全局扰动和互利因子作用的飞蛾扑火优化算法

靳储蔚[1]，李姗鸿[1]，张琳娜[2]，张达敏[1+]

（1. 贵州大学 大数据与信息工程学院，贵州 贵阳 550025；
2. 贵州大学 机械工程学院，贵州 贵阳 550025）

摘 要：为解决飞蛾扑火优化（moth-flame optimization，MFO）算法收敛速度慢、容易陷入局部最优等问题，提出一种飞蛾扑火优化（DBMFO）算法。使用 Bernoulli 混沌映射，提高初始种群的多样性；引入全局扰动因子，提高算法的全局搜索能力；使用互利因子对全局扰动后的位置再次进行更新，避免新的算法陷入局部最优，使得算法更快收敛。通过对 10 个基准函数进行仿真实验，确定迭代系数的取值，通过 Wilcoxon 秩和检验来验证算法性能，其结果表明，改进的 DBMFO 算法在求解的精确度以及收敛速度上均有明显提升。

关键词：群智能算法；飞蛾扑火优化；伯努利混沌映射；全局扰动因子；互利因子；10 个基准测试函数；秩和检验

中图法分类号： TP301.6 **文献标识号：** A **文章编号：** 1000-7024（2023）08-2297-08

doi： 10.16208/j.issn1000-7024.2023.08.008

Moth-flame optimization algorithm with global disturbance factor and mutually beneficial factor

JIN Chu-wei[1], LI Shan-hong[1], ZHANG Lin-na[2], ZHANG Da-min[1+]

(1. College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China;
2. College of Mechanical Engineering, Guizhou University, Guiyang 550025, China)

Abstract: To solve the problems of low convergence speed and easiness to fall into local optimum of moth-flame optimization (MFO) algorithm, a moth-flame optimization (DBMFO) algorithm was proposed. Bernoulli chaotic map was used to improve the diversity of the initial population. A global disturbance factor was introduced to improve the global search ability of the algorithm. The mutually beneficial factor was used to update the position after the global disturbance again, so as to avoid the new algorithm from falling into the local optimum, so that the algorithm converged faster. Through the simulation experiments of 10 benchmark functions, the value of the iteration coefficient is determined, and the performance of the algorithm is verified by the Wilcoxon rank-sum test. The results show that the improved DBMFO algorithm can significantly improve the solution accuracy and convergence speed.

Key words: swarm intelligence algorithm; moth-flame optimization; Bernoulli chaotic map; global disturbance factor; mutually beneficial factor; 10 benchmark functions; rank sum test

0 引 言

群智能优化算法[1]是受自然界生物种群的运动、捕食等行为所启发而提出的一类算法，当前，国内外已经提出了大量的群智能算法，且根据没有免费的午餐（no-free-lunch，NFL）定理[2]，即过去、现在或未来都不会出现一种算法能够很好地解决所有的优化问题，因此只有不断地提出新的群智能算法才能够解决这些现实优化问题。近几年已经出现了较多新颖的群智能算法[3-7]，这类算法的使用简单方便、参数少、运行时间短等特点，为其能够广泛运

收稿日期： 2022-05-30；**修订日期：** 2023-08-07

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

作者简介： 靳储蔚（1995-），男，辽宁锦州人，硕士研究生，研究方向为计算机应用技术、优化计算、信号与信息处理；李姗鸿（1998-），女，贵州遵义人，硕士研究生，研究方向为计算机应用技术、优化计算、信号与信息处理；张琳娜（1977-），女，贵州贵阳人，副教授，研究方向为工业产品缺陷检测、机器视觉等；+通讯作者：张达敏（1967-），男，贵州贵阳人，博士，教授，研究方向为计算机应用技术、智能算法、信号与信息处理等。E-mail：1203813362@qq.com

用在实际问题上提供了先决条件。

飞蛾扑火优化（moth-flame optimization，MFO）[8]算法是 Mirjalili 等提出的新型群智能算法，其基本思想是受飞蛾在夜间飞行所启发，但同其它智能优化算法相似，MFO 算法也具有收敛速度慢、求解的精度不高无法找到理论值、容易陷入局部最优不再随着迭代的进行向理论更优的位置搜索等问题。在 MFO 算法提出之后，不断的有研究者使用 MFO 算法对工程问题进行优化，文献［9］使用 MFO 算法优化认知无线电系统的参数；文献［10］用差分进化改进的 MFO 算法求解电力系统负荷经济调度；文献［11］使用差分进化改进的 MFO 算法进行特征选择。同时也有研究者在其原始算法基础上进行改进，如文献［12］将 Lévy 飞行引入飞蛾扑火优化算法以提高算法的收敛速度和求解精度；文献［13］将融合折射原理的反向学习引入 MFO 算法，提高种群的多样性跳出局部最优解；文献［14］使用二进制编码改进 MFO 算法以解决实际问题，虽然上述改进算法的求解精度和寻优能力相对原始算法有所提升，但其收敛速度以及求解精度仍然存在着较大的改进空间，因此本文提出一种基于全局扰动和互利因子的飞蛾扑火优化算法（DBMFO）。在算法初始化阶段使用 Bernoulli 混沌映射代替原始算法中随机产生的初始种群，使得初始化的种群具有更优的多样性；在算法更新公式中增加一个全局扰动因子，增加算法的全局搜索能力，避免算法陷入局部最优；采用互利因子增加种群多样性，获得更好的最优解。

