智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法

贾鹤鸣, 刘庆鑫, 刘宇翔, 王爽, 吴迪

引用本文:
贾鹤鸣,刘庆鑫,刘宇翔,王爽,吴迪. 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.
JIA Heming,LIU Qingxin,LIU Yuxiang,WANG Shuang,WU Di. Hybrid Aquila and Harris hawks optimization algorithm with dynamic opposition–based learning[J]. *CAAI Transactions on Intelligent Systems*, 2023, 18(1): 104–116.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202108031

DOI: 10.11992/tis.202108031
网络出版地址: https://kns.cnki.net/kcms/detail/23.1538.TP.20220601.1452.002.html

融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法

贾鹤鸣[1], 刘庆鑫[2], 刘宇翔[3], 王爽[1], 吴迪[4]

（1. 三明学院 信息工程学院, 福建 三明 365004; 2. 海南大学 计算机科学与技术学院, 海南 海口 570228; 3. 福州大学 物理与信息工程学院, 福建 福州 350108; 4. 三明学院 教育与音乐学院, 福建 三明 365004）

摘　要: 阿奎拉鹰优化算法 (Aquila optimizer, AO) 和哈里斯鹰优化算法 (Harris hawks optimization, HHO) 是近年提出的优化算法。AO 算法全局寻优能力强, 但收敛精度低, 容易陷入局部最优, 而 HHO 算法具有较强的局部开发能力, 但存在全局探索能力弱, 收敛速度慢的缺陷。针对原始算法存在的局限性, 本文将两种算法混合并引入动态反向学习策略, 提出一种融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法。首先, 在初始化阶段引入动态反向学习策略提升混合算法初始化性能与收敛速度。此外, 混合算法分别保留了 AO 的探索机制与 HHO 的开发机制, 提高算法的寻优能力。仿真实验采用 23 个基准测试函数和 2 个工程设计问题测试混合算法优化性能, 并对比了几种经典反向学习策略, 结果表明引入动态反向学习的混合算法收敛性能更佳, 能够有效求解工程设计问题。

关键词: 阿奎拉鹰优化算法; 哈里斯鹰优化算法; 动态反向学习; 混合优化; 基准函数; 管柱设计问题; 汽车碰撞设计问题; Wilcoxon 秩和检验

中图分类号: TP301.6　　**文献标志码**: A　　**文章编号**: 1673−4785(2023)01−0104−13

中文引用格式: 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.

英文引用格式: JIA Heming, LIU Qingxin, LIU Yuxiang, et al. Hybrid Aquila and Harris hawks optimization algorithm with dynamic opposition-based learning[J]. CAAI transactions on intelligent systems, 2023, 18(1): 104–116.

Hybrid Aquila and Harris hawks optimization algorithm with dynamic opposition-based learning

JIA Heming[1], LIU Qingxin[2], LIU Yuxiang[3], WANG Shuang[1], WU Di[4]

(1. School of Information Engineering, Sanming University, Sanming 365004, China; 2. School of Computer Science and Technology, Hainan University, Haikou 570228, China; 3. College of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China; 4. School of Education and Music, Sanming University, Sanming 365004, China)

Abstract: In recent years, optimization algorithms such as the Aquila optimizer (AO) and Harris hawks optimization (HHO) have been proposed. Although AO has strong global optimization capabilities, its convergence accuracy is low, and it is susceptible to local optimization. While the HHO algorithm has a strong local development capability, it suffers from flaws such as limited global exploration capabilities and slow convergence speed. Given the limitations of the original algorithms, this paper combines the two algorithms and proposes a hybrid Aquila and Harris Hawks algorithm with dynamic opposition-based learning. It introduces a dynamic opposition-based learning strategy and proposes a hybrid Aquila and Harris Hawks algorithm with dynamic opposition-based learning. First, a dynamic opposition-based learning strategy is introduced in the initialization phase to improve the initialization performance and convergence speed of the algorithm. Second, the hybrid algorithm retains the exploration mechanism of AO and the exploitation mechanism of HHO, which improves the algorithm's optimization ability. The simulation experiment compares several classical opposition-based learning strategies using 23 benchmark functions and two engineering design problems to test the optimization performance of the hybrid algorithm. The results show that the hybrid algorithm with dynamic opposition-based learning has better convergence performance and can effectively solve engineering design problems.

Keywords: Aquila optimizer; Harris Hawk optimization; dynamic opposition-based learning; hybrid optimization; benchmark function; tubular column design problem; car crash design problem; Wilcoxon rank-sum test

随着当代科技的飞速发展, 工程领域的优化问题日益复杂, 传统基于数学理论的优化方法 (例如牛顿下山法、梯度下降法) 已经无法有效解决此类问题。因此, 众多学者将目光抛向了元启发式优化算法 (meta heuristics, MAs)。MAs 是一

收稿日期: 2021−08−21.　**网络出版日期**: 2022−06−02.
基金项目: 福建省自然科学基金面上项目 (2021J011128); 福建省本科高校教育教学改革研究项目 (FBJG20210338); 三明市科技计划引导性项目 (2021-S-8); 三明学院教育教学改革重点项目 (J2010305); 三明学院高教研究课题 (SHE2013); 福建省农业物联网应用重点实验室开放研究基金项目 (ZD2101).
通信作者: 贾鹤鸣. E-mail: jiahemingluckу99@126.com.

类受自然现象 (例如生物群体行为、物理现象、进化法则) 启发的优化算法。与传统优化方法相比, MAs 不需要梯度信息, 灵活且易于实现, 被广泛应用于医学、计算机科学等领域的复杂优化问题[1-4]。常见的 MAs 有: 遗传算法 (genetic algorithm, GA)[5]、粒子群优化算法 (particle swarm optimization, PSO)[6]、灰狼优化算法 (grey wolf optimizer, GWO)[7]、多元宇宙优化算法 (multi-verse optimizer, MVO)[8]、正余弦优化算法 (sine cosine algorithm, SCA)[9]、鲸鱼优化算法 (whale optimization algorithm, WOA)[10]、樽海鞘群优化算法 (salp swarm algorithm, SSA)[11]、哈里斯鹰优化算法 (Harris hawks optimization, HHO)[12] 和阿奎拉鹰优化算法 (Aquila optimizer, AO)[13]。上述 MAs 在某些领域内展示了出色的优化效果, 但由于无免费午餐 (no-free-lunch, NFL) 定理[14] 证实了没有一种 MAs 能够解决所有的优化问题, 即每种 MAs 都有各自的优势和局限性, 仅针对部分问题有效。这激励着众多学者提出各种新颖或改进的 MAs 来解决不同类型的优化问题。

AO 是 Abualigah 等于 2021 年提出的一种新型 MAs。该算法模拟了阿奎拉鹰针对不同猎物所采取的多种攻击行为。AO 具有强大的全局寻优能力, 搜索效率高, 收敛速度快, 但局部开发能力较弱, 易受到局部极值的干扰。为了缓解上述问题, Al-qaness 等[15] 引入反向学习策略提高 AO 的搜索性能, 并将其应用到自适应模糊神经推理系统, 确定模型的最佳参数。Li 等[16] 引入自适应权重策略提高 AO 的局部开发能力, 并将改进算法应用于冷热电联产系统, 降低模型的运行成本。虽然上述文献提出的解决方案在一定程度上改善了 AO 的搜索性能, 但上述研究均是通过改进策略提升算法的性能, 忽略了混合算法所带来的优势特性 (即利用多种算法实现搜索过程中的多样化信息交流)。此外, AO 提出的时间较短, 其性能有待进一步提升。HHO 是在 2019 年由 Heidari 等提出的一种 MAs。该算法模仿了哈里斯鹰捕食猎物的一系列行为。HHO 根据猎物的逃逸能量, 实现算法对全局探索阶段和局部开发阶段的过渡。开发阶段根据猎物的能量和逃跑概率采取 4 种不同的进攻策略。实验表明该算法局部寻优能力较强, 采用 4 种不同的进攻策略的同时借助莱维游走使得 HHO 具有较强的抗局部最优值干扰能力, 能够有效避免算法出现“早熟”现象, 但前期全局探索能力较弱, 无法快速找到全局最优所在的区域, 导致算法收敛速度较慢。

基于上述分析, AO 和 HHO 算法存在各自的优势和局限性。为了改善 AO 的寻优性能, 本文利用混合算法的改进思路, 分别汲取了 AO 和 HHO 在全局探索和局部搜索方面的优势, 同时引入动态反向学习策略, 提出了一种融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法 (hybrid Aquila and Harris hawk optimization with dynamic opposition-based learning, DAHHO)。混合算法分别利用了 AO 的全局探索阶段和 HHO 的局部开发阶段进行迭代寻优, 并且引入动态反向学习策略提升种群初始化性能, 为后续的高效搜索奠定基础。仿真实验部分采用 23 个基准测试函数以及 2 个经典工程设计问题对改进算法进行性能测试, 并探讨了其他反向学习策略所带来的优化效果, 结果表明融合动态反向学习的混合算法寻优性能更优, 收敛速度和收敛精度均有较大提高, 为解决约束工程设计问题提供了一种行之有效的计算方法。

