控制与决策
Control and Decision

基于统计指导的飞蛾扑火算法求解大规模优化问题

刘小龙

引用本文:
刘小龙. 基于统计指导的飞蛾扑火算法求解大规模优化问题[J]. 控制与决策, 2020, 35(4): 901−908.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.1081

文章编号: 1001-0920(2020)04-0901-08 DOI: 10.13195/j.kzyjc.2018.1081

基于统计指导的飞蛾扑火算法求解大规模优化问题

刘 小 龙†

(华南理工大学 工商管理学院，广州 510641)

摘 要: 针对飞蛾扑火算法求解大规模优化问题较差的实际,借鉴差分进化算法中的变异思想,在飞蛾扑火算法中引入缩放因子和视距因子的概念,提出飞蛾直飞模型,并界定围绕历史最优飞蛾和当前随机飞蛾的直飞方式分别为局部寻优和全局寻优;设计3种不同类型的视距因子,从宏观上引导搜索算法启动全局探索和局部开发的时机,分析不同启动时机选择对飞蛾扑火算法在大规模问题上的优化精度影响,提出不同优化问题具有不同启动时机的思想;讨论飞蛾直飞和螺旋式飞行的3种组合策略下的优化效率,验证了所提出算法的较优性能,与现有文献改进算法在大规模优化问题上的改进效果进行对比,数值实验验证了改进算法的优越性和鲁棒性,拓展和丰富了原算法的应用范围.

关键词: 大规模优化问题；飞蛾扑火优化算法；函数优化；视距因子；缩放因子；统计指导

中图分类号: TP18 **文献标志码:** A

Moth-flame algorithm based on statistical guidance for large-scale optimization problems

LIU Xiao-long†

(School of Business Administration，South China University of Technology，Guangzhou 510641，China)

Abstract: In view of the fact that the moth-flame optimization (MFO) algorithm is poor in solving large-scale optimization problems, the direct flight model of moths is proposed based on the mutation idea of differential evolution algorithm (DE). The concept of the sight distance factor and the scaling factor is used, and the flame around the historical optimal moth and the current random moth is defined as local development and global exploration. Three different types of the sight distance factor are designed to guide the search algorithm and the global exploration and local development from the macroscopic point of view are started. The influence of different start-up time on the precision of the MFO algorithm in the large-scale problem is analyzed, and the idea that different optimization problems have different start-up opportunity for local development. The optimization efficiency of the three combined strategies is discussed on the model of direct flight and helical flight, and the better performance of the proposed algorithm is verified. Numerical experiments vprify the superiority and robustness of the improved algorithm, compared with the existing literature on large-scale optimization problems.

Keywords: large scale optimization problem；moth-flame optimization algorithm；function optimization；sight distance factor；scaling factor；statistical guidance

0 引 言

最优化问题是人们在科学研究和生产实践中经常遇到的问题. 传统以梯度为基础的最速下降法、线性规划、单纯形法等优化方法,在问题的目标函数是凸集、连续可微可导等情况下,具有较高的计算效率. 但现实中许多大规模、非线性、多极值、多约束、非凸性等现象,使得传统的优化方法难以进行数学建模,从而给以仿生为基础的群智能计算方法提供了广阔的舞台,并诞生了一批模拟生物行为、自然现象的启发式方法. 这些启发式方法都是基于群体智能,并利用“生成+检验”的自适应人工智能计算技术[1].

目前,主要的启发式仿生优化技术其灵感来源有3大类别,即模拟生物和人类进化过程、模拟自然和物理现象以及模拟生物群体的行为特征等. 借鉴大

收稿日期: 2018-08-06；**修回日期:** 2018-10-22.

基金项目: 国家自然科学基金项目(71071057, 71571072)；广州社科联基金项目(2018GZGJ02, 2016GZYB10).

责任编委: 冯俊娥.

†通讯作者. E-mail: xlliu@scut.edu.cn.

自然适者生存、优胜劣汰的进化规律,通过模拟生物和人类进化过程,研究者们将社会问题的优化过程看成是类似于生物进化的演变过程,提出了基于演化、进化的全局搜索算法,譬如ES进化策略、遗传算法、差分进化、演化算法等[2];模拟自然和物理现象,学者们提出了和声搜索优化、花朵授粉优化、模拟退火优化、万有引力优化、生物地理学优化、多元宇宙优化算法等[2-3];或者是针对生物群体的行为特征进行仿生,学者们提出了粒子群算法、蚁群算法、人工蜂群算法、布谷鸟算法、蝙蝠算法、鲸鱼优化算法、萤火虫算法等[2,4]. 这种源于生物系统或者自然界物理现象灵感的进化和群智能启发式方法在最优化领域得到了极大应用,数值实验表明这些启发式方法具有一定的优异性,但也存在着收敛精度不高、大规模问题的优化能力一般等问题[5].

