杨易莹, 潘大志. 基于变异算子和混沌映射改进灰狼优化算法[J]. 智能计算机与应用,2025,15(6):58-66. DOI:10.20169/j.issn.2095-2163.24121602

基于变异算子和混沌映射改进灰狼优化算法

杨易莹[1], 潘大志[2]
(1 西华师范大学 数学与信息学院, 四川 南充 637009; 2 西华师范大学 计算方法与应用研究所, 四川 南充 637009)

摘 要: 灰狼优化(GWO)算法是通过模拟狼群捕猎抽象出的一种群智能算法,在解决优化问题上具有良好的表现,但是仍存在求精精度低、收敛速度慢等问题。针对以上弊端,采用了混沌映射和反向学习策略对狼群进行初始化操作,引进非线性收敛因子改进策略和非线性权重调节策略提高了算法的收敛速度,避免算法陷入局部最优。为验证策略的有效性,选取了 23 种标准测试函数进行 50 次对比实验,实验结果表明,改进后的算法在单峰、多峰、低维、高维都有良好表现。

关键词: 灰狼优化算法; 混沌映射; 反向学习; 收敛因子; 函数优化
中图分类号: TP18 **文献标志码**: A **文章编号**: 2095-2163(2025)06-0058-09

The gray wolf optimization algorithm improved by mutation operator and chaotic mapping

YANG Yiying[1], PAN Dazhi[2]
(1 School of Mathematics and Information, China West Normal University, Nanchong 637009, Sichuan, China;
2 Institute of Computing Method and Application Software, China West Normal University, Nanchong 637009, Sichuan, China)

Abstract: Gray Wolf Optimization (GWO) algorithm is a swarm intelligence algorithm abstracted by simulating the hunting behavior of wolves. It performs well in solving optimization problems but still suffers from issues such as low precision and slow convergence speed. To address these drawbacks, chaotic mapping and reverse learning strategies are used to initialize the wolf population. Additionally, a nonlinear convergence factor improvement strategy and a nonlinear weight adjustment strategy are introduced to enhance the algorithm´s convergence speed and prevent it from getting trapped in local optima. To verify the effectiveness of these strategies, 23 standard benchmark functions are selected for 50 comparative experiments. The experimental results show that the improved algorithm performs well in single-peaked, multi-peaked, low-dimensional, and high-dimensional optimization problems.

Key words: grey wolf optimization algorithm; chaos mapping; reverse learning; convergence factor; functional optimization

0 引 言

通过对自然界中昆虫、兽群、鸟群、鱼群等社会性动物的观察,Beni 等学者[1]第一次提出了群体智能(swarm intelligence)的概念。在此基础上,从生物进化机理中受到启发,提出了许多用于解决优化问题的群智能算法。1991 年,Colorni 等学者[2]提出了蚁群算法(Ant Colony Optimization, ACO)。1995 年 Kennedy 等学者[3]和 Eberhart 等学者[4]提出了粒子群算法(Particle Swarm Optimization, PSO)。蚁群算法和粒子群算法为解决一类优化问题提供了新的思路,众多学者从生物学和社会学角度出发,又提出了一些有代表性的群智能算法。例如:2002 年,李晓磊等学者[5]通过研究鱼群的行为特点提出了鱼群算法(Artificial Fish Swarm Algorithm, AFSA)。Krishnanand 等学者[6]于 2005 年提出了人工萤火虫群优化(Glowworm Swarm Optimization, GSO)。2010 年,受蝙蝠回声定位行为的启发,Yang[7]提出了蝙蝠算法(Bat Algorithm, BA)。受狼群猎食行为的启发,Mirjalili 等学者[8]于 2014 年提出了灰狼优化算法(Grey Wolf Optimization, GWO)。2017 年,Saremi 等学者[9]提出了一种源自于蝗虫群体捕食行为的

基金项目: 国家自然科学基金(11871059); 四川省教育厅自然科学基金(18ZA0469)。
作者简介: 杨易莹(1999—),女,硕士研究生,主要研究方向:智能计算,组合优化。
通信作者: 潘大志(1974—),男,博士,教授,硕士生导师,主要研究方向:智能计算,算法设计。Email:pdzzj@126.com。
收稿日期: 2024-12-16

群智能优化算法：蝗虫优化算法（Grasshopper Optimization Algorithm，GOA）。2020 年，Xue 等学者[10]依据麻雀的觅食行为和反捕食行为，提出了麻雀优化算法（Sparrow Search Algorithm，SSA）。