1　阿奎拉鹰优化算法

阿奎拉鹰优化算法灵感来自阿奎拉鹰的捕食行为。阿奎拉鹰利用其极快的飞行速度与强有力的鹰爪攻击猎物, 并且能够根据猎物的不同转换狩猎策略。AO 通过迭代次数划分探索阶段和开发阶段, 若$t \leqslant \frac{2}{3}T$时, AO 执行全局探索, 否则执行局部开发。每个阶段的具体描述如下。

1.1　探索阶段

阿奎拉鹰在搜索空间内通过 2 种不同的飞行方式搜索猎物, 通过随机数 r_1 平衡采用不同策略的概率。

1.1.1　高空飞行搜索

阿奎拉鹰在高空中探索, 通过敏锐的眼睛观察位于其下方飞行的猎物, 其位置更新公式为

$$\boldsymbol{X}(t+1)=\boldsymbol{X}_{\text{best}}(t)\times\left(1-\frac{t}{T}\right)+(\boldsymbol{X}_{\text{M}}(t)-\boldsymbol{X}_{\text{best}}(t)\times r_2) \tag{1}$$

$$\boldsymbol{X}_{\text{M}}(t)=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{X}_i(t) \tag{2}$$

式中: $\boldsymbol{X}_{\text{best}}(t)$ 代表种群最佳位置; $\boldsymbol{X}_{\text{M}}(t)$ 代表种群平均位置; t 为当前迭代次数; T 为最大迭代次数; N 为种群规模; r_1 和 r_2 为分布在 0 到 1 之间的随机数。

1.1.2　环绕猎物飞行

确定猎物范围后, 阿奎拉鹰会在目标上方盘旋, 寻找合适的机会攻击猎物, 数学模型为

$$\boldsymbol{X}(t+1)=\boldsymbol{X}_{\text{best}}(t)\times\text{LF}(D)+\boldsymbol{X}_{\text{R}}(t)+(y-x)\times r_3 \tag{3}$$

式中: D 表示维度; $\boldsymbol{X}_{\text{R}}(t)$ 代表种群随机位置; r_3 是 0 到 1 的随机数; LF(D) 表示莱维飞行函数, 其公式为

$$\mathrm{LF}(D)=s\times\frac{u\times\sigma}{|v|^{\frac{1}{\beta}}}$$

$$\sigma=\left(\frac{\Gamma(1+\beta)\times\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\times\beta\times2^{\left(\frac{\beta-1}{2}\right)}}\right)^{\frac{1}{\beta}}$$

式中：s 是取值为 0.01 的常数；u 与 v 为 [0,1] 间的随机数；β=1.5。

$$\begin{cases}x=l\times\sin(\theta)\\ y=l\times\cos(\theta)\\ l=C+0.005\ 65\times D\\ \theta=-\omega\times D+\dfrac{3\pi}{2}\end{cases}$$

式中：C 是取值为 10 的搜索空间系数；ω 是取值为 0.005 的常数。

1.2　开发阶段

此时阿奎拉鹰已确定猎物位置，进而降低自身飞行高度以接近猎物，通过随机数 r_4 采取不同的狩猎策略。

1.2.1　低空飞行攻击

阿奎拉鹰接近猎物并通过强壮的鹰爪攻击猎物背部或颈部，其数学模型为

$$\boldsymbol{X}(t+1)=(\boldsymbol{X}_{\mathrm{best}}(t)-\boldsymbol{X}_M(t))\times\alpha-r_5+((\mathrm{UB}-\mathrm{LB})\times r_6+\mathrm{LB})\times\delta$$

式中：α 与 δ 是值为 0.1 的适应参数；UB 和 LB 分别代表种群上下界；r_5 与 r_6 为 0~1 的随机数。

1.2.2　地面近距离攻击

阿奎拉鹰在陆地游走接近猎物并发动攻击，数学模型为

$$\boldsymbol{X}(t+1)=\mathrm{QF}\times\boldsymbol{X}_{\mathrm{best}}(t)-(G_1\times\boldsymbol{X}(t)\times r_7)-G_2\times\mathrm{LF}(D)+r_8\times G_1$$

$$\mathrm{QF}(t)=t^{\frac{2\times r_9-1}{(1-T)^2}}$$

$$G_1=2\times r_{10}-1$$

$$G_2=2\times\left(1-\frac{t}{T}\right)$$

式中：QF(t)是质量函数，用于平衡其搜索策略；G_1 猎物的逃跑轨迹；G_2 是从 2 到 0 线性递减参数，表示阿奎拉鹰在空中追逐猎物的飞行斜率；r_7、r_8、r_9、r_{10} 为分布在 0 到 1 之间的随机数。

2　哈里斯鹰优化算法

哈里斯鹰是美国南部地区的一种猛禽，它们的觅食方式取决于狩猎环境以及猎物的逃逸形式。算法主要分为探索阶段、探索与开发转换阶段、开发阶段[17]。

2.1　探索阶段

哈里斯鹰随机栖息在搜索空间中，通过两种策略搜寻猎物，利用随机数平衡使用策略的概率，数学表述为

$$\boldsymbol{X}(t+1)=\begin{cases}\boldsymbol{X}_R(t)-r_{12}|\boldsymbol{X}_R(t)-2r_{13}\boldsymbol{X}(t)|\\ (\boldsymbol{X}_{\mathrm{best}}(t)-\boldsymbol{X}_M(t))-r_{14}(\mathrm{LB}+r_{15}(\mathrm{UB}-\mathrm{LB}))\end{cases}$$

式中：r_{12}、r_{13}、r_{14}、r_{15} 均为 0~1 的随机数。

2.2　探索与开发转换阶段

哈里斯鹰根据猎物的逃逸能量实现探索阶段与开发阶段的转换，猎物的逃逸能量 E 如图 1 所示，其计算公式为

$$E=2E_0\left(1-\frac{t}{T}\right)$$

式中：E_0 为−1~1 之间的随机数；当$|E|\geqslant1$时，HHO 执行探索模式；否则执行开发模式。

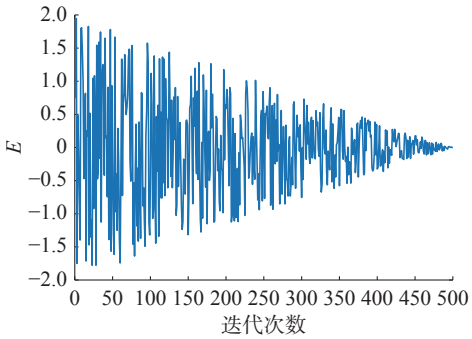


图 1　能量逃逸因子曲线图

Fig. 1　Curve of energy escape factor

2.3　开发阶段

2.3.1　软包围

当$r_{16}\geqslant0.5$且$|E|\geqslant0.5$时，猎物有足够的能量躲避追捕，此时哈里斯鹰通过软包围方式围捕猎物，计算公式为

$$\boldsymbol{X}(t+1)=\Delta\boldsymbol{X}(t)-E|J\boldsymbol{X}_{\mathrm{best}}(t)-\boldsymbol{X}(t)|$$

$$\Delta\boldsymbol{X}(t)=\boldsymbol{X}_{\mathrm{best}}(t)-\boldsymbol{X}(t)$$

$$J=2(1-r_{17})$$

式中：$\Delta\boldsymbol{X}(t)$为最优解与当前解的差值；J 为猎物随机跳跃的距离；r_{16}、r_{17} 是 0~1 的随机数。

2.3.2　硬包围

当$r_{16}\geqslant0.5$且$|E|<0.5$时，猎物的逃逸能量不足，此时哈里斯鹰通过硬包围方式进行狩猎，计算公式为

$$\boldsymbol{X}(t+1)=\boldsymbol{X}_{\mathrm{best}}(t)-E|\Delta\boldsymbol{X}(t)| \tag{4}$$

2.3.3　渐近式快速俯冲的软包围

当$r_{16}<0.5$且$|E|\geqslant0.5$时，猎物有足够的能量逃出包围圈，因此哈里斯鹰需要在攻击前形成一个更加智能的软包围圈，并在莱维游走与快速俯冲突袭之间选取最佳进攻策略 (根据适应度值评判)，计算公式为

$$\boldsymbol{Y}=\boldsymbol{X}_{\mathrm{best}}(t)-E|J\boldsymbol{X}_{\mathrm{best}}(t)-\boldsymbol{X}(t)| \tag{5}$$

$$\boldsymbol{Z}=\boldsymbol{Y}+\boldsymbol{S}\times\mathrm{LF}(D) \tag{6}$$

$$\boldsymbol{X}(t+1)=\begin{cases}\boldsymbol{Y}, & F(\boldsymbol{Y})<F(\boldsymbol{X}(t))\\ \boldsymbol{Z}, & F(\boldsymbol{Z})<F(\boldsymbol{X}(t))\end{cases} \tag{7}$$

式中：$\boldsymbol{S}$ 为 1×D 的位于 [0,1] 间的随机向量。

2.3.4 渐近式快速俯冲的硬包围

当$r_{16}<0.5$且$|E|<0.5$时, 猎物没有足够的能量逃逸, 哈里斯鹰在攻击前形成一个硬包围圈, 基于猎物位置和种群平均位置进行围捕, 从而减小种群与猎物的距离。若突袭失效, 则执行莱维游走 (根据适应度值评判), 数学表述为

$$\boldsymbol{Y}=\boldsymbol{X}_{\text{best}}(t)-E|J\boldsymbol{X}_{\text{best}}(t)-\boldsymbol{X}_{M}(t)| \tag{8}$$

