2025 年 第 25 卷 第 10 期
2025, 25(10): 04216-13
科 学 技 术 与 工 程
Science Technology and Engineering
ISSN 1671—1815
CN 11—4688/T

DOI:10.12404/j.issn.1671-1815.2404146
引用格式:刘衍平, 郑荣艳, 宋富洪, 等. 多策略改进的海马优化算法及应用[J]. 科学技术与工程, 2025, 25(10): 4216-4228.
Liu Yanping, Zheng Rongyan, Song Fuhong, et al. Multi-strategy improvedseahorse optimization algorithm and its application[J]. Science Technology and Engineering, 2025, 25(10): 4216-4228.

多策略改进的海马优化算法及应用

刘衍平[1], 郑荣艳[1*], 宋富洪[2], 廖彬[1]
(1. 贵州财经大学大数据统计学院, 贵阳 550025; 2. 贵州财经大学信息学院, 贵阳 550025)

摘 要 针对海马优化算法(seahorse optimization,SHO)存在的求解精度较低、容易早熟以及全局搜索能力不足等问题,设计了一种基于非线性惯性权重策略、改进的鲸鱼包围猎物策略以及改进的正余弦策略的多策略海马优化算法(multi-strategy seahorse optimization,MSHO)。首先,在 SHO 算法的运动行为中引入非线性惯性权重策略,以克服算法容易过早收敛的缺点;其次,将改进的鲸鱼包围猎物策略引入海马捕食成功的更新方程中,以降低算法陷入局部最优解的概率;然后,在算法的繁殖行为中引入改进的正余弦策略,以增强海马后代解的质量,进一步提升算法的全局寻优能力和稳定性。最后,为评估所提 MSHO 算法的性能,选取 SHO 算法,混沌的 SHO 算法、减法平均器算法、灰狼算法、海鸥算法、鲸鱼优化算法、粒子群算法与 MSHO 算法在 23 个基准测试函数上进行比较。实验结果表明,与其他 7 种算法相比,MSHO 算法在 20 个函数上表现出更高的收敛精度,在 16 个函数上表现出更强的稳定性。此外,为检验 MSHO 算法在工程问题上的应用能力,将算法应用于求解焊接梁、悬臂梁和压力容器设计问题。实验结果表明,相较于其他 7 种不同算法,MSHO 算法在这 3 类工程设计问题上表现出更好的搜索精度。

关键词 海马优化算法; 非线性惯性权重; 全局寻优; 测试函数; 工程问题
中图法分类号 TP301.6; **文献标志码** A

Multi-strategy Improved Seahorse Optimization Algorithm and Its Application

LIU Yan-ping[1], ZHENG Rong-yan[1*], SONG Fu-hong[2], LIAO Bin[1]
(1. College of Big Data Statistics, Guizhou University of Finance and Economics, Guiyang 550025, China;
2. College of Information, Guizhou University of Finance and Economics, Guiyang 550025, China)

[**Abstract**] In order to solve the problems of SHO (seahorse optimization), such as low accuracy, precocity and insufficient global search ability. MSHO (multi-strategy seahorse optimization) algorithm based on nonlinear inertial weight strategy, improved whale encircling strategy and improved sine and cosine strategy was MSHO designed. Firstly, the nonlinear inertia weight was introduced into the motion behavior of SHO algorithm to overcome the shortcoming that the algorithm is prone to premature convergence. Secondly, the improved strategy of whale encircling prey was introduced into the updated equation of seahorse hunting success to reduce the probability of the algorithm falling into the local optimal solution. Then, the improved sine-cosine strategy was introduced into the reproduction behavior of the algorithm to enhance the quality of the hippocampal progeny solution, and further improve the global optimization ability and stability of the algorithm. Finally, in order to evaluate the performance of the proposed MSHO algorithm, SHO algorithm, chaotic SHO algorithm, subtraction average algorithm, gray Wolf algorithm, Seagull algorithm, whale optimization algorithm, particle swarm algorithm and MSHO algorithm were compared on 23 benchmark test functions. The experimental results show that MSHO algorithm shows higher convergence accuracy on 20 functions and stronger stability on 16 functions compared with other 7 algorithms. In addition, in order to test the application ability of MSHO algorithm in engineering problems, the algorithm is applied to solve the design problems of welded beams, cantilever beams and pressure vessels. The experimental results show that MSHO algorithm has better search accuracy in these three kinds of engineering design problems than other 7 different algorithms.

[**Keywords**] seahorse optimization; nonlinear inertia weight; global optimization; test function; engineering problem

收稿日期: 2024-06-04; **修订日期**: 2025-01-16
基金项目: 国家自然科学基金(62061007);贵州省科技计划(黔科合基础-ZK[2023]一般 028,黔科合基础-ZK[2024]一般 693)
第一作者: 刘衍平(1981—),男,汉族,四川资阳人,博士,副教授。研究方向:博弈论、机器学习、优化算法等在无线资源管理中的应用。E-mail:liuyanping6@126.com。
* **通信作者**: 郑荣艳(1999—),女,汉族,贵州安顺人,硕士研究生。研究方向:数据挖掘、群智能优化算法。E-mail:z7725ry@163.com。

随着社会的飞速发展,数学、计算机科学、工程、管理等众多领域中的优化问题复杂度越来越高。因此,在面对这些复杂问题时,设计高效的优化算法变得尤为重要。群智能算法由于不受限于问题的梯度信息,并具有较强全局收敛性[1],因而在求解复杂优化问题等方面受到学者们的广泛关注。目前,学者们提出了大量的群智能算法,包括粒子群算法(particle swarm optimization, PSO)[2]、人工鱼群算法(artificial fish swarm algorithm, AFS)[3]、鲸鱼优化算法(whale optimization algorithm, WOA)[4]、灰狼算法(grey wolf optimizer, GWO)[5]、正余弦算法(sine cosine algorithm, SCA)[6]、蝴蝶优化算法(butterfly optimization algorithm, BOA)[7]、海鸥优化算法(seagull optimization algorithm, SOA)[8],海马优化算法(seahorse optimizer, SHO)[9]等。同时,这些算法在许多优化领域中也得到了大量应用,如短期风电功率预测[10],聚类优化[11],无人机航路规划[12]等。

SHO 算法是由 Zhao 等[9]提出的一种新型的群智能算法。然而,与大多数算法一样,SHO 算法也存在收敛精度较低以及逃避局部极值能力不足等问题[13-14]。为提升 SHO 算法的性能,研究学者们提出了一系列改进策略。其中,Özbay[15]在 SHO 算法中将随机值替换为 10 种混沌值,从而解决了 SHO 算法收敛速度慢和容易陷入局部最优值的缺点;舒奕彬等[16]提出了一种改进的 SHO 算法,将 Tent 映射以及逃逸能量调控策略融入 SHO 算法中,不仅加快了算法的收敛速度,还为 PID 控制系统的参数优化提供了重要的指导;Li 等[17]提出了一种改进的 SHO 算法,将 Tent 映射和 SCA 算法相结合,加快了 SHO 算法的收敛速度,并有效地提高了农业图像识别的精度;曹永娟等[18]将自适应云模型和混沌映射策略引入原 SHO 算法,成功降低了算法陷入局部最优解的风险,并显著提升了永磁同步电参数识别的准确性。这些改进的 SHO 算法在一定程度上平衡了原算法的全局利用和局部开发能力,但仍然存在一些问题,包括收敛速度慢、全局搜索能力弱等。因此,开发具有更强全局寻优能力的改进 SHO 算法具有重要现实意义。

为了进一步提升 SHO 算法的收敛准确性和全局优化性能,本文研究在原始 SHO 算法基础上设计基于非线性惯性权重策略,改进 WOA 包围猎物策略以及改进 SCA 策略的多策略海马优化算法(multi-strategy seahorse optimization, MSHO)。首先,在原 SHO 算法的运动行为中引入非线性惯性权重策略,以克服算法收敛速度过慢的缺点。这种策略利用非线性函数来动态调整惯性权重,使算法在早期迭代阶段加快搜索速度,在后期迭代阶段提高局部搜索能力,从而提升全局收敛性。其次,在原 SHO 算法的捕食行为中引入改进的 WOA 包围猎物策略,旨在防止算法陷入局部最优解。最后,将改进的 SCA 策略融入原算法的繁殖阶段,旨在有效提升算法的全局收敛性能。本文研究将 MSHO 算法应用于 23 个测试函数,以评估其鲁棒性和寻优能力。此外,还将 MSHO 算法应用于焊接梁、悬臂梁和压力容器问题,以考察其在实际工程问题中的适用性。通过这些实验,以期评估算法的全局寻优和收敛速度能力,并为实际的工程问题提供有效的解决方案。

1　海马优化算法

SHO 算法是 Zhao 等[9]借鉴了海马种群的运动、捕食和繁殖行为,而设计的一种新颖的群智能优化算法。

1.1　运动行为

海马在不同运动状态下可以采用正态随机值 $r_1 = \mathrm{randn}(0,1)$ 来调整行为。当 $r_1 > 0$ 时,海马进行螺旋运动。当 $r_1 \leqslant 0$ 时,海马进行布朗运动。两种运动的详细信息如下。

(1)螺旋运动:当 $r_1 > 0$ 时,海马的螺旋运动主要用于算法的局部勘探。在这种情况下,海马不断朝着最优个体 X_{elite} 的方向靠近,并用莱维飞行方程模拟海马的运动步长,以广泛搜索当前解空间。该过程的数学模型可表示为

$$X_{\mathrm{elite}} = \mathrm{argmin}[f(\cdot)]$$

$$X_{\mathrm{new}}^{1}(t+1) = X_i(t) + \mathrm{Levy}(\lambda)[(X_{\mathrm{elite}}(t) - X_i(t)]xyz + X_{\mathrm{elite}}(t) \tag{1}$$

