新型飞蛾火焰优化算法的研究

田 鸿[1],陈国彬[2],刘 超[3]
1.重庆人文科技学院,重庆 401524
2.重庆工商大学 融智学院,重庆 400033
3.贵州航天电器股份有限公司,贵阳 550009

摘 要:飞蛾火焰优化算法(Moth-Flame Optimization,MFO)是一种自然激励且易于实现的全局优化算法,在许多实际优化任务中表现出良好的性能。然而,MFO算法存在早熟收敛和容易陷入局部最优解的问题,针对这些不足,提出了一种Kent混沌动态惯性权值的改善飞蛾火焰优化算法(Ameliorative MFO,AMFO)。在AMFO算法中,引入Kent混沌映射搜索策略帮助当前最优解跳出局部最优;采用基于适应度值和迭代次数的动态惯性权值策略来平衡算法的开发和探索能力,以进一步提升MFO算法性能。在8个经典benchmark函数上验证AMFO算法的搜索精度和性能,并将其结果与标准飞蛾火焰优化算法、粒子群算法和差分进化算法进行比较,仿真结果表明AMFO算法具有较好的搜索性能。

关键词:群智能;飞蛾火焰优化算法;Kent混沌;动态惯性权值;数值函数
文献标志码:A **中图分类号:**TP18 doi:10.3778/j.issn.1002-8331.1805-0027

田鸿,陈国彬,刘超.新型飞蛾火焰优化算法的研究.计算机工程与应用,2019,55(16):138-143.
TIAN Hong, CHEN Guobin, LIU Chao. Research on new moth-flame optimization algorithm. Computer Engineering and Applications, 2019, 55(16):138-143.

Research on New Moth-Flame Optimization Algorithm

TIAN Hong[1], CHEN Guobin[2], LIU Chao[3]
1.Chongqing College of Humanities, Science & Technology, Chongqing 401524, China
2.Rongzhi College, Chongqing Technology and Business University, Chongqing 400033, China
3.Guizhou Aerospace Electronics Co., Ltd., Guiyang 550009, China

Abstract: Moth-Flame Optimization (MFO) algorithm, which is inspired by social behaviors of individuals in moth flame, is a nature-inspired and easy to implement global optimization algorithm. The MFO has shown good performance for many real-world optimization tasks. However, MFO has problems with premature convergence and easy trapping into local optimum solutions. In order to overcome these deficiencies, an Ameliorative MFO (AMFO) algorithm based on Kent chaotic dynamic inertia weight is proposed. In the AMFO algorithm, a Kent chaotic map search strategy is introduced to help the current optimal solution jump out of the local optimal solution. In addition, the dynamic inertia weight strategy based on fitness value and iteration number is used to balance the development and exploration ability and further improve the performance of MFO algorithm. The search accuracy and performance of AMFO algorithm are verified on 8 classical benchmark functions. The experimental results show that AMFO technique has superior search performance compared with standard MFO algorithm, particle swarm optimization algorithm and differential evolution algorithm.

Key words: swarm intelligence; moth-flame optimization algorithm; Kent chaotic; dynamic inertia weight; numerical function

基金项目:国家自然科学基金(No.61403331,No.61573306);重庆市物联网产业共性关键技术创新主题专项项目(No.cstc2015zdcy-ztzx40007)。

作者简介:田鸿(1972—),女,副教授,研究方向为智能算法、计算机网络、智能信息处理;陈国彬(1982—),男,博士研究生,讲师,主要研究方向为网络服务质量评价;刘超(1986—),通讯作者,男,博士,高级工程师,主要研究方向为人工智能技术、复杂工业系统的智能建模及优化控制。

收稿日期:2018-05-02 **修回日期:**2018-06-18 **文章编号:**1002-8331(2019)16-0138-06
CNKI **网络出版:**2018-10-26, http://kns.cnki.net/kcms/detail/11.2127.tp.20181025.0944.004.html

1　引言

近年来,最优化问题在统计物理、计算机科学、人工智能、模式识别等许多实际应用中经常遇到。在求解复杂非凸优化问题时,传统方法求解费时费力且极其困难。过去20年出现的大量群智能优化算法被认为是最有效的,一些著名的进化算法包括遗传算法(Genetic Algorithm,GA)[1]、粒子群优化(Particle Swarm Optimization,PSO)[2]、差分进化(Differential Evolution,DE)[3]。最近提出的优化算法包括磷虾群(Krill Herd,KH)[4]、基于生物地理学的优化(Biogeography Based Optimization,BBO)[5]、灰狼优化算法(Grey Wolf Optimization,GWO)[6]和飞蛾火焰优化算法(Moth-Flame Optimization,MFO)[7]等。这些群体智能算法能够解决许多实际的和具有挑战性的工程优化问题。然而,一种算法在解决特定问题时非常有效,却在解决另一组问题时容易陷入局部最优。