3 动态反向学习策略

对于元启发式优化算法来说, 种群初始化往往是在搜索空间内随机生成的, 只能保证种群在搜索空间内的分散程度, 无法保证初始解的质量。然而, 研究表明初始化的好坏将直接影响算法的收敛速度和解的精度。基于此问题, 国内外学者将多种策略引入初始化部分, 以提高算法初始化性能, 常见的有混沌初始化、反向学习和柯西随机生成等。文献 [18] 针对此问题提供了一种新思路, 将个体分布在非对称动态搜索空间内, 并利用贪婪策略选取最佳个体作为初始化种群, 能够有效提高解的质量。因此, 本文引入动态反向学习策略, 提高初始化解的质量, 计算方法如下为

$$\boldsymbol{X}_{\text{DOBL}}=\boldsymbol{X}_{\text{init}}+r_{18}\times(r_{19}\times(\text{LB}+\text{UB}-\boldsymbol{X}_{\text{init}})-\boldsymbol{X}_{\text{init}})$$

式中: $\boldsymbol{X}_{\text{init}}$ 代表通过随机生成的初始化种群; r_{18} 与 r_{19} 均为分布在 0~1 的随机数。首先, 算法分别生成原始初始化种群 $\boldsymbol{X}_{\text{init}}$ 与反向初始种群 $\boldsymbol{X}_{\text{DOBL}}$, 然后将两个种群合并为新种群 $\boldsymbol{X}_{\text{new}}=\{\boldsymbol{X}_{\text{DOBL}}\cup\boldsymbol{X}_{\text{init}}\}$。计算新种群的适应度值, 并利用贪婪策略使种群内部充分竞争, 选取最佳的 N 个个体作为初始化种群。通过此方法能够让种群更快地靠近最优解, 从而提升算法的收敛速度。

4 DAHHO 算法

4.1 改进混合算法理论分析

根据第 1 节的阐述, 阿奎拉鹰根据不同猎物会采取 4 种不同的捕食行为。在前期迭代中, 根据随机数 r_1 选择高空飞行搜索或环绕猎物飞行, 这两种探索方式主要针对快速移动的猎物。因此位置更新中分别考虑了阿奎拉鹰种群的最佳位置、平均位置和随机位置, 式 (1) 利用种群最佳位置和平均位置实现整体种群在搜索空间内的大范围搜索, 式 (3) 利用莱维飞行配合最佳位置实现搜索空间的大范围随机扰动, 体现算法较强的全局探索能力。当迭代次数$t>(\frac{2}{3}\times T)$时, 根据随机数 r_4 选择低空飞行攻击或地面近距离攻击策略, 这两种攻击方式主要针对移动速度较慢的猎物, 体现算法的局部搜索能力。然而, 根据生物特性来看, 阿奎拉鹰倾向于单独狩猎, 且地面移动能力较弱, 无法对地面上猎物实行有效攻击。从数学描述来看, 局部开发过程并没有对选定的搜索空间进行充分搜索, 其中莱维飞行的作用也很弱, 根据适应度进行位置更新加剧了局部最优停滞。此外, AO 算法仅是根据迭代次数进行阶段划分, 无法有效平衡算法全局阶段与局部阶段。因此, AO 算法全局探索阶段随机性较强, 种群在搜索空间覆盖面广, 不易错失关键的搜索信息, 而局部开发阶段容易陷入局部最优。HHO 则根据猎物的能量衰减实现全局到局部搜索的过渡。探索过程主要依靠最优个体信息, 没有与其他个体交流, 引发种群多样性下降, 导致收敛速度较慢。随着迭代次数的增加, 猎物的能量下降, 进入局部开发阶段, 根据猎物能量和逃逸概率采取 4 种不同的捕食策略。当逃逸概率 $r_{16}\geqslant 0.5$ 时, 通过猎物能量选择软包围或硬包围, 分别根据自身与猎物的距离和猎物的位置进行位置更新。否则, 通过猎物能量选择渐进式快速俯冲的软包围或硬包围, 两种方式均加入了莱维飞行项, 并根据适应度判断使用莱维游走还是快速俯冲攻击, 使算法能够有效跳出局部最优。总的来说, AO 的全局搜索和 HHO 的局部搜索分别是该两种算法的核心特点, 本文将 AO 的全局探索阶段和 HHO 的局部探索阶段相结合, 充分发挥两种算法的优势, 保留算法的全局探索能力、较快的收敛速度和跳出局部最优的能力。此外, 在初始化阶段引入动态反向学习策略, 进一步提升算法初始化种群质量, 使混合算法的收敛速度和精度都得到提高, 增强了算法的整体寻优性能。

DAHHO 算法的不同阶段如图 2 所示。

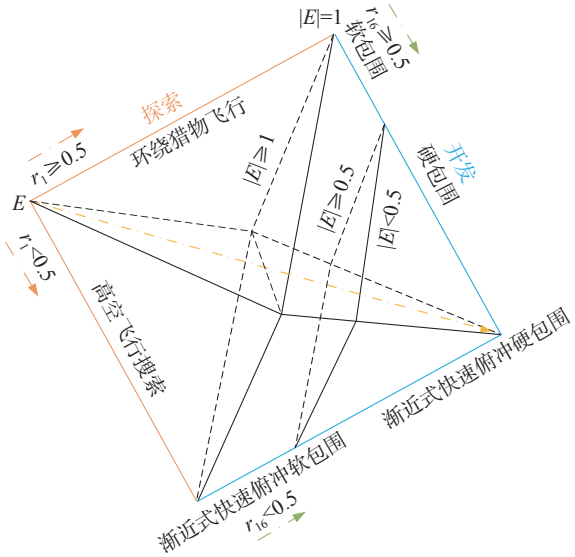


图 2　DAHHO 算法的不同阶段

Fig. 2　Different phases of DAHHO

DAHHO 算法步骤如下:

1）初始化种群，计算动态反向学习种群，根据贪婪策略保留较优个体进入主程序迭代。

2）计算种群适应度值,记录较优个体。

3）如果 $|E| \geqslant 1$，个体随机选择式 (1) 或式 (3) 开始探索行为。

4）如果 $|E| < 1$，种群根据逃逸猎物能量值与适应度值选择开发策略：

策略 1　软包围。当 $r_{16} \geqslant 0.5$ 且 $|E| \geqslant 0.5$，子群个体采用式 (11) 更新位置。

策略 2　硬包围。当 $r_{16} \geqslant 0.5$ 且 $|E| < 0.5$，子群个体采用式 (4) 更新位置。

策略 3　渐近式快速俯冲软包围。当 $r_{16} < 0.5$ 且 $|E| \geqslant 0.5$，子群个体采用式（5）、（6）、（7）更新位置。

策略 4　渐近式快速俯冲硬包围。当 $r_{16} < 0.5$ 且 $|E| < 0.5$，子群个体采用式（6）、（7）、（8）更新位置。

5）判断程序是否满足终止条件，满足则跳出循环，否则返回步骤 2。

6）输出最佳位置及适应度值。

4.2　计算复杂度分析

DAHHO 计算复杂度主要取决于以下 3 个过程：动态反向学习初始化，适应度评估以及种群位置更新。假设种群规模为 N，则初始化计算复杂度为 $O(2\times N)$。评价最佳位置以及更新种群位置向量的计算复杂度为 $O(N\times T+N\times T\times D)$。综上所述，DAHHO 的计算复杂度为 $O(N\times(2+T+T\times D))$。考虑到 NFL 定理，增加一些额外的计算复杂度来获取更好的解决方案是可取的。

5　实验仿真与分析

本文实验环境采用 Intel(R) Core(TM) i5-9 500 CPU，主频为 3.00 GHz，内存 24 GB 的电脑，操作系统为 64 位 Windows 10，编程语言为 MATLAB，版本 R2020b。使用 23 个基准测试函数对算法进行性能测试，可有效验证本文所提算法的寻优能力[4]。其中，F1~F7 为单峰测试函数，只有一个最优解，可评估算法的收敛速度与求解精度，F8~F24 是多峰测试函数，具有多个局部最优值与一个全局最优值，用于评估算法的全局搜索能力与避免陷入局部最优能力。

5.1　实验环境与参数设置

为了确保实验的公平性与可对比性，本文采用两组对比算法进行实验。第 1 组：选取 7 种不同的优化算法作为对比算法，这些算法被证实具有良好的寻优能力，分别是：AO[13]、HHO[12]、SSA[11]、WOA[10]、SCA[9]、MVO[8]、GWO[7] 和 GA[5]。各算法参数设置如表 1 所示。第 2 组：以混合算法为主体，在初始化阶段引入不同的经典反向学习策略，如引入反向学习[19](OBLAHHO)、随机反向学习[4](ROBLAHHO) 与准反向学习[20](QOBLAHHO)，以验证不同类型的反向学习策略对混合算法的改进效果。在实验设置方面，设定所有算法的种群规模 N=30，维度 D=30，最大迭代次数 T=500,各算法独立运行 30 次。本文分别采用平均值、标准差与 Wilcoxon 秩和检验作为算法性能评价指标。其中，平均值越小代表算法收敛精度高，标准差越小代表算法波动小，稳定性高。