1 飞蛾扑火优化算法

飞蛾扑火优化 MFO[8]算法是模拟飞蛾在自然界的群体行为而提出的元启发式智能算法，在 MFO 算法中，飞蛾飞行的空间即为目标问题的解的空间，一只飞蛾即为目标问题的一个解，火焰的位置即是目标问题的一个较优解，虽然两者都作为 MFO 算法中的解，但两者之间存在区别，火焰的位置代表着 MFO 局部找到的一个最优位置，在每一次迭代中，飞蛾的位置可能发生改变，但火焰的位置即上一次迭代的最优位置不会随着飞蛾的飞行而消失。在 MFO 算法中，飞蛾与火焰的位置分别用矩阵表示，飞蛾和火焰的适应度值分别用向量表示，向量中按适应度值的大小对飞蛾和火焰进行排序，飞蛾位置及火焰周围空间如图 1 所示。

MFO 算法中飞蛾的螺旋线飞行更新公式如下

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

$$D_i=|F_j-M_i| \tag{2}$$

其中，M_i 为第 i 只飞蛾的位置，F_j 是第 j 个火焰的位置，D_i 为当次迭代中飞蛾与火焰之间的距离，b 为对数螺旋线系数，在本文中为 1。在更新结束飞蛾的位置之后，使用式（3）使得火焰的数量随着迭代减少

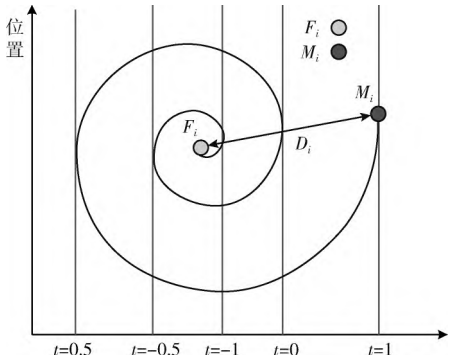

图 1 对数螺旋以及火焰周围空间

$$flame\ no=round\left(N-L\cdot\frac{N-1}{T}\right) \tag{3}$$

式中：N 为最大的火焰数量，L 为当前迭代数，T 是最大迭代次数，随着迭代的进行，火焰的数量逐步减少，在迭代末期，飞蛾仅根据最优的火焰位置更新位置。

2 改进的飞蛾扑火优化算法

2.1 Bernoulli 混沌映射

目前在群智能算法中使用的混沌映射[15]进行初始化种群，混沌有很多种，其主要使用 Logistic 混沌映射、Tent 混沌映射等[16]，但 Logistic 映射对初始参数敏感，而且遍历性较差，而 Tent 映射的混沌序列又存在小周期、不确定周期点等方面的不足。综上，本文采用 Bernoulli 混沌映射来初始化更加均匀的初始种群，相比传统初始化，使用 Bernoulli 混沌映射进行初始化可以在搜索空间中遍历得更加均匀，帮助算法更好的收敛，Bernoulli 混沌映射的定义如式（4）

$$X(i+1)=\begin{cases}\dfrac{X(i)}{1-\lambda}, & X(i)\in(0,1-\lambda]\\[2ex] \dfrac{X(i)-1+\lambda}{\lambda}, & X(i)\in(1-\lambda,1)\end{cases} \tag{4}$$

式（4）中当 λ 的值取 0.5 时，混沌性最好。由于 MFO 算法的模型简单，参数少，容易实现，而 Bernoulli 映射遍历性和随机性的特点使得算法拥有更加广泛的搜索范围。使用 Bernoulli 混沌映射代替传统 MFO 算法的随机生成初始化种群，提高了种群的多样性，使得初始的飞蛾种群分布更好，同时让改进算法有更多机会摆脱局部极值，从而在一定程度上提高了算法的性能。

2.2 全局扰动因子

传统 MFO 算法里，算法后期的收敛速度较慢，且随着火焰数量随着迭代次数不断减少，在迭代末期，算法容易受到火焰位置控制，这表明火焰位置会影响每只飞蛾的下一个位置，可知 MFO 全局搜索能力相对不足，也就是说 MFO 易陷入之前迭代的局部最优值，为了解决 MFO 算法的这个缺陷，本节引入全局扰动机制，从而增加飞蛾种群在迭代过程中的多样性以及随机性，全局扰动因子 μ 定义

如式（5）所示

$$\mu = \frac{t}{T_{\max}} \cdot \tanh\left(1 - \frac{t}{T_{\max}}\right) \cdot rand \tag{5}$$

其中，t 是当前迭代次数，$T_{\max}$ 是 MFO 算法的最大迭代次数，由式（5）可以看出全局扰动因子的影响，其扰动的趋势为迭代初期扰动力度逐渐增加，到达迭代中期扰动力度达到顶峰，此时算法的全局搜索能力最好局部搜索能力最弱，最后到迭代末期，扰动力度逐渐降低并且随着迭代次数到达最大，扰动最后消失，在这一阶段算法的全局搜索能力逐渐降低，局部搜索能力逐步提高，且由于每次迭代都会随机生成一个介于 0～1 之间的随机数作为随机系数对全局扰动因子进行影响，这更增加了全局扰动因子对 MFO 算法更新公式的影响的随机性和多样性。全局扰动因子的变化趋势如图 2 所示。

