DOI:10.19651/j.cnki.emt.2416865

基于混合策略改进的海马优化器及其应用*

康培培[1] 薛贵军[1,2] 谭全伟[1]
(1.华北理工大学电气工程学院 唐山 063210; 2.华北理工大学智能仪器厂 唐山 063000)

摘 要: 本文针对海马优化算法收敛精度低、全局搜索和局部开发不平衡、易陷入局部最优解等问题,提出了一种基于混合策略改进的海马优化器,记作 ISHO。首先,融合灰狼优化算法的搜索特点改进海马优化算法的运动行为,使其能够在搜索空间内更有效地进行全局搜索和局部开发;然后,结合精英反向学习策略细化搜索过程,从而提高收敛精度;最后对海马优化器捕食阶段的参数进行调整,使其具有更强的自适应性避免算法过早的陷入局部最优解。将 ISHO 与其他 6 种智能优化算法在 8 种测试函数上进行比较,实验表明该算法相较于其他算法有更好的收敛速度、收敛精度和稳定性。将改进的海马优化算法应用到解决工程约束问题上,进一步证明改进算法的实用性。

关键词: 海马优化算法;灰狼优化算法;精英反向学习策略;参数调整
中图分类号: TP391.9;TN03 **文献标识码**: A **国家标准学科分类代码**: 520.1040

Based on the improved sea-horse optimization algorithm with hybrid strategy and its applications

Kang Peipei[1] Xue Guijun[1,2] Tan Quanwei[1]
(1. School of Electrical Engineering, North China University of Science and Technology, Tangshan 063210, China;
2. Intelligent Instrument Factory of North China University of Science and Technology, Tangshan 063000, China)

Abstract: This paper addresses the issues of low convergence accuracy, imbalance between global and local search, and the tendency to get stuck in local optima in the Sea-horse Optimizer. An Improved Sea-horse Optimizer based on a hybrid strategy, denoted as ISHO, is proposed. Firstly, the search characteristics of the Grey Wolf Optimizer are integrated to improve the movement behavior of the SHO, enabling more effective global and local searches within the search space. Then, an elitism and reverse learning strategy is incorporated to refine the search process and enhance convergence accuracy. Finally, adjustments are made to the parameters of the predation phase of the SHO to give it stronger adaptability, avoiding premature convergence to local optima. The ISHO is compared with six other intelligent optimization algorithms on eight test functions. Experimental results show that the proposed algorithm has better convergence speed, accuracy, and stability compared to the other algorithms. Applying the improved seahorse optimization algorithm to solve engineering constraint problems further proves the practicality of the improved algorithm.

Keywords: sea-horse optimizer;grey wolf optimizer;elite opposition-based learning;parameter adjustment

0 引 言

海马优化算法(sea-horse optimizer, SHO)是由 Zhao 等[1]于 2022 年 9 月提出的一种新型元启发式算法,主要模拟了海马的运动行为、捕食行为和繁殖行为。相较于其他智能优化算法,SHO 具有参数少、易操作、应用范围广等特点。然而,不可避免的是现有的海马优化算法也存在一些需要改进的方面。首先,现有的 SHO 在搜索过程中过早地收敛到局部最优解,这限制了其全局搜索能力,存在早熟收敛风险;其次,在处理某些复杂的优化问题时,现有的 SHO 表现不够出色,例如在处理高维度、多约束、动态变化等问题时,存在着收敛速度慢、收敛精度低、容易陷入局部最优解等问题。随着实际应用场景的不断变化,对优化算法的收敛精度、适用性等方面提出了更高的要求,需要对优化算法进行改进以提升其性能,从而适应更广泛的应用需求。

收稿日期:2024-09-11
*基金项目:河北省自然科学基金(E2020209121)项目资助

针对上述问题，众多学者提出了不同的改进方法。Özbay[2]于 2023 年提出了基于混沌映射的改进海马优化算法，用 10 种不同的混沌映射对 SHO 的捕食阶段进行改进，使其参数产生混沌值而非随机值，降低了原始海马优化算法陷入局部最优的概率，提高了全局收敛速度。赵建萍[3]于 2023 年提出一种基于辛格尔混沌映射和失败者放逐的海马算法，用辛格尔混沌映射生成初始海马种群并在捕食阶段引入失败者放逐策略，动态繁殖策略用于动态影响父代和母代的权重，提高了算法收敛速度和寻优精度上均有较大提升。Houssein 等[4]于 2024 年将反向学习策略融入到海马优化器的捕食行为中，并在每个优化周期结束时使用了贪婪选择策略。通过增强算法的搜索能力，避免陷入局部最优，提升最优解的质量和多样性。王森和赵健[5]提出了一种混合正余弦算法的改进海马优化器，引入了变异过程和参数调整，提高了算法的搜索效率和解决复杂问题的能力。舒奕彬等[6]提出了基于 Tent 混沌映射和逃逸能量调控策略的改进型海马优化算法用于 PID 参数优化。

将不同的元启发式算法融合是一种常见的改进方法，众多学者也对此做出了多种尝试，得到了较好的效果。葛知著等[7]于 2021 年提出将黄金正弦算法的黄金正弦算子和飞蛾扑火优化算法的飞蛾扑火算子混合引入到乌鸦搜索算法中进行位置更新，提高了算法后期的局部收敛精度。张琳等[8]于 2022 年提出了一种多策略改进的麻雀搜索算法。融合了差分进化算法的差分和交叉操作，提高了麻雀搜索算法跳出局部最优解的能力。周晓华等[9]于 2023 年采用黄金正弦算法对秃鹰算法进行改进用于变压器故障诊断。王一诺等[10]于 2024 年提出了基于北方苍鹰算法的位置更新公式改进金豺优化算法勘探阶段的思想，增强了种群多样性和全局探索能力。

本文提出了一种融合灰狼优化算法、精英反向学习策略和参数调整的改进型海马优化算法，即 ISHO。首先，在结合灰狼优化算法头狼引导狼群位置更新捕食的思想改进海马的运动行为，提高海马的收敛速度，平衡算法的全局搜索和局部搜索；然后，应用精英反向学习策略使海马根据精英海马的位置进行位置更新，提高算法的稳定性和算法精度；最后，使用带有正弦函数特性的参数改进海马捕食行为中的步长参数，具有更快的收敛速度以提高算法的开发能力。通过基准测试函数以及工程设计问题证明了改进算法的有效性及实用性。

1　海马优化算法

1.1　种群初始化

海马优化器从总体种群初始化开始，假设每一只海马代表问题搜索解空间中的一个候选解，则整个海马种群（以 Seahorse 命名）可表示为：

$$Seahorse=\begin{bmatrix} x_1^1 & \cdots & x_1^{Dim} \\ \vdots & \ddots & \vdots \\ x_{pop}^1 & \cdots & x_{pop}^{Dim} \end{bmatrix} \tag{1}$$