式(1)中:$f(\cdot)$ 为给定问题的目标函数值;$x = \rho\cos(\theta)$、$y = \rho\sin(\theta)$ 和 $z = \rho\theta$ 分别表示螺旋运动下坐标的三维分量;$X_i(t)$ 表示第 t 次迭代下第 i 个海马个体的位置;X_{new}^{1} 表示海马进行运动行为后的新位置;$\rho = ue^{\theta v}$,其中常数 $u = 0.05$,$v = 0.05$,旋转角度 θ 为 $[0,2\pi]$ 范围内的随机数;$\mathrm{Levy}(\lambda)$ 表示莱维飞行的分布函数[19]。

$$\mathrm{Levy}(z) = s\frac{w\sigma}{|k|^{\frac{1}{\lambda}}} \tag{2}$$

式(2)中:$\lambda = 0.05$;$s = 0.01$;k、w 均为 $[0,1]$ 范围内的随机数。

$$\sigma = \frac{\Gamma(1+\lambda)\sin\frac{\pi\lambda}{2}}{\Gamma\left(\frac{1+\lambda}{2}\right)\lambda 2^{\frac{\lambda-1}{2}}} \tag{3}$$

(2)布朗运动:当 $r_1 \leqslant 0$ 时,海马的布朗运动主要体现了算法的探索能力。此时,引入布朗运动来模拟海马的另一个运动步长,其表达式为

$$X_{\text{new}}^1(t+1) = X_i(t) + \text{rand} l\beta_t [X_i(t) - \beta_t X_{\text{elite}}] \tag{4}$$

式(4)中:$l = 0.05$;rand 为随机数,rand $\in [0,1]$。

$$\beta_t = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{5}$$

结合式(1)和式(4),可以得到在第 t 次迭代时海马的新位置,其数学模型为

$$X_{\text{new}}^1(t+1) = \begin{cases} X_i(t) + \text{Levy}(\lambda)[(X_{\text{elite}}(t) - X_i(t)]xyz + X_{\text{elite}}(t), & r_1 > 0 \\ X_i(t) + \text{rand} l\beta_t \times [X_i(t) - \beta_t X_{\text{elite}}], & r_1 \leqslant 0 \end{cases} \tag{6}$$

1.2 捕食行为

在自然界中,海马捕食成功率高达 90%,充分体现了海马出色的捕食能力。因此,本文研究在 SHO 算法中设置随机数 $r_2 = 0.1$,以区分海马的两种捕食结果。具体来说,当 $r_2 > 0.1$ 时,表明海马成功接近了猎物(精英),并将以最快速度将其捕捉。当 $r_2 \leqslant 0.1$ 时,捕食失败,海马转而搜索整个空间。因此,整个捕食行为的数学模型为

$$X_{\text{new}}^2(t+1) = \begin{cases} \alpha[X_{\text{elite}} - \text{rand} X_{\text{new}}^1(t)] + (1-\alpha)X_{\text{elite}}, & r_2 > 0.1 \\ (1-\alpha)[X_{\text{new}}^1(t) - \text{rand}]X_{\text{elite}} + \alpha X_{\text{new}}^1(t), & r_2 \leqslant 0.1 \end{cases} \tag{7}$$

式(7)中:X_{new}^2 为海马进行捕食行为后的新位置;r_2 为[0,1]范围内的随机数;α 为一个线性递减函数。

$$\alpha = \left(1 - \frac{t}{T_{\max}}\right)^{\frac{2t}{T_{\max}}} \tag{8}$$

式(8)中:$T_{\max}$ 为最大迭代次数。

1.3 繁殖行为

在繁殖阶段中,根据计算得到的适应度值,将海马划分为雄性海马和雌性海马,其中适应度值较高的一半个体划分为父亲,剩余的另一半个体划分为母亲。其表达式为

$$\begin{cases} \text{fathers} = X_{\text{sort}}^2(p_{\text{op1}}), & p_{\text{op1}} \in (1,2,\cdots,p_{\text{op}}/2) \\ \text{mothers} = X_{\text{sort}}^2(p_{\text{op2}}), & p_{\text{op2}} \in (p_{\text{op2}}/2,\cdots,p_{\text{op}}) \end{cases} \tag{9}$$

式(9)中:X_{sort}^2 表示所有 X_{new}^2 进行适应度值升序后的排序结果;father 和 mother 分别为 SHO 算法中的雄性种群和雌性种群;p_{op} 为种群大小。

雄性海马和雌性海马通过随机交配方式产生后代。为简化 SHO 算法,假设每对海马仅繁殖一个后代。则第 i 个海马后代的表达式为

$$X_i^{\text{offspring}} = r_3 X_i^{\text{father}} + (1 - r_3) X_i^{\text{mather}} \tag{10}$$

式(10)中:$r_3 \in [0,1]$;X_i^{father} 和 X_i^{mother} 为从雄性种群和雌性种群中随机选择的个体,其中 i 为 $[1, p_{\text{op}}/2]$ 范围内的正整数。

由上述过程得到的种群规模为 $1.5p_{\text{op}}$。为避免算法复杂化,在最终得到的 $1.5p_{\text{op}}$ 种群中,选取适应度排名前 p_{op} 个个体作为新的种群。

2 多策略改进的海马优化算法

为提升 SHO 算法的收敛精度和全局寻优能力,在不改变 SHO 算法的框架上,本文提出基于非线性惯性权重策略、改进的 WOA 包围猎物策略以及改进的 SCA 策略的 MSHO 算法。

2.1 引入非线性惯性权重策略

根据算法当前的状态以及进展情况,非线性惯性权重能够调整参数的变化速率或幅度,从而在算法的勘探和开发过程中取得平衡。在算法中引入较大的惯性权重,使算法更注重全局搜索,以维持解的多样性和增强全局搜索能力;而引入较小的惯性权重使算法更注重局部搜索,以加速收敛到最优解[20]。在原始 SHO 算法的运动行为中,海马会逐渐收敛到种群中适应度值较好的个体,导致算法过早收敛的情况。由于 SHO 算法的运动行为在整个迭代寻优过程中是不可或缺的,故将非线性惯性权重 α_1 引入 SHO 算法运动行为中,通过迭代次数的增加使得 SHO 算法不容易过早收敛。α_1 的数学表达式为

$$\alpha_1 = 2 - \left(\frac{t}{T_{\max}}\right)^2 \tag{11}$$

式(11)中:t 为当前迭代次数。

α_1 逐渐从 2 非线性递减到 1。α_1 的变化情况如图 1 所示。

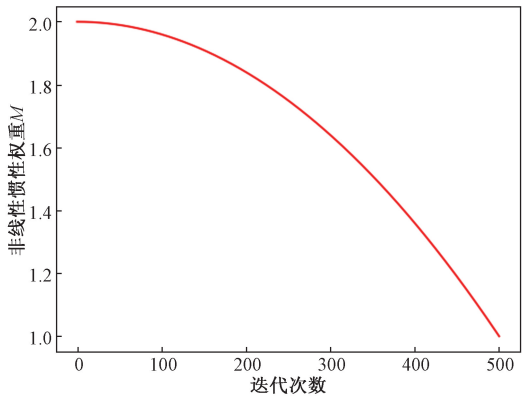


图 1 非线性惯性权重曲线图

Fig. 1 Nonlinear inertial weight curve

将式(11)代入式(1)和式(4)中,得到

$$X_{\text{new}}^{1}(t+1)=\alpha_1 X_i(t)+\text{Levy}(\lambda)[(X_{\text{elite}}(t)-X_i(t)]xyz+X_{\text{elite}}(t) \tag{12}$$

$$X_{\text{new}}^{1}(t+1)=\alpha_1 X_i(t)+\text{rand}l\beta_t[X_i(t)-\beta_t X_{\text{elite}}] \tag{13}$$

通过正态随机值 r_1 将式(12)和式(13)相结合,得到新的海马运动更新方程,其表达式为

$$X_{\text{new}}^{1}(t+1)=\begin{cases}\alpha_1 X_i(t)+\text{Levy}(\lambda)[X_{\text{elite}}(t)-X_i(t)]xyz+X_{\text{elite}}(t), & r_1>0\\ a_1 X_i(t)+\text{rand}l\beta_t\times[X_i(t)-\beta_t X_{\text{elite}}], & r_1<0\end{cases} \tag{14}$$

2.2　引入改进的 WOA 包围猎物策略

在 SHO 算法的捕食行为中,海马成功捕食的行为强调了算法的局部利用能力。此时,海马不断向着精英位置靠近,但这个过程中海马的位置更新方程较为简单,且伴随较大的随机性。这不仅影响算法的收敛精度,还容易使算法陷入局部最优解。为此本文研究将 WOA 算法的包围猎物机制引入海马成功捕食的更新方程[21]。此时,海马通过自身位置和精英(X_{elite})位置来更新其位置方程,有利于算法容易跳出局部最优。WOA 算法的包围猎物策略的表达式为

$$\begin{cases}D=|CX_{\text{elite}}(t)-X_i(t)|\\ X(t+1)=X_{\text{elite}}(t)-AD\end{cases} \tag{15}$$

式(15)中:X_i 为当前鲸鱼位置;A、C 为更新位置的辅助系数。

$$\begin{cases}A=2b\text{rand}-b\\ C=2\text{rand}\\ b=2-t\dfrac{2}{T}\end{cases} \tag{16}$$

为进一步增强 WOA 算法的局部利用能力,将权重 $M=1/2\sqrt{1+\cos(\pi t/T_{\max})}$ 引入 WOA 包围猎物机制[22],则表达式为

$$X(t+1)=X_{\text{elite}}(t)-MAD \tag{17}$$

式(17)中:M 表示非适应惯性权重随着当前迭代次数 t 的增加呈现前期缓慢下降,后期显著下降趋势;$T_{\max}$ 为最大迭代次数。

改进后的 WOA 算法的包围猎物策略,更有利于提高 SHO 算法的精度以及增强算法跳出陷入局部最优解的概率。将式(17)引入 SHO 算法中,替换原始 SHO 算法成功捕食更新方程。则改进后的海马位置更新方程为

$$X_{\text{new}}^{2}(t+1)=\begin{cases}X_{\text{elite}}(t)-MAD, & r_2>0.1\\ (1-\alpha)[X_{\text{new}}^{1}(t)-\text{rand}]X_{\text{elite}}+\alpha X_{\text{new}}^{1}(t), & r_2\leqslant 0.1\end{cases} \tag{18}$$

