文章编号：2095-2163(2021)07-0113-08 中图分类号：TP18 文献标志码：A

融合黄金正弦和随机游走的哈里斯鹰优化算法

聂春芳
（贵州大学 电气工程学院，贵阳 550025）

摘 要：针对哈里斯鹰优化算法收敛精度低、易陷入局部最优的问题，本文提出了融合黄金正弦和随机游走的哈里斯鹰优化算法。首先，该算法在哈里斯鹰的探索阶段融合黄金正弦优化算法，增强算法的全局探索能力；其次，使用一种非线性能量指数递减策略，平衡算法的全局探索和局部开发能力；然后，在哈里斯鹰的开发阶段引入高斯随机游走策略对猎物进行随机游走，提升算法的局部开发能力；最后，在 23 个测试函数上进行实验，评估改进后的哈里斯鹰优化算法的寻优性能。实验结果表明，所提算法具有更好的寻优速度和寻优精度。

关键词：哈里斯鹰优化算法；黄金正弦优化算法；随机游走

Harris Hawk Optimization Algorithm combining golden sine and random walk

NIE Chunfang
（The Electrical Engineering College, Guizhou University, Guiyang 550025, China）

【Abstract】 Aiming at the problem of low convergence accuracy and easy to fall into local optimal value of Harris Hawk Optimization algorithm, a Harris Hawk Optimization algorithm combining golden sine and random walk is proposed in this paper. Firstly, the algorithm integrates the golden sinusoidal optimization algorithm in the exploration stage of Harris Hawk to enhance the global exploration ability of the algorithm. Secondly, a non-linear energy exponential decline strategy is used to balance the exploration and development capabilities of the algorithm; Then, in the development phase of Harris Hawk, Gaussian random walk strategy is introduced to perform random walk on the rabbit to improve the local development capabilities of the algorithm. Finally, through experiments on 23 test functions, the optimization performance of the improved Harris Hawk Optimization algorithm is evaluated. Experimental results prove that the proposed algorithm has better optimization speed and accuracy.

【Key words】 Harris Hawk Optimization; golden sine optimization; random walk

0 引 言

哈里斯鹰优化算法（Harris Hawks Optimization，HHO）是一种新的基于群体的智能优化算法，该算法启发于美国亚利桑那州南部的猛禽哈里斯鹰的捕食行为。与常见的群体智能优化算法类似，HHO 也包括搜索和开发阶段。每个哈里斯鹰代表搜索空间的一个候选解，最优解则被视为猎物。由于 HHO 设计过程相对简单、算法整体的搜索能力优异[1]，目前被广泛用于电机控制[2]、电网负荷预测[3]、图像处理[4]等领域。

尽管 HHO 提高了种群多样性、加快了收敛速度，但是 HHO 存在算法收敛精度较低、易陷入局部最优的缺陷。为此，不少学者对其进行改进。文献[5]在原 HHO 中引入精英等级制度，增强种群多样性，并利用 tent 混沌映射调整算法参数、使用高斯随机游走策略跳出局部最优，提高了算法的寻优性能，但算法在固定维度测试函数上的表现不佳。文献[6]针对哈里斯鹰搜索和开发阶段的平衡，设计了 6 种不同的策略对逃逸能进行更新，最终证明指数递减策略要优于其它策略，但算法的整体收敛精度不高。文献[7]利用混沌映射增强种群多样性，并利用模拟退火算法对最优解进行优化，但测试结果并未到达理论最优，还存在很大的改进空间。

针对 HHO 的不足，本文从 3 个方面进行改进：

（1）在搜索阶段引入黄金正弦优化算法取代原算法的搜索策略，利用黄金正弦算法较强的遍历能力提高 HHO 的全局寻优能力。

（2）使用一种非线性指数递减策略更新逃逸能量，使算法在迭代后期也能进行全局探索。

（3）引入高斯随机游走策略对处于开发阶段的哈里斯鹰个体进行扰动，加快算法的收敛速度、提高算法的寻优能力。

1 哈里斯鹰优化算法

HHO 是 Heidari 教授于 2019 年提出的，该算法

作者简介：聂春芳（1995-），女，硕士研究生，主要研究方向：优化算法、新能源发电。

收稿日期：2021-04-26

主要分为 3 个阶段:探索阶段、探索和开发转换阶段、开发阶段[8]。对此拟做研究分述如下。

1.1 探索阶段

哈里斯鹰为捕获到猎物会随机栖息在某些位置并花费数小时去等待、观察、监视周围的沙漠地区。而栖息的位置分为 2 种,每种机会均等。研究推得的数学公式可表示为:

$$X(t+1)=\begin{cases}X_{rand}(t)-r_1\left|X_{rand}(t)-2r_2X(t)\right|, & q\geqslant 0.5\\ (X_{rabbit}(t)-X_m(t))-r_3(LB+r_4(UB-LB)), & q<0.5\end{cases} \tag{1}$$