其中，Dim 表示问题的维度，pop 表示种群大小，即海马个体的数量。

每个解在特定问题的下界和上界之间随机生成，分别用 LB 和 UB 表示。搜索空间 $[LB,UB]$ 中第 i 个 X_i 的表达式为：

$$\begin{cases} X_i=\left[x_i^1 \quad \cdots \quad x_i^{Dim}\right] \\ x_i^j=r\times(UB^j-LB^j)+LB^j \end{cases} \tag{2}$$

其中，r 为 $[0,1]$ 中的随机值。x_i^j 表示第 i 个个体的第 j 维。i 为 $1\sim Dim$ 之间的正整数，j 为 $[1,Dim]$ 之间的正整数。LB^j 和 UB^j 表示优化问题的第 j 个变量的下界和上界。

1.2　海马的运动行为

海马的不同运动模式大致遵循正态分布 $rand(0,1)$，将 $r_1=0$ 作为分界点，一半用于局部搜索，另一半用于全局搜索。所以运动可以分为以下两种情况：当正态随机值 r_1 位于分界点右侧（$r_1>0$）时，模拟海马随着海中漩涡的螺旋运动，主要实现 SHO 的局部搜索。海正沿着螺旋运动向精英个体 X_{elite} 移动，采用 $levy$ 飞行来模拟海马的运动步长，这种螺旋运动方式不断改变旋转角度，扩大了当前局部解的邻域。当（$r_1\leqslant 0$）时，模拟海马随海浪漂移作用下的布朗运动，实现 SHO 的全局搜索。因此，海马运动行为下的位置更新方式为：

$$X_{new}^1(t+1)=\begin{cases} X_i(t)+levy(\lambda)((X_{elite}(t)-X_i(t))xyz+X_{elite}(t)), & r_1>0 \\ X(t)+rand\times l\beta_t(X_i(t)-\beta_tX_{elite}), & r_1\leqslant 0 \end{cases} \tag{3}$$

式中：$x=\rho\cos\theta$，$y=\rho\sin\theta$，$z=\rho\theta$ 分别表示螺旋运动下的坐标 (x,y,z) 的三维分量，其中 $\rho=ue^{\theta v}$ 表示由对数螺旋常数 u 和 v 定义的长度。θ 为 $[0,2\pi]$ 之间的随机值。$levy(z)$ 为 $levy$ 飞行分布函数，由式计算。l 为常系数，β_t 是布朗运动的随机游走系数，本质上是一个服从标准正态分布的随机值。$levy(z)=s\times\dfrac{w\times\sigma}{|k|^{\frac{1}{\lambda}}}$ (4)

式中：λ 取 $[0,2]$ 之间的随机数（本文设为 $\lambda=1.5$）。s 值为 0.01。w 和 k 是介于 $[0,1]$ 之间的随机数。σ 由式（5）计算。

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

1.3　海马的捕食行为

海马捕食浮游动物和小型甲壳类动物有两种结果：成功和失败。考虑到海马捕食成功率在 90% 以上，由随机数 r_2 区分两种结果并将临界值设为 0.1。如果 $r_2>0.1$，则意味着海马的捕食成功，反之则捕食失败。捕食行为的数学表达式为：

$$X_{new}^{2}(t+1)=$$
$$\begin{cases}\alpha(X_{elite}-rand\times X_{new}^{1}(t))+(1-\alpha)X_{elite}, & r_2>0.1\\ (1-\alpha)(X_{new}^{1}(t)-rand\times X_{elite})+\alpha X_{new}^{1}(t), & r_2\leqslant 0.1\end{cases} \tag{6}$$

其中,$x_{new}^{1}(t)$ 表示海马在迭代 t 次时移动后的新位置,r_2 为[0,1]之间的随机数。α 用来调整海马狩猎猎物的移动步长,由式 (7)计算,其中 T 最大迭代次数。

$$\alpha=\left(1-\frac{t}{T}\right)^{\frac{2t}{T}} \tag{7}$$

1.4 海马的繁殖行为

在繁殖过程中雄性海马扮演着孕育后代的角色,SHO 算法采纳了一种策略,选取适应度值最高的个体,其中一半作为“父本”,余下的另一半作为“母本”。具体角色分配的数学表达式为:

$$\begin{cases}fathers=X_{sort}^{2}\left(1:\dfrac{pop}{2}\right)\\ mothers=X_{sort}^{2}\left(\dfrac{pop}{2}+1:pop\right)\end{cases} \tag{8}$$

其中,x_{sort}^{2} 表示适应度值按升序排列的所有 X_{new}^{2}。$fathers$ 和 $mothers$ 分别表示雄性和雌性种群。雄性和雌性随机交配以产生新的后代。假设每对海马只繁殖一个子代,其表达如下:

$$X_i^{offspring}=r_3X_i^{father}+(1-r_3)X_i^{mother} \tag{9}$$

其中,r_3 为[0,1]之间的随机数。i 是范围为[1,pop/2]的正整数。X_i^{father} 和 X_i^{mother} 分别从雄性种群和雌性种群中随机选择个体。

2 海马优化算法的改进

2.1 灰狼优化算法

灰狼优化算法[11](grey wolf optimizer, GWO)模拟了灰狼在自然界中的社会结构,它的社会等级可分为:α 首领狼,即当前找到的最佳解;β 狼,为 α 之后的次优解;δ 狼,为第三优解;ω 狼,即其他解。灰狼通过以下公式模拟包围猎物的行为:

$$D=|CX_p(t)-X(t)| \tag{10}$$
$$X(t+1)=X_p(t)-A\times D \tag{11}$$
$$A=2ar_3-a \tag{12}$$
$$C=2r_3 \tag{13}$$
$$a=2-\frac{2t}{T} \tag{14}$$

式中:D 表示灰狼与猎物之间的距离,$Xp(t)$为猎物(即最优解)当前的位置向量,$X(t)$为灰狼的当前位置,$X(t+1)$为灰狼下一个时刻的位置。A 和 C 用于调节探索和开发行为。由 α、β、δ 三只头狼来引导其他狼群更新个体位置,数学表达式如下:

$$\begin{cases}D_\alpha=|C_1\times X_\alpha-X|\\ D_\beta=|C_2\times X_\beta-X|\\ D_\delta=|C_3\times X_\delta-X|\end{cases} \tag{15}$$

$$\begin{cases}X_1=X_\alpha-A_1\times D_\alpha\\ X_2=X_\beta-A_2\times D_\beta\\ X_3=X_\delta-A_3\times D_\delta\end{cases} \tag{16}$$

$$X(t+1)=\frac{X_1+X_2+X_3}{3} \tag{17}$$

式中:D_α、D_β、D_δ 分别表示 3 只头狼与其他狼之间的距离,X_1、X_2、X_3 为中间量,式 (17)为改进后海马在运动阶段的最终位置更新公式,$X(t+1)$为灰狼个体在 3 只头狼引导后的位置。