式(18)中:$D=|CX_{\text{elite}}(t)-X_i(t)|$ 为海马精英个体所在位置和当前鲸鱼个体所在位置的距离。

2.3　引入改进的 SCA 策略

SHO 算法的繁殖行为中,主要由父母随机交配而产生后代,这种分配方案能够避免新的解决方案过度本地化的问题。此时,通过对海马后代进行一定的变换,可进一步增强解的质量。因此,本文研究将 SCA 的正弦更新方程融入 SHO 算法的繁殖行为[23]。这种策略有利于海马后代在探索空间过程中向精英位置靠近,使得改进后的 SHO 算法具有更强的全局探索能力。SCA 算法的表达式为

$$X(t+1)=\begin{cases}X_i(t)+g_1\text{sin}g_2|g_3X_{\text{elite}}(t)-X_i(t)|, & g_4>0.5\\ X_i(t)+g_1\text{cos}g_2|g_3X_{\text{elite}}(t)-X_i(t)|, & g_4<0.5\end{cases} \tag{19}$$

式(19)中:$X_i(t)$ 为当前解的位置;$X_{\text{elite}}(t)$ 为当前最优解的位置;$g_1=(1-t/T)^{2t/T}$ 为从 1 非线性递减到 0 的值;g_2 为[0,2π]范围内的一个随机值;g_3 为[-2,2]范围内的一个随机值;g_4 为[0,1]范围内的随机值。

为使得 SCA 算法具有更强的全局探索性能,故将非线性惯性权重 $M=1/2\sqrt{1+\cos(\pi t/T_{\max})}$ 引入 SCA 的正弦更新方程中。改进后的正弦更新方程为

$$X(t+1)=MX_i(t)+g_1\text{sin}g_2|g_3X_{\text{elite}}(t)-X_i(t)| \tag{20}$$

基于 SHO 算法得到的随机交配的后代,添加位置更新方程式(20),以增强 SHO 算法的收敛准确度和全局寻优能力。其数学模型为

$$\begin{aligned}&X_{\text{elite}}^{\text{offspring}}=\text{argmin}[f(X_i^{\text{offspring}})]\\ &X_{\text{new}}^{\text{offspring}}(t+1)=MX_i^{\text{offspring}}(t)+\\ &\quad g_1\text{sin}g_2|g_3X_{\text{elite}}^{\text{offspring}}(t)-X_i{}^{\text{offspring}}(t)|\end{aligned} \tag{21}$$

式(21)中:$X_i^{\text{offspring}}(t)$ 为当前海马后代的位置;$X_{\text{elite}}^{\text{offspring}}(t)$ 为当前精英后代所在位置;$f(\cdot)$ 为给定问题的目标函数值;i 为[0,p_{op}/2]范围内的正整数。

2.4　MSHO 算法的伪代码和流程图

利用式(14)和式(18)获得海马的新位置,并通过式(10)来生成海马后代,以及利用式(21)更新海马后代的位置。所提 MSHO 算法的流程图如图 2 所示,MSHO 算的伪代码如表 1 所示。

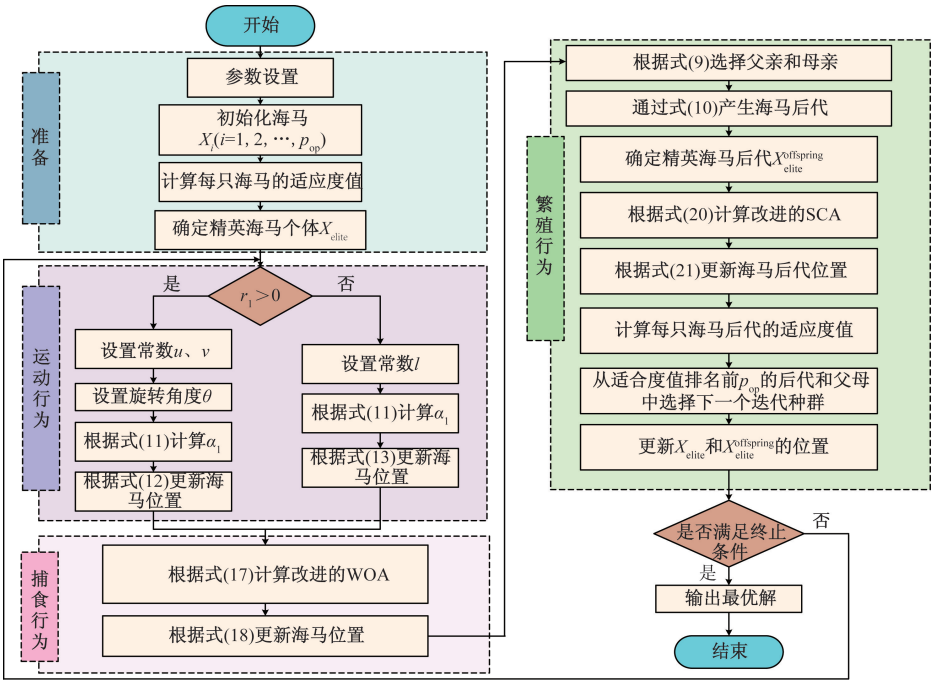


图 2 MSHO 算法的流程图

Fig. 2 Flow chart of MSHO algorithm

表 1 MSHO 算法的伪代码

Table 1 Pseudo-code ofMSHO algorithm

算法 1:MSHO 算法
输入:种群 p_{op}、最大迭代次数 T_{max}、变量维度 D_{im}
输出:最优解 X_{best}、最优适应度值 f_{best}
初始化海马个体 $X_i(i=1,2,\cdots,p_{op})$
计算每个海马适应度值
选取精英个体 X_{elite}
While $t < T_{max}$ do
  for $i=1:p_{op}$ do
    根据等式(11)计算非线性惯性权重 α_1
    if $r_1 > 0$ then
      根据等式(12)更新海马位置
    else
      根据等式(13)更新海马位置
    end if
  for $i=1:p_{op}$ do
    根据等式(17)计算改进的 WOA
    根据等式(18)更新海马位置
end for
修正变量边界
计算每个海马的适应度值
根据等式(9)选择 fathers 和 mothers
  for $i=1:p_{op}$ do
    根据式(10)计算海马后代
    确定精英后代 $X_{elite}^{offspring}$
    根据等式(20)计算改进的 SCA
    根据等式(21)更新海马后代位置
end for
修正变量边界
计算每个海马后代的适应度值
从适应度值排名前 p_{op} 的后代和父母中选择下一次迭代种群
更新 X_{elite} 和 $X_{elite}^{offspring}$ 的位置
  end while
  return X_{best}

3 结果验证分析

为有效验证所提出 MSHO 算法的性能,选取 23 个基准函数进行检验。首先,将 MSHO 算法在测试集上得到的数值结果与其他 7 种算法比较,包括 SHO 算法[9]、混沌海马优化算法(Chaotic seahorse optimization, CSHO)[15]、减法平均器算法(subtraction-average-based optimizer, SABO)[24]、GWO 算法[5]、SOA 算法[8]、WOA 算法[4]、PSO 算法[2],旨在更好地验证 MSHO 算法的开发和勘探能力。其次,通过绘制收敛曲线直观评价 MSHO 算法在局部开发和全局探索过程中的表现。最后,通过 wilcoxon 符号检进一步验证所提出的 MSHO 算法的有效性和优越性。

本次实验在 MATLAB R2022a 软件上实现,使用的操作系统为 MICROSOFT WINDOWS 11 64 位家庭版,计算机配置为 Intel(R) Core(TM) i5-12500H CPU 以及 16 GB RAM。

3.1 23 个测试函数

选取的 23 个测试函数如表 2 所示,包括 7 个单峰函数(F1 ~ F7)和 16 个多峰函数(F8 ~ F23)[9,14,15]。表 2 中,f_{min} 表示函数的最优值。单峰函数只有一个的全局最优值,可用于评估 MSHO 算法在局部开发能力方面的表现;而多峰函数拥有多个局部最优值,有助于更好地对 MSHO 算法的全局搜索能力进行评估。

7 种不同算法的参数大小如表 3 所示。为了保

证实验的公正性,实验参数设置:种群大小 30、最大迭代次数 500、试验次数 30[5,19,25]。最后,基于 30 次运行的实验结果,计算平均值和方差,以此作为评价标准来衡量算法的性能。

表 2　23 个测试函数

Table 2　23 test functions

编号	函数名称	范围	维度	f_{min}
F1	Sphere	[-100,100]	30	0
F2	Schwefel 2.22	[-10,10]	30	0
F3	Schwefel 1.2	[-100,100]	30	0
F4	Schwefel 2.21	[-100,100]	30	0
F5	Rosenbrock	[-30,30]	30	0
F6	Step	[-100,100]	30	0
F7	Quartic with noise	[-1.28,1.28]	30	0
F8	Schwefel 2.26	[-500,500]	30	-12 569.5
F9	Rastrigin	[-5.12,5.12]	30	0
F10	Ackley	[-32,32]	30	0
F11	Griewank	[-600,600]	30	0
F12	Penalized1	[-50,50]	30	0
F13	Penalized2	[-50,50]	30	0
F14	Shekel's Foxholes	[-65,65]	2	1
F15	Kowalik	[-5,5]	4	0.000 308
F16	Six-Hump Camel-Back	[-5,5]	2	-1.031 6
F17	Branin	[-5,10],[0,15]	2	0.398
F18	Goldstein-Price	[-2,2]	2	3
F19	Hartman's Family1	[0,1]	3	-3.86
F20	Hartman's Family2	[0,1]	6	-3.32
F21	Shekel's Family1	[0,10]	4	-10.15
F22	Shekel's Family2	[0,10]	4	-10.4
F23	Shekel's Family3	[0,10]	4	-10.536

表 3　不同算法参数设置

Table 3　Different algorithm parameter settings

算法	参数	值
SHO[9]	r_1,r_2	0,0.1
CSHO[15]	α	4
SABO[24]	v	[1,2]
GWO[5]	α	从 2 非线性递减到 0
SOA[8]	J_c	2
WOA[4]	α	从 2 非线性递减到 0
	r,l	[0,2],[-1,1]
PSO[2]	C_1,C_2	2,2
	w	从 0.9 非线性递减到 1