表 1　各算法参数设置

Table 1　Parameter setting of each algorithm

算法	参数名称	参数设置
AO[8]	搜索周期参数	C=10
HHO[7]	随机跳跃长度	J=[0,2]
SSA[16]	平衡参数	c_1=[1,0]
	收敛因子	a=[2,0]
WOA[6]	随机数	l=[−1,1]
	螺旋常数	b=1
SCA[17]	移动系数	a=2
MVO[18]	蠕虫存在率	WEP=[0.2,1]
	旅行距离率	TDR=[0.645 0,0]
GWO[19]	收敛因子	a=[2,0]
GA[20]	交叉概率	P_c=0.7
	变异概率	P_m=0.3
OBLAHHO	随机跳跃距离	J=[0,2]
ROBLAHHO	随机跳跃距离	J=[0,2]
QOBLAHHO	随机跳跃距离	J=[0,2]

5.2　求解精度分析

第 1 组实验结果如表 2 所示。其中，最佳结果用粗体显示。从单峰测试函数来说，DAHHO 在测试函数 F1~F5 与 F7 表现最佳，不仅求解精度最高，而且稳定性也超过了其余对比算法。值得注意的是，在求解 F1 与 F3 时，DAHHO 的标准差为 0，表现出优秀的求解稳定性。对于函数 F6，DAHHO 优化效果仅次于 SSA。在多峰测试函数中，DAHHO 也能表现出不错的效果。对于函数 F9~F11，DAHHO 可直接搜寻到理论最优值，且方差最小。F8 是最复杂的多峰测试函数，DAHHO 虽不能寻到全局最优值，但结果也仅次于 HHO 算法。对于函数 F12、13 与 F15，DAHHO 寻优精度与标准差皆优于其他算法。对于 F14，效果不及 MVO。对于 F20，GWO 达到了理论最优值，但标准差不如

MVO。对于函数 F16~F19、F21 和 F23, DAHHO 能够寻到全局最优值, 寻优结果超过大部分对比算法, 并且表现出不错的稳定性。从算法测试结果可以看出, DAHHO 并不能在所有测试函数上跑赢对比算法, 从侧面反映出本文选取的对比算法是具有可对比性与一定的参考价值。但 DAHHO 能够在大部分测试函数上表现出优秀的收敛精度以及稳定性。

表 2　DAHHO 与对比算法的基准函数测试结果

Table 2　Benchmark function results of DAHHO and contrast algorithm

函数	统计值	DAHHO	AO	HHO	SSA	WOA	SCA	MVO	GWO	GA
F1	平均值	**1.068×10^{-231}**	6.400×10^{-99}	8.200×10^{-98}	1.667×10^{-7}	7.250×10^{-74}	1.882×10^{1}	1.210	8.881×10^{-28}	7.103×10^{3}
	标准差	**0.000**	2.650×10^{-98}	4.330×10^{-97}	1.752×10^{-7}	3.560×10^{-73}	4.216×10^{1}	4.300×10^{-1}	1.838×10^{-27}	2.784×10^{3}
F2	平均值	**1.819×10^{-120}**	2.800×10^{-53}	4.060×10^{-52}	1.944	1.370×10^{-49}	1.800×10^{-2}	1.150	9.163×10^{-17}	3.539×10^{1}
	标准差	**4.849×10^{-120}**	1.530×10^{-52}	1.900×10^{-51}	1.477	7.420×10^{-49}	3.000×10^{-2}	1.390	5.910×10^{-17}	5.673
F3	平均值	**4.982×10^{-192}**	3.040×10^{-109}	5.060×10^{-69}	1.257×10^{3}	4.663×10^{4}	8.659×10^{3}	2.405×10^{2}	9.791×10^{-6}	9.976×10^{3}
	标准差	**0.000**	1.660×10^{-108}	2.770×10^{-68}	7.937×10^{2}	$1.357\mathrm{E}\times10^{5}$	6.034×10^{3}	1.105×10^{2}	2.516×10^{-5}	4.027×10^{3}
F4	平均值	**1.464×10^{-117}**	7.100×10^{-55}	1.290×10^{-47}	1.076×10^{1}	4.983×10^{1}	3.364×10^{1}	2.080	8.978×10^{-7}	2.747×10^{1}
	标准差	**7.860×10^{-117}**	3.880×10^{-54}	6.140×10^{-47}	3.439	2.536×10^{1}	1.124×10^{1}	8.000×10^{-1}	1.000×10^{-6}	4.449
F5	平均值	**4.617×10^{-4}**	6.500×10^{-3}	1.420×10^{-2}	2.359×10^{2}	2.793×10^{1}	1.586×10^{4}	3.680×10^{2}	2.721×10^{1}	3.352×10^{6}
	标准差	**7.029×10^{-3}**	9.000×10^{-3}	2.600×10^{-2}	4.215×10^{2}	5.200×10^{-1}	2.369×10^{4}	6.076×10^{2}	7.400×10^{-1}	1.755×10^{6}
F6	平均值	7.443×10^{-6}	1.500×10^{-4}	1.400×10^{-4}	**1.508×10^{-7}**	4.010×10^{-1}	2.506×10^{1}	1.260	8.188×10^{-1}	7.327×10^{3}
	标准差	1.300×10^{-5}	4.000×10^{-4}	1.900×10^{-4}	**1.290×10^{-7}**	2.340×10^{-1}	6.332×10^{1}	3.603×10^{-1}	3.573×10^{-1}	2.203×10^{3}
F7	平均值	**9.154×10^{-5}**	2.000×10^{-4}	1.000×10^{-4}	1.600×10^{-1}	2.000×10^{-3}	1.620×10^{-1}	1.489×10^{1}	2.323×10^{-3}	1.581
	标准差	**7.424×10^{-5}**	8.300×10^{-5}	1.000×10^{-4}	6.293×10^{-2}	2.800×10^{-3}	8.200×10^{-2}	1.028×10^{1}	1.537×10^{-3}	1.120
F8	平均值	-1.227×10^{4}	-8.154×10^{3}	**-1.253×10^{4}**	-7.544×10^{3}	-1.018×10^{4}	-3.775×10^{3}	-7.668×10^{3}	-5.702×10^{3}	-5.188×10^{3}
	标准差	1.280×10^{3}	4.021×10^{3}	**1.354×10^{2}**	6.536×10^{2}	2.017×10^{3}	2.577×10^{2}	8.095×10^{2}	6.549×10^{2}	8.136×10^{2}
F9	平均值	**0.000**	**0.000**	**0.000**	4.663×10^{1}	**0.000**	4.168×10^{1}	1.198×10^{2}	3.852	1.911×10^{2}
	标准差	**0.000**	**0.000**	**0.000**	1.363×10^{1}	**0.000**	3.315×10^{1}	3.638×10^{1}	4.545	2.379×10^{1}
F10	平均值	**8.882×10^{-16}**	8.880×10^{-16}	8.880×10^{-16}	2.583	3.840×10^{-15}	1.499×10^{1}	1.843	9.847×10^{-14}	1.339×10^{1}
	标准差	**0.000**	**0.000**	**0.000**	7.491×10^{-1}	2.480×10^{-15}	8.078	6.290×10^{-1}	1.663×10^{-14}	1.227
F11	平均值	**0.000**	**0.000**	**0.000**	1.590×10^{-2}	1.500×10^{-2}	9.300×10^{-1}	8.500×10^{-1}	5.621×10^{-3}	6.088×10^{1}
	标准差	**0.000**	**0.000**	**0.000**	1.224×10^{-2}	4.200×10^{-2}	3.330×10^{-1}	8.000×10^{-2}	1.033×10^{-2}	1.687×10^{1}
F12	平均值	**1.842×10^{-6}**	3.130×10^{-6}	6.610×10^{-6}	6.996	2.400×10^{-2}	2.638×10^{4}	2.048	3.669×10^{-2}	1.661×10^{5}
	标准差	**2.964×10^{-6}**	4.490×10^{-6}	7.650×10^{-6}	2.859	1.200×10^{-2}	7.601×10^{4}	1.430	1.645×10^{-2}	3.700×10^{5}
F13	平均值	**1.119×10^{-5}**	3.880×10^{-5}	8.040×10^{-5}	1.902×10^{1}	4.499×10^{-1}	8.043×10^{4}	2.242×10^{-1}	7.337×10^{-1}	3.813×10^{6}
	标准差	**2.121×10^{-5}**	4.540×10^{-5}	1.000×10^{-4}	1.356×10^{1}	2.272×10^{-1}	2.348×10^{5}	1.196×10^{-1}	2.986×10^{-1}	3.227×10^{6}
F14	平均值	2.706	2.010	1.097	1.230	2.862	2.056	**9.980×10^{-1}**	3.488	1.559
	标准差	2.528	2.714	3.033×10^{-1}	5.005×10^{-1}	3.320	1.006	**2.430×10^{-11}**	3.223	1.287
F15	平均值	**3.543×10^{-4}**	5.000×10^{-4}	4.000×10^{-4}	2.170×10^{-3}	6.000×10^{-4}	1.000×10^{-3}	4.300×10^{-3}	3.701×10^{-3}	2.223×10^{-3}
	标准差	**6.154×10^{-5}**	1.000×10^{-4}	1.000×10^{-4}	4.955×10^{-3}	4.000×10^{-4}	3.000×10^{-4}	1.100×10^{-2}	7.579×10^{-3}	4.953×10^{-3}
F16	平均值	**−1.032**	**−1.032**	**−1.032**	**−1.032**	**−1.032**	**−1.032**	**−1.032**	**−1.032**	**−1.032**
	标准差	**2.580×10^{-13}**	3.000×10^{-4}	4.830×10^{-9}	2.899×10^{-12}	2.170×10^{-9}	6.630×10^{-5}	2.640×10^{-7}	2.065×10^{-8}	6.454×10^{-10}