GWO 算法原理简单,操作方便。本文利用GWO 算法这种多个领头狼相互配合引导位置更新的思想对 SHO 算法的运动行为进行改进,将 α、β、δ 分别设置为适应度最好的 3 只海马的位置,将 X 换成海马优化算法中的精英海马个体 X_{elite},使得算法在全局搜索阶段具有更强的探索能力。动态参数 A 和 C 根据迭代次数逐步收敛到最优解,使得算法能在全局搜索和局部搜索之间找到平衡,更快地逼近最优解。

2.2 精英反向学习策略

精英反向学习[12]结合了精英策略和反向学习策略的思想,是反向学习策略衍生出的策略,通过精英个体的信息引导整个种群的学习过程。在每一次迭代优化过程中,算法选出最优秀的个体作为精英个体,其他个体的位置会根据精英个体的位置进行调整,以提高最优解的质量。位置更新公式为:

$$X_i^{new}=X_i+LR(X_{elite}-X_i) \tag{18}$$
$$LR=0.9e^{-\frac{t}{10T}} \tag{19}$$

式中:X_i 是第 i 个个体的当前位置,X_i^{new} 是更新后的个体位置。LR 为学习率,由式(19)给出。LR 在优化过程中随迭代次数逐渐减小,有助于算法在优化后期保持稳定,避免过度调整,提高算法的稳定性和可靠性。

2.3 参数调整策略

在捕食阶段,海马捕食猎物成功与否均受参数 α 的影响。它用以调整海马狩猎猎物的移动步长,因此 α 对算法的位置更新具有重要角色。合理调整算法的搜索步长是帮助算法逃离局部最优的有效方法[13],本文使用参数 γ 替换 α,由式 (20)给出。γ 在初期迅速减小,提高了算法的探索能力,防止过早收敛到局部最优解。参数 α 和参数 γ 随迭代过程的变化如图 1 所示。

$$\gamma=\sin\left(\frac{pi}{2}\times\left(\frac{T-t}{T}\right)^4\right) \tag{20}$$

改进后海马的捕食行为:

$$X_{new}^{2}(t+1)=$$
$$\begin{cases}\gamma(X_{elite}-rand\times X_{new}^{1}(t))+(1-\gamma)X_{elite}, & r_2>0.1\\ (1-\gamma)(X_{new}^{1}(t)-rand\times X_{elite})+\gamma X_{new}^{1}(t), & r_2\leqslant 0.1\end{cases} \tag{21}$$

2.4 改进海马优化算法流程图

改进后的海马优化算法流程图如图 2 所示。

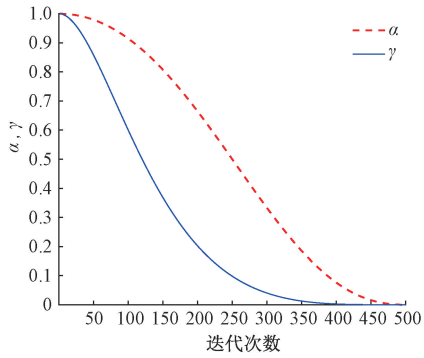

图 1　α、γ 随迭代变化的对比曲线

Fig. 1　Comparison curves of α and γ as they vary with iterations

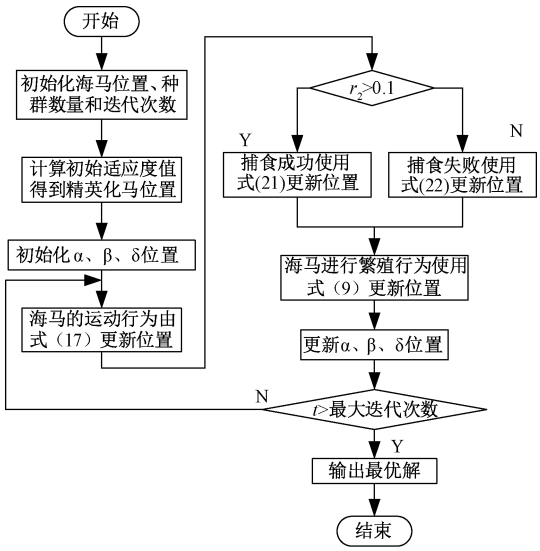

图 2　算法流程图

Fig. 2　algorithm flow char

3　实　　验

3.1　测试函数和实验参数设置

为了证实改进后的海马优化器的有效性，本文选取了 8 个广泛应用于智能优化算法性能测试的标准测试函数进行寻优求解，如表 1 所示，其中 F1～F5 为单峰函数，F6～F8 为多峰函数。在测试中，所有算法都采用相同的参数设置：维度为 30，最大迭代次数为 500 次，种群规模为 30。为了保持每个算法的竞争公平性和客观性，所有的算法在每个实验中都独立运行 30 次。记录最终平均适应度值、标准差和最优值。实验环境为：Intel(R) Core(TM) i5-9300H CPU，主频率 2.40 GHz，内存为 16 GByte，操作系统为 Windows10 64 位。所有测试均在同一环境下进行。

3.2　算法对比与分析

将改进的海马优化算法(ISHO)与 GWO、粒子群优化算法[14](particle swarm optimizer，PSO)、鲸鱼优化算法[15](whale optimization algorithm，WOA)、龙格库塔优化算法[16](RUNge kutta optimizer，RUN)、SHO 和文献[6]改进的海马优化算法(本文记为 TESHO)进行对比实验。记录 7 个算法在每个测试函数上训练得到的平均值、标准差和最优值，如表 2 所示。7 个算法在各测试函数上的收敛曲线如图 3 所示，其中横坐标为迭代次数，纵坐标为每个函数上的实际最优值，收敛曲线表示算法在当前迭代次数下搜索到的最优值。

在图 3 中，从单峰函数 F1、F2、F4 的仿真结果可以看出 ISHO 算法的收敛速度、收敛精度和稳定性均优于原始 SHO、TESHO 和其他优化算法。在 F3 测试函数的仿真结果中，虽然 ISHO 算法后期收敛速度变慢但前期收敛速度依然是最快的且最后仍能达到理论最优值。依然能够仿真

