文章编号：2095-2163(2020)12-0104-05 中图分类号：TP301.6 文献标志码：A

结合重心反向变异的飞蛾扑火优化算法

宋婷婷[1]，张琳娜[2]
（1 贵州大学 大数据与信息工程学院，贵阳 550025；2 贵州大学 机械工程学院，贵阳 550025）

摘 要：针对经典飞蛾扑火优化算法(MFO)在寻优过程中容易存在早熟及寻优精度低等问题，本文提出一种改进型飞蛾扑火优化算法(IMFO)。首先，使用佳点集初始化种群，使得初始种群分布更具遍历性；其次，引用惯性权重更新飞蛾位置，平衡算法的开发和探索能力；最后，采用重心反向变异策略对位置进行变异，跳出局部最优。选取 8 种测试函数进行测试，仿真结果表明：IMFO 算法具有更快的收敛效率、更强的全局寻优能力和鲁棒性。

关键词：佳点集；飞蛾扑火优化算法；惯性权重；重心反向变异；测试函数

Moth-flame Optimization Algorithm Fused with Centroid Opposition-based Mutation

SONG Tingting[1], ZHANG Linna[2]
(1 College of Big Data & Information Engineering, Guizhou University, Guiyang 550025, China;
2 College of Mechanical Engineering, Guizhou University, Guiyang 550025, China)

【Abstract】 In order to solve the problems of premature and low precision in the moth-flame optimization algorithm (MFO), an improved moth-flame optimization algorithm (IMFO) is proposed in this paper. Firstly, the good point set is used to initialize the population, which makes the initial population distribution more ergodic. Then, using inertial weights to update the moth position, balancing the development and exploration capability of the algorithm; The barycenter reverse mutation strategy is used to mutate the position to produce a new solution and jump out of the local optimum. Eight kinds of test functions were selected for testing. Simulation results show that IMFO algorithm has faster convergence, stronger global optimization ability and robustness.

【Key words】 Good point set; Moth-flame optimization algorithm; Inertia weight; Centroid opposition-based mutation; Test functions

0 引 言

飞蛾扑火优化算法(Moth-Flame Optimization, MFO)是一种仿生优化算法，由 Mirjalili 于 2015 年提出[1]，该算法具有参数较少、易于实现等优点，已经应用于解决工程、机器学习、医学和生物信息学等众多领域的优化问题[2-4]。例如：Yasir S 等提出一种基于 MFO 算法的聚类算法，用于车辆自组织网络以提升网络的可靠传输和整体性能[5]；将 MFO 算法和极限学习机结合，用于数字 X 照片的乳腺癌自动检测，对比其他算法，取得较好效果[6]。虽然该算法具有广阔的应用场景，但由于 MFO 算法存在自身空间搜索能力有限、求解时容易陷入局部最优等问题。

为了进一步提升 MFO 算法性能，本文提出一种基于融合重心反向变异的飞蛾扑火优化算法(IMFO)，该算法首先利用佳点集理论初始化种群，增加初始种群多样性；其次，引用正弦变化的惯性权重，修改位置更新公式，平衡算法的全局搜索与局部开发能力；最后，采用重心反向变异，增加种群多样性，避免算法陷入局部最优。

1 基本飞蛾扑火优化算法

飞蛾扑火优化算法本质上是一种群体智能优化算法，以飞蛾的位置表示候选解，飞蛾围绕火焰进行螺旋式位置更新，最终获得全局最优解。飞蛾种群 M 表示式(1)：

$$\boldsymbol{M}=\begin{bmatrix} m_{1,1} & \cdots & m_{1,d} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,d} \end{bmatrix}. \tag{1}$$

其中，n 为个体数量，d 为解的维度。所有飞蛾的适应度值储存在矩阵 OM 中，如式(2) 所示：

$$\boldsymbol{OM}=[OM_1 \quad OM_2 \quad \cdots \quad OM_n]^{\mathrm{T}}. \tag{2}$$