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

其中,t 为迭代次数;X_{rand} 为随机选择个体的位置;X_{rabbit} 为猎物的位置;X_m 为种群的平均位置;r_1、r_2、r_3、r_4 为 (0,1) 内的随机数;UB、LB 为种群活动范围的上下界;N 为种群个数。

1.2 探索到开发的转换

哈里斯鹰根据逃逸能量 E 来进行不同狩猎阶段的转换,当 $|E|\geqslant 1$ 时进行探索,$|E|<1$ 进行开发。此时会用到如下数学公式:

$$E=2E_0\left(1-\frac{t}{T}\right) \tag{3}$$

其中,E_0 为 (-1,1) 内的随机数,T 为最大迭代次数。

1.3 开发阶段

在此阶段中,哈里斯鹰对探索阶段中发现的猎物进行突袭。但猎物总试图从危险中逃脱,根据猎物逃逸的成功率 r 和逃逸能量 E,哈里斯鹰将分为 4 种策略捕获猎物。这里将给出分析表述如下。

1.3.1 软围攻

当 $r\geqslant 0.5$、$|E|\geqslant 0.5$ 时,猎物的逃逸能量足够,尝试采用误导性的跳跃逃离,但未能成功。此时哈里斯鹰的位置更新用式(4)~式(5)进行表示:

$$X(t+1)=\Delta X(t)-E\left|JX_{rabbit}(t)-X(t)\right| \tag{4}$$

$$\Delta X(t)=X_{rabbit}(t)-X(t) \tag{5}$$

$$J=2*(1-r_5) \tag{6}$$

其中,ΔX 为猎物与当前个体的插值;r_5 为 (0,1) 内的随机数;J 为猎物跳跃的距离。

1.3.2 硬围攻

当 $r\geqslant 0.5$、$|E|<0.5$ 时,猎物的逃逸能量不足,也无逃脱的机会。此时哈里斯鹰的位置更新用式(7)来表示:

$$X(t+1)=X_{rabbit}(t)-E\left|\Delta X(t)\right| \tag{7}$$

1.3.3 快速俯冲的软围攻

当 $r<0.5$、$|E|>0.5$ 时,猎物的逃逸能量足够,也可成功逃离。此时哈里斯鹰的围攻分为 2 个策略,若第一个围攻策略无效,则执行第二个围攻策略。策略内容参见如下。

(1)策略一。具体公式为:

$$Y=X_{rabbit}(t)-E\left|JX_{rabbit}(t)-X(t)\right| \tag{8}$$

(2)策略二。具体公式为:

$$Z=Y+\boldsymbol{S}\times LF(D) \tag{9}$$

其中,D 为求解问题的维数;$\boldsymbol{S}$ 为一个随机的 D 维向量;LF 为 *Levy* 飞行函数,如式(10)、式(11)所示:

$$LF(x)=0.01\times\frac{u\times\sigma}{|\nu|^{\frac{1}{\beta}}} \tag{10}$$

$$\sigma=\left(\frac{\Gamma(1+\beta)\times\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\times\beta\times 2^{\left(\frac{\beta-1}{2}\right)}}\right)^{\frac{1}{\beta}} \tag{11}$$

其中,u、δ 为(0,1)内的随机数,β 的值为 1.5。此时哈里斯鹰的位置更新用式(12)进行表示:

$$X(t+1)=\begin{cases}Y & F(Y)<F(X(t))\\ Z & F(Z)<F(X(t))\end{cases} \tag{12}$$

1.3.4 快速俯冲的硬围攻

当 $r<0.5$、$|E|<0.5$ 时,猎物的逃逸能量不够,但可成功逃离。此时哈里斯鹰的位置更新用式(13)~式(15)来表示:

$$X(t+1)=\begin{cases}Y & F(Y)<F(X(t))\\ Z & F(Z)<F(X(t))\end{cases} \tag{13}$$

$$Y=X_{rabbit}(t)-E\left|X_{rabbit}(t)-X_m(t)\right| \tag{14}$$

$$Z=Y+\boldsymbol{S}\times LF(D) \tag{15}$$

2 改进的哈里斯鹰优化算法

2.1 黄金正弦算法

黄金正弦算法(golden sine algorithm, Golden-SA)是 Tanyildizi 等人[9]于 2017 年提出的新型群智能算法。该算法引入黄金分割数,利用正弦函数进行迭代寻优可遍历正弦函数上的所有点,具有较强的全局搜索能力。Golden-SA 的核心位置更新方式,如式(16)所示:

$$X_i^{t+1}=X_i^t*\left|\sin(R_1)\right|+R_2*\sin(R_1)*\left|x_1*P_i^t-x_2*X_i^t\right| \tag{16}$$