其中，灰狼优化算法（GWO）具有结构简单、参数调节需求较少、适应性较强和收敛性较好的优点，并且在函数优化方面，已经证明 GWO 收敛速度和求解精度均优于 PSO 算法；因此，GWO 受到了国内外学者的广泛关注。但是，基本 GWO 算法也存在对初始种群的依赖性、以及过早收敛、易陷入局部最优等缺陷。针对上述问题，学者们分别提出了相应的策略来改善算法的性能。徐辰华等学者[11]提出了一种改进灰狼优化算法，利用 Cat 混沌映射进行初始化操作，结合高斯变异策略提高了种群多样性。魏政磊等学者[12]利用适应度值提出了基于自适应搜索策略的灰狼优化算法，加速算法收敛。王秋萍等学者[13]利用基于余弦规律和步长欧氏距离对灰狼优化算法进行了改进，加速算法收敛。蔡娟[14]将混沌映射以及高斯扰动策略与灰狼优化算法相结合，加强了局部搜索能力。Qiu 等学者[15]将镜像学习策略和粒子群算法融入灰狼优化算法，避免算法过早收敛。汪勇等学者[16]建立引导机制概率模型提高了灰狼算法的勘探能力，优化算法的开发能力。

上述改进虽然已经对 GWO 算法进行了优化，但是在算法精度方面仍有所欠缺。本文提出了基于混沌映射和反向学习策略的初始化操作来提高初始种群的多样性，引进随迭代次数变化的非线性收敛因子，同时引入基于种群质心的非线性权重调节策略以平衡算法的全局搜索能力和局部搜索能力。对 23 个标准测试函数进行仿真实验，并与其他改进的 GWO 算法进行比较，验证了本文改进算法的有效性。

1　基本灰狼算法

1.1　灰狼群体社会等级制度和捕食行为

GWO 是一种模仿灰狼社会等级制度和猎食行为的新型群智能算法。灰狼群体中具有严格的等级制度，如图 1 所示。图 1 中，处于金字塔第 1 层的领导者称为 α，是负责决策的最高级别灰狼；处于金字塔第 2 层的狼称为 β，是 α 的智囊团和接替者，主要任务是协助管理；处于金字塔中第 3 阶层的狼是 δ，听从 α 和 β 的指挥，负责捕猎、看护等；最底层的是 ω，负责平衡种群内部的关系。捕食过程由 α 负责指挥，β 和 δ 进行追捕围攻，ω 跟随此三者对猎物追踪围剿，最终完成捕食任务。

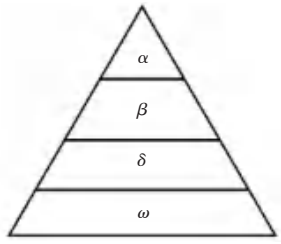

图 1　灰狼等级结构

Fig. 1　Rank structure of the gray wolf population

1.2　基本灰狼优化算法

在 n 维搜索空间中，假设狼群数目为 NP，第 i 只灰狼的位置可表示为 $X_i=(X_i^1,X_i^2,\cdots,X_i^d,\cdots,X_i^n)$，其中 X_i^d 为第 i 只灰狼在第 d 维的位置。狼群中的每一只狼都代表一个潜在解，α 是目前的最优解，β 和 δ 分别为优解和次优解，ω 是其他的候选解。第 i 只灰狼的位置更新公式如下：

$$D=|C\cdot X_P(t)-X_i(t)| \tag{1}$$

$$X_i(t+1)=X_P(t)-A\cdot D \tag{2}$$

其中，t 表示当前迭代次数；X_P 是猎物的位置；D 表示猎物与灰狼之间的距离；A 和 C 是系数向量，并有如下定义：

$$A=2a\cdot r_1-a \tag{3}$$

$$C=2r_2 \tag{4}$$

其中，r_1 和 r_2 都是 [0,1] 之间的随机变量，a 是收敛因子，随迭代次数线性地从 2 递减到 0，其表达式为：

$$a=2-2\left(1-\frac{t}{t_{\max}}\right) \tag{5}$$

其中，$t_{\max}$ 表示最大迭代次数。

在实际优化过程中，猎物的位置是不可知的，因此由 α，β 和 δ 引导其他灰狼的位置，使狼群更好地了解猎物位置。具体公式如下：

$$\begin{cases}D_\alpha=|C_1\cdot X_\alpha(t)-X_i(t)|\\ D_\beta=|C_2\cdot X_\beta(t)-X_i(t)|\\ D_\delta=|C_3\cdot X_\delta(t)-X_i(t)|\end{cases} \tag{6}$$

$$\begin{cases}X_1=X_\alpha-A_1\cdot D_\alpha\\ X_2=X_\beta-A_2\cdot D_\beta\\ X_3=X_\delta-A_3\cdot D_\delta\end{cases} \tag{7}$$

$$X_i(t+1)=\frac{X_1+X_2+X_3}{3} \tag{8}$$

2　改进的灰狼优化算法

2.1　种群初始化改进策略

在群智能算法中，在初始化阶段如果能使粒子

尽可能地遍历整个解空间,对后续寻优有至关重要的作用。GWO 算法在初始化阶段过于随意,可能导致狼群分布不均匀,使得寻优结果并不理想。混沌映射具有不可预测性、遍历性,将其应用于初始化阶段,可以使初始种群尽可能地覆盖在整个搜索空间中。