表 1　测试函数

Table 1　Test function

函数名称	表达式	定义域	f_{min}
Sphere	$F_1(x)=\sum_{i=1}^{n}x_i^2$	[−100,100]	0
Schwefel 2.22	$F_2(x)=\sum_{i=0}^{n}\lvert x_i\rvert+\prod_{i=0}^{n}\lvert x_i\rvert$	[−10,10]	0
Shifted Schwefel's Problem 1.2	$F_3(x)=\sum_{i=1}^{d}(\sum_{j=1}^{i}x_j)^2$	[−100,100]	0
Schwefel 2.21	$F_4(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	[−100,100]	0
Quartic	$F_5(x)=\sum_{i=1}^{n}ix_i^4+random[1,0)$	[−1.28,1.28]	0
Rastrigin	$F_6(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i+10)]$	[−5.12,5.12]	0
Ackley	$F_7(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)+\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	[−32,32]	0
Shifted Rotated Griewank's	$F_8(x)=\frac{1}{4\ 000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[−600,600]	0

表 2 测试结果对比

Table 2 Comparison of test results

函数	算法	平均值	标准差	最优值
F1	GWO	$1.103\ 5\times10^{-27}$	$1.553\ 3\times10^{-27}$	$1.557\ 6\times10^{-27}$
	PSO	2.681 9	1.223 4	5.347 9
	WOA	$9.356\ 4\times10^{-73}$	$3.453\ 2\times10^{-72}$	$5.173\ 3\times10^{-79}$
	RUN	4.794×10^{-165}	0	$5.008\ 8\times10^{-170}$
	SHO	$6.898\ 7\times10^{-141}$	$3.484\ 9\times10^{-140}$	$2.255\ 5\times10^{-146}$
	TESHO	**0**	**0**	**0**
	ISHO	**0**	**0**	**0**
F2	GWO	$1.313\ 9\times10^{-16}$	$8.492\ 5\times10^{-17}$	$3.187\ 8\times10^{-16}$
	PSO	4.274 3	0.9493 7	4.680 1
	WOA	$5.578\ 3\times10^{-52}$	$1.767\ 4\times10^{-51}$	$2.234\ 3\times10^{-54}$
	RUN	$1.038\ 1\times10^{-94}$	$5.335\ 4\times10^{-94}$	$9.770\ 4\times10^{-101}$
	SHO	$5.532\ 2\times10^{-78}$	$1.676\ 1\times10^{-77}$	$3.005\ 8\times10^{-80}$
	TESHO	$1.156\ 7\times10^{-175}$	0	$2.622\ 7\times10^{-178}$
	ISHO	**$1.141\ 3\times10^{-301}$**	**0**	**0**
F3	GWO	$1.751\ 9\times10^{-5}$	$3.675\ 7\times10^{-5}$	$1.204\ 1\times10^{-5}$
	PSO	187.705	38.527 6	162.571 3
	WOA	43 749.909 8	18 316.905 3	35 822.655 7
	RUN	$4.816\ 2\times10^{-150}$	$2.554\ 1\times10^{-149}$	$1.027\ 9\times10^{-158}$
	SHO	$2.391\ 3\times10^{-8}$	$1.222\ 1\times10^{-97}$	$2.009\ 4\times10^{-98}$
	TESHO	**0**	**0**	**0**
	ISHO	**$1.022\ 7\times10^{-279}$**	**0**	**0**
F4	GWO	8.507×10^{-7}	$1.087\ 3\times10^{-6}$	$7.180\ 1\times10^{-7}$
	PSO	1.944 6	$2.726\ 6\times10^{-1}$	1.941 1
	WOA	48.023	30.348 9	64.733 2
	RUN	$2.186\ 2\times10^{-77}$	$1.062\ 9\times10^{-76}$	$2.366\ 3\times10^{-83}$
	SHO	$1.685\ 7\times10^{-56}$	$6.077\ 1\times10^{-56}$	$3.224\ 8\times10^{-57}$
	TESHO	$6.127\ 3\times10^{-178}$	0	$1.818\ 7\times10^{-176}$
	ISHO	**$3.238\ 8\times10^{-277}$**	**0**	**$8.367\ 9\times10^{-298}$**
F5	GWO	$1.739\ 8\times10^{-3}$	$1.037\ 6\times10^{-3}$	$2.053\ 1\times10^{-3}$
	PSO	20.145 8	13.961 4	25.314 2
	WOA	$4.039\ 7\times10^{-3}$	$4.589\ 4\times10^{-3}$	$4.362\ 8\times10^{-3}$
	RUN	$2.783\ 7\times10^{-4}$	$2.770\ 7\times10^{-}$	3.168×10^{-4}
	SHO	1.204×10^{-4}	$9.767\ 7\times10^{-5}$	$4.860\ 1\times10^{-5}$
	TESHO	$1.158\ 2\times10^{-4}$	$1.760\ 4\times10^{-4}$	$4.018\ 5\times10^{-5}$
	ISHO	**$4.047\ 5\times10^{-5}$**	**$4.354\ 2\times10^{-5}$**	**$1.313\ 2\times10^{-5}$**
F6	GWO	4.441 4	3.407 3	7.154 1
	PSO	174.502 9	28.810 2	175.289 7
	WOA	$3.789\ 6\times10^{-15}$	$2.075\ 6\times10^{-14}$	0
	RUN	**0**	**0**	**0**
	SHO	**0**	**0**	**0**
	TESHO	**0**	**0**	**0**
	ISHO	**0**	**0**	**0**

续表 2

Continuation table 2

函数	算法	平均值	标准差	最优值
F7	GWO	$9.459\ 1\times10^{-14}$	$1.962\ 1\times10^{-14}$	$8.570\ 9\times10^{-14}$
	PSO	2.709 5	0.520 32	3.260 9
	WOA	$3.404\ 7\times10^{-15}$	$2.301\ 2\times10^{-15}$	$7.549\ 5\times10^{-15}$
	RUN	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
	SHO	$3.641\ 5\times10^{-15}$	1.084×10^{-15}	$3.996\ 8\times10^{-15}$
	TESHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
	ISHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
F8	GWO	$3.640\ 9\times10^{-3}$	7.849×10^{-3}	0
	PSO	$1.334\ 3\times10^{-1}$	$5.168\ 8\times10^{-2}$	$1.238\ 7\times10^{-1}$
	WOA	$9.863\ 1\times10^{-3}$	$5.402\ 2\times10^{-2}$	0
	RUN	**0**	**0**	**0**
	SHO	**0**	**0**	**0**
	TESHO	**0**	**0**	**0**
	ISHO	**0**	**0**	**0**