受自然界飞蛾导航启发,Mirjalili在2015年提出了飞蛾火焰算法(MFO)[7]。它是以飞蛾行为为基础,通过一种特殊的称作横向定向导航机制实现了勘探与开发的较好平衡,以获得全局优化性能。由于MFO的优良特性,近年来引起了众多研究者的关注[8-10]。文献[11]已成功地应用MFO来提取用于多晶硅太阳能电池/模块的三二极管模型的参数。实验结果表明,MFO算法在三种模型的参数提取过程中明显优于其他算法。文献[12]在机器学习领域提出了一种改进的基于探索/开发速率控制的飞蛾火焰优化算法(IMFO),用于特征选择。对比研究表明,IMFO能够很好地选择特征。文献[13]提出了Lévy飞行的飞蛾火焰算法,并成功地解决了工程优化问题。改进的MFO算法在特定的应用场合取得了良好效果,然而如Li等人所述,MFO的性能还有很大的改进空间[13]。在求解复杂高峰函数优化问题时,针对MFO易陷入局部最优而导致全局搜索精度不高的问题,设计一种新的改进MFO算法以提升MFO全局优化性能是必要的。

为了进一步提升MFO算法的全局优化性能,提出一种Kent混沌动态惯性权值的改善飞蛾火焰优化算法(Ameliorative MFO,AMFO)。在AMFO中,采用Kent混沌搜索策略对陷入局部最优的个体进行彻底搜索,增加其跳出局部最优的可能。其次,采用基于适应度值和迭代次数的动态惯性权值策略来平衡算法的开发和探索能力,以进一步提升MFO算法性能。为了验证AMFO算法的有效性,采用8个经典的基准函数对AMFO算法的搜索性能进行测试。与MFO算法、粒子群算法、差分进化算法进行对比,仿真结果验证了AMFO算法在复杂函数优化问题中具有良好的优化性能。

2　飞蛾火焰优化算法

2.1　种群初始化

在MFO算法中,飞蛾个体为优化问题的候选解,飞蛾在优化空间的位置代表求解优化问题的变量,通过在优化空间中改变位置向量来向全局最佳点靠拢。MFO算法的种群 M 由下列矩阵描述:

$$\boldsymbol{M}=[\boldsymbol{m}_1,\boldsymbol{m}_2,\cdots,\boldsymbol{m}_n]^{\mathrm{T}} \tag{1}$$

$$\boldsymbol{m}_i=[m_{i,1},m_{i,2},\cdots,m_{i,d}]^{\mathrm{T}}$$

式中,n 为飞蛾数量,d 为优化问题维数。飞蛾个体适应度值存储在 OM 矩阵中:

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

火焰为当前迭代所获得的最佳位置,式(3)描述了最优位置矩阵 F,其适应度值存放在 OF 中。

$$F=[f_1,f_2,\cdots,f_n]^{\mathrm{T}} \tag{3}$$

$$\boldsymbol{f}_i=[f_{i,1},f_{i,2},\cdots,f_{i,d}]^{\mathrm{T}}$$

$$OF=[OF_1\ OF_2\ \cdots\ OF_n]^{\mathrm{T}} \tag{4}$$

2.2　位置更新机制

飞蛾是一种奇特的昆虫,已经进化成利用月光在夜间飞行,并采用横向定位机制进行导航。飞蛾容易被人造光所欺骗,通常观察到飞蛾在灯光周围螺旋飞行。受自然界飞蛾行为启发提出了MFO算法,飞蛾个体围绕在火焰周围迭代更新其位置直到搜索到最佳解决方案为止,其数学描述分为捕焰行为和弃焰行为。

(1)捕焰行为。自然界中具有趋光特性的飞蛾 M_i 会朝着距离自身最近的亮光(火焰) F_j 移动[8]。选择式(5)描述的对数螺线作为飞蛾捕焰的移动轨迹:

$$S(M_i,F_j)=D_i\times \mathrm{e}^{bt}\times\cos(2\pi t)+F_j \tag{5}$$

式中,$S(M_i,F_j)$ 为更新后的飞蛾位置;b 为与螺线形状相关的常量;t 为随机数,取值区间为 $[-1,1]$,$t=-1$ 是最接近火焰,$t=1$ 是离火焰最远;$D_i=\left|F_j-M_i\right|$ 为飞蛾 M_i 到火焰 F_j 的距离。