常见混沌映射的有如下 3 种。

(1)Logistic 映射。数学公式为:

$$x_{i+1} = r \cdot x_i \cdot (1 - x_i) \tag{9}$$

其中,r 为控制参数,在 [0,4] 中取值。当 $r = 3.9$,初值 $x_0 = 0.5$,迭代 1 000 次后可以得到 Logistic 混沌映射的时间序列图,如图 2 所示。

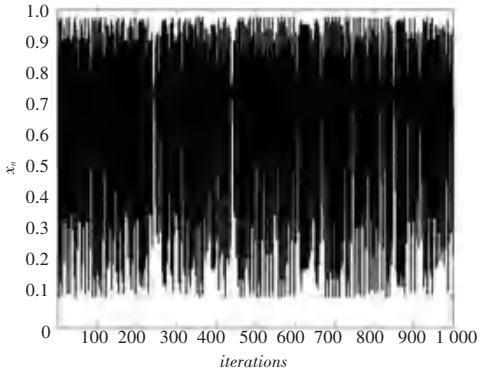

图 2 Logistic 混沌映射的时间序列图

Fig. 2 Time series plot of Logistic map

(2)Tent 映射,又称为帐篷映射。数学公式为:

$$x_{i+1} = \begin{cases} r \cdot x_n, & 0 \leqslant x_n \leqslant 0.5 \\ r \cdot (1 - x_n), & 0.5 \leqslant x_n \leqslant 1 \end{cases} \tag{10}$$

其中,r 表示控制参数,取值范围为 [0,2]。当 $r = 0.9$,初值 $x_0 = 0.4$,迭代 1 000 次后可以得到 Tent 映射的时间序列图,如图 3 所示。

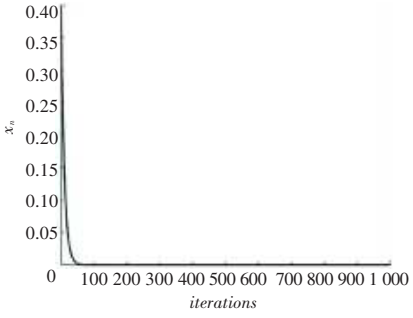

图 3 Tent 混沌映射的时间序列图

Fig. 3 Time series plot of Tent map

(3)Henon 映射。数学公式为:

$$x_{n+1} = 1 - a \cdot x_n^{\,2} + y_n \tag{11}$$

$$y_{n+1} = b \cdot x_n \tag{12}$$

其中,a 和 b 是参数,取 $a = 1.4$,$b = 0.3$,初始条件 $x_0 = 0.1$,$y_0 = 0.1$,迭代 1 000 次后得到 Henon 映射的时间序列图,如图 4 所示。

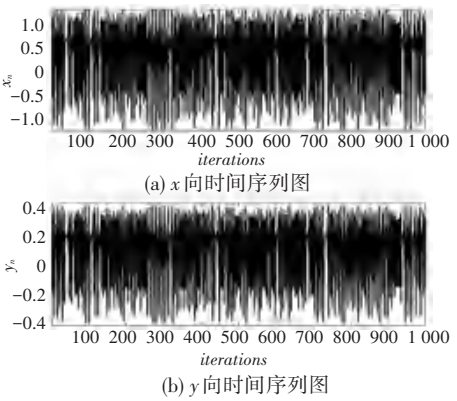

图 4 Henon 混沌映射的时间序列图

Fig. 4 Time series plot of Henon map

在图 2 中,不难发现 Logistic 映射容易在某些范围内具有较强的相关性,在边缘处不容易取值,容易导致分布不均匀;图 3 中可以看出,Tent 映射分布较为单一,随机性较弱;而 Henon 映射作为一种二维非线性混沌映射,具有较好的随机性和不可预测性。相较于 Logistic 映射和和 Tent 映射,分布更加均匀,随机性更强。因此,本文选择 Henon 映射,用于生成高质量的伪随机序列。

利用 Henon 映射产生给定区间内的 $10 \cdot N$ 个初始解后,再对每一个解按照反向学习策略生成一个反向学习解,扩大初始解的范围,提高种群的多样性。反向策略的计算公式如下:

$$X^* = ub + r \cdot (lb - X) \tag{13}$$

其中,X^* 表示任意一个解 X 反向学习解;NP 表示种群数;lb 表示区间最大值;ub 表示区间最小值;r 表示 (0,1) 之间的随机数。

接下来,将所有的初始解和反向解合并,按照适应度值进行升序(最小值)排列,选择适应度值较优的 N 个作为初始种群,并选取适应度值最小的 3 个解分别作为 α,β 和 δ。

2.2 非收敛因子改进策略

由灰狼算法的公式可以看出,A 对灰狼的全局搜索和局部搜索有很大的影响。A 是随着收敛因子 a 的变化而改变的,而收敛因子是线性递减的,初值 $a_{\text{ini}} = 2$,终值 $a_{\text{final}} = 0$。然而,GWO 算法在解决问题的过程中并非线性变化的,为了平衡灰狼优化算法的全局搜索和局部搜索能力,本文对 a 进行了如下的改进:

$$a = \begin{cases} 2 - e^{-2t}, & t \leqslant \dfrac{1}{2} t_{\max} \\ e^{-0.1(t - t_{\max})}, & t > \dfrac{1}{2} t_{\max} \end{cases} \tag{14}$$