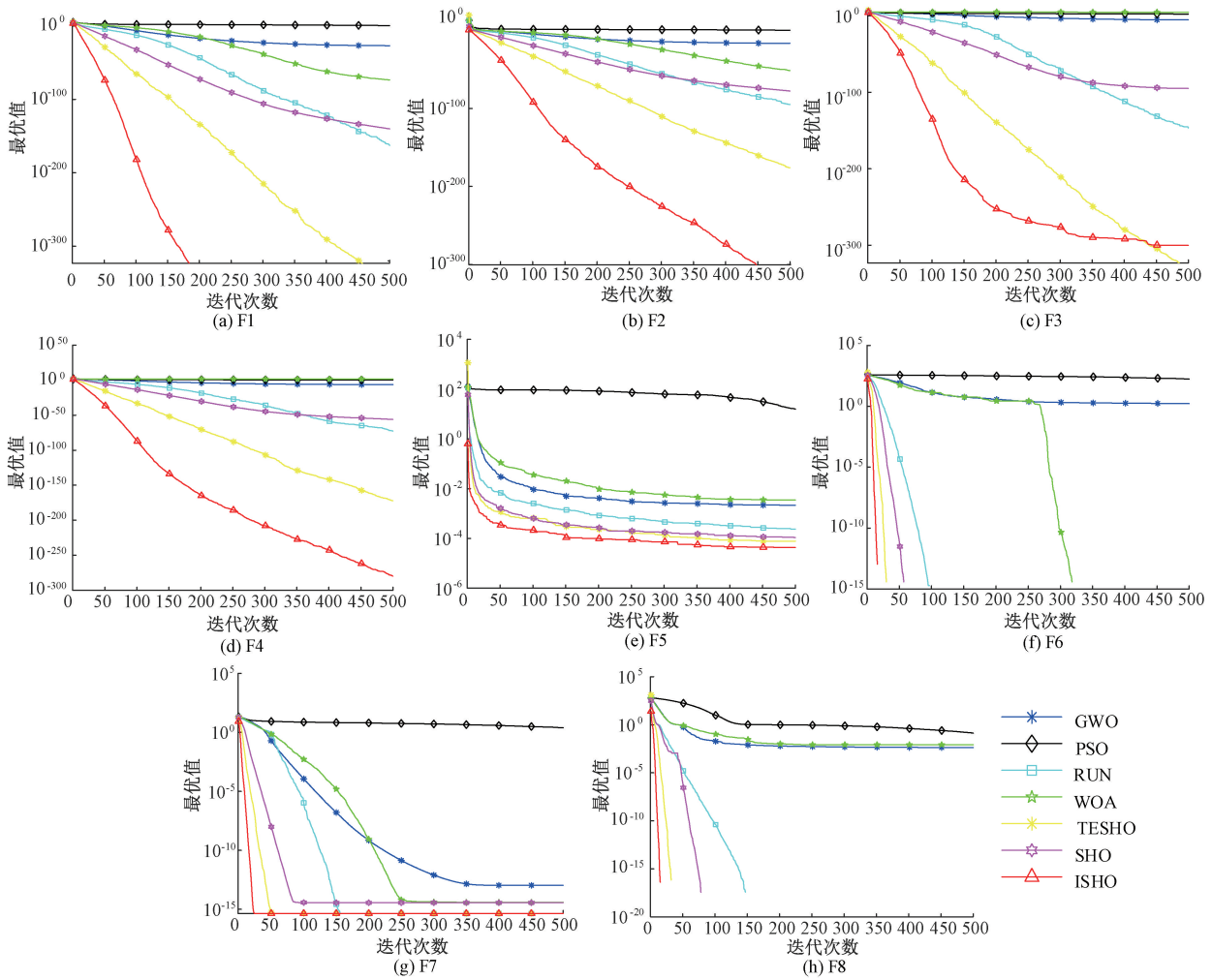

图 3　不同算法收敛曲线

Fig. 3　Convergence curves of different algorithms

结果在多数测试函数中 ISHO 都能够达到理论最优值。F5 测试函数中的巡游结果也是最好的。在多峰函数 F6、F7、F8 的仿真结果中,ISHO 算法的仿真结果最好,大约迭代了 30 次以内便找到了函数的理论最优值。在测试函数 F5、F6、F8 的仿真结果中,虽然 TESHO 与 ISHO 的收敛精度基本一致,但 ISHO 的收敛速度仍然由于 TESHO。

平均值反映了在实验中算法找到最优值的平均水平,平均值越小,说明算法整体上在多次实验中能够找到更小的目标值;标准差反映了算法的波动性和离散程度,标准差越小,意味着算法的结果在多次实验中波动较小,表现出更好的稳定性;最优适应度值更是反映了算法的性能,可以评估这些算法在搜索空间中找到最优解的质量。通过表 2 可以看出,在 7 种测试函数中,ISHO 算法的平均值,标准差和最优值在 7 种优化算法中是最小的,其中 F1、F2、F3、F6、F8 中均达到理论最优值。在处理单峰函数 F4、F5 时,ISHO 算法虽然没有达到理论最优值,但与其他 6 种算法相比 ISHO 算法的寻优值最小且从平均值和标准差可以看出 ISHO 算法性能更加稳定。在多峰测试函数 F6、F7、F8 中 ISHO 表现依然具有显著优势。

3.3 秩和检验

为了进一步验证文中提出改进算法的显著性,本节使用 Wilcoxon 秩和检验进行统计分析,比较 ISHO 与其他优化算法在的显著性水平下的差异性[17]。设定显著差异为 0.05,ISHO 算法与 GWO、PSO、WOA、RUN、SHO 和 TESHO 算法在对应函数上进行秩和检验的结果如表 3 所示。其中,P 值代表检验结果,S 值代表显著性判断结果。若 P 值小于 0.05,表示两种优化算法之间存在显著差异,此时 S 值为"1";若 P 值大于 0.05,则两种优化算法之间无显著差异,此时 S 值为"0";当 P 值为"NaN"时,表示算法在比较中所有值基本相同,导致无法进行有效的统计检验计算。由表 3 可知,除了函数 F8 中个别检验数据,ISHO 与 6 种算法在对应函数上的检验 P 值均小于 0.05,表明 ISHO 在 0.05 显著性水平上具有显著差异,体现了 ISHO 算法的有效性,具有更好的寻优能力。