(2)弃焰行为。MFO算法通过弃焰行为自适应减少火焰数量,直到保持一个最优的火焰位置为止,火焰减少过程如式(6)所示:

$$no_{\mathrm{flame}}=round\left(N-t\times\frac{N-1}{T}\right) \tag{6}$$

式中,t 和 T 分别为当前和最大迭代次数;N 为最大火焰数量。

3　改进的飞蛾火焰优化算法

非免费午餐优化定理表明没有一种优化算法可以解决所有的优化问题,MFO算法同样面临上述问题。MFO在处理复杂函数问题时易发生早熟收敛并陷入局部最优,因此需要对其进行改进,提高性能。混沌是指一个具有遍历性的非重复混沌动力学系统。Kent混沌

与Logistic混沌是同构的,已经证实二者能够以良好的分布进行准确搜索。文献[14-16]比较了二者的遍历性:Logistic概率分布中间均匀、两端偏高,造成Logistic映射遍历不均匀;而Kent映射在各区间呈均匀分布,良好的类随机性、遍历性和均匀分布特性有利于MFO算法在局部最优解周围进行挖掘以搜索到更优的解。除此之外,权值参数对MFO算法的全局和局部搜索有着重要影响,单一不变的固定权值将制约权值调整功能的发挥。为进一步提高收敛精度,除了考虑算法所处的迭代阶段以外,还应该将迭代过程中飞蛾的适应度值考虑进去,即提出一种与迭代阶段和飞蛾适应度值共同决定的动态惯性权值调整策略。

3.1 Kent混沌搜索策略

Kent混沌映射模型描述如下:

$$Z^{t+1}=\begin{cases}Z^t/a, 0<Z^t\leqslant a\\(1-Z^t)/(1-a), a<Z^t<1\end{cases} \tag{7}$$

式中,控制参数 $a\in(0,1)$,Kent映射的Lyapunov指数大于0,映射处于混沌状态。本文取 $a=0.4$,其概率密度函数在(0,1)内服从均匀分布,即 $\rho(Z)=1$。初始状态微小不确定性的发散比率可以由Lyapunov指数描述[14],此时Kent混沌的Lyapunov指数为0.696,大于经典Logistic的0.691。

在MFO算法中,假设经过连续 $limit$ 次迭代搜索后解没有得到明显改善,表明这个解陷入局部最优,因此采用Kent对其进行混沌优化。对MFO算法的当前最优解进行混沌优化,优化问题的解空间为 $[X_{\min},X_{\max}]$,Kent混沌优化步骤如下:

(1)利用式(8)将当前最优解 M_i 映射到Kent方程的定义域[0,1]内:

$$Z^0=\frac{M_i-M_{\min}}{M_{\max}-M_{\min}} \tag{8}$$

(2)生成混沌序列。用Kent方程迭代产生 $C_{\max}$ 个混沌变量序列 $Z_k(k=1,2,\cdots,C_{\max})$;

(3)利用载波操作先将 Z_k 放大,然后再加载到待搜索的个体 M_i 上,从而得到混沌算子操作后的原解空间的领域内的新的个体位置 U_k,其中 $k=1,2,\cdots,C_{\max}$:

$$U_k=M_i+\frac{M_{\max}-M_{\min}}{2}\times(2Z_k-1) \tag{9}$$

(4)计算 U_k 的适应度值 $f(U_k)$,并与 M_i 的适应度值比较,保留最好解。

3.2 动态惯性权值

惯性权值参数对算法的全局和局部搜索有着重要影响。MFO算法中权值大小决定了上次迭代中火焰个体对本次迭代中飞蛾个体的影响程度。迭代初期,希望有较高的全局搜索能力以探索新的解空间,跳出局部极值;迭代后期,则重视局部开发以加快收敛并发现精确解。若MFO算法采用线性递减惯性权值,对于复杂高维函数,当迭代次数较大时,每次迭代的惯性权值的变化量较小,将影响权值调整策略功能的发挥;同时,单一的变化模式,在迭代后期,飞蛾在陷入局部最优后将很难飞出。基于上述分析,除了考虑算法所处的迭代阶段以外,还应该将飞蛾的适应度值考虑进去,即权值大小由迭代次数和飞蛾的适应度值共同决定。动态惯性权值 w 描述如下:

$$w_{i,j}=\frac{\exp(f(j)/\mu)}{2.4+(\exp(-f(j)/\mu))^{iter}} \tag{10}$$

式中,μ 为第一次寻优过程的平均适应度值;$f(j)$ 为第 j 个飞蛾的适应度值;$iter$ 表示当前迭代次数。针对极小值优化问题,设置飞蛾优化解对应的函数值为适应度值,随着迭代次数增加,$f(j)$ 呈现非线性减小趋势,$2.4+(\exp(-f(j)/\mu))^{iter}$ 呈现非线性增加趋势,故权值 w 随着适应度值和迭代次数呈现非线性减小趋势。图1给出了动态惯性权值变化曲线。

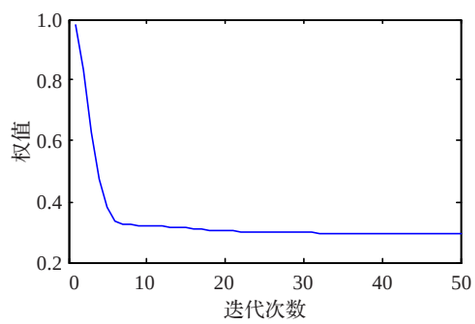

图1 动态惯性权值曲线

从图1中可以看出,开始迭代阶段,w 值接近1,随着迭代次数和适应度值的变化,迅速减小,在迭代次数为40时,w 值接近并稳定在0.294附近。改进后的位置更新公式如式(11)所示:

$$S(M_i,F_j)=w_{i,j}\times D_i\times e^{bt}\times\cos(2\pi t)+(1-w_{i,j})\times F_j \tag{11}$$

动态惯性权值随着迭代次数和适应度值非线性动态变化,人工飞蛾逐渐向适应度值较优的火焰运动,利于MFO算法的性能提升。

3.3 AMFO算法

基于上述描述,AMFO算法流程描述如下:

(1)初始化算法参数:飞蛾数量 n、维数 d、最大迭代次数 T 等;

(2)种群初始化,按式(1)格式初始化种群;

(3)计算种群中人工飞蛾和人工火焰的适应度值,并按适应度值排序;

(4)按照3.1节内容更新第一个火焰 F_1(当前为止最优解);

(5)利用式(6)更新火焰数量;

(6)按式(10)更新动态惯性权值;

(7)计算 $D_i=|F_j-M_i|$,按式(11)更新飞蛾位置;

(8)若满足终止条件，算法结束，获得最优解，否则返回步骤(3)。

4　AMFO算法性能测试

表1中8个经典测试函数用来验证AMFO算法性能，其中 f_1～f_5 为单峰函数；f_6～f_8 为多峰函数。Rastrigin函数具有大量的局部最优值，难以找到全局最优值；Ackley算法函数具有一个狭窄的全局最优池和许多较小的局部最优值，寻优困难；Griewank函数具有许多较小的局部最优值，难以达到理论最优值。为了说明AMFO算法的有效性，将AMFO与MFO算法、PSO算法和DE算法进行比较。PSO算法设置 $c_1=c_2=1.5$，$w'=0.9$，$w''=0.4$；DE算法设置 $F=0.6$，$Cr=0.85$；MFO和AMFO算法设置 $b=1$。四种优化算法种群规模为40，迭代次数为1 000。对于表1的8个测试函数，维度依次设置为10、30和50。四种算法均独立运行30次，记录并保存最佳解的平均值(Mean)和标准差(S.D.)。表2记录了PSO、DE、MFO和AMFO四种优化算法比较结果，表中最好结果用粗字体表示。

从表2中可以看出，PSO、DE和MFO算法性能相当，AMFO算法最优。针对高维单峰函数 f_1～f_5，AMFO搜索结果非常接近理论最优值；随着维数的上升，AMFO搜索的精度依然很高，标准差是四种优化算法中最小的，优化结果也更加稳定。虽然AMFO对 f_4、f_5 测试函数寻找到的最优解没有达到理论最优值，但是较其他三种算法仍具有很大的优势。对于多峰函数 f_6 和 f_8，MFO算法和PSO算法找到的最优解精度很低，DE算法仅在 f_6 为10维时搜索到理论最优值，而AMFO算法在10、30、50维时都能找到理论最优值，表明提出的混沌动态惯性权值策略能够使MFO算法跳出局部最优并实现全局最优。对于多峰函数 f_7，AMFO算法的优化精度最高，且针对10、30、50维情况下，AMFO算法优化结果非常稳定，均为 8.881×10^{-16}。比较AMFO与标准MFO，针对单峰函数 f_3，在10、30、50维情况下，AMFO算法由标准MFO优化的 1.472×10^{-7}、1.789×10^{4}、5.027×10^{4} 提升到 2.387×10^{-154}、8.231×10^{-81}、1.582×10^{-60}，且标准差也是最小的；针对多峰函数 f_8，在10、30、50维情况下，