针对大规模优化问题,不同学者从问题维度分解和算法集成等视角进行了研究. 现有论文提及的解决此类问题的框架基本上可以划分为两类:即合作型协同演化、非合作型协同演化[6]. 合作型协同演化方法主要通过降低待优化问题的维度,然后协同优化. 在合作型协同演化方法中可以嵌入多种启发式算法,针对大规模优化问题具有较好的鲁棒性. 例如文献[5]将整个种群分成多个子群,通过子群协同在搜索空间解决高维优化问题;文献[7]将协同思想引入差分进化算法实现对中等规模问题的优化. 但研究发现,如果待优化的问题变量属于“不可分问题”,也就是说待优化问题的变量与优化函数值之间存在相互关联时,合作型协同演化方法出现了优化性能下降的情况[8]. 非合作型协同演化方法将大规模全局优化问题的所有决策变量作为整体来处理,所利用的启发式算法包括前述粒子群算法、差分进化、遗传算法、人工蜂群算法、萤火虫算法等,改进策略包括多种群、竞争学习、群体多样性保持、算法融合、局部搜索、改进采样、采用新扰动算子、初始种群优化等[6]. 例如拓守恒[9]利用差分进化与模拟退火算法融合来优化大规模问题;欧阳海滨等[10]在和声搜索算法中引入反向学习策略来优化高维函数;龙文等[11]对鲸鱼优化进行参数改进实现大规模问题求解;姜天华[12]利用正交反向改进灰狼优化;罗家祥等[13]对差分进化、模拟退火和局部搜索链等3种搜索策略进行了融合研究,大规模优化实验表明取得了较好的优化效果.

2015年,Mirjalili[14]提出了飞蛾扑火优化算法(MFO),通过对29种基准测试函数的中低维度数值实验和实际复杂约束、未知空间搜索等工程问题的研究,得出MFO算法相对于粒子群优化、万有引力搜索、蝙蝠算法、花朵授粉算法、萤火虫算法、遗传算法等具有收敛速度更快、精度更高等优势. 该算法的这种并行性强、全局性优且不易陷入局部极值的性能特征,逐渐引起了学术界和工程界的关注,Yamany等[15]、Jangir等[16]、Allam等[17]、崔东文[18]、王子琪等[19]将其应用于感知器训练、电力系统有功和无功优化调度、混合经济排放调度、承压含水层参数反演和电力系统最优潮流计算中,也取得了相对较好的效果,但MFO算法针对大规模优化问题的国内外研究尚不多见. 因此,本文在协同演化框架下,基于融合技术和学习策略,对飞蛾扑火优化算法进行改进并将其应用于大规模优化问题. 数值实验研究表明,改进算法具有较强的适用性,稍作修改可应用于生产计划调度、图象分析处理和产品优化设计等诸多领域.

1 飞蛾扑火优化算法

1.1 飞蛾扑火思想

作为蝴蝶家族的特殊成员,飞蛾的夜间飞行模式颇为特殊. 若夜晚存在月光,则飞蛾通过维持与月亮相对固定的角度来飞行,因月亮距离飞蛾较遥远,则这种飞行可以保持直线;若近距离存在强光,譬如人类的灯光,由于光源相对于飞蛾的距离非常近,则飞蛾相对于光源保持一种固定夹角飞行,从而产生了螺旋式逼近的飞行路径,最后攀附于光源上,形成了“飞蛾扑火”现象. 通过模拟飞蛾与光源固定角度“横向定位”的飞行特性,文献[14]将飞蛾追逐灯光的行为模型化为飞蛾围绕灯源位置进行螺旋式位置更新.

1.2 飞蛾扑火算法的优化过程

文献[14]中,MFO算法假定飞蛾是待优化问题的解,通过螺旋式追逐火焰的位置来获得全局极值. 算法定义飞蛾种群为矩阵结构$M = (N, \dim)$,定义火焰结构为$F = (N, \dim)$. 其中N为飞蛾种群数,也是最大火焰的个数;dim为问题维度. 与其他算法不同的是,飞蛾扑火算法基于贪婪保留原理,火焰结构保存飞蛾飞行的历史最好的N个解,利用飞蛾的螺旋式飞行来更新待优化变量和火焰结构.