其中,t 为当前迭代次数;X_i^t 表示第 t 次迭代中第 i 个体的位置;R_1 为 $[0,2\pi]$ 内的随机数;R_2 为

$[0,\pi]$ 内的随机数；P_i^t 表示第 t 次迭代中个体 i 的最优位置；x_1、x_2 是根据黄金分割数所得的系数，$x_1 = -\pi + (1-\tau) * 2\pi$、$x_2 = -\pi + \tau * 2\pi$。

观察哈里斯鹰优化算法探索阶段的更新公式(1)可知原算法在 $q \geq 0.5$ 时的搜索过于随机，也未能与种群中的其它个体进行交流，导致算法的全局搜索能力变差，难以有效遍历整个解空间。因此，本文将黄金正弦优化算法融合到 HHO 的探索阶段，改进后的探索公式如式(17)所示：

$$X(t+1) = \begin{cases} X_i^t * |\sin(R_1)| + R_2 * \sin(R_1) * |x_1 * P_i^t - x_2 * X_i^t|, & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)), & q < 0.5 \end{cases} \tag{17}$$

2.2　非线性能量指数递减策略

在 HHO 中，逃逸能量 E 不仅控制着全局探索和局部开发的的转换，而且还决定着哈里斯鹰四种开发策略的选择。文献[4]已经指出了原始算法逃逸能量 E 后期恒小于 1，缺少全局探索，易使算法陷入局部最优。而文献[6]通过实验指出了指数递减策略为逃逸能量的最佳策略。因此本文定义了一种非线性能量指数递减策略，具体见式(18)、式(19)：

$$E = 2e^{-\left(\alpha \times \frac{t}{T}\right)} + \delta \tag{18}$$

$$\delta = randn \times \left(\sin^{\beta}\left(\frac{\pi}{2} \times \frac{t}{T}\right) + \cos\left(\frac{\pi}{2} \times \frac{t}{T}\right) - 1\right) \tag{19}$$

其中，$randn$ 为 $(0,1)$ 内的随机数；α 的值为 1.3；β 的值为 1.7。

2.3　高斯随机游走策略

高斯随机游走作为一种经典的随机游走模型，模型的开发能力比较强。在 HHO 的开发阶段引入高斯随机游走策略，对种群的最优个体施加扰动，生成新的个体，既利于增强算法的收敛速度，又可在算法陷入局部最优时帮助算法跳出局部最优。具体的高斯随机游走策略见式(20)、式(21)：

$$X(t+1) = Gussian(X_{rabbit}^t, \tau) \tag{20}$$

$$\tau = \cos\left(\left(\frac{\pi}{2} * \frac{t}{T}\right)^2\right) \times (X_{rabbit}^t - X_{rand}(t)) \tag{21}$$

其中，X_{rabbit}^t 为第 t 次迭代中猎物的位置，即最优个体的位置。

2.4　算法步骤

本文改进的哈里斯鹰优化算法(GSHHO)的伪代码详见如下。

```
Initialize the position of the hawks x_i.
Set maximum number of iterations T.
Set the dimensions of the optimization problem dim.
While (t < T)
    Check the boundary and Calculate the fitness value f_rabbit and identify the rabbit.
    Perform a Gaussian walk on the rabbit using Equations(20)-(21).
    For (i = 1: N) do
      If( |E| ≥ 1) then
        Update the position using Equation(17).
      Else If( |E| < 1) then
        If( r ≥ 0.5 and |E| ≥ 0.5) then
          Update the position using Equation(4)~(6).
        Else If( r ≥ 0.5 and |E| < 0.5) then
          Update the position using Equation(7).
        Else If( r < 0.5 and |E| ≥ 0.5) then
          Update the position using Equation(8)~(12).
        Else If( r < 0.5 and |E| < 0.5) then
          Update the position using Equation(13)~(15).
        End If
      End If
    End For
End While
```