其中,t 表示当前迭代次数,$t_{\max}$ 表示最大迭代次数。

收敛因子变化曲线如图 5 所示。由图 5 可以看出，原始收敛因子 a 的图像是线性递减的，而改进后的收敛因子 a 的图像是一条基于指数函数规律变化的曲线。从曲线的斜率可以看出，a 的值在迭代前期下降缓慢，可以使 A 较长时间地保持较大值，从而扩大搜索范围；在中期，曲线更加陡峭，a 的值变化较快，有助于提高搜索效率；而在迭代后期，a 的值下降缓慢，能够使 A 长时间保持较小值，提高算法的搜索精度。

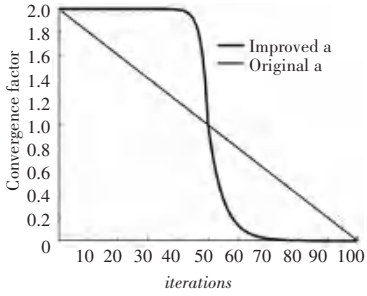

图 5　收敛因子变化图

Fig. 5　Convergence factor variation plot

2.3　引入权重调整策略

2.3.1　种群质心

种群质心[17]（Centroid of Population）是一个源于种群学和多目标优化中的概念，常用于描述种群中个体的位置分布特征。在生态学和种群学中，种群质心指的是某一物种在一定区域内所有个体的“重心”或“中心位置”[18]，即该物种在空间分布中的平均位置。在一些智能优化算法（如遗传算法[19]、粒子群优化算法[20]）中，种群质心通常代表当前种群的整体倾向，可以作为搜索的参考点，帮助引导整个种群朝着更优的区域进行探索，使算法能更快速地找到全局最优解或接近最优解，提高算法的收敛速度。本文通过设计种群质心 $C_g(t)$ 引入到灰狼算法中，以调整各灰狼的位置。其中，质心的定义公式如下：

$$C_g(t) = \sum_{i=1}^{N} R_i(t) \cdot X_i(t) \tag{15}$$

$$R_i(t) = \frac{1}{N-1} \cdot \left(1 - \frac{f(X_i(t))}{\sum_{i=1}^{N} X_i(t)}\right) \tag{16}$$

其中，$f(X_i(t))$ 表示第 i 只灰狼第 t 次迭代时的适应度值。当 $f(X_i(t))$ 越小时，式（16）中的 $R_i(t)$ 越大，$X_i(t)$ 在生成质心时占比越大。

2.3.2　权重调整策略

在 GWO 算法的迭代过程中，由于每一只灰狼在搜索时仅依靠 α，β 和 δ 狼的经验，灰狼之间缺乏合作和交流，种群的多样性也随之下降，使得狼群快速地向某一区域搜索猎物后容易陷入局部最优。为了更好地保持种群多样性并加快算法的收敛速度，本文提出了基于种群质心的位置更新方法，增加了灰狼种群内部的联系，利用潜在猎物的位置，可以知道猎物与 α，β 和 δ 狼的距离分别为多少，通过距离重新分配 3 只狼的权重，公式如下：

$$X_i(t+1) = k_1 \cdot (w_1 \cdot X_1 + w_2 \cdot X_2 + w_3 \cdot X_3) + (1 - k_1) \cdot (k_2 \cdot C_g(t) + k_3 \cdot (X_\alpha - X_i(t))) \tag{17}$$

$$\begin{cases} w_1 = \dfrac{f_\alpha}{f_\alpha + f_\beta + f_\delta} \\ w_2 = \dfrac{f_\beta}{f_\alpha + f_\beta + f_\delta} \\ w_3 = \dfrac{f_\delta}{f_\alpha + f_\beta + f_\delta} \end{cases} \tag{18}$$

其中，f_α，f_β，f_δ 分别表示与 α，β 和 δ 狼的适应度值；k_2 表示给定常数 0.002；k_1 和 k_3 表示 $(0,1]$ 之间的随机数。

3　算法步骤

综上所述，本文改进的灰狼优化算法（HGWO）步骤如下：

步骤 1　设置算法参数，种群规模 NP，最大迭代次数等参数。

步骤 2　在搜索空间内按照 Henon 混沌映射初始化灰狼种群，根据反向学习策略得到反向解，选取适应度最小的 NP 只灰狼作为初始种群。

步骤 3　计算种群中所有灰狼个体的适应度值并排序，选择最好的 3 只狼，分别记为 α，β 和 δ，记录其适应度值和位置。

步骤 4　利用新的收敛因子和位置更新公式更新所有灰狼的位置。

步骤 5　重复步骤 3、步骤 4，直至达到最大迭代次数。

4　仿真实验及分析

4.1　测试函数

一般对于群智能算法，常选用存在很多局部最优陷阱且较难找到全局最优值的函数进行算法的验证，本文选用 23 个基准测试函数对算法性能进行验证。其中，$F1 \sim F7$ 为单峰函数，$F8 \sim F13$ 为多峰函数，$F14 \sim F23$ 是给定区间内的多峰函数，表 1 给出了每个函数的表达式、维度、取值范围以及最佳值。算法独立运行 50 次，求得实验的平均值用来反映算法在给定迭代次数下所能达到的收敛精度，标准差用来反映算法的稳定性。