该螺旋飞行满足3个条件,即螺旋函数的初始点为飞蛾位置、飞蛾飞行的终点为火焰位置以及螺旋函数的波动采用对数螺旋结构,同时要求螺旋函数的波动范围不超过搜索空间. 螺旋式飞行表明飞蛾可环绕在火焰周围而不仅是在它们之间的空间,从而保

障了MFO算法具有较好的跳出局部极值能力. 飞蛾的螺旋式飞行按照下式进行:

$$X_j(l+1)=|X_j(l)-F_j(l)|\mathrm{e}^{bt}\cos(2\pi t)+F_j(l). \quad (1)$$

其中: X 为飞蛾种群, F 为火焰种群, j 为选定的问题维度, l 为当前迭代步, b 为限定螺旋式飞行的形状 $(b=1)$, t 为飞蛾与火焰的螺旋飞行系数. 火焰种群的个数和螺旋飞行系数 t 按照下式确定:

$$\mathrm{Flame_{no}}=\mathrm{round}\Big(N-l\times\frac{N-l}{L}\Big), \quad (2)$$

$$t=\Big(-\frac{l+L}{L}-1\Big)\times\mathrm{rand}+1. \quad (3)$$

其中: L 为最大迭代步; 算法执行中, 火焰种群个数由 N 线性递减到1; rand为均匀分布随机数, 当 $t=-1$ 时, 表示飞蛾与火焰最近, 当 $t=1$ 时, 表示飞蛾与火焰最远. 飞蛾与火焰的距离度量随着迭代的进行, 由外围逐渐螺旋式靠近火焰中心, 从而完成对火焰周边区域的搜索. 按照文献[14]的基本思想, 绘制MFO算法的流程结构如图1所示.

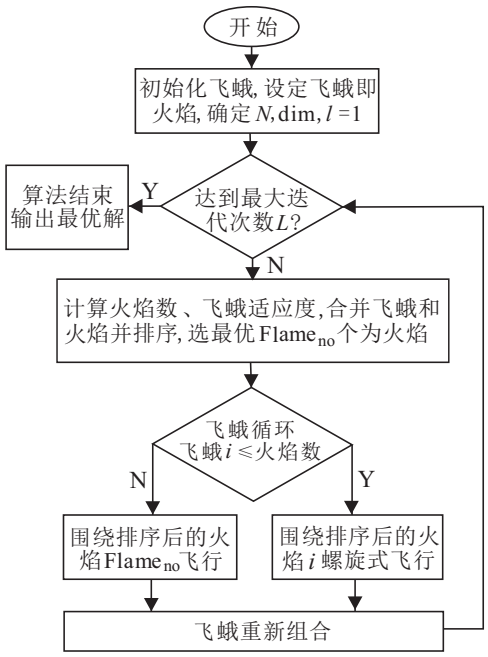

图1　MFO算法流程结构

图1中, MFO算法将最好的飞蛾位置保存为火焰位置, 飞蛾按照排序后的火焰和最差火焰螺旋式飞行, 获得新一轮的优化值. 基于贪婪保留原理, 新迭代将火焰位置和飞蛾位置合并, 选择最好的多个精英作为新的火焰, 精英个数随着迭代线性递减到1, 新的飞蛾继续围绕排序后的火焰和最差火焰螺旋式飞行, 获得另一轮的优化值. 上述按照适应度值进行排序, 选取最优精英的适应度值所对应位置保存为火焰, 相当于剔除了劣解, 因此MFO算法具有较高精度的局部开发能力.

2　改进飞蛾扑火优化算法

2.1　飞蛾扑火算法的问题分析

MFO算法采用了一种基于火焰矩阵的位置更新机制, 火焰矩阵 F 是将当前找到的精英解作为下一代火焰的位置, 因此, 火焰矩阵 F 包含了截止到目前飞蛾群体找到的最优解集合. 迭代过程中, 飞蛾群体和排序后的火焰一一对应, 这一机制提升了飞蛾在选定火焰周围的局部开发能力. 但是, 任何事物都有两面性, 基于精英保留的贪婪策略, 也会使得火焰矩阵 F 过早陷入局部极值位置而使得飞蛾无法跳出. 譬如在第 t 次迭代时, 如果飞蛾的新位置优于火焰位置, 则意味着火焰需要全部更新, 而更新前的火焰离全局最优位置较近. 当火焰更新完后, 飞蛾就错过了在全局最优位置继续寻优的可能. 另外, 迭代后期, 精英个数减少, 围绕最差火焰的飞蛾增加, 导致了群体多样性的丧失. 这就解释了文献[14]中, MFO针对 f_3、f_4、f_5、f_8、f_9 等函数 $(D=100$ 维) 难以获得全局极值的原因.

为了弥补上述易陷入局部极值的缺陷, 平衡算法的全局搜索与局部开发能力, MFO算法提出一种自适应机制的点亮火焰概念, 同时界定最差火焰就是最后一个火焰. MFO算法迭代中, 可点亮的火焰种群数随着迭代递减, 到最后只剩下最优的单个火焰.