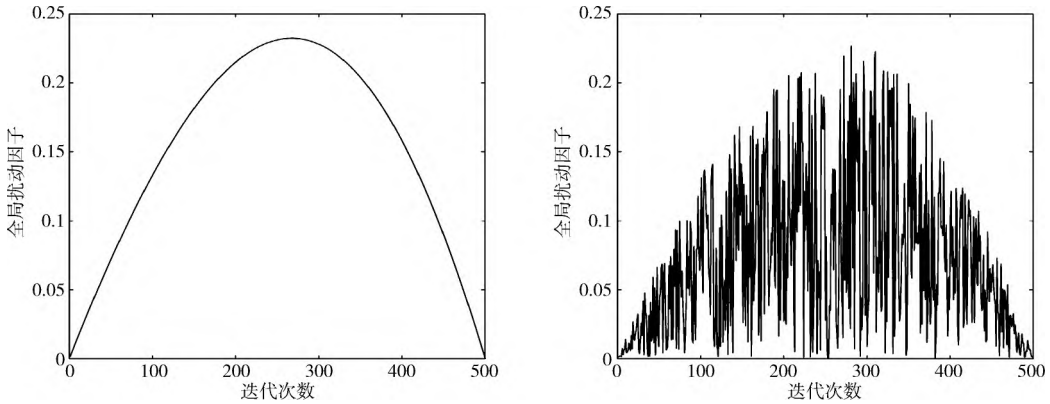

图 2　全局扰动因子 μ 变化曲线

图 2 中，左图代表没有受到随机系数影响的全局扰动因子，其曲线较为光滑，对全局扰动因子的影响缺乏随机性，右图代表受到随机系数影响的全局扰动因子，其随着迭代次数的增加，按其原有光滑曲线变化的趋势，不断波动，使得全局扰动因子更具随机性与多样性

$$S(M_i, F_j) = \mu \cdot (D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j) \tag{6}$$

式（6）为引入了全局扰动因子后的位置更新公式，与原来位置更新式（1）只使用螺旋线飞行方式产生的新飞蛾位置直接替换原飞蛾位置相比，增加了 MFO 算法的随机性和多样性，以获得比较不错的全局最优解。

2.3　互利因子

标准 MFO 算法的飞蛾的螺旋线飞行更新公式产生的新飞蛾位置直接替换原飞蛾位置，存在以下缺点：第 i 只飞蛾的位置会根据螺旋线飞行方式选择第 j 个火焰的位置和当前种群最优飞蛾位置进行更新，对随机选择的第 j 个火焰的位置个体依赖性较强，缺乏与其它个体学习的部分。在引入了 2.2 节中所提出的全局扰动因子之后，新的 MFO 算法的随机性与多样性得到了提升，但算法在对于部分测试函数的优化效果欠佳，因此引入互利因子[17]进行完善，互利因子对完成全局扰动后的位置再次进行更新，之后对改进后的新位置与改进前位置进行比较，选择其间更为优秀的位置，如下

$$\delta = \sqrt{t / T_{\max}} \tag{7}$$

$$b = \frac{M_A + M_B}{2} \tag{8}$$

$$S(M_i, F_j)_{new} = S(M_i, F_j) + \delta \cdot (F_{Best} - b \cdot r) \tag{9}$$

其中，式（7）的 δ 为收缩因子，对互利因子更新公式的影响进行收缩，保证更新后的位置处于一个合理的范围内，避免存在过大的偏差；式（8）中的 b 为互利因子，代表在搜索空间中，随机两个飞蛾的位置的共生量，即通过两个随机的位置找到一个新的位置；式（9）为引入了被收缩因子收缩之后的互利因子后的位置更新公式，r 为位于区间 1 到 2 之间的随机数，与全局扰动的 MFO 位置更新式（6）相比，引入收缩因子和互利因子，使其不再局限在前一个火焰位置搜索，即增加了飞蛾的种群多样性，以获得最好的全局最优解。

2.4　DBMFO 算法步骤

由 2.1、2.2、2.3 可得基于全局扰动和互利因子的飞蛾扑火优化算法（DBMFO）的步骤如下：

步骤 1　初始化算法参数，建立搜索空间的矩阵。

步骤 2　在搜索空间中使用 Bernoulli 混沌映射初始化飞蛾种群，计算飞蛾的适应度值并进行排序。

步骤 3　使用改进后的更新式（6）对飞蛾的位置进行更新，更新火焰位置及其适应度值。

步骤 4　使用互利因子影响的更新位置式（9）对步骤 3 的结果再一次进行更新，并判断该步骤是否使飞蛾位置更优秀，若结果更好则使用新的位置，否则保持步骤 3 的位置。