表1　测试函数

序号	函数	范围	最优解
f_1	Sphere	$[-100,100]$	0
f_2	Schwefel Problem2.22	$[-10,10]$	0
f_3	Schwefel Problem1.2	$[-100,100]$	0
f_4	Schwefel Problem2.21	$[-100,100]$	0
f_5	Quartic	$[-1.28,1.28]$	0
f_6	Rastrigin	$[-5.12,5.12]$	0
f_7	Ackley	$[-32,32]$	0
f_8	Griewank	$[-600,600]$	0

表2　比较PSO、DE、MFO和AMFO结果

序号	维数	PSO		DE		MFO		AMFO	
		平均值	均方差	平均值	均方差	平均值	均方差	平均值	均方差
f_1	10	3.208×10^{-9}	4.702×10^{-8}	1.603×10^{-25}	1.304×10^{-25}	6.784×10^{-31}	6.784×10^{-30}	**8.264×10^{-192}**	0
	30	2.056×10^{-2}	1.521×10^{-2}	2.256×10^{-8}	9.025×10^{-9}	3.548×10^{3}	5.876×10^{3}	**2.472×10^{-107}**	1.118×10^{-106}
	50	3.665×10^{-1}	8.256×10^{-2}	1.528×10^{-2}	3.329×10^{-2}	8.032×10^{3}	7.696×10^{3}	**2.281×10^{-94}**	1.012×10^{-93}
f_2	10	2.982×10^{-3}	3.799×10^{-3}	8.643×10^{-15}	3.982×10^{-15}	5.824×10^{-19}	6.775×10^{-16}	**2.465×10^{-100}**	1.557×10^{-99}
	30	1.296	1.867×10^{-1}	2.745×10^{-5}	2.445×10^{-6}	3.107×10^{1}	2.621×10^{1}	**8.023×10^{-66}**	5.102×10^{-65}
	50	4.345	2.063	5.071×10^{-2}	1.056×10^{-2}	5.946×10^{1}	2.966×10^{1}	**3.468×10^{-59}**	8.001×10^{-59}
f_3	10	2.134×10^{-6}	2.782×10^{-6}	7.379×10^{1}	2.203×10^{1}	1.472×10^{-7}	5.162×10^{-7}	**2.387×10^{-154}**	1.032×10^{-153}
	30	2.568	9.267×10^{-1}	2.029×10^{4}	2.448×10^{3}	1.789×10^{4}	1.464×10^{4}	**8.231×10^{-81}**	3.612×10^{-80}
	50	3.458×10^{1}	1.557×10^{1}	6.667×10^{4}	5.311×10^{3}	5.027×10^{4}	1.973×10^{4}	**1.582×10^{-60}**	7.083×10^{-60}
f_4	10	2.374×10^{-3}	4.018×10^{-3}	3.125×10^{-3}	1.617×10^{-3}	1.995×10^{-1}	3.823×10^{-1}	**9.633×10^{-89}**	4.363×10^{-87}
	30	8.047×10^{-1}	7.965×10^{-1}	7.265	6.758×10^{-1}	3.286×10^{1}	1.311×10^{1}	**2.578×10^{-48}**	3.451×10^{-48}
	50	3.714	5.003×10^{-1}	2.017×10^{1}	5.338	5.779×10^{1}	5.479	**1.022×10^{-39}**	1.114×10^{-38}
f_5	10	1.028×10^{-3}	1.663×10^{-3}	1.661×10^{-2}	5.198×10^{-3}	5.024×10^{-3}	3.501×10^{-3}	**4.362×10^{-5}**	3.012×10^{-5}
	30	3.129×10^{-2}	2.068×10^{-2}	1.625×10^{-1}	3.029×10^{-2}	2.502	6.153	**1.487×10^{-5}**	1.372×10^{-5}
	50	6.596×10^{-1}	1.167×10^{-1}	4.246×10^{-1}	8.459×10^{-2}	1.921×10^{1}	1.531×10^{1}	**1.632×10^{-4}**	1.087×10^{-4}
f_6	10	8.258	3.697	0	0	2.398×10^{1}	1.524×10^{1}	**0**	0
	30	4.201×10^{1}	6.893	4.301×10^{-1}	5.518	1.488×10^{2}	3.231×10^{1}	**0**	0
	50	6.331×10^{1}	5.597×10^{1}	2.412×10^{2}	12.814	2.943×10^{2}	5.739×10^{1}	**0**	0
f_7	10	2.568	6.557×10^{-1}	4.561×10^{-13}	1.053×10^{-13}	8.356×10^{-15}	4.673×10^{-14}	**8.881×10^{-16}**	0
	30	4.137	5.453×10^{-1}	3.936×10^{-5}	4.389×10^{-6}	3.479×10^{-2}	5.562×10^{-3}	**8.881×10^{-16}**	0
	50	8.074	1.259	1.774×10^{-1}	3.026×10^{-2}	1.036×10^{1}	1.673	**8.881×10^{-16}**	0
f_8	10	4.937	1.634	1.189×10^{-5}	5.232×10^{-5}	1.412×10^{-1}	7.219×10^{-2}	**0**	0
	30	4.369×10^{1}	3.467	1.446×10^{-6}	2.806×10^{-6}	1.362×10^{1}	3.306×10^{1}	**0**	0
	50	1.067×10^{1}	5.691	4.358×10^{-2}	2.027×10^{-2}	7.762×10^{1}	6.028×10^{1}	**0**	0