MFO算法界定: 如果飞蛾序列大于可点亮火焰标号, 则飞蛾围绕最差火焰螺旋式飞行. 这一机制设定有一定的合理性, 但是迭代前期的飞蛾差异较大, 基于最差火焰的飞行不需要跳出局部最优; 迭代中后期, 最差火焰与最优火焰的差距很小, 基于排序火焰和最差火焰的位置更新难以跳出局部最优, 仅仅增加了局部寻优的精度. 这就解释了文献[14]中针对 f_1、f_2 等函数寻优精度相对较高的原因. 为了提高MFO算法的寻优精度和全局寻优能力, 并将MFO算法拓展到高维空间 $(D=1\,000)$, 以下将从3个方面进行算法改进.

2.2　改进飞蛾扑火优化算法设计

2.2.1　视距因子 A 和基于统计指导的搜索机制

为了让算法具有自适应能力, MFO算法中定义了可点亮火焰数目, 通过前述分析可知这一机制没有提高算法全局搜索能力. 因此, 本文提出飞蛾的视距因子 A, 当飞蛾的视距因子 A 大于某固定值时执行全局搜索, 否则执行局部搜索, 尝试以此来提高算法全局寻优性能. 通过反复探索性实验, 本文设计线性、凹型和凸型3种不同类型变化的视距因子 A, A 随迭代收缩至0, 表达式如下:

$$A_1 = 1 - \frac{l}{L}, \tag{4}$$

$$A_2 = 1.5 - 1.5 \times \left(\frac{l}{L}\right)^{1/6}, \tag{5}$$

$$A_3 = 1 - \left(\frac{l}{L}\right)^{6}. \tag{6}$$

另外,设计随视距因子A变化而宏观指导MFO算法启动局部开发的时机,而且这一时机人为可调,便于不同实际问题的适应. 一般问题,经多次试验研究发现,前期8%左右的全局搜索、后期92%左右的局部搜索时机基本可获得问题较优解. 现尝试利用这种启动时机试验值从宏观上指导飞蛾的全局和局部飞行搜索.

2.2.2 带差分思想的飞蛾搜索

根据达尔文的进化理论和孟德尔的遗传学说,物种在传承时,父代的基本特征都会被其后代所继承,但后代也会表现出与其父代不完全相同的特征,这种父子两代之间的差异就是变异. 基于这种思想,Storn等[20]提出了差分进化,通过群体内个体间的合作与竞争指导优化搜索,该算法由变异、交叉和选择进行驱动,其中变异操作被设计用于利用或更好地探索搜索空间,现有文献针对变异的操作有5种类型,分别是DE/rand/1、DE/best/1、DE/current to best/1、DE/best/2和DE/rand/2,具体表达式见相关文献. 基于上述思想,本文借助差分变异和差分中的缩放因子概念,定义了飞蛾的直线飞行方式,设计飞蛾围绕火焰F的直飞公式如下:

$$X_j(l+1) = F_j(l) + P \times |P \times F_j(l) - X_j(l)|. \tag{7}$$

式(7)表示飞蛾执行局部开发,F为当前最优的火焰矩阵,P为基于高斯分布的缩放因子. 在缩放因子设计时,如果按照标准差分进化的思想,将缩放因子设计为正值,就会导致飞蛾寻优只能在飞蛾与火焰之间进行. 因此,本文利用高斯分布变量P的正负取值特征,使得飞蛾寻优在火焰周围进行,从而增加了获得全局极值的机会. 为了避免改进算法的过早收敛,探讨飞蛾的全局探索功能,定义飞蛾围绕随机选定飞蛾进行飞行为局部开发,用下式表示:

$$X_j(l+1) = X_j^m(l) + P|P \times X_j^m(l) - X_j(l)|. \tag{8}$$

其中:相关符号同前,m表示随机飞蛾.

2.2.3 基于高斯分布的缩放因子

分布估计的算法思想源于遗传算法,在求解松散分布的问题时,交叉操作常常导致了算法的局部收敛和早熟,为了避免这一缺陷,分布估计使用抽样概率模型产生新群体,替换遗传计算中的交叉和变异算子,从而形成带有“全局操控性”的分布估计思想. 在现有分布估计算法中,都采用高斯分布作为描述连续解空间的概率模型,故本文探讨利用具有高斯分布特征的变量P作为飞蛾飞行目的地的缩放因子,其中$P = \mathrm{rand}\, n$.

通过反复实验发现,对火焰和飞蛾的差距执行单个P值缩放,收敛速度较慢,为此在火焰本身也引入缩放因子,使之在3倍标准差之内随机取值,可以拓宽火焰周边的区域,从而在提高全局位置获取能力的同时,提高寻优精度.