在 MFO 中，搜索空间中火焰位置矩阵 $\boldsymbol{F}$ 与飞蛾位置矩阵 $\boldsymbol{M}$ 同维，表示为式(3)：

基金项目：贵州省科学技术基金项目（黔科合基础[2020]1Y254）；国家自然科学基金(62062021，61872034)；贵州省自然科学基金(黔科合基础[2019]1064)。

作者简介：宋婷婷(1995-)，女，硕士研究生，主要研究方向：计算机应用技术、进化算法。

收稿日期：2020-10-15

$$\boldsymbol{F} = \begin{bmatrix} F_{1,1} & \cdots & F_{1,d} \\ \vdots & \ddots & \vdots \\ F_{n,1} & \cdots & F_{n,d} \end{bmatrix}. \tag{3}$$

飞蛾围绕火焰进行螺旋位置更新，公式如(4)：

$$M_i = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j. \tag{4}$$

式中，M_i 表示第 i 个飞蛾；F_j 代表第 j 火焰；$D_i = |F_j - M_i|$ 表示第 i 个飞蛾与第 j 个火焰之间的距离；b 是大于 0 的对数螺旋的常数；t 是[- 1,1] 的随机数。

火焰数量 *falme no* 在迭代过程中自适应地减少，如式(5)所示：

$$flame\ no = round\left(N - l * \frac{N-1}{T}\right). \tag{5}$$

式中，l 为当前迭代次数；N 为种群数；T 为最大迭代次数。

2　改进飞蛾扑火优化算法

2.1　佳点集初始化种群

有研究发现，初始化种群的多样性有利于提高算法寻优性能[7-8]。原始 MFO 算法随机生成初始种群，种群多样性不够，会影响算法收敛速度和精度。本文利用佳点集理论初始化种群，保证初始种群在解空间分布更加均匀，具体实现方法如下：

在 dim 维空间取 n 个点的佳点集，表示为式(6)

$$P_n(k) = \{(\{r^1 k\},\cdots,\{r^i k\},\cdots,\{r^t k\}),k = 1,2,\cdots,n\}. \tag{6}$$

其中，$r^i = \{2\cos(2\pi i/p)\}$；$1 \leqslant i \leqslant t$；$P$ 是满足 $p \geqslant 2t + 3$ 的最小素数。

2.2　引入动态惯性权重

对于群智能优化算法，能很好的协调算法的全局搜索和局部搜索能力一直是研究热点。由式(4)可知，下一代飞蛾位置围绕火焰进行更新，同时受当前飞蛾位置影响，不能很好的平衡算法的全局搜索和局部搜索能力，容易陷入局部最优。因此受 PSO 算法启发，本文在飞蛾位置更新方式中引入动态变化的惯性权重 w，新的位置更新公式为式(7)：

$$M_i = w \cdot D_i \cdot e^{bt} \cdot \cos(2\pi t) + (1 - w) \cdot F_j, \tag{7}$$

$$w = 2 \times (1 - \sin(0.5 \times \pi \times (t/t_{\max}))) \times rand(). \tag{8}$$

其中，t 为当前迭代次数；$t_{\max}$ 为最大迭代次数；$rand(\ )$ 为[0,1] 间随机数，使得搜索具有无偏性，用以提高种群多样性。随着算法迭代，$t/t_{\max}$ 变化近似于 0 到 1，使得 w 最开始较大，在迭代后期，w 逐渐趋于 0，使算法不易陷入局部最优且收敛加快。

2.3　重心反向变异

为提升种群多样性，扩大搜索范围，常用的办法就是引入变异策略，产生变异飞蛾，帮助算法逃离局部最优。本文引用重心反向学习策略，在算法迭代前期，种群个体差异较大，变异飞蛾的产生可以探索更多的区域，在算法迭代后期，种群个体数减少，变异飞蛾仍然能保留多样性[9]。重心定义如下：