续表 2

函数	统计值	DAHHO	AO	HHO	SSA	WOA	SCA	MVO	GWO	GA
F17	平均值	**3.980×10^{-1}**	**3.980×10^{-1}**	**3.980×10^{-1}**	3.979×10^{-1}	**3.980×10^{-1}**	**3.996×10^{-1}**	**3.980×10^{-1}**	3.979×10^{-1}	3.979×10^{-1}
	标准差	**1.073×10^{-13}**	1.000×10^{-4}	6.490×10^{-6}	1.437×10^{-11}	1.320×10^{-5}	1.646×10^{-3}	1.880×10^{-7}	6.391×10^{-7}	2.875×10^{-10}
F18	平均值	**3.000**	3.043	**3.000**	3.000	**3.000**	**3.000**	5.700	**3.000**	**3.000**
	标准差	**2.153×10^{-12}**	4.000×10^{-2}	3.830×10^{-7}	2.116×10^{-10}	5.570×10^{-5}	1.000×10^{-4}	1.479×10^{1}	6.101×10^{-5}	1.962×10^{-10}
F19	平均值	**−3.861**	−3.860	−3.860	−3.860	−3.860	−3.870	−3.860	−3.860	−3.860
	标准差	**3.449×10^{-9}**	6.000×10^{-3}	3.300×10^{-3}	1.152×10^{-7}	1.300×10^{-2}	1.000×10^{-2}	3.580×10^{-6}	4.141×10^{-3}	2.539×10^{-4}
F20	平均值	−3.074	−3.170	−3.115	−3.220	−3.250	−2.910	−3.290	**−3.268**	−3.233
	标准差	1.311×10^{-1}	1.020×10^{-1}	1.200×10^{-1}	5.836×10^{-2}	8.300×10^{-2}	3.500×10^{-1}	**5.390×10^{-2}**	6.952×10^{-2}	8.950×10^{-2}
F21	平均值	**-1.015×10^{1}**	**-1.015×10^{1}**	−5.381	−8.240	−7.583	−2.729	−7.043	−9.476	−9.479
	标准差	**3.025×10^{-3}**	9.900×10^{-3}	8.770×10^{-1}	3.254	2.572	2.105	3.077	1.751	1.747
F22	平均值	**-1.022×10^{1}**	-1.040×10^{1}	−5.080	−7.978	−8.554	−3.171	−8.069	-1.011×10^{1}	−8.836
	标准差	9.700×10^{-1}	8.000×10^{-2}	4.230×10^{-1}	3.114	2.897	1.757	3.199	**9.898×10^{-4}**	2.916
F23	平均值	**-1.054×10^{1}**	**-1.054×10^{1}**	−5.120	−8.723	−5.902	−3.975	−9.012	-1.026×10^{1}	−9.690
	标准差	**3.641×10^{-3}**	1.400×10^{-2}	4.940×10^{-1}	3.122	3.020	1.836	2.864	1.481	2.221

第 2 组实验结果如表 3 所示。从中可以看出，无论在求解单峰测试函数问题上还是多峰测试函数，引入动态反向学习策略的混合算法在求解精度和稳定性都强于其他反向学习策略。这是因为动态反向学习策略将个体扩展到非对称搜索空间中，更有利于个体充分探索目标区域，增加了总体寻到全局最优解的概率，从而有效提高算法寻优性能。

表 3　DAHHO 与不同类型反向学习混合算法的基准函数测试结果

Table 3　Benchmark function results of DAHHO and different types of opposition-based learning hybrid algorithms

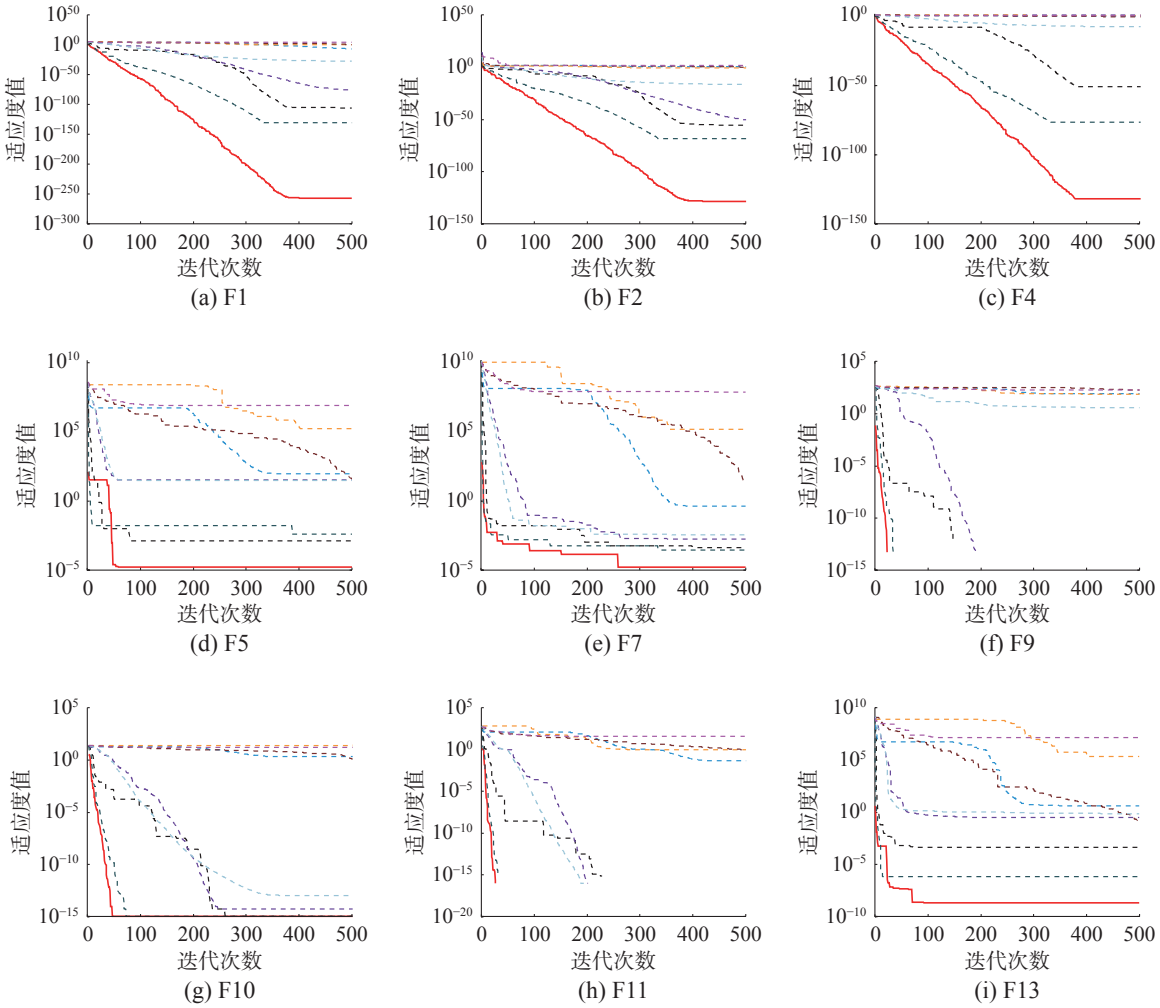
函数	统计值	DAHHO	OBLAHHO	ROBLAHHO	QOBLAHHO
F2	平均值	**1.82×10^{-120}**	5.93×10^{-119}	1.75×10^{-118}	4.76×10^{-119}
	标准差	**4.85×10^{-120}**	3.23×10^{-118}	8.86×10^{-112}	2.57×10^{-118}
F3	平均值	**4.98×10^{-192}**	5.64×10^{-189}	7.76×10^{-190}	8.19×10^{-173}
	标准差	**0.00**	0.00	0.00	0.00
F4	平均值	**1.46×10^{-117}**	1.75×10^{-115}	7.57×10^{-116}	1.69×10^{-117}
	标准差	**7.86×10^{-117}**	8.42×10^{-111}	4.09×10^{-113}	5.49×10^{-117}
F6	平均值	**1.87×10^{-5}**	1.96×10^{-5}	3.42×10^{-5}	6.35×10^{-6}
	标准差	**3.72×10^{-5}**	3.84×10^{-5}	7.64×10^{-5}	9.73×10^{-6}
F7	平均值	**8.15×10^{-5}**	8.53×10^{-5}	8.90×10^{-5}	1.14×10^{-4}
	标准差	**7.42×10^{-5}**	7.03×10^{-5}	8.28×10^{-5}	1.33×10^{-4}
F8	平均值	**-1.23×10^{4}**	-1.23×10^{4}	-1.22×10^{4}	-1.22×10^{4}
	标准差	**1.28×10^{3}**	1.38×10^{3}	1.47×10^{3}	1.52×10^{3}
F13	平均值	**1.12×10^{-5}**	2.42×10^{-5}	1.18×10^{-5}	4.17×10^{-4}
	标准差	**2.12×10^{-5}**	4.35×10^{-5}	1.67×10^{-5}	5.92×10^{-4}
F19	平均值	**−3.86**	−3.86	−3.86	−3.86
	标准差	**3.45×10^{-3}**	7.20×10^{-3}	3.82×10^{-3}	4.36×10^{-3}