2.2.4 IMFO算法步骤和执行框架

综上所述,提出IMFO算法的步骤如下.

step 1: 定义问题维度D、群体个数N、最大迭代步数L、视距因子A、飞行系数t.

step 2: 计算群体适应度,标识历史最优飞蛾为火焰F.

step 3: 算法循环,根据式(3)和(4)更新螺旋飞行系数t和视距因子A.

step 4: 视距因子判断. A大于92%执行全局探索,随机选定一半飞蛾直线飞行,按照式(8)更新飞蛾位置;另一半飞蛾螺旋式飞行,按照下式:

$$X_j(l+1) = |PX_j^m(l) - X_j(l)|\mathrm{e}^{bt}\cos(2\pi t) + X_j^m(l) \tag{9}$$

更新飞蛾位置,m为随机飞蛾.

step 5: 局部开发判断. 当A小于92%时,执行局部开发,随机选定一半飞蛾围绕火焰位置直线飞行,按照式(7)更新飞蛾位置;另一半飞蛾围绕火焰位置螺旋式飞行,按照下式:

$$X_j(l+1) = |PX_j^m(l) - F|\mathrm{e}^{bt}\cos(2\pi t) + F \tag{10}$$

更新飞蛾位置.

step 6: 计算群体适应度、更新火焰位置.

step 7: 循环判断. l小于L,返回step 3;否则结束.

3 数值实验

3.1 测试函数和验证指标

龙文等[11]、Mahdavi等[21]、Bolufe等[22]对大规模优化问题($D = 1\,000$)进行研究,使用了基准测试函数进行对比分析. 为探讨IMFO的性能,本文利用文献[11, 13-14, 21]等涉及的15个标准测试函数进行实验研究,函数表达式详见文献[11].

文献[11]中的函数最小值均为0,包括单峰和多峰函数,给定算法收敛阈值为1e-08(f_5、f_6、f_7阈值设为1、1e-04和1e-02),如果算法收敛到阈值范围内,则

认定算法获得全局最优解.

在所有实验设置中,考虑到尽可能少地节省计算资源,除非特别说明,均设定飞蛾数目为100,最大迭代次数为500,IMFO最大计算次数为50 000,所有算法测试30次,统计各算法均值、标准差. 就现有比较实验而言,统计各测试函数中IMFO算法进入到收敛阈值之内的次数和实际测试次数之比,得寻优成功率SR. 30次测试的寻优结果都是源于总体($U=0$)的抽样样本,使用t检验统计30次测试数据的最小显著性P值.

3.2　视距因子A和缩放因子P的影响分析

为验证A的影响,对A_1、A_2、A_3影响算法性能的情况进行实验. 在算法框架中,A_1线性下降到0,A_2凹型下降至0,A_3凸型下降到0,因此当固定A值为92%时,不同视距因子A所影响的仅仅是从那一个迭代步开始执行局部搜索,因此本文认为调整A的取值和选择A的凹凸形态是一样的意义.

另外,基于差分思想下的缩放因子P影响着搜索精度,现有文献建议取定值,且介于[0,2]之间,本文采用具有高斯分布的P值,P值在[−1,1]之间的可能性有68.27%,利用$P=(1,\mathrm{Rand}\,n)$两种可能进行数值试验,P值对算法的影响结果见表1.

表1　缩放因子P对寻优精度的影响

函数	$P=1$	$P=\mathrm{Rand}\,n$	函数	$P=1$	$P=\mathrm{Rand}\,n$
f_1	1.67e+05	0	f_9	1.19e+01	8.89e-16
f_2	3.13e+02	7.70e-161	f_{10}	2.15e+03	0
f_3	8.71e-05	0	f_{11}	3.08e+02	7.99e-165
f_4	7.49e-05	0	f_{12}	1.09e+03	3.45e-06
f_5	8.21e+04	3.82e-05	f_{13}	2.56e+01	0
f_6	1.81e-01	1.11e-04	f_{14}	1.09e+03	1.48e-07
f_7	1.09e+07	3.59e-11	f_{15}	1.83e-164	0
f_8	2.49e+03	0			

由表1可知,基于高斯分布的P值缩放因子除了f_{12}测试函数外,所有基准测试函数都明显提高了寻优精度3个等级以上,从而验证了高斯分布缩放因子设计的有效性.

3.3　开始局部搜索的时机影响

基于经验,表1设定了局部开始的时机为A下降到0.92,此时迭代次数执行了$N(1-92\%)=40$次. 现利用15个大规模函数($D=1\,000$)测试A下降的时机选择和精度影响,实验数据如表2所示.

由表2可知,不同寻优时机选择影响着改进算法的寻优精度,但改进算法针对现有函数的寻优等级差别相距不是太大,说明现有测试函数的局部陷阱设计相对较为平缓. 文献[11]在测试函数f_5中进入了局部最优无法跳出,本文算法均能获得e-05等级以上的求解精度,但本文算法在求解f_{12}和f_{14}时的精度要劣于文献[11],且f_{12}的求解精度相对于阈值1e-08而言稍微偏低.