3　仿真结果和分析

本文仿真环境为 64 位的 Windows 10 操作系统、Intel Core i5-4210M CPU、Matlab 2019b，为验证 GSCHHO 的寻能力能力，将融合黄金正弦优化的哈里斯鹰优化算法(GSHHO)同粒子群算法(PSO)[10]、黄金正弦优化算法(GoldSA)、改变逃逸能量的哈里斯鹰优化算法(MHHO)[6]和混沌精英哈里斯鹰优化算法(CEHHO)[5]同时在表 1 的 23 个测试函数中进行对比。

3.1　实验参数设置

为确保仿真实验的合理性，各算法种群规模都设为 30，迭代次数设为 500，所有算法均独立运行 30 次取平均值后作为实验结果。各算法的参数设置见表 2。

3.2　实验结果与分析

表 3 给出了 6 个群智能优化算法在 23 个基准测试函数上的 30 次寻优的平均值和标准差，黑色加粗字体为对应测试函数上平均值最佳的优化算法。其中，$F_1 \sim F_7$ 为单峰测试函数，主要用于评价算法的开发能力。由表 3 可知，对于 $F_1 \sim F_4$，GSHHO 相比于原 HHO 的寻优精度有着明显提升，其平均值均可以收敛到最优值 0；对于 $F_5 \sim F_7$，GSHHO 的平均寻优值虽未收敛到最优值，但相比于原 HHO 其收敛精度分别提高了 2、2、1 个数量级。相比于其

余 4 种对比算法在 $F_1 \sim F_7$ 上的寻优表现,GSHHO 的平均寻优能力均排在首位。结合图 1 单峰测试函数的平均收敛曲线可知,对于 $F_1 \sim F_4$, GSHHO 的收敛速度要弱于 GoldSA,但由于 GSHHO 在哈里斯鹰的探索阶段用黄金正弦优化取代了原本的寻优公式,优化了算法的全局遍历能力,故 GSHHO 的收敛速度要强于原 HHO 和改进后的 MHHO、CEHHO。而在哈里斯鹰的开发阶段添加的高斯随机游走策略强化了算法的局部开发能力,因此 GSHHO 的收敛速度虽弱于 GoldSA 但却总能先收敛到 0;对于 $F_5 \sim F_6$, GSHHO 的收敛速度也明显优于其余 5 种算法;F_7 由于篇幅限制,未在此列出。总地来说,在单峰测试函数上 GSHHO 相比于原 HHO 和其余对比算法有着更好的寻优精度和收敛速度。

表 1 测试函数

Tab. 1 Test function

分类	测试函数	维数	范围	最优值
单峰测试函数	$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[100,100]	0
	$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[10,10]	0
	$F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	[100,100]	0
	$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant n \}$	30	[100,100]	0
	$F_5(x) = \sum_{i=1}^{n-1} [100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[30,30]	0
	$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[100,100]	0
	$F_7(x) = ix_i^4 + random[0,1]$	30	[128,128]	0
多峰测试函数	$F_8(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[500,500]	$-418.982\,9 * n$
	$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[5.12,5.12]	0
	$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	30	[32,32]	0
	$F_{11}(x) = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[600,600]	0
	$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_{i-1})^2 [1 + \sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < a \end{cases}$	30	[50,50]	0
	$F_{13}(x) = 0.1\{\sin^2(3\pi x_i) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_i - 1)^2 [1 + \sin^2(2\pi x_i)]\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	30	[50,50]	0
固定维测试函数	$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	[-65,65]	1
	$F_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[-5,-5]	0.000 30
	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,-5]	-1.031 6
	$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	[-5,-5]	0.398
	$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
	$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_i - p_{ij})^2)$	3	[1,3]	-3.86
	$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_i - p_{ij})^2)$	6	[0,1]	-3.32
	$F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	[0,10]	-10.153 2
	$F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	[0,10]	-10.402 8
	$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	[0,10]	-10.536 3

表 2　各优化算法主要参数

Tab. 2　The main parameters of each optimization algorithm

算法	参数
PSO	$c_1 = c_2 = 2$, $V_{max} = 1$, $V_{min} = -1$, $\omega_{max} = 0.9$, $\omega_{min} = 0.2$
GoldSA	$a = -\pi$, $b = \pi$
HHO、MHHO、CEHHO、GSHHO	/