续表 3

函数	统计值	DAHHO	OBLAHHO	ROBLAHHO	QOBLAHHO
F21	平均值	**-1.02×10^{1}**	-9.98	-1.01×10^{1}	-9.98
	标准差	**3.02×10^{-3}**	9.30×10^{-1}	5.93×10^{-3}	9.30×10^{-1}
F23	平均值	**-1.05×10^{1}**	-1.05×10^{1}	-1.05×10^{1}	-1.05×10^{1}
	标准差	**3.64×10^{-3}**	5.50×10^{-3}	8.70×10^{-3}	1.15×10^{-2}

5.3　收敛性分析

本文借助各算法求解基准测试函数的迭代曲线来验证 DAHHO 的收敛性。图 3 展示了各算法求解部分基准测试函数的收敛曲线。由图中可以看出, 在求解单峰函数 F1、F2 和 F4 时, MVO、SSA、SCA 和 GA 均过早收敛到局部最优解, 而 DAHHO 随着迭代次数的增加, 收敛精度不断提高, 在迭代初期便超过了所有对比算法, 在迭代末期收敛到更接近理论最优值的全局最优值。对于函数 F5, 可以看出 DAHHO 在迭代初期收敛精度便强于对比算法, 可知引入动态反向学习策略成功提升算法收敛速度。对于 F7, DAHHO 算法的收敛曲线呈阶梯状下降, 证明了本文所提算法均能跳出不同程度的局部最优解, 也证明了 HHO 不错的开发能力。在函数 F9 和 F11, AO、HHO 与 WOA 均收敛到理论最优值, 但收敛速度不及 DAHHO。对于函数 F10, 仅有 DAHHO、AO 和 HHO 收敛到全局最优值, 但 AO 和 HHO 的收敛速度不及 DAHHO。对于函数 F13, WOA、GWO、AO 与 HHO 均陷入不同程度的局部最优值, MVO 虽能不断跳出, 但收敛精度太低, DAHHO 在收敛前期就表现出优秀的寻优性能, 具有较强的避免早熟收敛能力。对于函数 F15、F22 与 F23, DAHHO 均保持了最快的收敛速度和最佳的搜索精度。综上所述, 混合算法有效利用了 AO 和 HHO 的优势特性, 在基准测试函数上表现出更佳的优化性能, 同时借助动态反向学习策略, 提升了种群的初始化性能, 为后续种群搜索奠定了基础, 证实了 DAHHO 的收敛性。



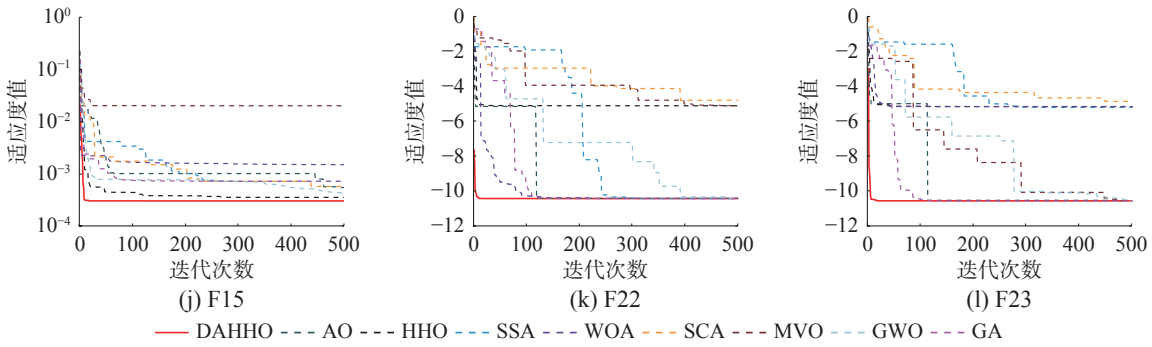


图 3　各种算法收敛曲线

Fig. 3　Convergence curves of various algorithm

5.4　Wilcoxon 秩和检验

上述仿真中，仅采用了平均值与标准差进行评价，为了进一步验证改进算法有效性，本文采用 Wilcoxon 秩和检验进行统计检验[21]。显著性水平设置为 5%，若 p 值小于 5%，则拒绝零假设，说明两种算法之间具有显著性差异，否则说明两种算法性能相差不大。Wilcoxon 秩和检验 p 值结果如表 4 所示，因为算法无法与自身对比，故表中不再列出 DAHHO 的 p 值，其中 p>0.05 已用粗体展示，NaN 表示数据无效，即实验样本数据相同，算法性能相当。

表 4　各算法 Wilcoxon 秩和检验结果

Table 4　Wilcoxon rank sum test results of each algorithm

函数	AO	HHO	SSA	WOA	SCA	MVO	GWO	GA
F1	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F2	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F3	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F4	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F5	2.628×10^{-2}	1.356×10^{-4}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F6	3.691×10^{-3}	3.357×10^{-5}	5.760×10^{-4}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F7	4.450×10^{-2}	3.615×10^{-2}	3.392×10^{-6}	7.477×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F8	2.329×10^{-5}	4.903×10^{-2}	3.392×10^{-6}	2.329×10^{-5}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	4.143×10^{-6}
F9	NaN	NaN	6.866×10^{-7}	4.923×10^{-2}	6.866×10^{-7}	6.866×10^{-7}	6.646×10^{-7}	6.866×10^{-7}
F10	NaN	NaN	6.866×10^{-7}	9.954×10^{-7}	6.866×10^{-7}	6.866×10^{-7}	5.902×10^{-7}	6.866×10^{-7}
F11	NaN	NaN	6.866×10^{-7}	$\mathbf{3.507\times10^{-1}}$	6.866×10^{-7}	6.866×10^{-7}	3.841×10^{-2}	6.866×10^{-7}
F12	2.253×10^{-2}	3.230×10^{-3}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F13	1.282×10^{-2}	1.404×10^{-3}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}
F14	$\mathbf{2.808\times10^{-1}}$	4.902×10^{-2}	1.611×10^{-3}	$\mathbf{2.290\times10^{-1}}$	$\mathbf{5.897\times10^{-1}}$	2.823×10^{-3}	6.700×10^{-4}	1.102×10^{-3}
F15	6.152×10^{-6}	$\mathbf{3.401\times10^{-1}}$	3.392×10^{-6}	4.795×10^{-3}	3.392×10^{-6}	3.392×10^{-6}	$\mathbf{1.249\times10^{-1}}$	1.140×10^{-2}
F16	3.392×10^{-6}	$\mathbf{5.202\times10^{-1}}$	1.140×10^{-2}	4.450×10^{-2}	3.392×10^{-6}	3.392×10^{-6}	3.392×10^{-6}	2.739×10^{-6}
F17	4.143×10^{-6}	4.800×10^{-2}	3.275×10^{-6}	1.620×10^{-3}	3.392×10^{-6}	$\mathbf{5.615\times10^{-1}}$	2.253×10^{-2}	6.866×10^{-7}
F18	3.392×10^{-6}	$\mathbf{2.808\times10^{-1}}$	6.152×10^{-6}	1.892×10^{-4}	6.152×10^{-6}	2.329×10^{-5}	1.935×10^{-5}	2.668×10^{-6}
F19	1.866×10^{-3}	2.134×10^{-1}	3.392×10^{-6}	1.249×10^{-2}	3.230×10^{-3}	1.356×10^{-4}	$\mathbf{7.716\times10^{-1}}$	1.563×10^{-6}
F20	4.197×10^{-2}	3.401×10^{-2}	4.225×10^{-4}	5.452×10^{-3}	$\mathbf{1.140\times10^{-2}}$	4.806×10^{-5}	9.662×10^{-5}	2.622×10^{-4}
F21	$\mathbf{1.711\times10^{-1}}$	3.392×10^{-6}	9.709×10^{-2}	9.662×10^{-5}	3.392×10^{-6}	2.019×10^{-2}	1.150×10^{-2}	$\mathbf{1.968\times10^{-1}}$
F22	2.019×10^{-2}	3.192×10^{-6}	6.709×10^{-4}	1.330×10^{-5}	3.392×10^{-6}	4.150×10^{-2}	$\mathbf{6.187\times10^{-1}}$	6.115×10^{-4}
F23	1.099×10^{-5}	3.392×10^{-6}	4.150×10^{-4}	4.143×10^{-6}	3.392×10^{-6}	1.057×10^{-2}	4.795×10^{-3}	5.553×10^{-6}

由表 4 可知，大部分 p 值均小于 5%，表明 DAHHO 算法优越性在统计检验上是显著的。可以认为 DAHHO 算法相比于对比算法，表现出更好的寻优能力。

通过分析算法收敛精度、收敛曲线与统计性检验，可以得出结论：基于两种算法的优势特性、

本文改进算法 DAHHO 的全局探索能力、局部开发能力与避免早熟收敛能力均得到增强, 此外加入动态反向学习有效提升了算法的收敛速度, 表现出更优秀的收敛性能及稳定性。