表2　局部寻优的启动时机对寻优精度的影响

A	f_1	f_2	f_3	f_4	f_5
0.92	0	7.70e-161	0	0	3.82e-05
0.88	6.88e-303	7.05e-162	0	0	9.48e-06
0.84	9.58e-314	1.18e-153	1.93e-308	5.81e-304	1.40e-06
0.80	4.51e-296	3.89e-151	3.37e-162	1.75e-301	1.75e-06
A	f_6	f_7	f_8	f_9	f_{10}
0.92	1.11e-04	3.59e-11	0	8.89e-16	0
0.88	1.65e-04	1.36e-10	0	8.89e-16	0
0.84	6.23e-05	7.81e-12	0	8.89e-16	0
0.80	1.16e-04	1.49e-10	0	8.89e-16	0
A	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0.92	7.99e-165	3.45e-06	0	1.48e-07	0
0.88	1.02e-161	2.10e-05	0	2.05e-10	6.42e-317
0.84	1.79e-126	1.06e-03	0	7.36e-08	3.22e-308
0.80	1.00e-129	7.15e-06	0	2.93e-09	2.94e-293

3.4　直线和螺旋飞行的效率分析

标准MFO算法设计了飞蛾的螺旋式飞行,本文定义飞蛾围绕随机飞蛾和火焰的飞行方式分别为全局探索和局部开发;本文基于差分思想设计了飞蛾的直线飞行路线,定义飞蛾围绕火焰直飞为局部开发,围绕随机飞蛾直飞为全局探索. 现对IMFO算法步骤和执行框架中,各种搜索策略进行界定,实验1是飞蛾按照螺旋式飞行执行全局和局部搜索,实验2是飞蛾按照本文提出的直线飞行策略执行全局和局部搜索,实验3是将螺旋式飞行和直飞策略进行组合. 设置全局/局部概率为8%/92%,并对这些策略组合进行实验,实验结果见表3.

表3　不同实验策略配置下的寻优精度等级($D=1\,000$)

策略	f_1	f_2	f_3	f_4	f_5
实验1	6.6e-161	3.03e-98	5.52e+161	5.25e-301	1.08e-02
实验2	1.04e-118	1.38e-63	2.05e-120	2.08e-203	3.86e-203
实验3	0	7.70e-161	0	0	3.82e-05
策略	f_6	f_7	f_8	f_9	f_{10}
实验1	1.05e-04	9.89e-08	0	4.44e-15	0
实验2	1.91e-04	4.07e-07	0	8.89e-16	0
实验3	1.11e-04	3.59e-11	0	8.89e-16	0
策略	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
实验1	7.17e-05	1.94e-06	0	1.21e-07	0
实验2	1.26e-63	1.18e-05	0	1.42e-03	3.26e-223
实验3	7.99e-165	3.45e-06	0	1.48e-07	0

由表3可知, f_5 测试函数下的组合策略性能变差, f_1、f_2、f_3、f_4、f_7、f_{11} 测试函数下的组合策略性能更好, f_6、f_8、f_9、f_{10}、f_{12}、f_{13}、f_{14} 和 f_{15} 测试函数下的组合策略与其他策略性能相当. 总的来说,螺旋和直飞的组合策略较为有效,因此本文尝试利用组合策略优化大规模问题,并与不同启发式算法进行比较.

3.5 不同启发式优化算法的比较

针对文献[11]中的测试函数,设置局部寻优的启动时机 A 值为0.92,利用组合策略的IMFO算法求解15个大规模优化问题,统计30次实验结果见表4.

表4 IMFO求解大规模优化问题的性能指标($D=1\,000$)

指标	f_1	f_2	f_3	f_4	f_5
Mean	3.93e-148	7.69e-87	2.46e-149	1.43e-275	0.037 9
Std	1.79e-149	3.36e-86	1.00e-148	0	0.101 7
SR	100 %	100 %	100 %	100 %	100 %
指标	f_6	f_7	f_8	f_9	f_{10}
Mean	7.55e-05	1.17e-05	6.06e-14	3.49e-15	0
Std	9.57e-05	3.63e-05	3.32e-13	2.07e-15	0
SR	100 %	100 %	100 %	100 %	100 %
指标	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
Mean	1.92e-137	3.45e-06	0	1.48e-07	0
Std	7.19e-138	2.49e-06	0	2.12e-06	0
SR	100 %	60 %	100 %	73.33 %	100 %

由表4可知,除了 f_{12} 和 f_{14} 外,IMFO算法的寻优成功率均为100 %,15个测试函数的平均寻优成功率为95.55 %,稍高于文献[11]中15个测试函数的寻优平均成功率84 %. 在寻优精度上,有4个测试函数的寻优精度稍低于文献[11],11个函数的寻优精度高于文献[11].