设 $(m_{1j},m_{2j},\cdots,m_{nj})$ 为 n 个飞蛾在第 j 维的取值，种群个体数为 N，则飞蛾种群在第 j 维的重心为式(9)，种群重心为 $Z = (z_1,z_2,\cdots,z_j)$。

$$Z_j = \frac{x_{1j} + x_{2j} + \cdots + x_{nj}}{n}. \tag{9}$$

重心反向变异：设 $x_i = (x_{i1},x_{i2},\cdots,x_{iD})$ 为飞蛾 i，飞蛾维度是 D 维，选取的变异维度是第 j 维，则飞蛾 i 对应的重心反向解为 $x_{op\_i} = (x_{op\_i1},x_{op\_i2},\cdots,x_{op\_iD})$，由式(10)确定：

$$x_{op\_i} = 2 \times k \times Z_j - x_i. \tag{10}$$

其中，k 是收缩因子，取值为[0,1] 间的随机数。迭代过程中，对飞蛾种群的每一只飞蛾选取某一维度 j 进行变异，变异结果与上一代位置进行比较，保留较优变异。

2.4　IMFO 算法步骤

Step 1　设定算法参数：种群规模 N、维度 dim、最大迭代次数 $t_{max\_}$；

Step 2　参考式(6)佳点集理论初始化种群位置；

Step 3　计算飞蛾和火焰的适应度值，计算火焰个数，合并飞蛾和火焰，并按适应度值大小进行排序；

Step 4　利用式(7)进行飞蛾位置更新；

Step 5　根据式(10)进行重心反向变异，保存较优变异；

Step 6　若满足终止条件，则输出最优个体值和全局最优解，算法结束；否则，返回 Step 3。

3　实验结果分析

3.1　与基本算法的函数优化结果比较

为了验证 IMFO 算法的有效性和鲁棒性，本文将 IMFO 算法和 MFO 算法、GWO 算法[10]、SCA 算法[11]、PSO 算法[12]等基本的原始优化算法进行比较。用表 1 中的 8 种标准测试函数对算法进行数值检验，测试函数维度取值包含 10~200 的维度，可以更全面的检测算法性能。

表 1 8 个测试函数的基本信息

Tab. 1 Basic information of the 8 test functions

函数	名称	维度	定义域	最值
F_1	Sphere	60	[-100,100]	0
F_2	Schwefel's Problem 2.22	120	[-10,10]	0
F_3	Schwefel's Problem 1.2	10	[-100,100]	0
F_4	Schwefel's Problem 2.21	200	[-100,100]	0
F_5	Quartic	30	[-1.28,1.28]	0
F_6	Rastrigin	10	[-5.12,5.12]	0
F_7	Ackley	60	[-32,32]	0
F_8	Griewank	120	[-600,600]	0

将 5 种算法独立运行 30 次,种群规模为 30,迭代次数为 1000,维度设定见表 1,取其最优值、平均值、最差值和标准差,对 5 种算法进行比较,最优结果已经标粗,见表 2。