表 3　测试函数实验结果

Tab. 3　Experimental results of test function

函数		PSO	GoldSA	HHO	MHHO	CEHHO	GSHHO
F_1	平均值	1.29E-05	1.01E-249	1.30E-97	3.47E-128	**0.00E+00**	**0.00E+00**
	标准差	2.18E-05	0.00E+00	6.05E-97	1.87E-127	0.00E+00	0.00E+00
F_2	平均值	7.44E-03	3.04E-130	5.94E-52	1.15E-70	6.33E-59	**0.00E+00**
	标准差	9.92E-03	1.64E-129	1.56E-51	3.74E-70	2.12E-58	0.00E+00
F_3	平均值	3.25E+01	1.22E-232	9.61E-70	3.23E-117	1.84E-88	**0.00E+00**
	标准差	8.25E+01	0.00E+00	5.17E-69	1.73E-116	8.49E-88	0.00E+00
F_4	平均值	4.82E-01	5.69E-111	2.24E-48	1.37E-68	2.79E-58	**0.00E+00**
	标准差	4.82E-01	3.06E-110	1.05E-47	6.99E-68	9.26E-58	0.00E+00
F_5	平均值	4.79E+01	4.24E-03	1.64E-02	1.35E-02	4.09E-03	**8.89E-04**
	标准差	3.66E+01	7.43E-03	2.64E-02	1.41E-02	1.05E-02	1.45E-03
F_6	平均值	8.25E-06	3.91E-04	1.02E-04	5.90E-05	2.18E-05	**6.44E-06**
	标准差	1.58E-05	9.23E-04	1.03E-04	1.35E-04	3.91E-05	5.88E-06
F_7	平均值	8.97E-02	1.19E-04	1.03E-04	1.37E-04	1.26E-04	**9.62E-05**
	标准差	3.07E-02	1.59E-04	7.45E-05	1.35E-04	1.14E-04	7.62E-05
F_8	平均值	-2.78E+03	**-1.26E+04**	**-1.26E+04**	**-1.26E+04**	**-1.26E+04**	**-1.26E+04**
	标准差	5.00E+02	1.98E-01	1.26E+00	7.30E-01	2.69E-01	2.10E-01
F_9	平均值	4.66E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	标准差	1.17E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{10}	平均值	1.29E-03	**8.88E-16**	**8.88E-16**	**8.88E-16**	**8.88E-16**	**8.88E-16**
	标准差	8.60E-04	9.86E-32	9.86E-32	9.86E-32	9.86E-32	9.86E-32
F_{11}	平均值	4.42E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	标准差	7.08E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{12}	平均值	7.87E-01	1.39E-05	5.88E-06	7.26E-06	1.82E-06	**7.10E-07**
	标准差	9.35E-01	2.23E-05	8.88E-06	1.17E-05	3.12E-06	6.94E-07
F_{13}	平均值	3.30E-03	3.39E-05	1.55E-04	6.87E-05	2.77E-05	**7.15E-06**
	标准差	5.03E-03	9.09E-05	2.66E-04	9.19E-05	8.31E-05	8.16E-06
F_{14}	平均值	1.59E+00	1.10E+00	1.39E+00	1.46E+00	1.16E+00	**1.03E+00**
	标准差	1.36E+00	3.84E-01	9.40E-01	9.45E-01	3.70E-01	1.78E-01
F_{15}	平均值	5.42E-04	5.33E-04	3.51E-04	3.81E-04	**3.37E-04**	3.42E-04
	标准差	3.74E-04	4.49E-04	3.95E-05	6.63E-05	2.86E-05	3.63E-05
F_{16}	平均值	**-1.03E+00**	-1.02E+00	**-1.03E+00**	**-1.03E+00**	**-1.03E+00**	**-1.03E+00**
	标准差	0.00E+00	8.98E-03	4.15E-09	2.08E-05	1.57E-08	4.58E-10
F_{17}	平均值	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**
	标准差	1.11E-16	8.22E-04	1.69E-05	5.34E-04	2.58E-06	2.35E-06
F_{18}	平均值	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**
	标准差	1.92E-15	9.64E+00	9.81E-07	1.48E-03	5.55E-06	7.95E-08
F_{19}	平均值	**-3.86E+00**	-3.80E+00	**-3.86E+00**	-3.85E+00	**-3.86E+00**	**-3.86E+00**
	标准差	2.66E-15	6.69E-02	2.02E-03	2.03E-02	5.45E-03	5.08E-03
F_{20}	平均值	**-3.28E+00**	-2.98E+00	-3.12E+00	-3.02E+00	-3.10E+00	-3.11E+00
	标准差	5.60E-02	3.30E-01	8.61E-02	1.09E-01	1.07E-01	9.46E-02
F_{21}	平均值	-5.57E+00	**-1.01E+01**	-5.39E+00	-5.05E+00	-5.49E+00	**-1.01E+01**
	标准差	3.54E+00	1.49E-02	1.25E+00	5.03E-03	1.32E+00	6.23E-02
F_{22}	平均值	-7.25E+00	-8.77E+00	-5.08E+00	-5.08E+00	-6.22E+00	**-1.01E+01**
	标准差	3.44E+00	3.55E+00	5.73E-03	3.52E-03	2.06E+00	1.19E+00
F_{23}	平均值	-7.59E+00	**-1.05E+01**	-5.01E+00	-5.12E+00	-5.80E+00	**-1.05E+01**
	标准差	3.66E+00	1.08E-01	5.86E-01	4.89E-03	1.72E+00	4.78E-02