表 3 Wilcoxon 秩和检验结果

Table 3 Wilcoxon rank sum test results

函数		ISHO vs GWO	ISHO vs PSO	ISHO vs WOA	ISHO vs RUN	ISHO vs SHO	ISHO vs TESHO
F1	P	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	NaN
	S	1	1	1	1	1	0
F2	P	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$
	S	1	1	1	1	1	1
F3	P	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	NaN
	S	1	1	1	1	1	0
F4	P	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	$6.386\,4\times10^{-5}$
	S	1	1	1	1	1	1
F5	P	$1.826\,7\times10^{-4}$	$18\,267\times10^{-4}$	$2.461\,3\times10^{-4}$	$18\,267\times10^{-4}$	0.017 3	0.427 4
	S	1	1	1	1	1	0
F6	P	$6.294\,4\times10^{-5}$	$6.386\,4\times10^{-5}$	NaN	NaN	NaN	NaN
	S	1	1	0	0	0	0
F7	P	$6.340\,3\times10^{-5}$	$6.386\,4\times10^{-5}$	0.005 7	NaN	$1.593\,8\times10^{-5}$	NaN
	S	1	1	1	0	1	0
F8	P	0.077 9	$6.386\,4\times10^{-5}$	0.368 1	NaN	NaN	NaN
	S	0	1	0	0	0	0

3.4 改进策略分析

为了进一步分析各个策略之间的相互关系,进行了消融实验,测试改进策略的有效性。将融合灰狼优化算法改进的 SHO 记为 wSHO,将引入精英反向学习策略改进的 SHO 记为 eSHO,将进行参数调整的 SHO 记为 pSHO。将 ISHO 算法与 SHO 算法、eSHO 算法、pSHO 算法、wSHO 算法对表 1 中的测试函数进行寻优测试。每个函数的独立运行次数为 30 次,迭代次数为 500 次,测试函数维度为 30。记录测试结果如表 4 所示,函数收敛曲线图如图 4 所示。

在图 4 中,eSHO、pSHO、wSHO 算法比原始 SHO 算法的收敛速度更快,结合 3 种改进策略的 ISHO 算法的收敛速度更是得到有效地提升。从表 4 的数据中可以看出,在 F1、F2、F6、F8 四个测试函数中,ISHO 均达到了理论最优值。其他测试函数中虽然 ISHO 未达到最优值,但相较于 SHO 和其他 3 种改进策略相比,ISHO 收敛速度和寻优精度明显更优。在 F6、F7、F8 三个测试函数中 5 个算法的寻优能力都很强。但在迭代次数相同的条件下,ISHO 具

表 4　SHO、CSHO、PSHO、WSHO、ISHO 寻优结果比较

Table 4　Comparison of optimization results for SHO, CSHO, PSHO, WSHO and ISHO

函数	算法	平均值	标准差	最优值
F1	SHO	$5.201\ 9\times10^{-140}$	$1.624\ 5\times10^{-139}$	$7.807\ 5\times10^{-145}$
	eSHO	**0**	**0**	**0**
	pSHO	$1.435\ 3\times10^{-222}$	0	$5.780\ 2\times10^{-251}$
	wSHO	$1.237\ 8\times10^{-306}$	0	$6.098\ 7\times10^{-309}$
	ISHO	**0**	**0**	**0**
F2	SHO	$4.241\ 7\times10^{-78}$	$7.889\ 5\times10^{-78}$	$3.773\ 5\times10^{-77}$
	eSHO	1.164×10^{-197}	0	$2.384\ 8\times10^{-212}$
	pSHO	$8.670\ 3\times10^{-124}$	$2.643\ 5\times10^{-123}$	$1.626\ 8\times10^{-122}$
	wSHO	$2.192\ 1\times10^{-153}$	$6.931\ 3\times10^{-153}$	$3.065\ 7\times10^{-164}$
	ISHO	**0**	**0**	**0**
F3	SHO	$4.175\ 2\times10^{-99}$	$1.046\ 5\times10^{-98}$	$7.396\ 7\times10^{-101}$
	eSHO	$1.661\ 4\times10^{-226}$	0	4.602×10^{-313}
	pSHO	$3.346\ 8\times10^{-80}$	$1.058\ 3\times10^{-79}$	$2.541\ 6\times10^{-110}$
	wSHO	$1.615\ 6\times10^{-150}$	$5.109\ 1\times10^{-150}$	$3.231\ 3\times10^{-149}$
	ISHO	$\mathbf{1.701\ 7\times10^{-290}}$	**0**	$\mathbf{1.096\ 8\times10^{-310}}$
F4	SHO	$6.779\ 2\times10^{-56}$	$2.088\ 1\times10^{-55}$	$1.532\ 6\times10^{-56}$
	eSHO	$2.390\ 9\times10^{-172}$	0	$2.447\ 3\times10^{-181}$
	pSHO	$4.718\ 1\times10^{-89}$	$1.155\ 9\times10^{-88}$	1.733×10^{-96}
	wSHO	$7.570\ 6\times10^{-133}$	1.708×10^{-132}	$9.391\ 8\times10^{-137}$
	ISHO	$\mathbf{8.193\ 5\times10^{-281}}$	**0**	$\mathbf{2.523\ 8\times10^{-283}}$
F5	SHO	$1.430\ 3\times10^{-4}$	$1.040\ 3\times10^{-4}$	$6.871\ 5\times10^{-5}$
	eSHO	$5.335\ 2\times10^{-5}$	$2.933\ 7\times10^{-5}$	$7.457\ 4\times10^{-5}$
	pSHO	$1.000\ 5\times10^{-4}$	$7.366\ 8\times10^{-5}$	$1.278\ 2\times10^{-5}$
	wSHO	$4.995\ 7\times10^{-5}$	$4.930\ 5\times10^{-5}$	$1.861\ 5\times10^{-4}$
	ISHO	$\mathbf{3.653\ 3\times10^{-5}}$	$\mathbf{2.952\ 9\times10^{-5}}$	$\mathbf{1.259\ 69\times10^{-5}}$
F6	SHO	**0**	**0**	**0**
	eSHO	**0**	**0**	**0**
	pSHO	**0**	**0**	**0**
	wSHO	**0**	**0**	**0**
	ISHO	**0**	**0**	**0**
F7	SHO	$3.996\ 8\times10^{-15}$	0	$3.996\ 8\times10^{-15}$
	eSHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
	pSHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
	wSHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
	ISHO	$\mathbf{4.440\ 9\times10^{-16}}$	**0**	$\mathbf{4.440\ 9\times10^{-16}}$
F8	SHO	**0**	**0**	**0**
	eSHO	**0**	**0**	**0**
	pSHO	**0**	**0**	**0**
	wSHO	**0**	**0**	**0**
	ISHO	**0**	**0**	**0**