表 2 基准函数优化结果对比

Tab. 2 Comparison of benchmark function optimization results

	算法	最佳值	平均值	最差值	标准差
F_1	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	2.83E+01	1.11E+04	4.05E+04	1.28E+04
	GWO	3.33E-41	3.12E-39	3.64E-38	6.69E-39
	SCA	1.30E+00	4.37E+02	2.99E+03	6.56E+02
	PSO	1.15E-03	1.93E-02	1.31E-01	2.65E-02
F_2	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	1.52E+02	2.23E+02	3.29E+02	4.20E+01
	GWO	8.54E-17	2.37E-16	6.35E-16	1.19E-16
	SCA	5.04E-02	4.66E+00	2.35E+01	5.98E+00
	PSO	1.09E+01	2.51E+01	4.89E+01	8.88E+00
F_3	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	1.71E-09	5.00E+02	5.00E+03	1.53E+03
	GWO	6.70E-64	5.62E-53	1.23E-51	2.31E-52
	SCA	8.01E-17	2.07E-09	4.04E-08	7.58E-09
	PSO	8.42E-16	1.02E-11	1.94E-10	3.69E-11
F_4	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	9.45E+01	9.69E+01	9.85E+01	1.05E+00
	GWO	2.61E+00	1.30E+01	2.64E+01	6.20E+00
	SCA	8.92E+01	9.51E+01	9.75E+01	1.54E+00
	PSO	1.54E+01	1.73E+01	2.07E+01	1.37E+00
F_5	IMFO	**9.24E-07**	**5.01E-05**	**2.02E-04**	**5.58E-05**
	MFO	2.56E-02	4.15E+00	2.43E+01	7.24E+00
	GWO	1.94E-04	9.37E-04	2.81E-03	5.40E-04
	SCA	3.37E-03	2.85E-02	8.01E-02	2.15E-02
	PSO	3.43E-02	7.52E-02	1.36E-01	2.59E-02
F_6	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	3.98E+00	2.40E+01	7.18E+01	1.60E+01
	GWO	0.00E+00	3.82E-01	7.06E+00	1.49E+00
	SCA	0.00E+00	2.84E-01	8.52E+00	1.56E+00
	PSO	9.95E-01	4.01E+00	6.96E+00	1.56E+00
F_7	IMFO	**8.88E-16**	**8.88E-16**	**8.88E-16**	**0.00E+00**
	MFO	1.75E+01	1.96E+01	2.00E+01	6.89E-01
	GWO	3.29E-14	4.00E-14	5.06E-14	3.49E-15
	SCA	1.26E-01	1.67E+01	2.06E+01	7.36E+00
	PSO	3.82E-02	1.13E+00	2.18E+00	6.27E-01
F_8	IMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	MFO	1.74E+02	4.81E+02	8.77E+02	1.50E+02
	GWO	0.00E+00	3.61E-04	1.08E-02	1.98E-03
	SCA	1.04E+01	8.21E+01	2.71E+02	7.18E+01
	PSO	5.68E-02	9.84E-02	1.59E-01	2.80E-02

由表 2 数据可以看出,对于 8 种测试函数来说,IMFO 算法的最佳值和平均值均优于其他 4 种算法,说明 IMFO 算法的寻优精度和寻优能力较好。特别是对函数 F_1、F_2、F_3、F_4、F_6 和 F_8 的寻优上,最佳值、最差值、平均值、标准差都为 0,说明 IMFO 算法在 30 次运行都能达到最优值。通过设置的测试函数维度不同,如 F_1, F_4 属于单峰可分函数,F_3 属于单峰不可分函数,算法寻优精度都有所下降,但总体来看,虽然 IMFO 算法求解 F_5 没有达到理论最优值,但其寻优精度与其它算法精度相比仍能最大达到 1E - 05 级的误差。对于 F_6, F_7, $F_8$3 个多峰函数,除 F_7 外,IMFO 算法在其余两个测试函数均达到了理论最优值,并且在 F_7 函数的寻优精度上,IMFO 算法与其他算法精度相比最多达到 1E - 17 级误差。由此可见,IMFO 算法在求解单峰、多峰、高维和低维基准函数最优值有着明显优势。

另外,Wilcoxon 秩和检验是一种具有统计意义的检验方法, p 值见表 3。IMFO/MFO, IMFO/GWO, IMFO/PSO, IMFO/SCA 分别代表基准函数的 IMFO 算法与其他 4 种算法在秩和检验的 p 值。"+"、"="、"-"分别表示 IMFO 算法在对应测试函数统计结果显著性优于、相当于、弱于对比算法。表 3 的 p 值基本都小于 0.05,因此 IMFO 与其他算法之间的差异是显著的,说明改进效果的有效性。