表 1 标准测试函数

Table 1 Standard test functions

函数	表达式	维数	范围	理论值
$F1$	$F1 = \sum_{i=1}^{n} x_i^2$	30	$[-100,100]$	0
$F2$	$F2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10,10]$	0
$F3$	$F3 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100,100]$	0
$F4$	$F4 = \max_i \{ \lvert x_i \rvert , 1 \leqslant i \leqslant 30 \}$	30	$[-100,100]$	0
$F5$	$F5 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30,30]$	0
$F6$	$F6 = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100,100]$	0
$F7$	$F7 = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	$[-1.28,1.28]$	0
$F8$	$F8 = -\sum_{i=1}^{n} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	$[-500,500]$	−12 569.5
$F9$	$F9 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12,5.12]$	0
$F10$	$F10 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32,32]$	0
$F11$	$F11 = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]$	0
$F12$	$F12 = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} \sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m, & x_i \geqslant -a \end{cases}$	30	$[-50,50]$	0
$F13$	$F13 = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(3\pi x_i)]\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	30	$[-50,50]$	0
$F14$	$F14 = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65.536,65.536]$	1
$F15$	$F15 = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$	4	$[-5,5]$	0.000 307 5
$F16$	$F16 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]$	−1.031 628 5
$F17$	$F17 = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5,10]\times[0,15]$	0.398
$F18$	$F18 = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2,2]$	3
$F19$	$F19 = -\sum_{i=1}^{4} c_i \exp\left[\left(-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right)\right]$	3	$[0,1]$	−3.86
$F20$	$F20 = -\sum_{i=1}^{4} c_i \exp\left[\left(-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right)\right]$	6	$[0,1]$	−3.32
$F21$	$F21 = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	$[0,10]$	−10.152 3
$F22$	$F22 = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	$[0,10]$	−10.402 8
$F23$	$F23 = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^{\mathrm{T}} + c_i]^{-1}$	4	$[0,10]$	−10.536 3

4.2　结果分析

一般而言，种群规模越大，算法的搜索精度越高，但是种群数目过大又会导致算法运行时间过长，本文针对 23 种测试函数设置了合适的算法参数，即：种群规模 $NP = 100$，迭代次数 $t_{max} = 100$。

为了验证每一个改进策略对算法的影响，在 GWO[8] 算法中添加了策略 1、策略 2 和策略 3 三种不同的组合。在策略 1 中，GWO 算法中只添加了 Henon 混沌映射和反向学习策略初始化种群。在策略 2 中，GWO 算法中只改变了控制参数。在策略 3 中，GWO 算法只改进了种群新的位置搜索机制。选择 7 个标准测试函数（$F1 \sim F7$）来验证每种改进策略的影响，实验结果见表 2。实验结果表明，不论是平均值、还是方差，每一个策略得出的结果均优于 GWO 算法，尤其是策略 3 在单峰测试函数 $F1 \sim F4$ 以及 $F7$ 上表现优异，结果提升了几个数量级。

表 2　不同策略在标准测试函数下的实验结果

Table 2　The experimental results of different strategies on standard test functions

函数	测试指标	GWO[8]	策略 1	策略 2	策略 3
*F*1	平均值	2.156 0e-05	3.935 0e-06	6.563 6e-06	1.225 3e-66
	方差	8.274 7e-06	3.498 2e-06	4.578 6e-06	4.677 8e-67
*F*2	平均值	9.725 5e-04	2.727 9e-04	1.149 5e-04	1.575 0e-33
	方差	3.023 6e-04	8.514 1e-05	3.624 1e-05	1.209 1e-33
*F*3	平均值	26.336 2	2.589 5e-149	2.596 7e-109	1.744 6e-118
	方差	13.278 5	8.122 2e-150	4.825 0e-109	2.460 0e-118
*F*4	平均值	0.191 4	0.080 6	0.052 2	4.315 5e-31
	方差	0.118 2	0.012 9	0.036 5	2.877 9e-31
*F*5	平均值	28.067 6	27.460 4	26.729 0	27.866 0
	方差	0.944 1	0.701 7	0.384 3	0.006 5
*F*6	平均值	1.392 7	0.730 0	0.283 3	0.078 0
	方差	0.666 1	0.305 1	0.264 5	0.255 2
*F*7	平均值	0.004 7	0.005 8	0.001 8	8.758 3e-05
	方差	0.002 3	4.453 9e-04	7.732 2e-04	9.577 3e-05