6　工程设计优化问题

为了验证 DAHHO 对现实工程问题的优化性能, 本文引入管柱设计问题和汽车碰撞设计问题对算法进行测试。实验参数设置 5.1 节所述相同。

6.1　管柱设计问题

管柱设计问题[22]是一个常用的工程设计问题。模型如图 4 所示。该问题试图使管柱设计成本最小化, 其需要优化的变量有两个, 分别是直径 d 与厚度 t。数学公式为

$$\begin{cases} f(d,t)=9.8dt+2d \\ g_1=\dfrac{P}{\pi dt\sigma_y}-1\leqslant 0, \\ g_2=\dfrac{8PL^2}{\pi^3 Edt(d^2+t^2)}-1\leqslant 0 \\ g_3=\dfrac{2.0}{d}-1\leqslant 0, \\ g_4=\dfrac{d}{14}-1\leqslant 0, \\ g_5=\dfrac{0.2}{t}-1\leqslant 0, \\ g_6=\dfrac{t}{0.8}-1\leqslant 0 \end{cases}$$

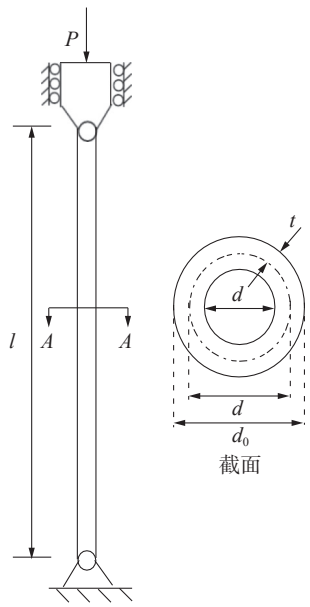


图 4　管柱设计问题

Fig. 4　Tubular column design problem

表 5 列出了 DAHHO 及对比算法求解管柱设计问题的实验结果。通过数据可以看出, DAHHO 相比于 WOA 有较大提升。相比于其他算法, DAHHO 也有不同程度的提升。图 5 展示了各算法求解管柱设计问题收敛曲线, 可以看出 DAHHO 具有较快的收敛速度。总体来看, DAHHO 在求解管柱设计问题时能够提供优秀的解决方案。

表 5　各算法求解管柱设计问题优化结果

Table 5　Optimization results of each algorithm for tubular column design

算法	d	t	结果
DAHHO	**5.4512**	**0.2920**	**26.5313**
AO	5.4652	0.2934	26.6788
HHO	5.4998	0.2894	26.6285
SSA	5.4779	0.2906	26.5887
WOA	6.1227	0.2599	27.8744
SCA	5.523	0.2887	26.7019
MVO	5.4847	0.2909	26.6370
GWO	5.4536	0.2919	26.5375
GA	5.374	0.3092	27.0627

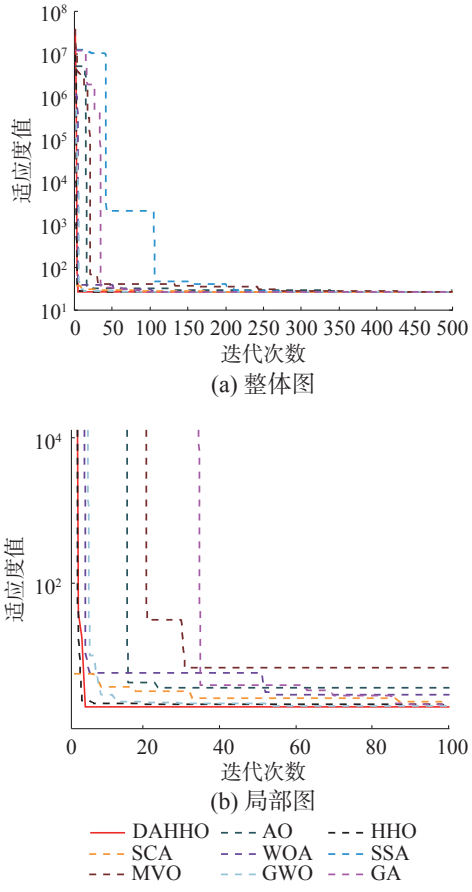


图 5　各算法求解管柱设计问题收敛曲线

Fig. 5　Convergence curve of each algorithm for tubular column design problem

6.2　汽车碰撞设计问题

汽车碰撞设计问题是一个经典工程设计问题, 与我们的日常生活息息相关[23]。设计模型如图 6 所示。该问题对汽车侧面进行撞击, 若不断

增强汽车侧面耐撞性能，则汽车重量也随之增加，从而提高汽车日常使用成本。因此，汽车碰撞设计问题试图使汽车安全性能与总重量达到一定的平衡。本问题需要最小化一个具有 11 个设计变量的目标函数，分别是：B 柱内板厚度 x_1、B 柱加强筋厚度 x_2、地板内侧厚度 x_3、横梁厚度 x_4、门梁厚度 x_5、车门腰线加固 x_6、车顶纵梁厚度 x_7、B 柱内件材质 x_8、内板内侧材料 x_9、屏障高度 x_{10} 和障碍物撞击位置 x_{11}，具体数学模型为

目标函数：$f(x)=1.98+4.90x_1+6.67x_2+6.98x_3+4.01x_4+1.78x_5+2.73x_7$，

约束条件：

$g_1(x)=1.16-0.371\,7\,0.009\,31x_2x_4-0.009\,31x_2x_{10}-0.484x_3x_9+0.013\,43x_6x_{10}\leqslant 1$，

$g_2(x)=0.261-0.015\,9x_1x_2-0.188x_1x_8-0.019x_2x_7+0.014\,4x_3x_5+0.000\,875\,7x_5x_{10}+0.080\,405x_6x_9+0.001\,39x_8x_{11}+0.000\,015\,75x_{10}x_{11}\leqslant 0.32$，

$g_3(x)=0.214+0.008\,17x_5-0.131x_1x_8-0.070\,4x_1x_9+0.030\,99x_2x_6-0.018x_2x_7+0.020\,8x_3x_8+0.121x_3x_9-0.003\,64x_5x_6+0.000\,771\,5x_5x_{10}-0.000\,535\,4x_6x_{10}+0.001\,21x_8x_{11}\leqslant 0.32$，

$g_4(x)=0.074-0.061x_2-0.163x_3x_8+0.001\,232x_3x_{10}-0.166x_7x_9+0.227x_2^2\leqslant 0.32$，

$g_5(x)=28.98+3.818x_3-4.2x_1x_2+0.020\,7x_5x_{10}+6.63x_6x_9-7.7x_7x_8+0.32x_9x_{10}\leqslant 32$，

$g_6(x)=33.86+2.95x_3+0.179\,2x_{10}-5.057x_1x_2-11.0x_2x_8-0.021\,5x_5x_{10}-9.98x_7x_8+22.0x_8x_9\leqslant 32$，

$g_7(x)=46.36-9.9x_2-12.9x_1x_8+0.110\,7x_3x_{10}\leqslant 32$

$g_8(x)=4.72-0.5x_4-0.19x_2x_3-0.012\,2x_4x_{10}+0.009\,325x_6x_{10}+0.000\,191x_{11}^2\leqslant 4$

$g_9(x)=10.58-0.674x_1x_2-1.95x_2x_8+0.020\,54x_3x_{10}-0.019\,8x_4x_{10}+0.028x_6x_{10}\leqslant 9.9$

$g_{10}(x)=16.45-0.489x_3x_7-0.843x_5x_6+0.043\,2x_9x_{10}-0.055\,6x_9x11-0.000\,786x_{11}^2\leqslant 15.7$

变量范围：$0.5\leqslant x_1\sim x_7\leqslant 1.5, x_8, x_9\in(0.192, 0.345)$，$-30\leqslant x_{10}, x_{11}\leqslant 30$。

表 6 列出了各算法求解汽车碰撞设计问题所得实验结果，可以看出 DAHHO 得到的解决方案是最佳的。同时图 7 展示了各算法求解该问题的收敛曲线，可以看出 DAHHO 具有较快的收敛速度和跳出局部极值能力，说明本文算法 DAHHO 对于求解这类工程问题具有良好的性能。

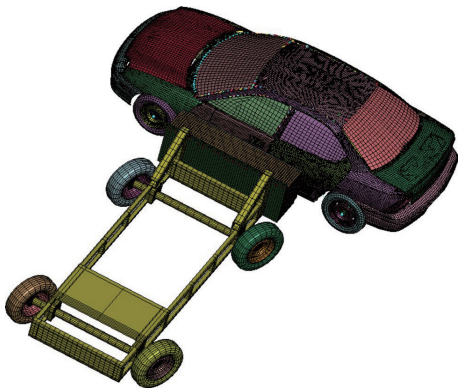


图 6　汽车碰撞设计问题

Fig. 6　Car crash design problem

表 6　各算法应用汽车碰撞设计问题优化结果

Table 6　Algorithms applied to the optimization results of car crush design problem