表 3 Wilcoxon 秩和检验 p 值

Tab. 3 The p value of Wilcoxon rank sum test

函数	IMFO/MFO	IMFO/GWO	IMFO/SCA	IMFO/PSO
F_1	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_2	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_3	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_4	1.21E-12	1.21E-12	1.21E-12	1.21E-12
F_5	3.02E-11	3.69E-11	3.02E-11	3.02E-11
F_6	1.21E-12	1.61E-01	1.10E-02	1.15E-12
F_7	1.21E-12	4.51E-13	1.21E-12	1.21E-12
F_8	1.21E-12	3.34E-01	1.21E-12	1.21E-12
+/=/-	8/0/0	6/0/2	8/0/0	8/0/0

为直观比较 5 种算法的迭代效果,图 1~图 8 给出了 5 种算法分别在 8 个测试函数中的收敛曲线。可知,IMFO 算法整体在寻优精度和收敛速度上都优于其他 4 种算法。在求解 F_1、F_3 和 F_4 时, PSO 和 GWO 算法表现不佳,MFO 算法几乎与 SCA 算法重合,但在原始飞蛾扑火算法加入本文的改进策略后,IMFO 算法能明显提高算法的搜索精度,且优于其他 4 种算法,说明改进策略是有效的。由图 5 可知,其他 4 种算法在迭代前期、甚至中后期收敛曲线伴随不同程度的震荡,容易陷入局部最优,而 IMFO 的收敛精度和速度都优于其他算法。IMFO 算法求解

F_1、F_2、F_3、F_4、F_6 和 F_8 都能达到理论最优值，虽然从表 2 得出 SCA 算法在求解 F_6 时也能达到理论最优 0，但是从图 6 能看出，IMFO 算法和 SCA 算法在达到相同精度的搜索精度时，IMFO 算法在迭代前期即能收敛，说明本文改进点能有效加快算法收敛速度，尤其图 7 和图 8，收敛曲线下降速率快，具有更高的收敛精度。