步骤 5　判断迭代次数是否达到上限，若是则停止迭代，得到最优位置以及其适应度值，否则重复执行步骤 3～步骤 5，直到满足终止迭代条件，算法流程如图 3 所示。

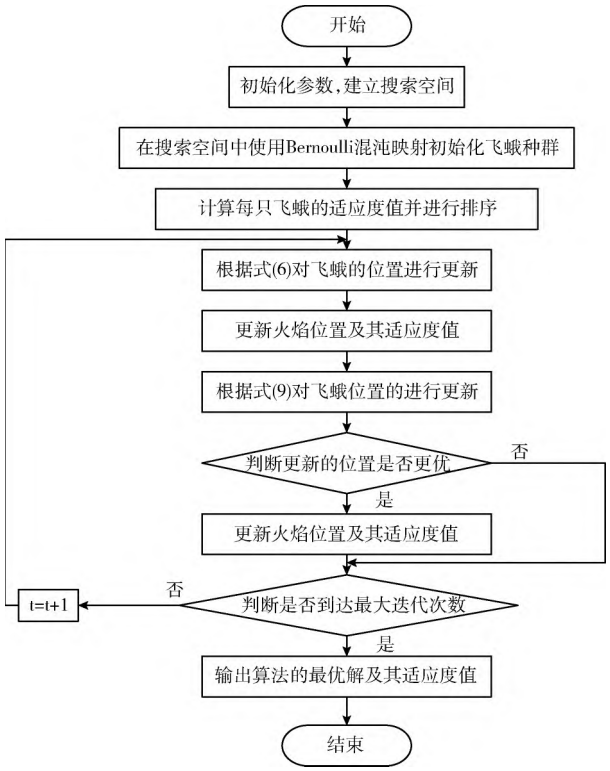

图 3 算法流程

3 仿真实验和结果分析

3.1 函数选取与参数设置

本文采用 MATLAB R2018a 进行实验仿真，运行环境为 64 位 Windows 10 操作系统，处理器类型为 Intel Core i7-6700。

为了验证本文所提算法的性能，同时验证本文提出的策略的相互关系，将本文提出的基于全局扰动和互利因子的飞蛾扑火优化算法（DBMFO）算法与仅引入全局扰动的飞蛾扑火优化（DMFO）算法、仅使用互利因子改进的飞蛾扑火优化（BMFO）、传统飞蛾扑火优化（MFO）算法、正弦余弦算法（SCA）[3]、蝴蝶优化算法（BOA）[4]、蜜獾算法（SSA）[5]进行对比实验，实验引入 10 个具有不同寻优特征的经典函数，如表 1 所示，因为 F1～F6 是单峰函数，局部最优即全局最优，用来检验本文提出的 DBMFO 算法的收敛速度和收敛精度；其中 F7～F10 是多峰函数，具有多个局部极值，尤其 F8 和 F9 的变量之间相互关联，使算法很难搜索到全局最优，用来测试算法的跳出局部最优的能力。在实验中，将使用 30 次独立运行进行实验以测试算法性能，其中，种群个数为 30，最大迭代次数为 500。另外，测试函数的维度也是影响算法寻优的一个关键因素，因此表 1 所列出的测试函数的维度从 10 到 200 不等，可以更加全面验证算法性能。

表 1 测试函数

	名称	维度	搜索区间	理论值	类型
F1	Sphere	20	[－100,100]	0	单峰
F2	Schwefel 2.22	100	[－10,10]	0	单峰
F3	Schwefel 1.2	10	[－100,100]	0	单峰
F4	Schwefel 2.21	60	[－100,100]	0	单峰
F5	Rosenbrock	30	[－30,30]	0	单峰
F6	Quartic	10	[－1.28,1.28]	0	单峰
F7	Rastrigin	200	[－5.12,5.12]	0	多峰
F8	Ackley	50	[－32,32]	0	多峰
F9	Griewank	20	[－600,600]	0	多峰
F10	Penalized	30	[－50,50]	0	多峰

实验中的 10 个基准测试函数如表 1 所示。实验中各算法的基本参数见表 2。

表 2 算法参数

算法名称	参数
DBMFO	b=1 μ=[0,0.25] r=[1,2]
DMFO	b=1 μ=[0,0.25]
BMFO	b=1 r=[1,2]
MFO	b=1
SCA	a=2
BOA	P=0.8 c=0.01 a=0.1
SSA	—

本文所提出的 DBMFO 算法同其它传统算法以及本文提出的各种策略对应的算法相比较的数据结果见表 3。

由表 3 可以看出，在与其它的传统群智能优化算法相比时，F1 到 F4 这 4 个单峰函数上 DBMFO 算法的表现都是远超过其它的传统群智能算法，且其寻优都到达了最优值，且其寻优稳定性好，在 30 次实验中均未出现个别寻优值偏离理论值的现象；在 F5 和 F6 这两个单峰函数上，虽然 DBMFO 算法没有找到测试函数的理论值，但其优化的精度相比于传统的 MFO 算法、SCA、SSA 都有至少 2 个数量级的提升，且相比于优化性能较为优秀的 BOA，本文提出的改进算法也能与之优化精度持平；在 F7 与 F9 这两个多峰函数上，DBMFO 算法同样能够找到测试函数的理论值，值得注意的是虽然函数 F7 的维度是 200 维，但 DBMFO 算法并没有因此受到过多的影响，同时在寻得理论值的同时，30 次数据的标准差为 0，即其寻优过程非常稳定；在 F8 与 F10 这两个多峰函数上，虽然 DBMFO 算法没有搜索到算法的理论值，但其相比与其它的几个传统算法有着较大的改进，其中在 F8 函数上，在 30 次实验中每一次都在迭代后期陷入了局部最优值 8.88E－16，其寻优精度相比于除 BOA 以外的其它传统算法提高了 16 个数量级，相比于 BOA，其精度提高了 7 个数量级，在 F10 函数