AMFO算法由标准MFO优化的1.412×10^{-1}、1.362×10^{1}、7.762×10^{1}提升到0、0、0,标准差为0,依然最小。上述数据表明,经过Kent混沌动态惯性权值改善的AMFO算法较标准MFO算法在优化精度和稳定性上有了极大提升。

为了直观比较四种算法的性能,图2~图9给出四种算法在8个测试函数维数为30时的收敛曲线。明显地可以发现,AMFO算法较其余三种算法有更好的优化精度。图7和图9直观显示出AMFO算法收敛到全局最优解,而MFO、DE、PSO三种算法均没有搜索到全局最优。比较AMFO和MFO曲线,AMFO算法因采用了Kent混沌策略和动态惯性权值策略,表现出更好的收敛精度,其全局优化性能有了显著提升。

在AMFO算法中,*limit*值为判断陷入局部最优的阈值,即当算法连续迭代*limit*次后,当前最优解仍然没有得到改善,则认为算法陷入局部最优。为了说明Kent混沌中*limit*参数对AMFO算法的性能影响,针对表1中函数f_5,*limit*分别设置为5、10、15、20,表3列出了AMFO算法在不同*limit*下的性能情况。从表3中可以看出,AMFO算法在$limit=5$的情况下,耗时较其余几种情况多几秒,当*limit*较小,更容易将当前最优解认为陷入局部最优,也就会进行更多次的Kent混沌搜索,将增加算法的耗时。在$limit=10$、15和20的情况下,AMFO算法耗时基本相当。针对函数f_5,当$limit=10$时,AMFO算法优化得到的平均值为1.487×10^{-5},是四种*limit*参数中最优的。过小的*limit*,AMFO算法进行Kent混沌搜索次数增加,将会提升算法精度,却增加耗