3.2　统计结果分析

如表 4 所示,给出了 MSHO 算法与其余 7 种算法在表 2 中的函数上得到的平均值(Mean)和方差(Std)。如表 3 所示,MSHO 算法在 6 个单峰函数上表现最佳均值,仅在 F7 函数上的均值较低于 CSHO、SABO 和 SHO 的均值。这说明与其余 7 种算法相比,MSHO 算法在单峰函数上表现出更优秀的局部开发能力;MSHO 算法在 14 个多峰函数上展现出最佳均值,仅在 F14 函数上的略高于 PSO 的均值,在 F23 上的均值略高于 GWO 的均值。这说明 MSHO 算法同时具备较强的全局搜索能力。从方差角度分析,在 23 个基准测试函数上,MSHO 算法仅在函数 F5、F7、F14、F16、F18、F22、F23 上的方差略高于其他 7 种算法。换句话说,在 16 个基准测试函数上,MSHO 算法具有较强的稳定性。

如图 3 所示,给出了 MSHO 算法和其余 7 种算法在部分基准函数上得到的寻优结果所绘制的收敛曲线图。在单峰函数 F1、F3 和 F6 上 MSHO 算法跳出局部最优,且在函数 F1、F3 上达到最优值。在单峰函数 F5 上,MSHO 算法与大部分算法差不多,没能及时跳出局部最优。在多峰函数 F8、F15、F18 上,MSHO 算法在 100 次迭代内就收敛到函数最优解。在多峰函数 F10、F12 上,MSHO 算法的收敛曲线更低,说明其收敛精度更高。总的来说,MSHO 算法与其他算法相比,具有较强的局部开发和全局探索性能。

为探索 MSHO 算法与其他算法性能的差异性,采用 Wlicoxon 符号检验作为工具进行检测。如表 5 所示,给出了 MSHO 算法与竞争算法在 95% 的置信区间内得到的 P 值。通过比较表 5 中数据发现:MSHO 算法在 20 个函数上的性能优于 SHO,CSHO,SABO,在 F9 和 F11 上的性能与这 3 种算法类似,但在 F7 上的性能不如它们;MSHO 算法在 23 个函数上的性能显著优于 GWO 和 SOA;MSHO 算法在 22 个函数上的性能优于 WOA,仅在函数 F11 上的性能相似于 WOA;MSHO 算法在 22 个函数上的性能优于 PSO,仅在函数 F23 上的性能差于 PSO。这侧面证明了多策略的 MSHO 算法是有效的,其性能是显著优于其他算法的。

3.3　比较 MSHO 算法 3 种策略的有效性

为更进一步评估 MSHO 算法的性能,将原始的 SHO 算法,非线性权重策略的 SHO 算法(SHO algorithm with nonlinear weight strategy, NSHO),改进 WOA 包围猎物策略的 SHO 算法(SHO algorithm based on improved WOA strategy for encircling prey, WSHO)以及改进 SCA 策略的 SHO 算法(SHO algorithm based on improved SCA strategy, SSHO)进行比较。由于文章篇幅的限制,故仅选取表 2 中具有 30 维的 13 个函数(F1 ~ F13)进行测试。同样,为确保实验公平性,设置种群大小 30,迭代次数 500,试验次数 30。如表 6 所示,给出了 MSHO、SHO、NSHO、WSHO 和 SSHO 算法在函数(F1 ~ F13)上进行 30 次实验所得的平均值(Mean)、方差(Std)和平均时间。

表 4 不同算法在 23 个函数上的测试结果

Table 4 Test results of different algorithms on 23 functions

函数	指标	MSHO	SHO	CSHO	SABO	GWO	SOA	WOA	PSO
F1	Mean	**0**	3.08×10^{-141}	1.11×10^{-149}	4.87×10^{-197}	1.50×10^{-27}	2.14×10^{-12}	2.08×10^{-73}	2.54×10^{-1}
	Std	**0**	1.38×10^{-140}	3.37×10^{-149}	0	5.09×10^{-27}	2.67×10^{-12}	5.53×10^{-73}	1.58×10^{-1}
F2	Mean	**0**	1.17×10^{-78}	6.69×10^{-81}	5.86×10^{-111}	8.05×10^{-17}	1.60×10^{-8}	1.31×10^{-47}	4.24×10^{-1}
	Std	**0**	1.83×10^{-78}	1.36×10^{-80}	1.80×10^{-110}	8.54×10^{-17}	1.45×10^{-8}	3.53×10^{-47}	1.82×10^{0}
F3	Mean	**0**	5.26×10^{-99}	1.08×10^{-113}	2.06×10^{-25}	1.50×10^{-5}	1.49×10^{-4}	4.24×10^{4}	1.93×10^{3}
	Std	**0**	2.05×10^{-98}	2.60×10^{-113}	9.21×10^{-25}	3.02×10^{-5}	4.31×10^{-4}	1.51×10^{4}	1.57×10^{3}
F4	Mean	**0**	2.91×10^{-56}	7.75×10^{-64}	1.34×10^{-77}	5.64×10^{-7}	4.37×10^{-3}	4.41×10^{1}	7.57×10^{0}
	Std	**0**	1.18×10^{-55}	2.49×10^{-63}	1.69×10^{-77}	7.70×10^{-7}	1.47×10^{-2}	3.19×10^{1}	1.47×10^{0}
F5	Mean	$\mathbf{2.26\times10^{1}}$	2.81×10^{1}	2.81×10^{1}	2.84×10^{1}	2.71×10^{1}	2.84×10^{1}	2.78×10^{1}	1.73×10^{2}
	Std	7.68×10^{0}	6.31×10^{-1}	6.38×10^{-1}	$\mathbf{3.60\times10^{-1}}$	7.51×10^{-1}	4.54×10^{-1}	4.22×10^{-1}	9.16×10^{1}
F6	Mean	$\mathbf{3.51\times10^{-4}}$	3.17×10^{0}	2.88×10^{0}	2.73×10^{0}	8.81×10^{-1}	3.13×10^{0}	3.49×10^{-1}	4.61×10^{-1}
	Std	$\mathbf{3.25\times10^{-4}}$	5.53×10^{-1}	5.51×10^{-1}	5.23×10^{-1}	3.67×10^{-1}	3.68×10^{-1}	1.84×10^{-1}	7.68×10^{-1}
F7	Mean	1.09×10^{-4}	1.03×10^{-4}	$\mathbf{9.56\times10^{-5}}$	9.61×10^{-5}	2.30×10^{-3}	3.21×10^{-3}	2.93×10^{-3}	4.38×10^{-2}
	Std	1.07×10^{-4}	7.08×10^{-5}	9.27×10^{-5}	$\mathbf{7.64\times10^{-5}}$	1.15×10^{-3}	3.56×10^{-3}	3.46×10^{-3}	1.19×10^{-2}
F8	Mean	$\mathbf{-1.26\times10^{4}}$	-6.02×10^{3}	-6.55×10^{3}	-3.18×10^{3}	-5.84×10^{-3}	-5.06×10^{3}	-1.01×10^{4}	-7.58×10^{3}
	Std	$\mathbf{3.82\times10^{0}}$	6.37×10^{2}	4.85×10^{2}	3.45×10^{2}	9.33×10^{2}	6.50×10^{2}	1.93×10^{3}	7.35×10^{2}
F9	Mean	**0**	0	0	0	3.82×10^{0}	3.25×10^{0}	2.81×10^{0}	5.14×10^{1}
	Std	**0**	0	0	0	4.22×10^{0}	5.61×10^{0}	1.54×10^{-1}	1.56×10^{1}
F10	Mean	$\mathbf{4.44\times10^{-16}}$	4.00×10^{-15}	3.76×10^{-15}	4.00×10^{-15}	9.92×10^{-14}	1.93×10^{1}	3.46×10^{-15}	6.72×10^{-1}
	Std	**0**	0	9.01×10^{-16}	0	1.73×10^{-14}	3.64×10^{0}	2.38×10^{-15}	5.90×10^{-1}
F11	Mean	**0**	0	0	0	7.06×10^{-3}	1.34×10^{-2}	0	3.93×10^{-1}
	Std	**0**	0	0	0	1.09×10^{-2}	2.09×10^{-2}	0	2.01×10^{-1}
F12	Mean	$\mathbf{1.32\times10^{-5}}$	2.49×10^{-1}	1.59×10^{-1}	2.19×10^{-1}	3.84×10^{-2}	3.38×10^{-1}	1.13×10^{-1}	3.15×10^{-1}
	Std	$\mathbf{1.36\times10^{-5}}$	7.79×10^{-2}	6.05×10^{-2}	8.59×10^{-2}	1.58×10^{-2}	1.57×10^{-1}	3.80×10^{-1}	5.14×10^{-1}
F13	Mean	$\mathbf{3.15\times10^{-4}}$	1.95×10^{0}	1.93×10^{0}	2.75×10^{0}	5.56×10^{-1}	2.05×10^{0}	5.51×10^{-1}	5.18×10^{-1}
	Std	$\mathbf{2.70\times10^{-4}}$	3.27×10^{-1}	3.65×10^{-1}	5.49×10^{-1}	2.26×10^{-1}	2.03×10^{-1}	2.75×10^{-1}	4.43×10^{-1}
F14	Mean	1.78×10^{0}	5.11×10^{0}	5.95×10^{0}	3.25×10^{0}	4.97×10^{0}	1.82×10^{0}	2.61×10^{0}	$\mathbf{1.03\times10^{0}}$
	Std	2.96×10^{0}	4.28×10^{0}	4.62×10^{0}	1.80×10^{0}	4.53×10^{0}	1.04×10^{0}	2.39×10^{0}	$\mathbf{1.81\times10^{-1}}$
F15	Mean	$\mathbf{3.40\times10^{-4}}$	5.09×10^{-4}	1.90×10^{-3}	1.27×10^{-3}	4.37×10^{-3}	1.13×10^{-3}	7.70×10^{-4}	1.36×10^{-3}
	Std	$\mathbf{1.67\times10^{-4}}$	3.57×10^{-4}	5.09×10^{-3}	2.83×10^{-3}	8.21×10^{-3}	2.84×10^{-4}	5.11×10^{-4}	3.62×10^{-3}
F16	Mean	$\mathbf{-1.03\times10^{0}}$	-1.03×10^{0}	-1.03×10^{0}	-1.02×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}	-1.03×10^{0}
	Std	1.27×10^{-10}	7.33×10^{-9}	2.30×10^{-8}	1.94×10^{-2}	1.59×10^{-8}	2.35×10^{-6}	1.09×10^{-9}	$\mathbf{6.18\times10^{-16}}$
F17	Mean	$\mathbf{3.98\times10^{-1}}$	3.98×10^{-1}	3.99×10^{-1}	5.21×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}	3.98×10^{-1}
	Std	$\mathbf{1.72\times10^{-7}}$	1.28×10^{-3}	2.33×10^{-3}	2.09×10^{-1}	1.50×10^{-6}	4.64×10^{-4}	3.67×10^{-5}	0
F18	Mean	$\mathbf{3.00\times10^{0}}$	3.00×10^{0}	3.00×10^{0}	4.17×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}	3.00×10^{0}
	Std	4.74×10^{-9}	5.16×10^{-9}	6.32×10^{-10}	2.15×10^{0}	2.35×10^{-5}	1.84×10^{-4}	2.66×10^{-4}	$\mathbf{2.31\times10^{-15}}$
F19	Mean	$\mathbf{-3.86\times10^{0}}$	-3.86×10^{0}	-3.86×10^{0}	-3.65×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.86×10^{0}	-3.84
	Std	$\mathbf{2.36\times10^{-6}}$	3.80×10^{-3}	3.51×10^{-3}	1.86×10^{-1}	2.19×10^{-3}	2.01×10^{-3}	5.09×10^{-3}	1.41×10^{-1}
F20	Mean	$\mathbf{-3.28\times10^{0}}$	-3.03×10^{0}	-3.04×10^{0}	-3.26×10^{0}	-3.26×10^{0}	-2.86×10^{0}	-3.20×10^{0}	-3.27×10^{0}
	Std	$\mathbf{5.94\times10^{-2}}$	2.04×10^{-1}	1.72×10^{-1}	8.00×10^{-2}	7.74×10^{-2}	5.84×10^{-1}	1.01×10^{-1}	6.51×10^{-2}
F21	Mean	$\mathbf{-8.96\times10^{0}}$	-5.71×10^{0}	-6.84×10^{0}	-5.04×10^{0}	-9.65×10^{0}	-3.62×10^{0}	-8.74×10^{0}	-5.57×10^{0}
	Std	$\mathbf{2.18\times10^{0}}$	2.47×10^{0}	2.47×10^{0}	2.64×10^{0}	1.56×10^{0}	4.17×10^{0}	2.52×10^{0}	3.59×10^{0}
F22	Mean	$\mathbf{-1.00\times10^{1}}$	-5.48×10^{0}	-5.05×10^{0}	-4.94×10^{0}	-1.00×10^{1}	-6.16×10^{1}	-7.53×10^{0}	-7.12×10^{0}
	Std	1.34×10^{0}	1.84×10^{0}	2.72×10^{0}	2.86×10^{-1}	$\mathbf{8.46\times10^{-4}}$	4.46×10^{0}	3.05×10^{0}	3.82×10^{0}
F23	Mean	-9.82×10^{0}	-5.29×10^{0}	-5.48×10^{0}	-4.69×10^{0}	$\mathbf{-1.01\times10^{1}}$	-8.37×10^{0}	-6.93×10^{0}	-6.96×10^{0}
	Std	1.81×10^{0}	1.88×10^{0}	2.35×10^{0}	$\mathbf{1.25\times10^{0}}$	1.81×10^{0}	4.44×10^{0}	3.28×10^{0}	3.92×10^{0}