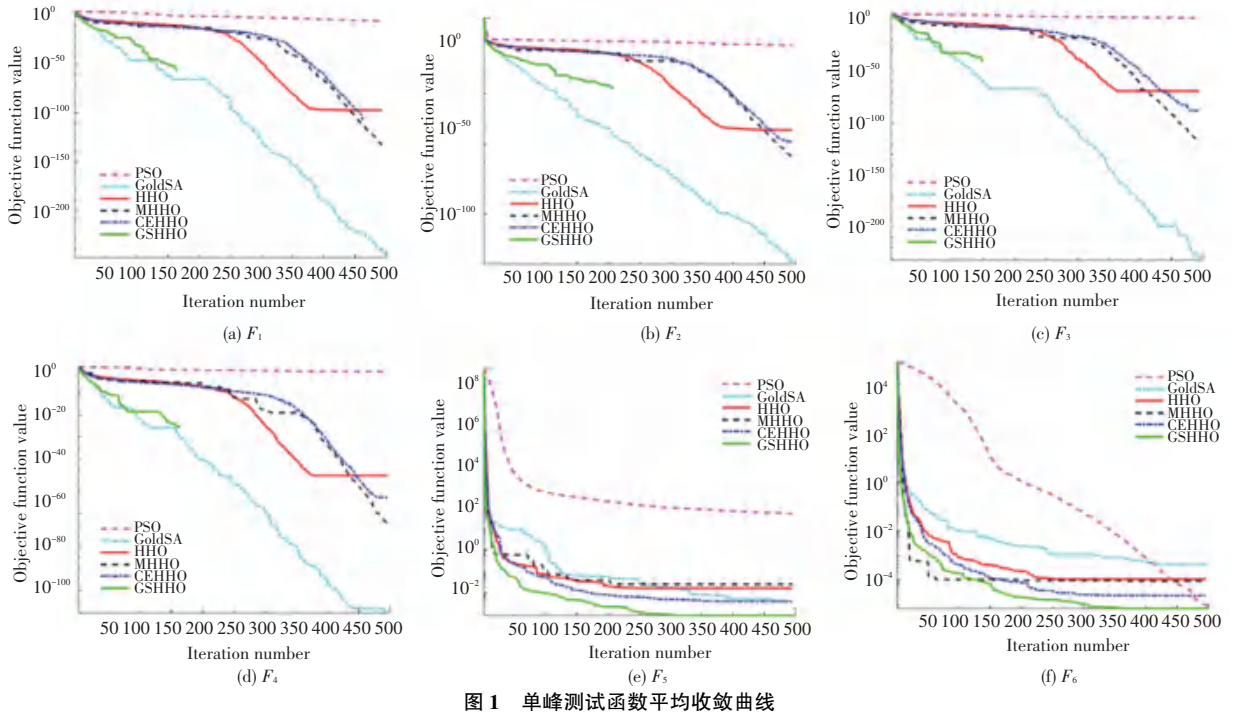

图 1 单峰测试函数平均收敛曲线

Fig. 1 Average convergence curve of unimodal test functions

$F_8 \sim F_{13}$ 为多峰测试函数，主要用于评价算法的全局探索能力，以判断算法能否跳出局部最优值。多峰测试函数平均收敛曲线如图 2 所示。观察 6 个算法在多峰测试函数上的寻优平均值可知，GSHHO 在 $F_8 \sim F_{11}$ 上的平均寻优值同原 HHO 保持一致，但在 $F_{12} \sim F_{13}$ 上其平均寻优值分别提高了 1、2 个数量级；相比较于其余对比算法，GSHHO 的平均寻优值也均排在首位或并列首位。这说明改进算法在探索和开发转换阶段引入的非线性指数递减策略平衡了算法的全局探索能力和局部开发能力，一定程度上避免了算法陷入局部最优的问题。从图 2 也可直观地看出本文所改进的哈里斯鹰优化算法的整体寻优能力要优于 PSO、GoldSA、HHO、MHHO 和 CEHHO。