表 3　不同算法的结果比较

	算法	最优值	最差值	平均值	标准差	平均用时/s
	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.2089
	DMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.1537
	BMFO	1.55E−04	1.09E−02	2.24E−03	2.75E−03	0.1999
F1	MFO	1.37E−04	8.78E−03	1.29E−03	1.73E−03	0.1356
	SCA	4.93E−06	1.32E+00	5.02E−02	2.41E−01	0.2911
	BOA	1.09E−11	1.48E−11	1.27E−11	8.28E−13	0.1586
	SSA	4.51E−09	1.76E−08	8.01E−09	3.16E−09	0.2722
	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.4134
	DMFO	2.23E−171	4.02E−170	9.45E−171	—	0.3059
	BMFO	4.90E+02	6.99E+02	5.49E+02	5.58E+01	0.3868
F2	MFO	1.55E+02	3.38E+02	2.36E+02	4.11E+01	0.3058
	SCA	4.98E−01	1.58E+01	7.11E+00	4.56E+00	0.3194
	BOA	8.50E+38	1.04E+52	5.37E+50	2.02E+51	0.2498
	SSA	3.39E+01	6.96E+01	4.93E+01	9.05E+00	0.2869
	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.3437
	DMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.1761
	BMFO	1.52E−03	6.66E+03	3.92E+02	1.50E+03	0.3384
F3	MFO	1.91E−03	5.00E+03	5.00E+02	1.50E+03	0.1846
	SCA	1.80E−09	1.84E−01	8.46E−03	3.39E−02	0.1768
	BOA	9.14E−12	1.37E−11	1.12E−11	1.04E−12	0.3049
	SSA	3.45E−09	4.86E−06	2.83E−07	9.35E−07	0.2020
	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.3153
	DMFO	3.77E−168	4.45E−166	7.10E−167	—	0.2239
	BMFO	1.00E+02	1.00E+02	1.00E+02	0.00E+00	0.2731
F4	MFO	7.50E+01	9.40E+01	8.83E+01	3.57E+00	0.2267
	SCA	6.64E+01	8.39E+01	7.73E+01	4.34E+00	0.2276
	BOA	5.46E−09	7.43E−09	6.33E−09	4.04E−10	0.1845
	SSA	1.70E+01	2.96E+01	2.23E+01	3.60E+00	0.2193
	DBMFO	**2.71E+01**	**2.79E+01**	**2.75E+01**	**1.98E−01**	0.2920
	DMFO	2.73E+01	2.80E+01	2.75E+01	2.03E−01	0.1710
	BMFO	1.93E+02	9.04E+04	1.03E+04	2.72E+04	0.2800
F5	MFO	2.63E+02	9.03E+04	7.38E+03	2.22E+04	0.1990
	SCA	1.47E+02	1.90E+05	2.59E+04	3.87E+04	0.1841
	BOA	2.89E+01	2.90E+01	2.89E+01	3.10E−02	0.2537
	SSA	2.45E+01	3.78E+03	4.65E+02	9.02E+02	0.1846
	DBMFO	**7.41E−07**	**3.03E−04**	**8.02E−05**	8.44E−05	0.2852
	DMFO	5.20E−06	3.84E−04	8.04E−05	**8.18E−05**	0.1578
	BMFO	2.33E−03	2.18E−02	8.65E−03	5.18E−03	0.2812
F6	MFO	3.06E−03	2.79E−02	9.50E−03	5.21E−03	0.1563
	SCA	2.06E−04	1.21E−02	3.49E−03	3.17E−03	0.1582
	BOA	3.99E−04	1.98E−03	1.19E−03	4.05E−04	0.2507
	SSA	2.66E−03	2.86E−02	1.23E−02	6.70E−03	0.1758

表 3(续)