注:加粗数值为最佳结果。

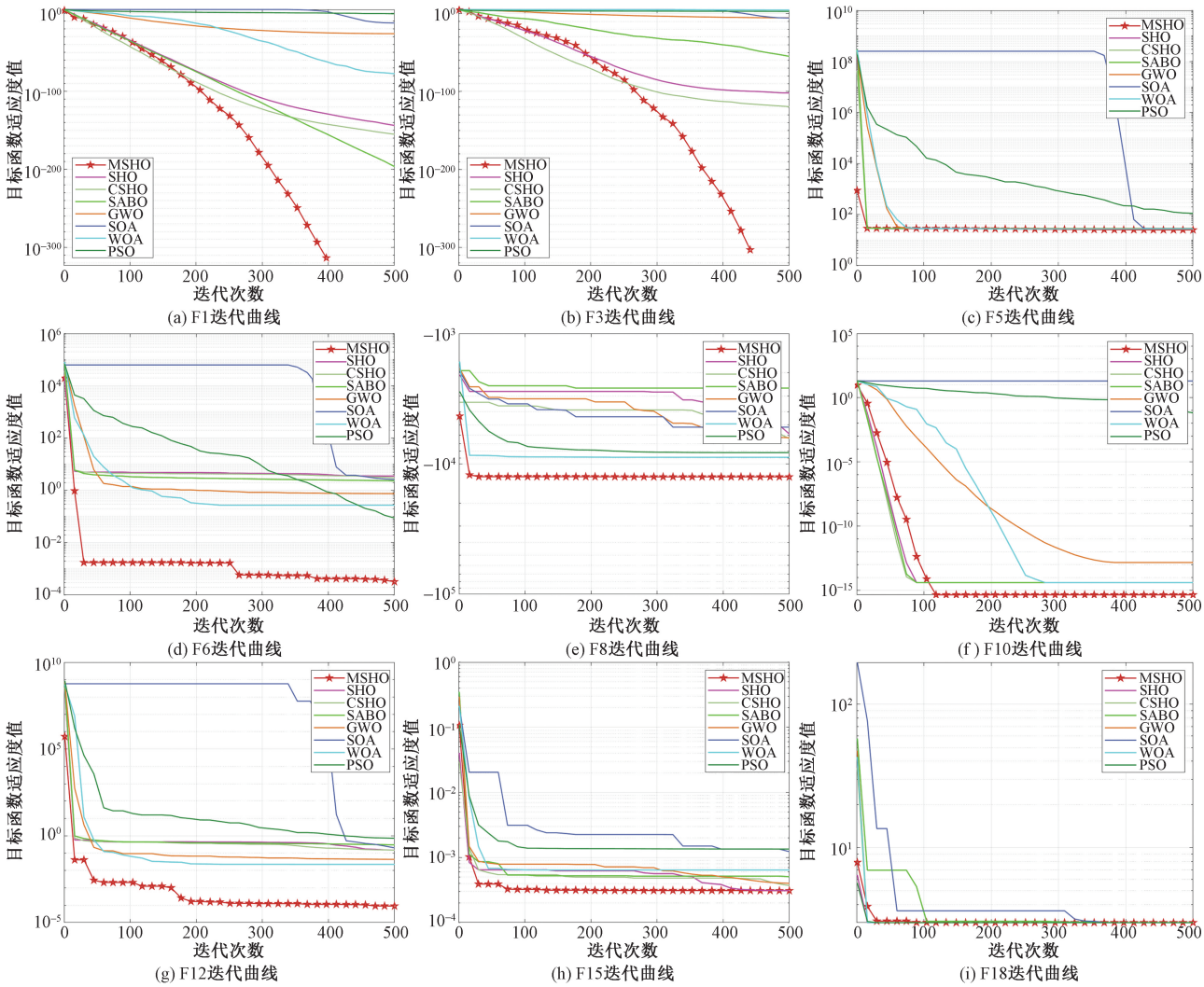


图 3　部分函数的收敛曲线图

Fig. 3　Convergence diagram of some functions

表 5　Wilcoxon 符号检验对比

Table 5　Test comparison of Wilcoxon signed-rank

函数	*P*						
	MSHO/SHO	MSHO/CSHO	MSHO/SABO	MSHO/GWO	MSHO/SOA	MSHO/WOA	MSHO/PSO
F1	1.21×10^{-12}	1.21×10^{-12}	3.02×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F5	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F6	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F7	6.63×10^{-1}	8.53×10^{-2}	9.12×10^{-2}	3.02×10^{-11}	4.50×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F8	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.50×10^{-11}	3.02×10^{-11}
F9	NaN	NaN	NaN	1.18×10^{-12}	1.21×10^{-12}	8.15×10^{-2}	1.21×10^{-12}
F10	1.17×10^{-13}	1.17×10^{-13}	1.69×10^{-14}	1.06×10^{-12}	1.21×10^{-12}	3.66×10^{-8}	1.21×10^{-12}
F11	NaN	NaN	NaN	1.10×10^{-2}	1.21×10^{-12}	NaN	1.21×10^{-12}
F12	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F13	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F14	5.37×10^{-11}	4.87×10^{-11}	2.96×10^{-11}	2.96×10^{-11}	6.76×10^{-5}	2.96×10^{-11}	3.08×10^{-11}
F15	7.66×10^{-5}	3.18×10^{-3}	5.97×10^{-9}	1.52×10^{-3}	6.52×10^{-9}	3.35×10^{-8}	4.86×10^{-3}
F16	1.36×10^{-7}	1.19×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.70×10^{-8}	1.01×10^{-11}

续表

函数	P						
	MSHO/SHO	MSHO/CSHO	MSHO/SABO	MSHO/GWO	MSHO/SOA	MSHO/WOA	MSHO/PSO
F17	4.62×10^{-10}	3.16×10^{-10}	3.02×10^{-11}	2.67×10^{-9}	3.02×10^{-11}	1.07×10^{-7}	1.21×10^{-12}
F18	6.01×10^{-8}	6.36×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.50×10^{-11}
F19	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	5.49×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.19×10^{-10}
F20	7.12×10^{-9}	3.09×10^{-6}	3.18×10^{-4}	1.24×10^{-3}	3.02×10^{-11}	1.87×10^{-5}	5.89×10^{-4}
F21	1.10×10^{-8}	2.02×10^{-8}	3.02×10^{-11}	1.17×10^{-4}	1.10×10^{-8}	6.05×10^{-7}	1.14×10^{-3}
F22	8.89×10^{-10}	4.62×10^{-10}	6.07×10^{-11}	2.68×10^{-4}	1.73×10^{-6}	3.52×10^{-7}	4.04×10^{-2}
F23	1.33×10^{-10}	3.16×10^{-10}	5.49×10^{-11}	1.07×10^{-7}	2.92×10^{-9}	5.00×10^{-9}	6.61×10^{-1}
数量	+20/≈2/-1	+20/≈2/-1	+20/≈2/-1	+23/≈0/-0	+23/≈0/-0	+22/≈1/-0	+22/≈0/-1