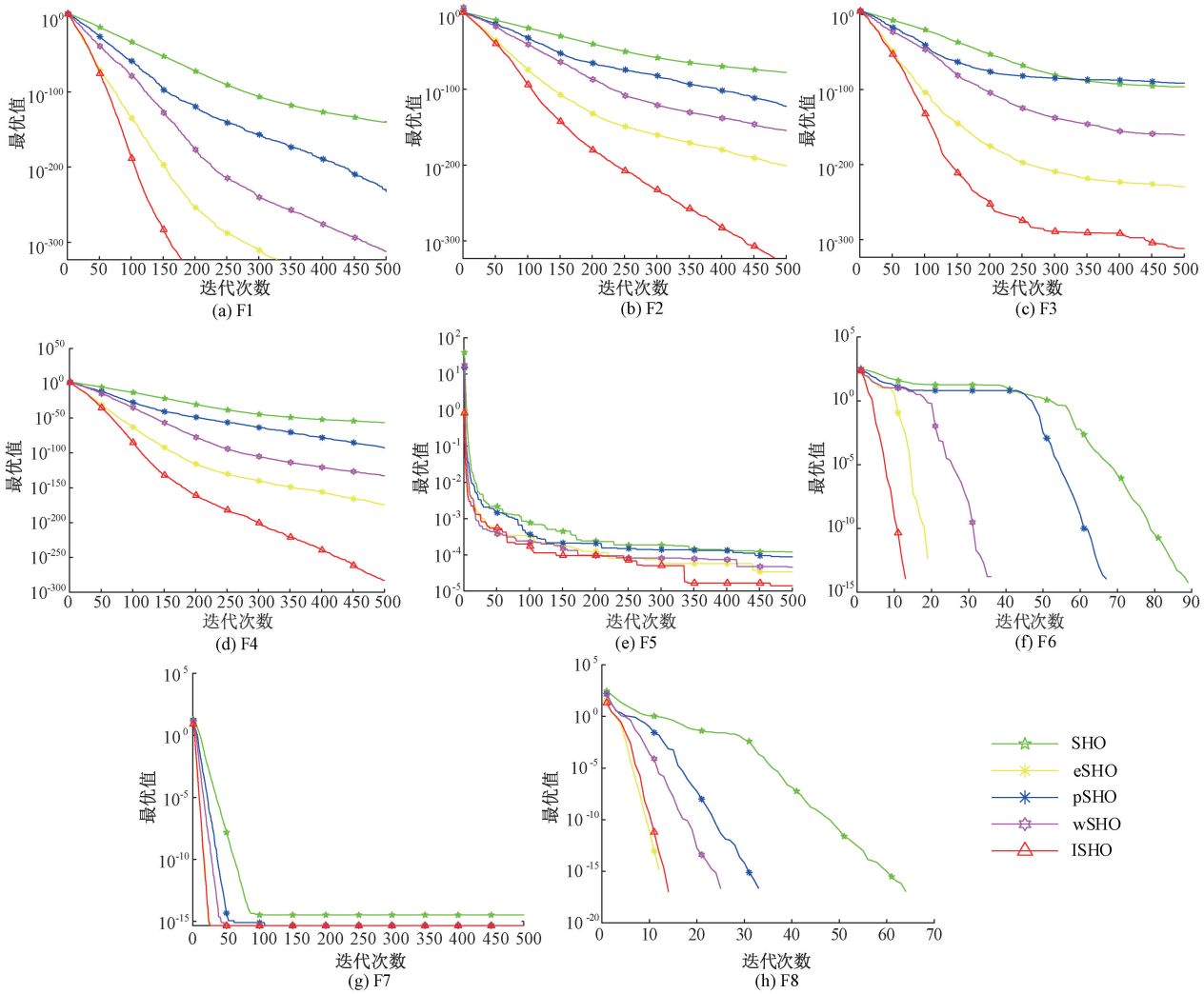

图 4 SHO、eSHO、pSHO、wSHO、ISHO 测试函数收敛曲线

Fig. 4 Convergence curves of test functions for SHO, eSHO, pSHO, wSHO, and ISHO

有更快的收敛速度。3 种单一改进策略相较于 SHO 均有提升,但次于混合策略改进的 ISHO,说明每个策略都发挥了作用且都是有效的。

3.5 工程约束优化问题求解

本文通过求解拉压弹簧设计问题评估改进的海马优化算法在实际问题求解中的优化性能[18]。拉伸/压缩弹簧设计问题是基于 3 个优化变量和 4 个约束条件,使拉压弹簧的重量最小。目标函数为弹簧重量,3 个优化变量分别为:弹簧线圈直径 d、弹簧簧圈直径 D、绕线圈数 N,其数学模型表达式如式(22)~(24)所示。

$$\min f(x_1,x_2,x_3)=(x_3+2)x_1^{\ 2}x_2 \tag{22}$$

约束条件为:

$$g_1(x)=1-\frac{x_2^{\ 3}x_3}{71\ 785x_1^{\ 4}}\leqslant 0 \tag{23}$$

$$g_2(x)=\frac{x_2(4x_2-x_1)}{12\ 566x_1^{\ 3}(x_2-x_1)}+\frac{1}{5\ 108x_1^{\ 2}}-1\leqslant 0 \tag{24}$$

变量范围为 $0.05\leqslant x_1\leqslant 2$,$0.25\leqslant x_2\leqslant 1.3$,$2.0\leqslant x_3\leqslant 15.0$。

优化后的结果如表 5 所示,实验结果可以看出 ISHO 算法在解决拉伸/压缩弹簧设计问题时表现的性能优于其余算法,这说明经过 3 种策略改进后的海马优化算法可以有效地处理有约束的问题。

表 5　拉伸/压缩弹簧设计问题各算法优化结果

Table 5　Optimization results of each algorithm for the tension/compression spring design problem

算法	设计变量			弹簧质量
	d	D	N	
GWO	0.055 6	0.467 5	6.408 7	0.012 550
PSO	0.053 4	0.345 9	13.589 7	0.013 650
WOA	0.052 3	0.379 5	11.073 9	0.012 681
RUN	0.051 9	0.348 6	10.458 9	0.011 586
SHO	0.052 1	0.345 7	10.654 4	0.011 873
TESHO	0.043 8	0.256 9	9.758 2	0.010 956
ISHO	**0.037 4**	**0.207 1**	**8.589 2**	**0.009 854**

4　结　　论

为解决海马优化算法收敛精度低，易陷入局部最优等问题，本文提出了一种结合灰狼优化算法、精英反向学习策略和参数调整的改进型海马优化器。受灰狼优化算法中头狼引导狼群进行位置更新的启发改进海马的运动行为，使算法能够在搜索空间内更有效地进行全局搜索和局部开发，结合精英反向学习策略细化搜索过程，利用学习率动态调整海马个体向最优精英个体运动，提高收敛精度，避免算法过度寻优，海马优化器捕食阶段的步长参数进行调整，提高算法的全局搜索能力，避免算法过早的陷入局部最优解。通过对比试验、秩和检验和消融实验分析，所提出的改进算法具有较好的收敛速度、收敛精度和跳出局部最优解的能力，充分证明改进算法的有效性。在解决实际工程设计问题时同样具有实用性。未来，计划采用 ISHO 算法解决其他实际问题。

参考文献

[1] ZHAO SH J, ZHANG T R, MA SH L, et al. Seahorse optimizer: A novel nature-inspired meta-heuristic for global optimization problems[J]. Applied Intelligence, 2022, 53(10): 11833-11860.