	算法	最优值	最差值	平均值	标准差	平均用时/s
F7	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.6463
	DMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.5317
	BMFO	4.50E+03	4.72E+03	4.59E+03	5.85E+01	0.9449
	MFO	1.93E+03	2.48E+03	2.23E+03	1.17E+02	0.5629
	SCA	8.70E+01	1.03E+03	5.31E+02	2.40E+02	0.5934
	BOA	0.00E+00	2.85E−09	1.51E−10	5.83E−10	0.4890
	SSA	7.00E+02	9.63E+02	8.40E+02	7.22E+01	0.5029
F8	DBMFO	**8.88E−16**	**8.88E−16**	**8.88E−16**	**0.00E+00**	0.3010
	DMFO	**8.88E−16**	**8.88E−16**	**8.88E−16**	**0.00E+00**	0.2144
	BMFO	1.85E+01	2.00E+01	1.97E+01	3.86E−01	0.3413
	MFO	9.52E+00	2.00E+01	1.89E+01	2.40E+00	0.2236
	SCA	5.04E+00	2.06E+01	1.71E+01	5.82E+00	0.2365
	BOA	5.03E−09	6.67E−09	5.87E−09	3.26E−10	0.2277
	SSA	3.33E+00	8.37E+00	4.99E+00	1.30E+00	0.2237
F9	DBMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.2689
	DMFO	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.1609
	BMFO	5.61E−04	9.00E+01	3.07E+00	1.64E+01	0.2901
	MFO	9.77E−04	9.02E+01	9.04E+00	2.70E+01	0.1690
	SCA	3.43E−04	6.93E−01	2.83E−01	1.97E−01	0.1699
	BOA	4.70E−13	5.74E−12	2.12E−12	1.24E−12	0.2464
	SSA	3.13E−08	6.14E−02	1.93E−02	1.69E−02	0.1881
F10	DBMFO	**3.20E−04**	**1.16E−02**	**3.86E−03**	**3.31E−03**	0.9470
	DMFO	4.89E−04	1.92E−02	4.61E−03	4.90E−03	0.4078
	BMFO	7.29E−01	9.50E+06	3.17E+05	1.93E+06	0.9579
	MFO	2.12E+00	2.56E+08	8.53E+06	4.60E+07	0.4038
	SCA	5.78E−01	1.44E+06	6.86E+04	2.66E+05	0.4443
	BOA	4.01E−01	1.00E+00	6.61E−01	1.51E−01	0.8359
	SSA	1.59E+00	1.86E+01	7.33E+00	3.40E+00	0.4133

上其精度相比于 BOA 以外的其它算法提高了至少 7 个数量级，对于 BOA 仅提高了 2 个数量级；综上，虽然相比于其它的各种传统群智能算法 DBMFO 算法对各个测试函数精度以及稳定性的提升不尽相同，但总的来说，DBMFO 算法在求解各种基准函数都具有一定的优势。

其次，在算法的改进策略里，由表 3 可以清楚地看到，仅引入全局扰动这一策略主导了 DBMFO 算法的效果，虽然 DMFO 算法相比于传统的 MFO 算法已经有了相当可观的精度提升，但其在一些单峰函数上如 F2 和 F4，单一的全局扰动策略在 500 次迭代下不足以找到测试函数的最优值，在进行进一步测试后发现，DMFO 算法在测试函数 F2 上要达到 DBMFO 算法所达到的效果在 30 次实验里最多需要 1195 次迭代，在测试函数 F4 上需要 1216 次迭代，虽然其 500 次迭代的精度超过了 E−150，但这相比于 DBMFO 的效果仍有一定的差距，若算法未来解决实际问题，这样细微的差距可能会造成不利的影响。反观 BMFO 算法，从表 3 可以看到，仅仅使用互利因子对于传统 MFO 算法的改进并不明显，BMFO 算法在多峰函数上运行的耗时非常高，且其在测试函数 F4 上，甚至陷入了局部最优，虽然 BMFO 对算法的寻优精度和寻优稳定性有着诸多的不利影响，但当互利因子与全局扰动因子相结合后，所得的 DBMFO 算法在部分函数上的寻优精度以及寻优稳定性取得了明显的提升，同时，由于全局扰动因子改进后的更新公式更新的位置有利于互利因子的应用，DBMFO 算法在高维、多峰的测试函数上的运行并没有过多的运行时间，其运行时间相比 BMFO 算法有减少。综上，全局扰动因子与互利因子的结合是有一定意义且有必要的，改进后的 DBMFO 算法有效地解决了传统 MFO 算法寻优精度、寻优速度、鲁棒性不足的缺点。