注：+表示 MSHO 算法的性能优于竞争算法($P<5\%$)；≈表示 MSHO 算法的性能与其竞争算法相差不大(P = NaN)，其中 NaN 为无穷大或没有值；-表示 MSHO 算法的性能差于竞争算法($P>5\%$)。

表 6 不同改进策略的测试结果

Table 6 Test results of different improvement strategies

函数	指标	MSHO	SHO	NSHO	WSHO	SSHO
F1	Mean	**0**	3.08×10^{-141}	9.38×10^{-93}	6.93×10^{-47}	1.99×10^{-101}
	Std	**0**	1.38×10^{-140}	5.14×10^{-92}	2.48×10^{-46}	9.68×10^{-117}
	时间/s	2.344	0.816 57	0.784 5	0.818 54	0.875 8
F2	Mean	**0**	1.17×10^{-78}	2.04×10^{-58}	1.16×10^{-30}	9.44×10^{-59}
	Std	**0**	1.83×10^{-78}	6.90×10^{-58}	2.45×10^{-30}	3.60×10^{-74}
	时间/s	0.792 83	0.260 09	0.198 28	0.182 12	0.247 36
F3	Mean	**0**	5.26×10^{-99}	2.29×10^{-48}	5.90×10^{-2}	6.85×10^{-47}
	Std	**0**	2.05×10^{-98}	1.25×10^{-47}	3.09×10^{-1}	3.96×10^{-62}
	时间/s	0.833 82	0.327 35	0.319 85	0.321 55	0.433 91
F4	Mean	**0**	2.91×10^{-56}	1.60×10^{-32}	1.94×10^{-6}	3.97×10^{-31}
	Std	**0**	1.18×10^{-55}	7.25×10^{-32}	5.71×10^{-6}	2.67×10^{-46}
	时间/s	0.691 18	0.188 1	0.169 79	0.195 44	0.220 55
F5	Mean	$\mathbf{2.26\times10^{1}}$	2.81×10^{1}	2.77×10^{1}	2.69×10^{1}	2.80×10^{1}
	Std	7.68×10^{0}	6.31×10^{-1}	4.69×10^{-1}	1.28×10^{0}	$\mathbf{1.81\times10^{-14}}$
	时间/s	0.791 98	0.239 31	0.177 02	0.272 36	0.340 58
F6	Mean	$\mathbf{3.51\times10^{-4}}$	3.17×10^{0}	2.30×10^{0}	1.21×10^{0}	2.34×10^{0}
	Std	3.25×10^{-4}	5.53×10^{-1}	4.61×10^{-1}	4.48×10^{-1}	$\mathbf{9.03\times10^{-16}}$
	时间/s	0.669 2	0.175 06	0.160 46	0.235 12	0.307 25
F7	Mean	1.09×10^{-4}	1.03×10^{-4}	1.88×10^{-4}	1.82×10^{-3}	$\mathbf{3.98\times10^{-5}}$
	Std	1.07×10^{-4}	7.08×10^{-5}	1.74×10^{-4}	1.22×10^{-3}	$\mathbf{3.45\times10^{-20}}$
	时间/s	0.859 29	0.270 71	0.248 46	0.257 38	0.339 92
F8	Mean	$\mathbf{-1.26\times10^{4}}$	-6.02×10^{3}	-6.45×10^{3}	-8.22×10^{3}	-6.85×10^{3}
	Std	3.82×10^{0}	6.37×10^{2}	3.86×10^{2}	8.68×10^{2}	$\mathbf{3.70\times10^{-12}}$
	时间/s	1.642 5	0.189 92	0.178 41	0.184 32	0.217 83
F9	Mean	**0**	**0**	**0**	1.09×10^{-8}	**0**
	Std	**0**	**0**	**0**	3.25×10^{-8}	**0**
	时间/s	1.918 8	0.658 07	0.665 27	0.241 01	0.213 82
F10	Mean	$\mathbf{4.44\times10^{-16}}$	4.00×10^{-15}	2.34×10^{-15}	6.63×10^{-1}	4.00×10^{-15}
	Std	**0**	**0**	1.80×10^{-15}	3.63×10^{0}	**0**
	时间/s	1.665 6	0.502 69	0.256 42	0.171 46	0.198 3
F11	Mean	**0**	**0**	**0**	4.58×10^{-3}	**0**
	Std	**0**	**0**	**0**	1.91×10^{-2}	**0**
	时间/s	0.863 77	0.216 24	0.225 95	0.229 53	0.244 82
F12	Mean	$\mathbf{1.32\times10^{-5}}$	2.49×10^{-1}	1.62×10^{-1}	8.46×10^{-2}	2.84×10^{-1}
	Std	1.36×10^{-5}	7.79×10^{-2}	6.26×10^{-2}	1.02×10^{-1}	$\mathbf{1.13\times10^{-16}}$
	时间/s	1.106 2	0.450 54	0.409 76	0.439 14	0.536 67
F13	Mean	$\mathbf{3.15\times10^{-4}}$	1.95×10^{0}	1.80×10^{0}	1.03×10^{0}	2.70×10^{0}
	Std	2.70×10^{-4}	3.27×10^{-1}	3.56×10^{-1}	2.32×10^{-1}	$\mathbf{4.52\times10^{-16}}$
	时间/s	1.118 5	0.441 88	0.512 92	0.495 23	0.688 4

注：加粗数值为最佳结果。

如表 6 所示,NSHO 算法仅在函数 F5、F6、F8、F12 上的均值和方差优于 SHO 算法,说明在算法中添加单一的权重策略对算法性能的提升没有太大的影响。WSHO 算法在函数 F5、F6、F8、F12、F13 上的均值优于 SHO 算法,在函数 F6、F13 上的方差优于 SHO 算法。这说明具有非线性权重的 WOA 策略有助于减小 SHO 算法陷入局部最优解的概率,以及提升算法的稳定性,同时也表明海马通过自身位置和精英位置来更新其位置方差的策略是有效的。SSHO 算法在函数 F5、F6、F7、F8 上的均值优于 SHO 算法,函数 F5 ~ F13 上的方差优于 SHO 算法。这说明具有非线性权重的 SCA 策略提升了原始 SHO 算法的全局寻优能力和稳定性。根据上述分析可知,MSHO 算法的性能优于 SHO 算法的性能的关键在于 3 种策略的共同作用。通过策略与策略的相互融合,使得 MSHO 算法具备出色的局部开发和全局探索能力。

3.4　MSHO 算法时间复杂度分析

算法的复杂度是用来衡量算法执行时间或空间资源消耗的指标。它用于描述算法在输入规模增加时,执行时间或空间需求如何增加[15]。本节主要对 MSHO 算法的时间复杂度进行讨论分析。

SHO 算法中,假设 n 表示海马种群大小,D 表示海马种群维度,则 T 次迭代后,SHO 算法的时间复杂度为 $O(TnD)$。基于多策略的 MSHO 算法:将非线性惯性权重策略引入 SHO 算法的运动行为所需时间为 nD;将改进的 WOA 包围猎物策略引入 SHO 算法的捕食行为中,由于没有引进新的循环,故所需时间为 nD;将改进的 SCA 策略引入 SHO 算法的繁殖行为中,由于也没有引进新的循环,故所需时间也为 nD。通过计算,T 次迭代后 MSHO 算法的时间复杂度为 $O(3TnD)$。

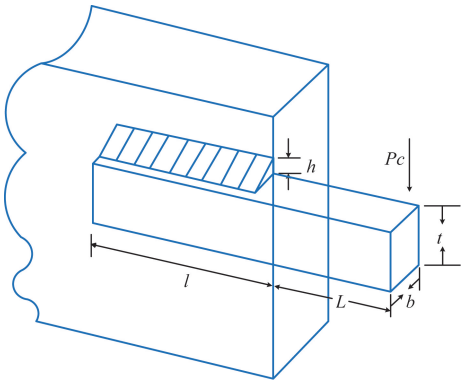
MSHO 算法与 SHO 算法经过 30 次实验后得到的平均运行时长,如表 6 所示。通过计算发现,MSHO 算法的平均运行时长约为 SHO 算法的 3 倍,这说明由多策略共同作用的 MSHO 算法是以牺牲时间的方式来提升原始算法的精度和稳定性的。这也证明了上述分析的可行性。

4　实际工程应用分析

为验证 MSHO 算法在工程优化问题上的适用性,将所提 MSHO 算法与文献中的算法在焊接梁、悬臂梁、压力容器问题上进行测试。实验中采用的参数设置:种群大小 30,最大迭代次数 500,实验次数 30[15]。

4.1　焊接梁问题

焊接梁问题关键在于,如何在给定的约束条件下,实现生产成本最小化[26]。图 4 所示为焊接梁的示意图。



h 为焊缝厚;t 为焊接钢筋高;b 为焊接钢筋厚;l 为焊接钢筋长度

图 4　焊接梁示意图

Fig. 4　Welded beam diagram

此问题涉及的目标函数和约束条件为:

$$\text{Consider: } x = [x_1, x_2, x_3, x_4] = [h, l, t, b] \tag{22}$$

$$\text{Minmize } f(x) = 1.104\,71x_1^2x_2 + 0.048\,11x_3x_4(14 + x_2) \tag{23}$$

$$\text{s.t.} \begin{cases} \dfrac{61}{x_1^3} + \dfrac{37}{x_2^3} + \dfrac{19}{x_3^3} + \dfrac{7}{x_4^3} + \dfrac{1}{x_5^3} - 1 \leqslant 0 \\ 0.01 \leqslant x_1, x_2, x_3, x_4, x_5 \leqslant 100 \end{cases} \tag{24}$$

$$\text{s.t.} \begin{cases} g_1(x) = \tau(x) + \tau_{\text{maks}} \leqslant 0 \\ g_2(x) = \sigma(x) + \sigma_{\text{maks}} \leqslant 0 \\ g_3(x) = \delta(x) + \delta_{\text{maks}} \leqslant 0 \\ g_4(x) = x_1 - x_4 \leqslant 0 \\ g_5(x) = P - P_c(x) \leqslant 0 \\ g_6(x) = 0.125 - x_1 \leqslant 0 \\ g_7(x) = 0.104\,71x_1^2 + 0.048\,11x_3x_4(14 + x_2) - 5 \leqslant 0 \\ 0.1 \leqslant x_1, x_4 \leqslant 2.0,\ 0.1 \leqslant x_2, x_3 \leqslant 10.0 \end{cases} \tag{25}$$