本文基于均值和标准差，对 HGWO 算法的性能与其他群智能算法性能进行比较和分析，结果见表 3。其中，HGWO 的结果以粗体突出显示。表 3 的结果表明，与基本 GWO[8] 算法、CEOGWO[14] 算法、Cat-IGWO[11] 算法、IGWO[15] 算法、SAGWO[12] 算法相比，本文提出的 HGWO 算法在 23 个基准测试函数得出的结果均能一致收敛到问题的理论最优值，并且得到的结果优于其他算法，因此，HGWO 算法在精度上具有优越性。

为了验证 HWGO 算法具有更快的收敛速度，选择 7 个标准测试函数（$F1 \sim F7$）进行实验。结果如图 6~图 12 所示。实验表明，改进的灰狼优化算法具有更好的寻优能力，在收敛速度上更有优势。

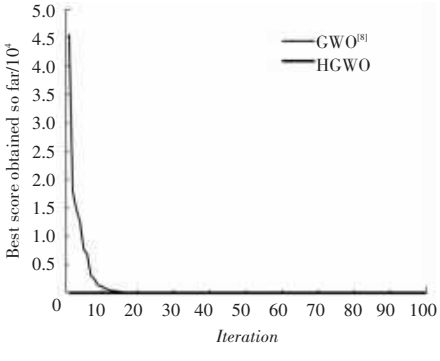

图 6　*F*1 上的收敛曲线

Fig. 6　Convergence curve of function *F*1

表 3 不同算法在标准测试函数下的实验结果

Table 3 The experimental results of different algorithms on standard test functions

函数	测试指标	GWO[8]	CEOGWO[14]	Cat-IGWO[11]	IGWO[15]	SAGWO[12]	HGWO
*F*1	平均值	2.156 0e-05	3.206 4e-04	3.834 3e-65	56.808 7	1.508 5e-37	**1.277 0e-68**
	方差	8.274 7e-06	1.312 1e-04	8.416 7e-65	25.390 8	8.950 1e-65	**1.355 4e-68**
*F*2	平均值	9.725 5e-04	0.002 0	1.570 7e-35	2.683 2	9.127 0e-22	**5.624 5e-36**
	方差	3.023 6e-04	8.149 3e-04	3.281 6e-35	0.119 2	7.504 3e-36	**2.598 9e-36**
*F*3	平均值	26.336 2	3.241 5e-74	3.048 6e-50	1.926 6e-119	2.007 80	**1.667 2e-125**
	方差	13.278 5	2.641 5e-74	4.505 3e-50	4.307 9e-119	3.200 4e-51	**2.944 8e-125**
*F*4	平均值	0.191 4	38.061 0	8.222 8e-33	2.994 6	2.429 6e-23	**6.204 2e-37**
	方差	0.118 2	18.023 6	1.544 7e-32	1.016 5	3.721 9e-30	**3.270 5e-37**
*F*5	平均值	28.067 6	58.983 7	28.767 6	876.381 8	28.750 80	**27.172 1**
	方差	0.944 1	20.278 1	0.112 0	915.652 6	0.148 10	**0.039 6**
*F*6	平均值	1.392 7	3.247 2	4.502 5	50.198 8	3.715 70	**0.649 2**
	方差	0.666 1	0.630 3	0.362 5	50.198 8	0.220 30	**0.549 6**
*F*7	平均值	0.004 7	0.016 2	1.142 2e-04	1.809 8e-04	5.944 5e-03	**2.227 3e-05**
	方差	0.002 3	0.009 7	1.075 1e-04	2.019 9e-04	5.342 1e-03	**2.843 1e-05**
*F*8	平均值	-5.935 7e+03	-5.322 3e+03	-4.130 4e+03	-2.806 0e+03	-3.607 0e+03	**-1.254 4e+04**
	方差	1.557 3e+03	645.890 5	506.824 0	583.744 1	1.207 1e+03	**2.346 5**
*F*9	平均值	17.169 7	143.527 5	0	22.714 6	0.453 33	**0**
	方差	7.090 6	77.589 2	0	8.239 8	0.346 7e-01	**0**
*F*10	平均值	9.788 7e-04	20.202 9	4.440 9e-16	2.820 1	4.440 9e-12	**3.996 8e-15**
	方差	1.523 0e-04	0.053 5	0	0.533 6	0.231 00	**0**
*F*11	平均值	0.009 2	0.412 8	0	0.533 6	0.946 7e-04	**0**
	方差	0.020 6	0.148 4	0	0.404 1	1.346 0e-01	**0**
*F*12	平均值	0.085 7	0.511 0	0.640 1	1.824 9	0.665 70	**0.032 0**
	方差	0.059 4	0.199 1	0.043 3	0.461 9	0.087 10	**0.010 7**
*F*13	平均值	0.986 0	2.144 3	2.689 6	5.968 4	3.827 00	**0.538 4**
	方差	0.230 2	0.349 3	0.125 3	2.051 3	0.059 10	**0.251 4**
*F*14	平均值	1.988 9	1.204 8	8.475 2	21.126 9	2.320 70	**1.357 9**
	方差	1.400 9	0.625 0	5.695 0	29.817 8	1.145 50	**0.381 2**
*F*15	平均值	0.004 5	8.192 2e-04	4.456 4e-04	0.030 2	3.718 5e-03	**3.075 2e-04**
	方差	0.008 9	1.542 4e-04	6.515 3e-05	0.028 6	4.330 5e-04	**8.485 0e-05**
*F*16	平均值	-1.031 6	-1.031 6	-1.031 6	-0.725 2	-1.031 50	**-1.031 6**
	方差	5.533 7e-08	7.701 5e-05	6.333 6e-05	0.436 9	5.924 5e-02	**1.587 4e-05**
*F*17	平均值	0.403 8	0.402 8	0.400 4	0.401 8	0.499 60	**0.398 1**
	方差	0.013 2	0.004 0	0.001 1	0.003 6	5.896 4e-04	**0.000 1**
*F*18	平均值	3.000 2	3.000 2	3.000 1	14.827 5	3.700 00	**3.000 0**
	方差	3.214 0e-04	2.681 8e-04	5.310 2e-05	13.985 2	2.451 4e-03	**1.499 3e-05**
*F*19	平均值	-3.861 4	-3.858 6	-3.835 3	-3.691 2	-3.823 20	**-3.859 9**
	方差	0.003 2	0.032 0	0.016 9	0.106 3	0.021 60	**0.006 2**
*F*20	平均值	-3.260 9	-3.027 5	-2.821 7	-2.246 7	-8.915 30	**-3.319 0**
	方差	0.865 0	0.223 3	0.154 7	0.623 9	0.098 80	**0.155 9**
*F*21	平均值	-10.134 7	-5.055 2	-3.769 4	-3.595 6	-3.896 30	**-10.531 3**
	方差	0.008 4	0.004 3	0.530 8	0.841 6	0.808 80	**0.001 8**
*F*22	平均值	-9.321 1	-5.087 7	-4.091 7	-3.237 7	-3.851 10	**-10.401 3**
	方差	2.366 8	0.142 0	0.611 3	0.848 8	0.480 50	**0.001 0**
*F*23	平均值	-10.513 5	-5.128 5	-4.372 0	-3.474 3	-3.668 20	**-10.530 6**
	方差	0.010 0	9.362 2e-16	0.338 6	1.379 6	0.810 90	**0.002 8**

