616.
- [9] MIRJALILI S, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69: 46-61.
- [10] MIRJALILI S, LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.
- [11] DHIMAN G, KUMAR V. Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems [J]. Knowledge-Based Systems, 2018, 165: 169-196.
- [12] HU Hui, CAI Zhaoquan, HU Song, et al. Constrained solution of CEC2017 with monarch butterfly optimisation [J]. Int. J. of Wireless and Mobile Computing, 2019, 16 (2): 138-145.
- [13] 贾连春, 崔 春, 姜良芹. 可变参数的三杆桁架体系优化仿真 [J]. 低温建筑技术, 2011, 33 (6): 74-75.
- [14] 刘 远, 黄墨宇, 金 晶. 一种悬臂梁结构设计优化方案 [J]. 船海工程, 2016, 45 (6): 34-37.
- [15] 熊海根. 浅谈机械设计中优化设计理论与方法选择 [J]. 南方农机, 2015, 46 (9): 48-49.
- [16] 汪逸晖, 高 亮. 乌鸦搜索算法的改进及其在工程约束优化问题中的应用 [J]. 计算机集成制造系统, 2021, 27 (7): 1871-1883.
- [17] 原杨飞, 党乾龙, 徐 伟, 等. 动态罚函数法求解约束优化问题 [J]. 计算机工程与应用, 2022, 58 (4): 83-90.
- [18] 许雨晴, 周芳宇, 刘 茜. 求解不等式约束问题的一类新的精确罚函数方法 [J]. 山东师范大学学报: 自然科学版, 2018, 33 (4): 406-413.
- [19] 郑芳英. 求解不等式约束极大极小值问题的罚函数方法 [J]. 浙江理工大学学报, 2014, 31 (9): 559-564.