3.2 测试函数收敛曲线

根据实验所得数据，各个算法以及策略分别 10 个测试函数独立运行 30 次的平均收敛曲线如图 4 所示。

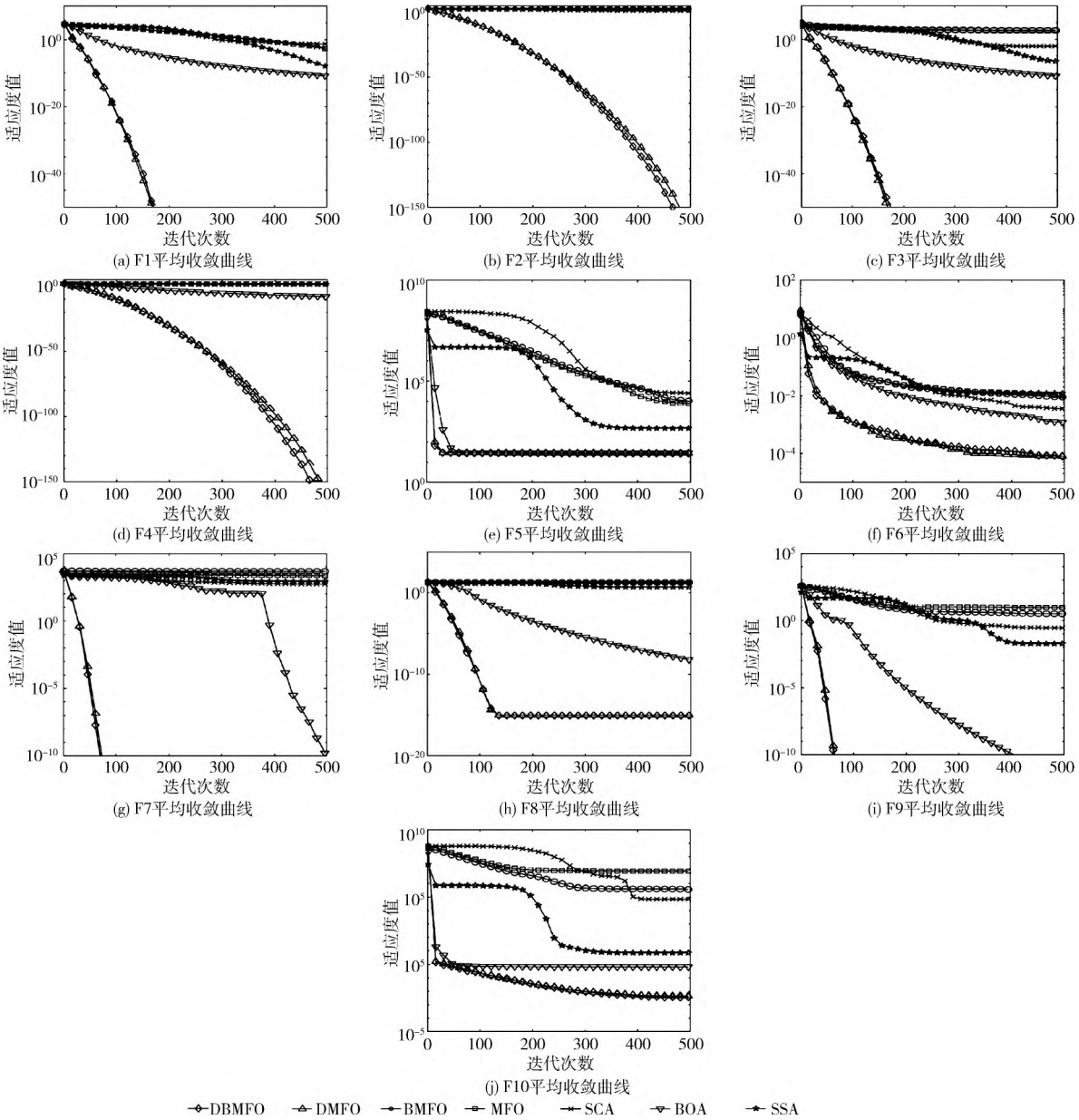

图 4　不同算法的平均收敛曲线

3.3　Wilcoxon 秩和检验

虽然在 30 次独立实验所得到的平均值与标准差虽然有一定的参考意义，但在多次实验中，可能会出现某一次实验的效果极优或极差的情况，这是实验数据的平均值与标准差无法体现的，因此在评估改进的算法的性能时，仅仅依据平均值与标准差是不够的，进行统计检验来验证本文所提出的改进算法是非常有必要的，在 5%的显著性水平下进行 Wilcoxon 秩和检验[18]从而判断 DBMFO 算法在某些特定问题上有显著的性能提升。

表 4 列出所有测试函数中 DBMFO 算法与其它算法的 Wilcoxon 秩和检验的 p 值，其中，每个数据代表 DBMFO 算法与该数据对应的算法在对应的测试函数中相比的 p 值，由于改进算法无法与算法本身进行比较，因此表中第二列均为 N/A，由于 p 小于 0.05 即可认为是拒绝零假设的有利证据，即认为“改进算法与其对比的算法是有显著区别”这一说法是错误的概率小于 5%，因此，在表中的数据越小就越能够验证改进的算法与之对比的算法区别越大，结合表 3 的数据，即可综合判断改进算法的效果，在表中唯有 DBMFO 算法与 BOA 在 F7 上的 Wilcoxon 秩和检验的 p 值大于 0.05，这是由于虽然 DBMFO 算法在 F7 上已经稳定寻找到了理论值，但传统 BOA 在 30 次独立实验中也有找到理论值，由于秩和测试的定义，两者数据的差别不太大，因此导致秩和检验的 p 值大于 0.05，除此之外，DBMFO 算法与其它算法在各个测试函数的 Wilcoxon 秩和检验的 p

值均小于 0.05，验证本文所提出的改进算法在统计上是具有优越性的。

表 4 Wilcoxon 秩和检验的 p 值

	DBMFO	MFO	SCA	BOA	SSA
F1	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F2	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F3	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F4	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F5	N/A	3.02E−11	3.02E−11	3.02E−11	1.07E−07
F6	N/A	3.02E−11	5.49E−11	3.02E−11	3.02E−11
F7	N/A	1.21E−12	1.21E−12	8.15E−02	1.21E−12
F8	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F9	N/A	1.21E−12	1.21E−12	1.21E−12	1.21E−12
F10	N/A	3.02E−11	3.02E−11	3.02E−11	3.02E−11