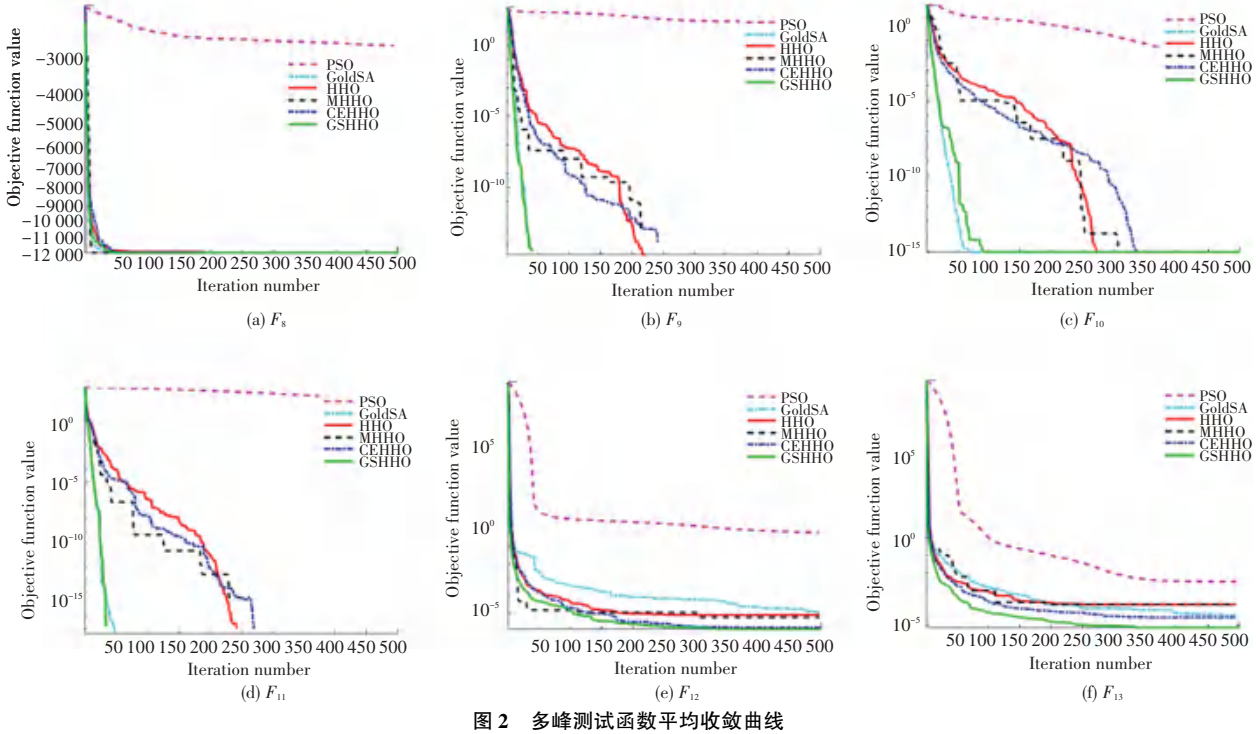

图 2 多峰测试函数平均收敛曲线

Fig. 2 Average convergence curve of multimodal test functions

F_{14} ~ F_{23} 为固定维测试函数，主要用于评价算法在不同维度、不同函数上的寻优能力。由表 3 可知，在 10 个固定维度的测试函数中，GSHHO 的平均寻优值有 8 个是最佳的，而原 HHO 算法只有 4 个是最佳的，其余对比算法 PSO、GoldSA、MHHO、CEHHO 的平均寻优值分别有 5、4、3、5 个为最佳。为更直观地表明 GSHHO 的寻优能力，绘制了各优化算法在固定维测试函数上的平均收敛曲线。限于篇幅，本文仅列出了各算法在 F_{14}、F_{15}、F_{19}、F_{21}、F_{22}、F_{23} 上的平均收敛曲线，如图 3 所示。由图 3 可知，GSHHO 除在 F_{19} 上的收敛速度弱于 PSO 外，在其余固定维的测试函数上均收敛最快。总体而言，GSHHO 在固定维测试函数上的整体寻优能力更佳。