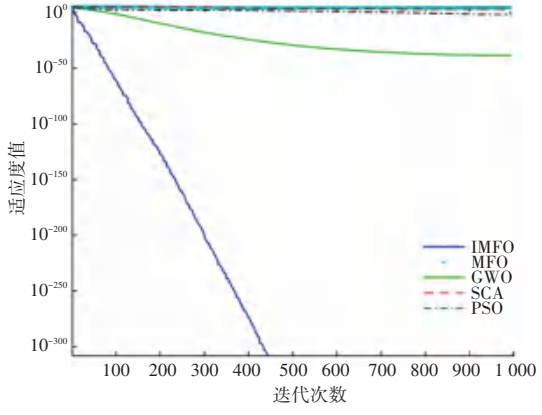

图 1　F_1 函数平均收敛曲线

Fig. 1　The average convergence curve of F_1 function

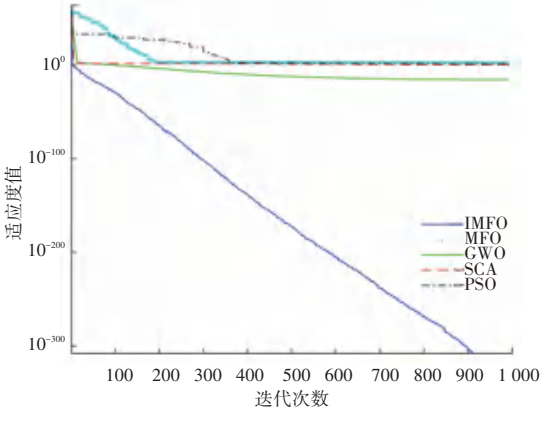

图 2　F_2 函数平均收敛曲线

Fig. 2　The average convergence curve of F_2 function

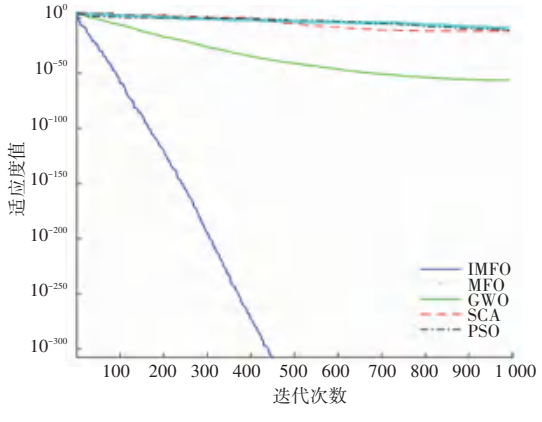

图 3　F_3 函数平均收敛曲线

Fig. 3　The average convergence curve of F_3 function

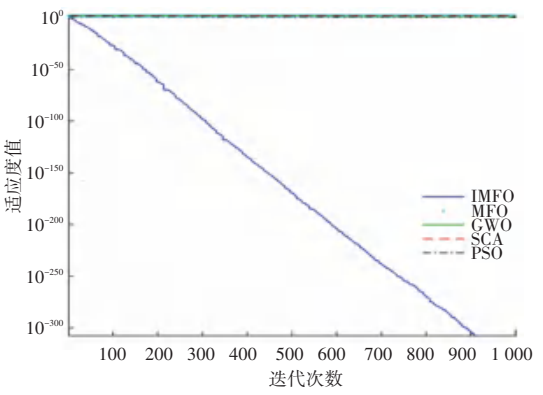

图 4　F_4 函数平均收敛曲线

Fig. 4　The average convergence curve of F_4 function

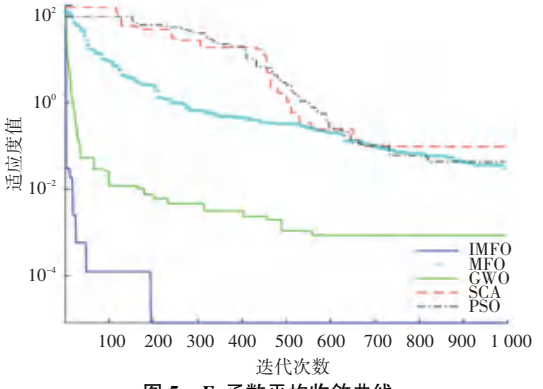

图 5　F_5 函数平均收敛曲线

Fig. 5　The average convergence curve of F_5 function

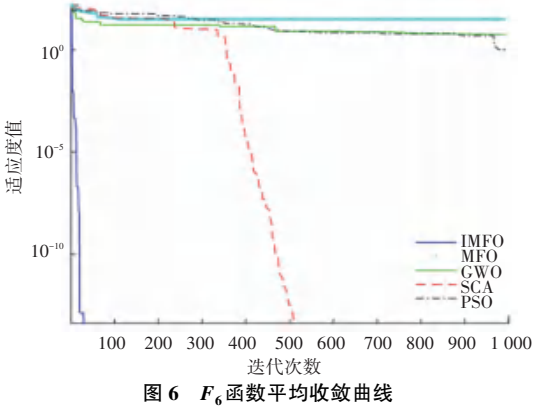

图 6　F_6 函数平均收敛曲线

Fig. 6　The average convergence curve of F_6 function

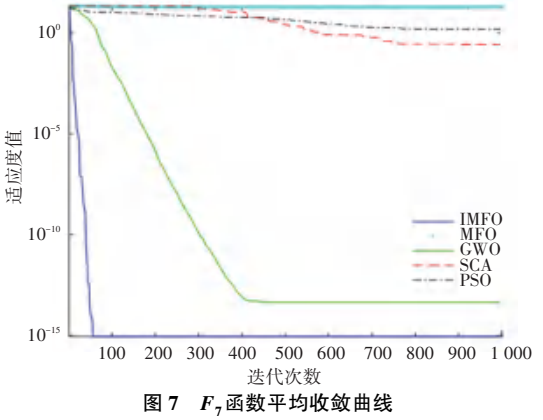

图 7　F_7 函数平均收敛曲线

Fig. 7　The average convergence curve of F_7 function

（下转第 115 页）