[2] ÖZBAY F A. A modified seahorse optimization algorithm based on chaotic maps for solving global optimization and engineering problems [J]. Engineering Science and Technology, an International Journal, 2023, 41:101408.

[3] 赵建萍. 基于 MATLAB 的改进海马算法[J]. 运筹与模糊学,2023, 13(4): 3462-3475.
ZHAO J P. Improved seahorse algorithm based on MATLAB [J]. Operations Research and Fuzzy Science, 2023, 13(4): 3462-3475.

[4] HOUSSEIN H E, SAAD R M, ÇELIK E, et al. An enhanced sea-horse optimizer for solving global problems and cluster head selection in wireless sensor networks[J]. Cluster Computing, 2024, 27(6): 7775-7802.

[5] 王淼,赵健. 一种混合正余弦算法的改进海马优化器[J]. 辽宁科技大学学报, 2024, 47(2): 138-145.
WANG M, ZHAO J. An improved seahorse optimizer using a hybrid sine cosine algorithm[J]. Journal of Liaoning University of Science and Technology, 2024, 47(2): 138-145.

[6] 舒奕彬,李立君,张振翩,等. 基于改进海马优化算法的 PID 参数优化[J]. 机床与液压, 2024, 52(13): 189-194.
SHU Y B, LI L J, ZHANG ZH C, et al. PID parameter optimization based on improved seahorse optimization algorithm [J]. Machine Tools and Hydraulics, 2024, 52(13): 189-194.

[7] 葛知著,张达敏,张琳娜,等. 混合策略改进的乌鸦搜索算法[J]. 计算机应用研究, 2021, 38(11): 3334-3339.
GE ZH ZH, ZHANG D M, ZHANG L N, et al. Improved crow search algorithm with hybrid strategy[J]. Computer Application Research, 2021, 38(11): 3334-3339.

[8] 张琳,汪廷华,周慧颖. 一种多策略改进的麻雀搜索算法[J]. 计算机工程与应用, 2022, 58(11): 133-140.
ZHANG L, WANG T H, ZHOU H Y. Multi strategy improved sparrow search algorithm [J]. Computer Engineering and Applications, 2022, 58(11): 133-140.

[9] 周晓华,冯雨辰,陈磊,等. 改进秃鹰搜索算法优化 SVM 的变压器故障诊断研究[J]. 电力系统保护与控制, 2023, 51(8): 118-126.
ZHOU X H, FENG Y CH, CHEN L, et al. Research on improving bald eagle search algorithm and optimizing SVM for transformer fault diagnosis[J]. Power System Protection and Control, 2023, 51(8): 118-126.

[10] 王一诺,郑焕祺,杨胜坤,等. 基于改进金豺算法的无

人机三维航迹规划[J]. 现代电子技术, 2024, 47(16): 103-109.
WANG Y N, ZHENG H Q, YANG SH K, et al. Three dimensional trajectory planning of unmanned aerial vehicles based on improved golden jackal algorithm[J]. Modern Electronic Technology, 2024, 47(16): 103-109.

[11] MIRJALILI S, MIRJALILI M S, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, DOI:10.1007/978-3-030-61111-8_5.

[12] 吴智祥,刘杰,覃涛,等. 多策略改进的精英金豺优化算法[J/OL]. 计算机工程与科学, 1-15. http://kns.cnki.net/kcms/detail/43.1258.TP.20240627.1446.004.html.
WU ZH X, LIU J, QIN T, et al. Elite golden jackal optimization algorithm with multi strategy improvement [J/OL]. Computer Engineering and Science, 1-15. http://kns.cnki.net/kcms/detail/43.1258.TP.20240627.1446.004.html.

[13] 王佳荣,周超. 改进PSO及其在机器人路径规划中的应用[J]. 计算机仿真, 2024, 41(5): 436-440,445.
WANG J R, ZHOU CH. Improving PSO and its application in robot path planning [J]. Computer Simulation, 2024, 41(5): 436-440,445.

[14] 丁同,傅晓锦. 基于改进粒子群优化算法和极限学习机的锂离子电池健康状态预测[J]. 国外电子测量技术, 2023, 42(5): 163-173.
DING T, FU X J. Lithium ion battery health status prediction based on improved particle swarm optimization algorithm and extreme learning machine [J]. Foreign Electronic Measurement Technology, 2023, 42(5): 163-173.

[15] HU H, BAI Y, XU T. A whale optimization algorithm with inertia weight [J]. WSEAS Transactions on Computers, 2016, 15:319-326.

[16] AHMADIANFAR I, HEIDARI ALI A, GANDOMI A, et al. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method[J]. Expert Systems With Applications, 2021, 181:115079.

[17] 杜晓昕,郝田茹,王波,等.基于双重随机扰动的大猩猩算法及工程应用[J/OL].北京航空航天大学学报, 1-17. https://doi.org/10.13700/j.bh.1001-5965.2023.0404.
DU X X, HAO T R, WANG B, et al. Gorilla algorithm and engineering application based on double random perturbation [J/OL]. Journal of Beihang University, 1-17. https://doi.org/10.13700/j.bh.1001-5965.2023.0404.

[18] 潘劲成,李少波,周鹏,等. 改进正弦算法引导的蜣螂优化算法[J]. 计算机工程与应用, 2023, 59(22): 92-110.
PAN J CH, LI SH B, ZHOU P, et al. Improved sine algorithm guided beetle optimization algorithm [J]. Computer Engineering and Applications, 2023, 59(22): 92-110.

作者简介

康培培(通信作者),硕士研究生,主要研究方向为复杂系统建模与控制。

E-mail:kangpeipei1218@163.com

薛贵军,高级工程师,主要研究方向为流量计领域仪器仪表和热力热电领域自动化控制系统的研究。

E-mail:xueguijun@126.com

谭全伟,硕士研究生,主要研究方向为复杂系统建模与控制。

E-mail:2431303407@qq.com