式中:

$$\begin{cases} \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\dfrac{x^2}{2R} + (\tau'')^2} \\ \tau' = \dfrac{P}{\sqrt{2}x_1x_2} \\ \tau'' = \dfrac{MR}{J} \end{cases} \tag{26}$$

$$\begin{cases} M = P\left(L + \dfrac{x_2}{2}\right) \\ R = \sqrt{\dfrac{x_2^2}{4} + \left(\dfrac{x_1 + x_3}{2}\right)^2} \\ J = 2\left\{\sqrt{2}x_1x_2\left[\dfrac{x_2^3}{4} + \left(\dfrac{x_1 + x_3}{2}\right)^2\right]\right\} \end{cases} \tag{27}$$

$$\begin{cases}\sigma(x) = \dfrac{6PL}{x_4x_3^2} \\ \delta(x) = \dfrac{4PL^3}{Ex_4x_2^2} \\ P_c(x) = \dfrac{4.013E\sqrt{x_3^2x_4^6}}{L^2}\left(1-\dfrac{x_3}{2L}\sqrt{\dfrac{E}{4G}}\right)\end{cases} \tag{28}$$

式中:p = 27 215 kg,L = 0.355 6 m,E = 1.44 × 10^9 Pa,G = 5.75 × 10^8 Pa,τ_{max} = 6.5 × 10^5 Pa,σ_{max} = 1.44 × 10^6 Pa,δ_{max} = 0.006 35 m;τ_{max}、σ_{max}、δ_{max} 分别表示 τ、σ、δ 的最大限度。

如表 7 所示,给出了 MSHO 算法和其他不同算法在焊接梁设计问题上进行 30 次实验,得到的最优成本值和最佳变量。通过比较数据发现,MSHO 算法具有最小成本(1.696 33),其对应的最优变量组合为(0.204 8,3.240 7,9.066 6,0.205 6)。图 5 清晰地显示了所提 MSHO 算法与其他算法在此问题上的结果比较。这说明 MSHO 算法更适用于求解该问题,更节约成本。

表 7 焊接梁结果比较

Table 7 Comparison of welded beam results

算法	h	l	t	b	优化成本
MSHO	0.204 80	3.240 70	9.066 6	0.205 60	1.696 33
SHO	0.205 85	3.469 46	9.032 7	0.205 91	1.725 90
CSHO	0.205 70	3.469 90	9.034 6	0.205 80	1.724 90
SABO	0.205 70	3.470 40	9.036 6	0.205 73	1.724 80
GWO	0.205 63	3.472 437	9.041 2	0.205 70	1.725 70
SOA	0.203 06	3.198 80	9.372 5	0.204 18	1.729 20
WOA	0.192 98	3.622 60	8.835 4	0.221 65	1.809 30
PSO	0.231 67	2.943 30	8.630 3	0.231 92	1.806 00

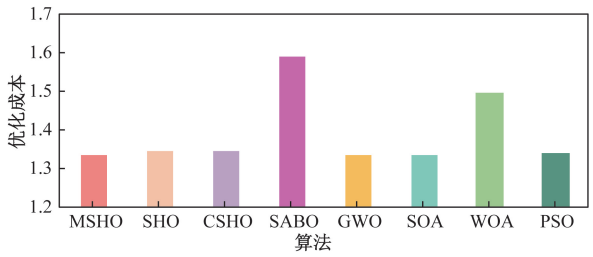


图 5 焊接梁结果比较

Fig. 5 Comparison of welded beam results

4.2 悬臂梁问题

悬臂梁设计问题的主要目标是找到使得悬臂的总重量达到最小值[27]。如图 6 所示,x_j 表示悬臂块的长度,其中 j = 1,2,3,4,5。

此问题涉及的目标函数和约束条件为

$$\text{Minmize } f(x) = 0.062\,24(x_1+x_2+x_3+x_4+x_5)$$

$$\text{s.t.}\begin{cases}\dfrac{61}{x_1^3}+\dfrac{37}{x_2^3}+\dfrac{19}{x_3^3}+\dfrac{7}{x_4^3}+\dfrac{1}{x_5^3}-1 \leqslant 0 \\ 0.01 \leqslant x_1,x_2,x_3,x_4,x_5 \leqslant 100\end{cases} \tag{29}$$

如表 8 所示,给出了 MSHO 算法和不同算法在悬臂梁设计上经过 30 次实验,得到的优化成本极其最优变量组合。通过比较数据发现,相较于其他算法,MSHO 算法得到的优化成本最低为 1.336 8,对应的组合优化变量为(5.937,5.298,4.545,3.553,2.143)。图 7 清晰地显示了所提 MSHO 算法与其他算法在此问题上的结果比较。这说明 MSHO 算法在求解该问题上具有较好的寻优能力,以及重要的实际应用意义。

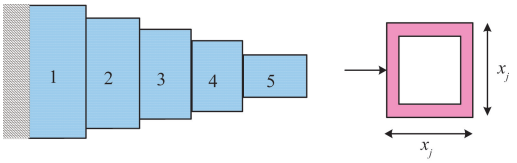


图 6 悬臂梁示意图

Fig. 6 Cantilever diagram

表 8 悬臂梁结果比较

Table 8 Comparison of cantilever beam result

算法	x_1	x_2	x_3	x_4	x_5	优化成本
MSHO	5.937	5.298	4.545	3.553	2.143	1.336 8
SHO	5.506	5.437	4.473	4.702	3.909	1.349 2
CSHO	6.094	4.984	4.904	3.337	2.298	1.345 6
SABO	7.509	7.383	3.124	4.243	3.307	1.591 4
GWO	5.952	5.323	4.466	3.521	2.217	1.337 0
SOA	6.160	5.553	4.330	3.375	2.106	1.339 8
WOA	4.808	6.878	6.369	3.158	2.881	1.499 7
PSO	6.086	4.969	4.822	3.505	3.609	1.342 9

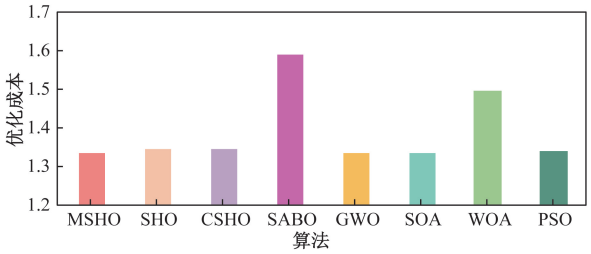


图 7 悬臂梁结果比较

Fig. 7 Comparison of cantilever beam results

4.3 压力容器问题

该问题的目标是使得压力容器的生产成本最小化[28]。如图 8 所示,T_h 为圆柱头部为半球形的壁厚、T_s 为圆柱体底部厚度的厚度、L 为不考虑半球形的圆柱部分的截面长度、R 为圆柱体的内壁半径,这 4 个变量即为压力容器问题需要优化的变量。

其中设计到的成本函数及约束条件为

$$\text{Consider: } x = [x_1,x_2,x_3,x_4] = [T_s,T_h,R,L] \tag{30}$$

$$\text{Minmize } f(x) = 0.622\,4x_1x_3x_4 + 1.778\,1x_2x_3^2 + 3.166\,1x_1^2x_4 + 19.84x_1^2x_3 \tag{31}$$

$$\text{s.t.}\begin{cases} g_1(x) = -x_1 + 0.019\,3x_3 \leqslant 0 \\ g_2(x) = -x_2 + 00.009\,54x_3 \leqslant 0 \\ g_3(x) = -\pi x_3^2 x_4 - (4/3)\pi x_3^3 + 1\,296\,000 \leqslant 0 \\ g_4(x) = x_4 - 240 \leqslant 0 \\ 0.062\,5 \leqslant x_1, x_2 \leqslant 99 \times 0.062\,5, 10 \leqslant x_3, x_4 \leqslant 200 \end{cases} \tag{32}$$

表9给出了MSHO算法和不同算法在压力容器问题上经过30次实验后得到的优化成本和最优变量组合。通过比较数据发现,相较于其他7种算法,MSHO算法得到更低的优化成本为5 885.310,对应的最优变量组合为(0.778,0.384,40.322,199.958)。如图9所示,展示了8种算法在压力容器上得到的优化成本比较图。这说明MSHO算法对于压力容器问题的优化设计有一定的参考价值和实际应用意义。

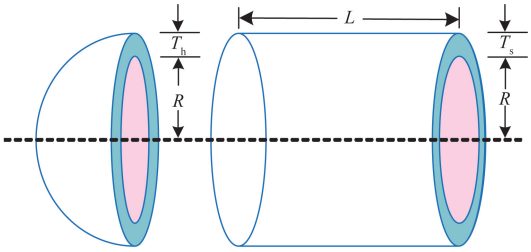


图8　压力容器示意图

Fig. 8　Pressure vessel diagram

表9　压力容器结果比较

Table 9　Comparison of pressure vessel results

算法	T_s	T_h	R	L	优化成本
MSHO	0.778	0.384	40.322	199.958	5 885.310
SHO	0.798	0.392	41.101	189.397	5 951.417
CSHO	0.786	0.417	40.493	197.592	6 015.931
SABO	1.024	0.518	51.671	88.863	6 763.558
GWO	0.865	0.431	44.809	145.808	6 069.623
SOA	0.823	0.423	42.351	174.956	6 096.149
WOA	0.878	0.431	44.651	147.372	6 173.661
PSO	0.866	0.428	44.909	144.659	6 054.651