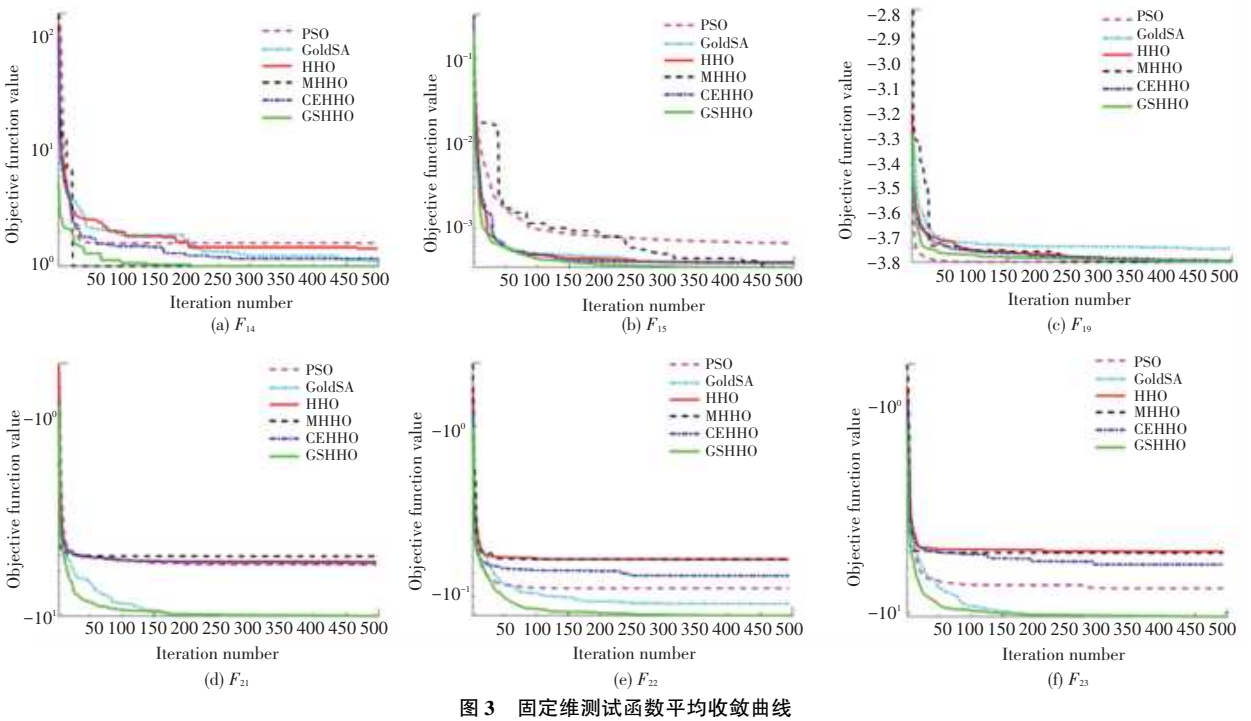

图 3　固定维测试函数平均收敛曲线

Fig. 3　Average convergence curve of the fixed－dimensional test functions

为更加形象地看出各算法性能的优劣，根据表 3 中各算法寻优的平均值和标准差，绘制了不同优化算法排序对比的雷达图，如图 4 所示。其中，算法收敛精度最高的算法排名最佳，相同收敛精度下标准差越小、排名越高。由图 4 可知，GSHHO 所围的面积最小，这说明在 23 个测试函数中本文改进的 GSHHO 的寻优性能最佳。

4　结束语

本文针对哈里斯鹰优化算法进行改进，利用黄金正弦优化较强的遍历能力，增强了算法探索阶段的全局寻优能力；采用一种非线性的指数递减策略，使算法在前期侧重于全局寻优，后期侧重于局部开发、但也能进行少量的全局探索，避免算法陷入局部最优；同时利用高斯随机游走策略对猎物进行扰动，增强算法的局部开发能力，加快收敛速度。本文选取了单峰、多封、固定维共 23 个测试函数进行实验，结果表明，本文改进的哈里斯鹰优化算法较原 HHO 和其余对比算法 PSO、GoldSA、MHHO、CEHHO 有着更好的寻优能力。

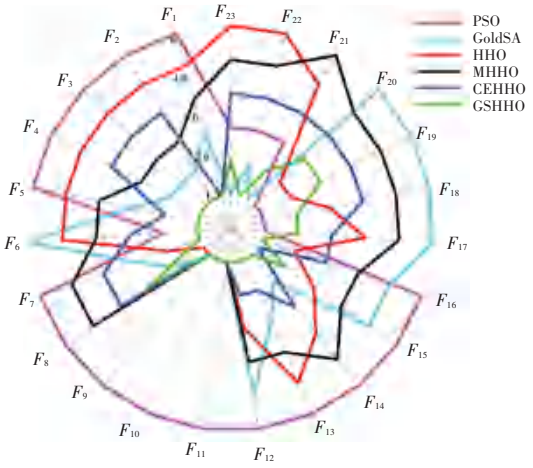

图 4　不同优化算法排序对比图

Fig. 4　Comparison of ranking of different optimization algorithms

（下转第 123 页）