算法	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	结果
DAHHO	**0.5046**	**1.2507**	**0.5000**	**1.1138**	**0.5006**	**0.5009**	**0.5004**	**0.3427**	**0.2768**	**3.8389**	**6.9421**	**22.9975**
AO	0.5901	1.2008	0.5189	1.1813	0.5137	0.7646	0.5853	0.3450	0.1920	0.8966	2.0208	23.7403
HHO	0.5000	1.3342	0.5000	1.1407	0.5483	0.5000	0.5000	0.2000	0.3052	1.7723	8.2050	23.7233
SSA	0.5000	1.2329	0.5000	1.1635	0.5000	1.2465	0.5000	0.3439	0.2696	0.4011	0.3779	23.0528
WOA	0.5000	1.3311	0.5000	1.2461	0.6878	0.5000	0.5526	0.1920	0.1920	0.2747	1.2092	24.5159
SCA	0.5000	1.3338	0.5000	1.5000	0.5000	0.5083	0.5000	0.1920	0.3450	0.1834	18.8321	25.0711
MVO	0.5000	1.2390	0.5000	1.1503	0.5000	0.8584	0.5023	0.3447	0.1986	1.5950	3.4810	23.0468
GWO	0.5000	1.2416	0.5000	1.1408	0.5000	0.5639	0.5003	0.3450	0.2956	2.0066	5.1117	23.0202
GA	0.5907	1.3087	0.5767	1.1444	1.0940	0.5000	0.9016	0.1920	0.2401	0.2869	8.1420	26.6149

通过对以上 2 种不同类型的经典工程问题进行测试，充分说明了本文改进算法 DAHHO 在求解此类工程问题上具有较好的应用潜力，能够提供优良可信的解决方案。

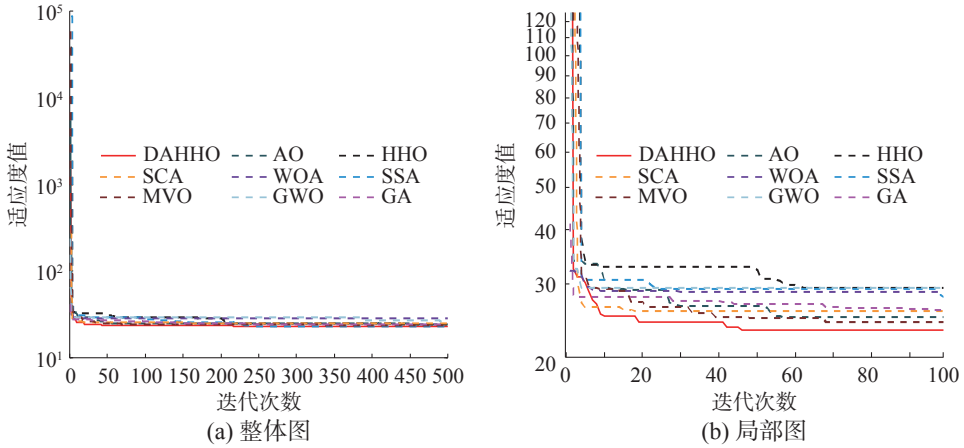


图 7　各算法求解汽车碰撞设计问题收敛曲线

Fig. 7　Convergence curve of each algorithm for car crush design problem design problem

7　结束语

本文针对 AO 和 HHO 两种新型优化算法的优势互补特点，提出了融合动态反向学习的阿奎拉鹰与哈里斯鹰混合优化算法。DAHHO 保留了原 AO 算法较强的全局探索能力，同时将 HHO 较强的局部开发能力融入 AO 的局部位置更新中，使算法有效避免局部最优。此外，初始化阶段引入动态反向学习策略，增加初始化种群多样性，提高算法的收敛速度。仿真实验部分对 23 个基准测试函数及 2 个工程设计问题进行了验证。同时，在混合算法的基础上引入其他经典反向学习策略进行对比，结果表明引入动态反向学习的混合算法 DAHHO 的寻优性能更佳，在整体上也明显优于 GWO、WOA 和 SSA 等经典 MAs，具有更佳的收敛速度、收敛精度、鲁棒性和工程适用性，为解决约束工程问题提供了一种行之有效的计算方法。未来的研究考虑将 DAHHO 应用到图像多阈值分割或特征选择中，并结合实际问题作进一步研究。

参考文献：

[1] 贾鹤鸣, 李瑶, 孙康健. 基于遗传乌燕鸥算法的同步优化特征选择 [J]. 自动化学报, 2022, 48(6): 1601–1615.
JIA Heming, LI Yao, SUN Kangjian. Simultaneous feature selection optimization based on hybrid sooty tern optimization algorithm and genetic algorithm[J]. Acta automatica sinica, 2022, 48(6): 1601–1615.

[2] 贾鹤鸣, 姜子超, 李瑶. 基于改进秃鹰搜索算法的同步优化特征选择 [J]. 控制与决策, 2022, 37(2): 445–454.
JIA Heming, JIANG Zichao, LI Yao. Simultaneous feature selection optimization based on improved bald eagle search algorithm[J]. Control and decision, 2022, 37(2): 445–454.

[3] 贾鹤鸣, 姜子超, 李瑶, 等. 基于改进斑点鬣狗优化算法的同步优化特征选择 [J]. 计算机应用, 2021, 41(5): 1290–1298.
JIA Heming, JIANG Zichao, LI Yao, et al. Simultaneous feature selection optimization based on improved spotted hyena optimizer algorithm[J]. Journal of computer applications, 2021, 41(5): 1290–1298.

[4] 贾鹤鸣, 刘宇翔, 刘庆鑫, 等. 融合随机反向学习的黏菌与算术混合优化算法 [J]. 计算机科学与探索, 2022, 16(5): 1182–1192.
JIA Heming, LIU Yuxiang, LIU Qingxin, et al. Hybrid algorithm of slime mould algorithm and arithmetic optimization algorithm based on random opposition-based learning[J]. Journal of frontiers of computer science and technology, 2022, 16(5): 1182–1192.

[5] KATOCH S, CHAUHAN S S, KUMAR V. A review on genetic algorithm: past, present, and future[J]. Multimedia tools and applications, 2021, 80(5): 8091–8126.

[6] KENNEDY J, EBERHART R. Particle swarm optimization[C]//Proceedings of ICNN'95 - International Conference on Neural Networks. Perth: IEEE, 1995: 1942−1948.

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

[8] MIRJALILI S, MIRJALILI S M, HATAMLOU A. Multiverse optimizer: a nature-inspired algorithm for global optimization[J]. Neural computing and applications, 2016, 27(2): 495–513.

[9] MIRJALILI S. SCA: a sine cosine algorithm for solving optimization problems[J]. Knowledge-based systems, 2016, 96: 120–133.

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

[11] MIRJALILI S, GANDOMI A H, MIRJALILI S Z, et al.

Salp swarm algorithm: a bio-inspired optimizer for engineering design problems[J]. Advances in engineering software, 2017, 114: 163–191.

[12] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: algorithm and applications[J]. Future generation computer systems, 2019, 97: 849–872.

[13] ABUALIGAH L, YOUSRI D, ABD ELAZIZ M, et al. Aquila optimizer: a novel meta-heuristic optimization algorithm[J]. Computers & industrial engineering, 2021, 157: 107250.

[14] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE transactions on evolutionary computation, 1997, 1(1): 67–82.

[15] AL-QANESS M A A, EWEES A A, FAN Hong, et al. Modified Aquila optimizer for forecasting oil production[J]. Geo-spatial information science, 2022, 25(4): 519–535.

[16] LI Xiaoyan, MOBAYEN S. Optimal design of a PEMFC-based combined cooling, heating and power system based on an improved version of Aquila optimizer[J]. Concurrency and computation:practice and experience, 2022, 34(15): e6976.

[17] 刘小龙, 梁彤缨. 基于方形邻域和随机数组的哈里斯鹰优化算法 [J/OL]. 控制与决策, 2021. [2021−08−19]. https: //doi. org/10.13195/j. kzyjc. 2021.0478.
LIU Xiaolong, LIANG Tongying. Harris hawk optimization Algorithm based on square neighborhood and random array[J/OL]. Control and decision, 2021. [2021−08−19].https://doi.org/10.13195/j.kzyjc.2021.0478.

[18] SUN Pu, LIU Hao, ZHANG Yong, et al. An improved atom search optimization with dynamic opposite learning and heterogeneous comprehensive learning[J]. Applied soft computing, 2021, 103: 107140.

[19] HOUSSEIN E H, HUSSAIN K, ABUALIGAH L, et al. An improved opposition-based marine predators algorithm for global optimization and multilevel thresholding image segmentation[J]. Knowledge-based systems, 2021, 229: 107348.

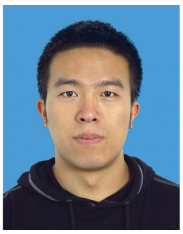
[20] FAN Qian, CHEN Zhenjian, XIA Zhanghua. A novel quasi-reflected Harris hawks optimization algorithm for global optimization problems[J]. Soft computing, 2020, 24(19): 14825–14843.

[21] TAHERI A, RAHIMIZADEH K, RAO R V. An efficient balanced teaching-learning-based optimization algorithm with individual restarting strategy for solving global optimization problems[J]. Information sciences, 2021, 576: 68–104.

[22] GANDOMI A H, YANG Xinshe, ALAVI A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems[J]. Engineering with computers, 2013, 29(1): 17–35.

[23] YILDIZ B S, PHOLDEE N, BUREERAT S, et al. Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems[J]. Engineering with computers, 2022, 38(5): 4207–4219.

作者简介：



贾鹤鸣，教授，主要研究方向为元启发式优化算法与工程应用。主持福建省自然科学基金等 10 余项。发表学术论文 60 余篇。

刘庆鑫，硕士研究生，主要研究方向为元启发式优化算法、多阈值图像分割。

刘宇翔，硕士研究生，主要研究方向为元启发式优化算法。