为了广泛对比,参考文献[11]及其他几种改进算法的大规模优化问题测试结果,按照年份进行排序,分别是综合学习粒子群(CLPSO)、协方差矩阵进化策略(CMAES)、自适应差分(SaDE)、量子粒子群(SMQPSO)、组合差分(CoDE)、动态和声搜索(DIHS)、改进鲸鱼优化算法(IWOA),上述算法在测试函数选择中均涉及到文献[11]中的7种函数,最大迭代步(MaxFEs)为 5×10^6,各种算法的参数设置详见文献[11]. 现将本文改进IMFO算法的MaxFEs从 5×10^4 调整为 5×10^6,对这7种函数进行数值对比实验,实验结果见表5.

表5 不同算法求解大规模问题的寻优精度($D=1\,000$)

文献对比算法	f_1	f_2	f_5	f_8
CLPSO	3.14e-21	7.73e-02	7.45e+02	4.96e+03
CMAES	7.06e-01	1.78e+0	3.26e+03	3.60e+02
SaDE	2.22e-01	1.57e-02	3.05e+03	8.22e+02
CoDE	6.06e-02	3.16e-03	2.21e+03	1.06e+03
DIHS	1.98e-23	5.24e-12	1.21e+03	2.34e+02
IWOA	0	0	9.90e+02	0
本文IMFO	0	0	8.28e-08	0
文献对比算法	f_9	f_{10}	f_{12}	
CLPSO	1.95e+01	2.78e-15	1.90e+03	
CMAES	2.13e-01	4.94e-01	1.19e+01	
SaDE	1.21e+01	9.25e-01	1.15e+02	
CoDE	9.19e+00	1.23e+00	9.94e+01	
DIHS	4.02e-13	2.50e-15	9.97e-26	
IWOA	8.88e-16	0	0	
本文IMFO	8.89e-16	0	1.54e-06	

由表5可知,本文IMFO在上述测试函数中具有一定的比较优势,除了 f_{12} 测试函数低于DIHS、IWOA外,其他测试函数不比其他优化算法差. 其中, f_5 函数中,其他改进算法明显陷入局部极值,而本文提出的算法跳出了局部极值,达到了1e-08精度等级; f_{12} 测试函数中,本文算法的寻优精度相对较低,但也达到了1e-06的水平.

文献[14]使用了13个中等规模和10个小规模基准函数进行比较研究,前13个基准函数与文献[11]中的某8个基准函数是一样的,文献[14]中剩下的5个基准函数 f_3、f_4、f_6、f_8、f_{13} 寻优成功率相对偏低、缺乏对比研究,本文利用MFO和IMFO继续深入研究,这些函数的表达式见文献[14]. 另外,本文发现文献[14]有部分笔误,所给出的框架和提供的代码中,螺旋式飞行系数 t 等参数设置应该处于飞蛾 i 和维度 j 的循环之间而非内循环.

对比实验中,为了公平起见,所有算法的最大适应度计算次数一样(MaxFEs = 50 000). 此时,参数设置同前面一样,飞蛾数目为100,最大迭代次数为500,开始局部搜索的启动时机 $A=0.92$,但阈值设置为1e-08. 利用IMFO和MFO针对前述文献[14]与文献[11]有差别的5个基准函数进行30次实验,统计各算法的性能测试指标,即最优值的均值、最优值的标准差、最优值成功率,显著性 P 值(t 检验),实验结果见表6.

表6　大规模优化问题的数值对比实验($D=1\,000$)

	Mean	Std	SR	P值	
IMFO	f_3	1.50e+05	7.10e+05	36.67 %	0.254 2
	f_4	3.88e-12	2.07e-11	100 %	0.313 5
	f_6	0.048 4	0.120 4	33.33 %	0.035 6
	f_8	−418 998	0.259 2	93.33 %	0.097 8
	f_{13}	0.017 7	0.057 6	36.67 %	0.102 5
MFO	f_3	7.73e+07	1.57e+07	0	4.17e-22
	f_4	65.953 8	13.060 2	0	2.14e-22
	f_6	168.831 8	6.742 2	0	2.41e-42
	f_8	−318 500	89 096	0	9.80e-07
	f_{13}	326.453 7	218.887 4	0	5.23e-09

由表6可知,IMFO显著提高了MFO的寻优精度和成功率,且具有较好的鲁棒性. 30次测试中,IMFO针对5个测试函数分别有12、30、10、28、12次进入了1e-08的阈值范围以内,除了f_6以外其他测试函数的显著性P值大于0.05,不拒绝总体均值为0的假设;而MFO算法中,所有基准函数的成功率SR为0,P值全部远小于0,寻优均值与总体均值的差异较大,精度和鲁棒性相对较差,故而得出MFO不适合求解大规模优化问题,IMFO较为适合求解大规模优化问题的结论.

3.6　算法时间复杂度比较

IMFO和MFO算法的时间复杂度与问题维度D、种群大小N、最大迭代步L和算法执行策略有关,其详细的计算复杂度见表7.