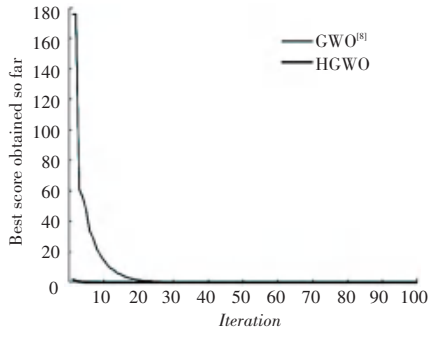

图 7　*F*2 上的收敛曲线

Fig. 7　Convergence curve of function *F*2

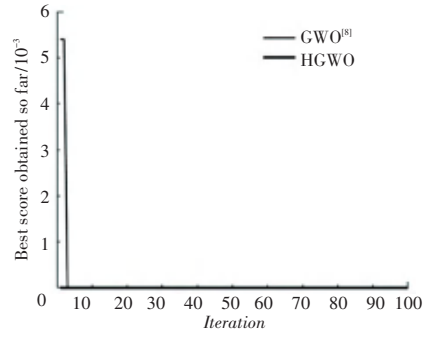

图 8　*F*3 上的收敛曲线

Fig. 8　Convergence curve of function *F*3

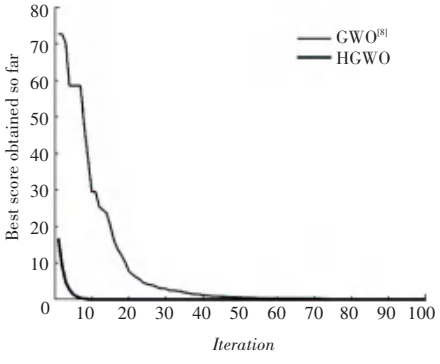

图 9　*F*4 上的收敛曲线

Fig. 9　Convergence curve of function *F*4

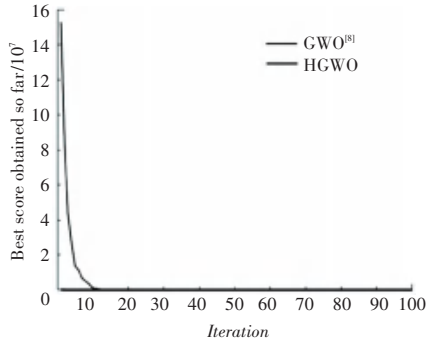

图 10　*F*5 上的收敛曲线

Fig. 10　Convergence curve of function *F*5

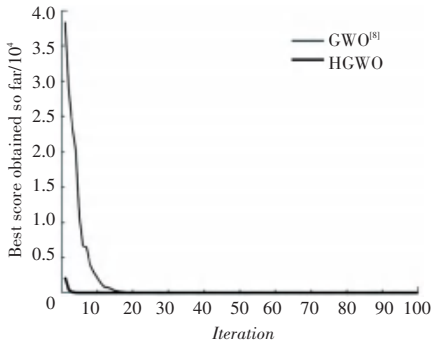

图 11　*F*6 上的收敛曲线

Fig. 11　Convergence curve of function *F*6

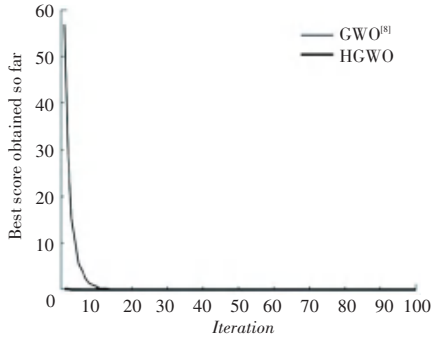

图 12　*F*7 上的收敛曲线

Fig. 12　Convergence curve of function *F*7

5　结束语

为了解决基本灰狼优化(GWO)算法存在收敛速度慢且易陷入早熟收敛等问题,本文提出一种基于 Henon 混沌映射和权重策略的改进灰狼优化算法(HGWO 算法),使得种群初始化更加均匀,在求解复杂优化问题的全局最优解时寻优精度更好。通过对 23 个标准高维测试函数进行仿真测试证实了算法是有效且可行的,并且实验结果表明, HGWO 算法在这几种测试函数上均获得了较高的寻优精度,在鲁棒性以及快速跳出局部最优等方面显著改善了算法的优化性能,能够有效地处理函数优化问题。

参考文献

[1] BENI G, WANG J. Swarm intelligence in cellular robotic systems [C]//Proceedings of NATO Advanced Workshop on Robots and Biological Systems. Cham:Springer, 1989:425-428.

[2] COLORNI A, DORIGO M, MANIEZZO V. Distributed optimization by ant colonies [C]//Proceedings of the First European Conference on Artificial Life. Amsterdam: Elsevier, 1991:134-142.

[3] KENNEDY J, EBERHART R. Particle Swarm Optimization [C]// Proceedings of International Conference on Neural Networks. Piscataway, NJ: IEEE,1995:1942-1948.

[4] EBERHART R, KENNEDY J. A new optimizer using particle swarm theory [C]// Proceedings of the 6th International Symposium on Micro Machine and Human Science. Piscataway, NJ: IEEE, 1995:39-43.
[5] 李晓磊,邵之江,钱积新. 一种基于动物自治体的寻优模式:鱼群算法[J]. 系统工程理论与实践,2002(11):32-38.