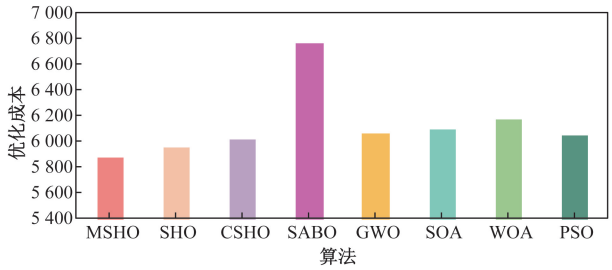


图9　压力容器结果比较

Fig. 9　Comparison diagram of pressure vessel results

5　结论

针对SHO算法存在全局收敛性不强,收敛精度较差等问题,设计了一种融合多策略改进的MSHO算法。首先,在SHO算法的运动行为中引入非线性惯性权重策略;其次,在算法的捕食行为中引入改进的WOA包围猎物策略,以替换海马捕食成功的位置更新方程;然后,在算法的繁殖阶段引入改进的SCA策略,用于改善海马后代解的质量。通过所提3种改进策略的共同作用,MSHO算法的收敛精度得到了增强,同时其全局寻优能力和稳定性也得到了提升。最后,将MSHO算法应用于23测试函数,并与7种算法(SHO、CSHO、SABO、GWO、SOA、WOA、PSO)进行比较。实验数据表明,MSHO算法在20个函数上的平均精度值优于其余7种算法,在16个函数上的平均方差低于其余7种算法,这说明MSHO算法在寻优能力和稳定性方面更优秀。通过wilcoxon符号检验,证实了所提MSHO算法的性能优于这些算法,其改进策略是有效的。此外,将MSHO算法应用于焊接梁、悬臂梁以及压力容器设计,发现MSHO算法在这3个现实世界的工程优化问题上表现出更低的生产成本,说明MSHO算法在处理这些工程问题上具有更好的潜力。基于上述分析,未来还可以尝试将MSHO算法应用于其他优化领域,如机器学习的参数调优、图像处理、神经网络训练、特征选择等问题。

参　考　文　献

[1] 李大海, 詹美欣, 王振东. 混合策略改进的麻雀搜索算法及其应用[J]. 计算机应用研究, 2023, 40(2): 404-412.
Li Dahai, Zhan Meixin, Wang Zhendong. Hybrid strategy improved sparrow search algorithm and its application[J]. Computer Application Research, 2023, 40(2): 404-412.

[2] 王娜, 吴延凯, 许娜. 基于粒子群优化算法的农业机器人控制策略研究[J]. 农机化研究, 2025, 47(1): 205-209.
Wang Na, Wu Yankai, Xu Na. Research on control strategy of agricultural robot based on particle swarm optimization algorithm[J]. Journal of Agricultural Mechanization Research, 2025, 47(1): 205-209.

[3] 何凯琳, 张正军, 位雅, 等. 基于人工鱼群的自适应密度峰值聚类算法[J]. 计算机工程与设计, 2024, 45(1): 110-119.
He Kailin, Zhang Zhengjun, Wei Ya, et al. Adaptive density peak clustering algorithm based on artificial fish swarm[J]. Computer Engineering and Design, 2024, 45(1): 110-119.

[4] 罗超雷, 徐哈宁, 肖慧, 等. 基于鲸鱼优化混合神经网络的滑坡位移预测[J]. 科学技术与工程, 2024, 24(16): 6610-6616.
Luo Chaolei, Xu Haning, Xiao Hui, et al. Landslide displacement prediction based on whale optimization hybrid neural network[J]. Science Technology and Engineering, 2024, 24(16): 6610-6616.

[5] 陈晓梅, 周博, 蔡烨. 基于改进灰狼算法的微网多主体主从博弈策略[J]. 科学技术与工程, 2024, 24(18): 7701-7709.
Chen Xiaomei, Zhou Bo, Cai Ye. Multi-agent master-slave game strategy in micronetworks based on improved grey wolf algorithm[J]. Science Technology and Engineering, 2024, 24(18):

7701-7709.

[6] 付宇, 艾寒冰, 姚振岸, 等. 基于正余弦算法的瑞利波频散曲线反演[J]. 物探与化探, 2023, 47(6): 1467-1478.
Fu Yu, AI Hanbing, Yao Zhenan, et al. Inversion of Rayleigh wave dispersion curve based on sine-cosine algorithm[J]. Geophysical and Geochemical Exploration, 2023, 47(6): 1467-1478.

[7] Wang Y, Zhao K, Hao Y, et al. Short-term wind power prediction using a novel model based on butterfly optimization algorithm-variational mode decomposition-long short-term memory[J]. Applied Energy, 2024, 366. DOI: 10.1016/j. apenergy. 2024.123313.

[8] Huang Q, Ding H, Razmjooy N. Oral cancer detection using convolutional neural network optimized by combined seagull optimization algorithm[J]. Biomedical Signal Processing and Control, 2024, 87. DOI: 10.1016/j. bspc. 2023.105546.

[9] Zhao S, Zhang T, Ma S, et al. Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems[J]. Applied Intelligence, 2023, 53(10): 11833-11860.

[10] 杨仁峥, 黄艳国, 何烜. 基于SSA-BiLSTM和奇异谱分析的短期风电功率预测[J]. 科学技术与工程, 2024, 24(22): 9392-9399.
Yang Renzheng, Huang Yanguo, He Xuan. Short-term wind power prediction based on SSA-BiLSTM and singular spectrum analysis[J]. Science Technology and Engineering, 2024, 24(22): 9392-9399.

[11] 程宁, 李超. 基于粒子群算法的无线传感网络大数据聚类优化方法[J]. 传感技术学报, 2023, 36(8): 1316-1322.
Cheng Ning, Li Chao. Clustering optimization method of big data in wireless sensor networks based on particle swarm optimization[J]. Chinese Journal of Sensor Technology, 2023, 36(8): 1316-1322.

[12] 吴坤, 谭劭昌. 基于改进鲸鱼优化算法的无人机航路规划[J]. 航空学报, 2020, 41(S2): 107-114.
Wu Kun, Tan Shaochang. UAV route planning based on improved whale optimization algorithm[J]. Acta Aeronautica Sinica, 2020, 41(S2): 107-114.

[13] 吕鑫, 慕晓冬, 张钧, 等. 混沌麻雀搜索优化算法[J]. 北京航空航天大学学报, 2021, 47(8): 1712-1720.
Lü Xin, Mu Xiaodong, Zhang Jun, et al. Chaotic sparrow search optimization algorithm[J]. Journal of Beijing University of Aeronautics and Astronautics, 2021, 47(8): 1712-1720.

[14] 邱仲睿, 苗虹, 曾成碧. 多策略融合的改进黏菌算法[J]. 计算机应用, 2023, 43(3): 812-819.
Qiu Zhongrui, Miao Hong, Zeng Chengbi. Improved slime mold algorithm based on multi-strategy fusion[J]. Journal of Computer Applications, 2023, 43(3): 812-819.

[15] Özbay F A. A modified seahorse optimization algorithm based on chaotic maps for solving global optimization and engineering problems[J]. Engineering Science and Technology, 2023, 41. DOI: 10.1016/j. jestch. 2023.101408.

[16] 舒奕彬, 李立君, 张振翮, 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 Yibin, Li Lijun, Zhang Zhenhe, et al. PID parameter optimization based on improved hippocampal optimization algorithm[J]. Machine Tool & Hydraulics, 2024, 52(13): 189-194.

[17] Li Z, Qu S, Xu Y, et al. Enhanced sea horse optimization algorithm for hyperparameter optimization of agricultural image Recognition[J]. Mathematics, 2024, 12. DOI: 10.3390/math12030368.

[18] 曹永娟, 陆壮壮, 蔡骏, 等. 改进海马优化算法的永磁同步电机多参数辨识[J]. 仪表技术与传感器, 2024(2): 79-85, 92.
Cao Yongjuan, Lu Zhuangzhuang, Cai Jun, et al. Multi-parameter identification of permanent magnet synchronous motor with improved seahorse optimization algorithm[J]. Instrument Technique and Sensor, 2024(2): 79-85, 92.

[19] 奚金明, 郑荣艳. 基于自适应权重和莱维飞行的改进海鸥优化算法[J]. 计算机系统应用, 2023, 32(12): 171-179.
Xi Jinming, Zheng Rongyan. Improved gull optimization algorithm based on adaptive weights and Levy flight[J]. Applications of Computer Systems, 2023, 32(12): 171-179.

[20] Fan Q, Chen Z, Li Z, et al. A new improved whale optimization algorithm with joint search mechanisms for high-dimensional global optimization problems[J]. Engineering with Computers, 2021, 37: 1851-1878.

[21] Dey B, Bhattacharyya B. Comparison of various electricity market pricing strategies to reduce generation cost of a microgrid system using hybrid WOA-SCA[J]. Evolutionary Intelligence, 2022, 15(3): 1587-1604.

[22] Zhang J, Wang J S. Improved whale optimization algorithm based on nonlinear adaptive weight and golden sine operator[J]. IEEE Access, 2020, 8: 77013-77048.

[23] 郑洪清, 冯文健, 周永权. 融合正弦余弦算法的蝴蝶优化算法[J]. 广西科学, 2021, 28(2): 152-159.
Zheng Hongqing, Feng Wenjian, Zhou Yongquan. Butterfly optimization algorithm with fusion of sine and cosine algorithm[J]. Guangxi Science, 2021, 28(2): 152-159.

[24] Trojovsk P, Dehghani M. Subtraction-average-based optimizer: a new swarm-inspired metaheuristic algorithm for solving optimization problems[J]. Biomimetics, 2023, 8(2). DOI: 10.3390/biomimetics8020149.

[25] 刘成汉, 何庆. 融合多策略的黄金正弦黑猩猩优化算法[J]. 自动化学报, 2023, 49(11): 2360-2373.
Liu Chenghan, He Qing. Golden sinusoidal chimp optimization algorithm with multi-strategy fusion[J]. Acta Automatica Sinica, 2023, 49(11): 2360-2373.

[26] 刘景森, 袁蒙蒙, 李煜. 基于改进樽海鞘群算法求解工程优化设计问题[J]. 系统仿真学报, 2021, 33(4): 854-866.
Liu Jingsen, Yuan Mengmeng, Li Yu. Engineering optimization design problem based on improved Salpa group algorithm[J]. Journal of System Simulation, 2021, 33(4): 854-866.

[27] Askari Q, Younas I, Saeed M. Political optimizer: a novel socio-inspired meta-heuristic for global optimization[J]. Knowledge-Based Systems, 2020, 195. DOI: 10.1016/j. knosys. 2020.105709.

[28] 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 Jincheng, Li Shaobo, Zhou Peng, et al. Dung beetle optimization algorithm guided by improved sinusoidal algorithm[J]. Computer Engineering and Applications, 2023, 59(22): 92-110.