4 结束语

针对传统 MFO 算法的缺点，首先引进 Bernoulli 混沌映射初始化飞蛾种群，使得搜索范围内种群分布更加均匀，提高种群的多样性；然后引入全局扰动因子，增加飞蛾种群在迭代过程中的多样性和随机性，提高算法的寻优精度和寻优稳定性；最后，使用互利因子与全局扰动因子结合，进一步地提高算法的寻优精度、寻优速度以及寻优稳定性。通过对比实验证明，改进的算法在具有良好的全局以及局部寻优能力的同时，也具有较好的鲁棒性。在未来的研究中，计划将改进的算法应用于认知无线电系统的参数优化问题上，改进算法的实际应用。

参考文献：

[1] Alomari Abdullah. Swarm intelligence optimization techniques for obstacle-avoidance mobility-assisted localization in wireless sensor networks [J]. IEEE Access, 2018, 6: 22368-22385.

[2] Lockett AJ, Miikkulainen R. A probabilistic reformulation of no free lunch: Continuous lunches are not free [J]. Evolutionary Computation, 2017, 25 (3): 503-528.

[3] Mirjalili S, Gandomi AH, Mirjalili SZ, et al. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems [J]. Advances in Engineering Software, 2017, 114 (6): 163-191.

[4] ARORA S, SINGH S. Butterfly optimization algorithm: A novel approach for global optimization [J]. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2019, 23: 715-734.

[5] Hashim Fatma A. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems [J]. Mathematics and Computers in Simulation, 2022, 192: 84-110.

[6] Saremi S, Mirjalili S, Lewis A. Grasshopper optimization algorithm: Theory and Application [J]. Advances in Engineering Software, 2017, 105: 30-47.

[7] Jahani, Ehsan, Chizari, et al. Tackling global optimization problems with a novel algorithm-mouth brooding fish algorithm [J]. Applied Soft Computing, 2018, 62: 987-1002.

[8] Singh P, Prakash S. Optical network unit placement in fiber-wireless (FiWi) access network by moth-flame optimization algorithm [J]. Optical Fiber Technology, 2017, 36 (jul): 403-411.

[9] Kiranjot Kaur, Munish Rattan, Manjeet Singh Patterh. Nakagami-m fading affected multiuser cognitive radio system design using moth flame optimization [J]. Wireless Personal Communications, 2020, 112 (prepublish): 1-19.

[10] LI Rong, HE Xingshi, YANG Xinshe. Application of moth algorithm based on differential evolution in power scheduling [J]. Computer Engineering and Applications, 2021, 57 (13): 258-268 (in Chinese). [李荣，贺兴时，杨新社. 基于差分进化的飞蛾算法在电力调度中的应用 [J]. 计算机工程与应用，2021，57 (13)：258-268.]

[11] Mohamed Abd Elaziz. Opposition-based moth-flame optimization improved by differential evolution for feature selection [J]. Mathematics and Computers in Simulation, 2020, 168 (C): 48-75.

[12] LI Zhiming, MO Yuanbin. Optimization algorithm of moths to fire based on Lévy flight [J]. Computer Engineering and Design, 2017, 38 (3): 807-813 (in Chinese). [李志明，莫愿斌. 基于 Lévy 飞行的飞蛾扑火优化算法 [J]. 计算机工程与设计，2017，38 (3)：807-813.]

[13] WANG Guang, JIN Jiayi. Moth to fire algorithm based on refraction principle reverse learning [J]. Computer Engineering and Applications, 2019, 55 (11): 46-51 (in Chinese). [王光，金嘉毅. 融合折射原理反向学习的飞蛾扑火算法 [J]. 计算机工程与应用，2019，55 (11)：46-51.]

[14] Reddy SK, Panwar LK, Panigrahi BK, et al. Solution to unit commitment in power system operation planning using binary coded modified moth flame optimization algorithm (BMMFOA): A flame selection based computational technique [J]. Journal of Computational Ence, 2017, 25 (MAR): 298-317.

[15] Eneko Osaba. A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy [J]. Soft Computing, 2017, 21 (18): 5295-5308.

[16] TENG Zhijun, LV Jinling, GUO Liwen, et al. An improved algorithm for hybrid grey wolf optimization based on Tent map [J]. Journal of Harbin Institute of Technology, 2018, 50 (11): 40-49 (in Chinese). [滕志军，吕金玲，郭力文，等. 一种基于 Tent 映射的混合灰狼优化的改进算法 [J]. 哈尔滨工业大学学报，2018，50 (11)：40-49.]

[17] WANG Yirou, ZHANG Damin, FAN Ying. Mutually beneficial adaptive satin-blue gardener bird optimization algorithm for non-uniform variation [J]. Computer Engineering and Science, 2020, 42 (12): 2233-2241 (in Chinese). [王依柔，张达敏，樊英. 非均匀变异的互利自适应缎蓝园丁鸟优化算法 [J]. 计算机工程与科学，2020，42 (12)：2233-2241.]

[18] RSMD Barros, Hidalgo J, Cabral D. Wilcoxon rank sum test drift detector [J]. Neurocomputing, 2018, 275: 1954-1963.