[6] KRISHNANAND K N, GHOEE D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]//Proceedings of IEEE Swarm Intelligence Symposium. Piscataway, NJ: IEEE, 2005:84-91.
[7] YANG Xinshe. A new metaheuristic bat-inspired algorithm[J]. Nature Inspired Cooperative Strategies for Optimization,2010,284:65-74.
[8] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer [J]. Advance in Engineering Software, 2014, 69(3):46-61.
[9] SAREMI S, MIRJALILI S, LEWIS A. Grasshopper optimization algorithm: Theory and application[J]. Advance in Engineering Software, 2017, 105: 30-47.
[10] XUE Jiankai, SHEN Bo. A novel swarm intelligence optimization approach: sparrow search algorithm [J]. Systems Science & Control Engineering, 2020, 8(1):22-34.
[11] 徐辰华,李成县,喻昕,等. 基于 Cat 混沌与高斯变异的改进灰狼优化算法[J]. 计算机工程与应用,2017,53(4):1-9.
[12] 魏政磊,赵辉,韩邦杰,等. 具有自适应搜索策略的灰狼优化算法[J]. 计算机科学,2017,44(3):259-263.
[13] 王秋萍,王梦娜,王晓峰. 改进收敛因子和比例权重的灰狼优化算法[J]. 计算机工程与应用,2019,55(21):60-65.
[14] 蔡娟. 混沌映射与精英高斯扰的非线性灰狼优化算法[J]. 计算机工程与设计,2022,43(1):186-195.
[15] QIU Yihui, YANG Xiaoxiao, CHEN Shuixuan. An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems[J]. Scientific Reports, 2024,14(1):14190.
[16] 汪勇, 康澳明, 艾学轶,等. 一种具有制导机制的灰狼优化算法[J]. 统计与决策,2023,39(24):35-40.
[17] 汪永生,李均利. 质心粒子群优化算法[J]. 计算机工程与应用,2011,47(3):34-37.
[18] DAS S, KODURU P, GUI M, et al. Adding local search to particle swarm optimization [C]//Proceedings of IEEE International Conference on Evolutionary Computation. Piscataway, NJ: IEEE, 1995:39-43.
[19] KATOCH S, CHAUHAN S S, KUMAR V. A review on genetic algorithm: Past, present, and future[J]. Multimedia Tools and Applicatons,2021,80(5):8091-8126.
[20] 陈寿文. 基于质心和自适应指数惯性权重改进的粒子群算法[J]. 计算机应用,2015,35(3):675-679.