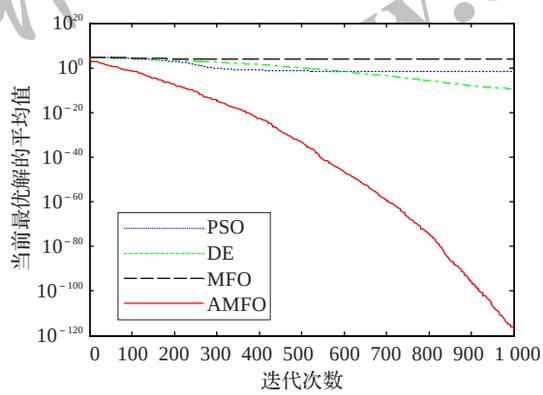

图2　函数f_1的收敛曲线

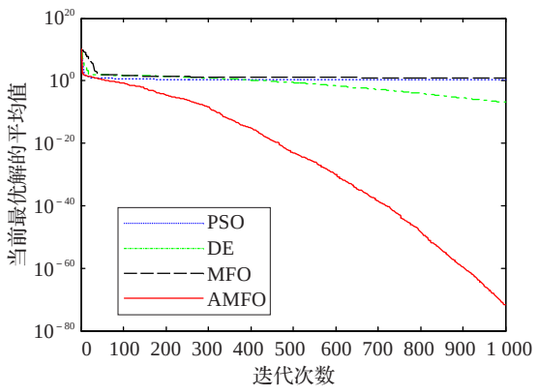

图3　函数f_2的收敛曲线

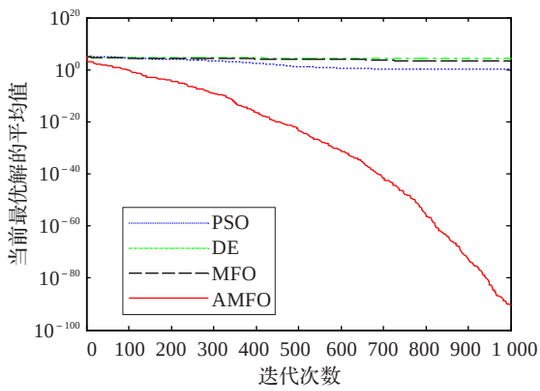

图4　函数f_3的收敛曲线

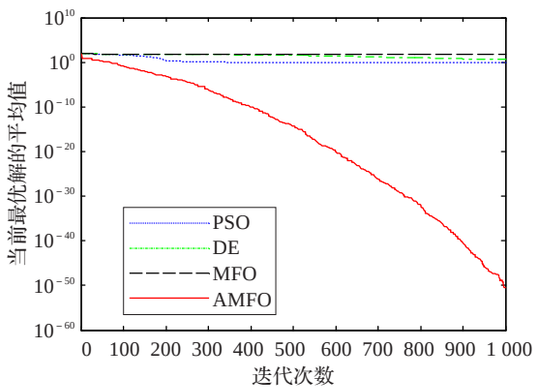

图5　函数f_4的收敛曲线

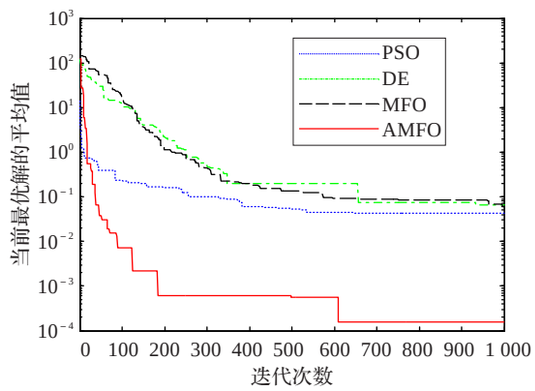

图6　函数f_5的收敛曲线

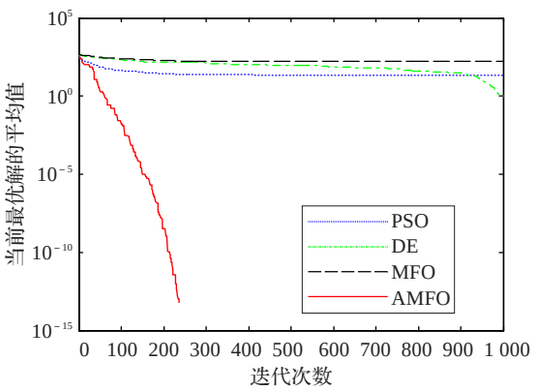

图7　函数f_6的收敛曲线

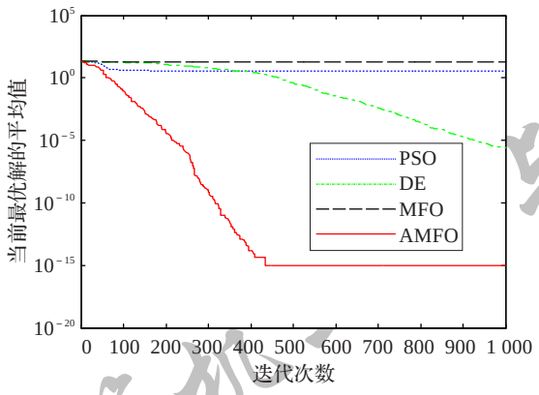

图8　函数 f_7 的收敛曲线

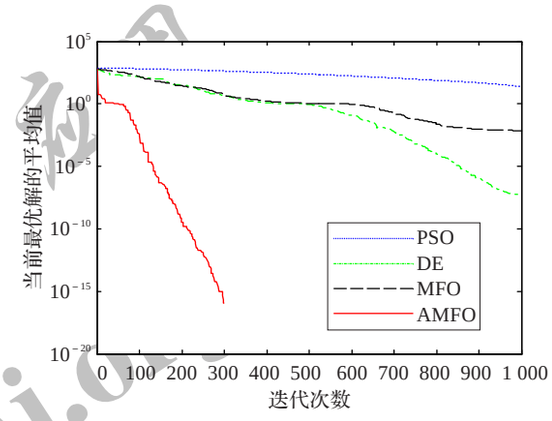

图9　函数 f_8 的收敛曲线

时；相反，将会降低算法精度，却减少耗时。综合考虑，文中将 *limit* 参数设置为10。

表3　*limit* 对AMFO算法性能影响（$n=30$）

函数	*limit*	AMFO		
		平均值	均方差	耗时/s
f_5	5	2.393×10^{-3}	7.583×10^{-5}	11.14
	10	1.487×10^{-5}	1.372×10^{-5}	7.03
	15	7.243×10^{-5}	4.227×10^{-4}	7.15
	20	6.368×10^{-3}	8.773×10^{-4}	6.84

综上所述，AMFO算法能够有效地找到全局最优值或接近理论最优值，提出的AMFO算法的改进策略是成功的，针对函数优化问题，AMFO是一个非常高效的优化算法。

5　结束语

针对MFO算法收敛精度不高、易陷入局部最优的问题，将MFO算法与Kent混沌搜索策略和动态惯性权值调整策略相结合，提出一种改善的MFO算法（AMFO）。Kent混沌搜索策略利用当前为止最优解跳出局部最优；基于适应度值与迭代次数的动态惯性权值平衡算法的探索和开发能力，利于AMFO算法全局寻优。通过8个经典的10维、30维和50维测试函数验证AMFO算法性能，将其结果与MFO、PSO、DE算法进行比较，在8个经典测试函数中，AMFO算法均取得了最好的优化结果。

参考文献：

[1] Gupta D K，Vasudev K L，Bhattacharyya S K.Genetic algorithm optimization based nonlinear ship maneuvering control[J].Applied Ocean Research，2018，74：142-153.

[2] Chen Ke，Zhou Fengyu，Liu Aling.Chaotic dynamic weight particle swarm optimization for numerical function optimization[J].Knowledge-Based Systems，2018，139：23-40.

[3] Tong Lyuyang，Dong Minggang，Jing Chao.An improved multi-population ensemble differential evolution[J].Neurocomputing，2018，290：130-147.

[4] 孔祥鑫，周炜，王晓丹.基于改进磷虾群算法的SVDD参数优化[J].计算机工程与应用，2017，53(22)：137-142.

[5] 范会联，曾广朴.带自适应迁入的生物地理学优化算法[J].计算机应用研究，2015，32(12)：3642-3645.

[6] Heidari A A，Pahlavani P.An efficient modified grey wolf optimizer with Lévy flight for optimization tasks[J].Applied Soft Computing，2017，60：115-134.

[7] Mirjalili S.Moth-flame optimization algorithm：a novel nature-inspired heuristic paradigm[J].Knowledge-Based Systems，2015，89：228-249.

[8] 崔东文.飞蛾火焰优化算法在承压含水层参数反演中的应用[J].长江科学院院报，2016，33(7)：28-33.

[9] 吴伟民，李泽熊，林志毅，等.飞蛾纵横交叉混沌捕焰优化算法[J].计算机工程与应用，2018，54(3)：136-141.

[10] Rebecca N S，Mohd H S，Zuriani M.Optimal reactive power dispatch solution by loss minimization using moth-flame optimization technique[J].Applied Soft Computing，2017，59：210-222.

[11] Allam D，Yousri D A，Eteiba M B.Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm[J].Energy Conversion and Management，2016，123：535-548.

[12] Emary E，Zawbaa H M.Impact of chaos functions on modern swarm optimizers[J].PLoS One，2016，11(7)：1-26.

[13] Li Zhiming，Zhou Yongquan，Zhang Sen.Lévy-flight moth-flame algorithm for function optimization and engineering design problems[J].Mathematical Problems in Engineering，2016：1-22.

[14] 刘建军，石定元，武国宁.基于Kent映射的混合混沌优化算法[J].计算机工程与设计，2015，36(6)：1498-1503.

[15] 张彩宏，潘广贞.融合禁忌搜索的混合果蝇优化算法[J].计算机工程与设计，2016，37(4)：907-913.

[16] 孙宪坤，陈涛，韩华.基于Kent混沌测量矩阵的压缩感知图像重构算法[J].计算机应用与软件，2017，34(4)：213-220.