表7　算法的时间复杂度比较

策略	计算复杂度	
	MFO操作	IMFO操作
初始化	$O(N)$	$O(N)$
火焰选择	$O(L)$	$O(L)$
火焰数	$\geqslant 1$	$=1$
参数更新	$O(L)$	$O(L)$
朝火焰i飞	$O(L\times$火焰数$)$	$O(L\times N/2)$
朝最差火焰飞	$O(L\times(N-$火焰数$))$	$O(L\times N/2)$
飞蛾合并	$O(L)$	$O(L)$

表7中,MFO和IMFO的循环次数相同,其时间复杂度差别在于组合策略选择. 以单次循环为例,MFO算法中,飞蛾围绕火焰矩阵和最差火焰飞行,其时间复杂度计算为

$$O(L\times \mathrm{Flame_{no}})+O(L\times(N-\mathrm{Flame_{no}}))=O(L\times N).$$

而IMFO算法中,组合策略的时间复杂度计算为

$$O(L\times N/2)+O(L\times N/2)=O(L\times N).$$

很明显,二种算法的时间复杂度一样,唯一的差异在于式(1)和(7)的计算时间,而对于计算机来说,二者的计算时间可以忽略不计.

4　结　论

飞蛾扑火优化算法是新近提出的新型仿生元启发式算法,对部分中低维度优化问题具有一定的适用性,但大规模优化问题的适用性较差. 针对飞蛾扑火优化算法的不足,本文基于视距因子、缩放因子、统计指导、飞蛾直飞等理念,探索设计了改进型IMFO算法,对比了3种不同组合策略的寻优效率. 选取现有文献提及的大规模优化基准测试函数进行多角度反复对比研究,研究结果表明IMFO具有较高的寻优精度和较好的问题适用性,在大规模优化问题上表现出了相对优越的寻优性能. 本文研究结果拓展和丰富了MFO算法在大规模优化问题上的应用范围,稍作修改可以应用于约束优化、多目标优化和实际的工程问题之中.

参考文献(References)

[1] 刘小龙. 改进多元宇宙算法求解大规模实值优化问题[J]. 电子与信息学报, 2019, 41(7): 1666-1673.
(Liu X L. Application of improved multiverse algorithm to large scale optimization problems[J]. Journal of Electronics & Information Technology, 2019, 41(7): 1666-1673.)

[2] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[3] Mirjalili S, Mirjalili S M, Hatamlou A. Multi-verse optimizer: A nature-inspired algorithm for global optimization[J]. Neural Computing and Applications, 2016, 27(2): 495-513.

[4] 周凌云, 丁立新, 马懋德, 等. 一种正交反向学习萤火虫算法[J]. 电子与信息学报, 2019, 41(1): 202-209.
(Zhou L Y, Ding L X, Ma M D, et al. Orthogonal opposition based firefly algorithm[J]. Journal of Electronics & Information Technology, 2019, 41(1): 202-209.)

[5] 刘睿, 梁静. 求解大规模问题的协同进化动态粒子群优化算法[J]. 软件学报, 2018, 29(9): 2595-2605.
(Liu R, Liang J. Co evolution dynamic particle swarm optimization algorithm for large-scale problems[J]. Journal of Software, 2018, 29(9): 2595-2605.)

[6] 梁静, 刘睿, 瞿博阳, 等. 进化算法在大规模优化问题中的应用综述[J]. 郑州大学学报: 工学版, 2018, 39(3): 15-21.
(Liang J, Liu R, Qu B Y, et al. Application of evolutionary algorithms in large-scale optimization problems[J].

Journal of Zhengzhou University: Engineering Science, 2018, 39(3): 15-21.)

[7] 王旭, 赵曙光. 解决高维优化问题的差分进化算法[J]. 计算机应用, 2014, 34(1): 179-181.
(Wang X, Zhao S G. Differential evolution algorithm for solving high-dimensional optimization problems[J]. Computer Application, 2014, 34(1): 179-181.)

[8] Liu J, Tang K. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution[C]. 2013 Proceedings of Intelligent Data Engineering and Automated Learning (IDEAL 2013). Berlin: Springer, 2013: 350-357.

[9] 拓守恒. 基于DE和SA的Memetic高维全局优化算法[J]. 计算机系统应用, 2012, 21(2): 93-97.
(Tuo S H. High dimensional global optimization algorithm of Memetic based on SA and DE[J]. Computer System Application, 2012, 21(2): 93-97.)

[10] 欧阳海滨, 高立群, 邹德旋. 反向学习和声搜索算法优化高维函数问题[J]. 小型微型计算机系统, 2014, 35(3): 571-578.
(Ouyang H B, Gao L Q, Zou D X. Inverse learning and sound search algorithms for optimizing high dimensional functions[J]. Minicomputer System, 2014, 35(3): 571-578.)

[11] 龙文, 蔡绍洪, 焦建军, 等. 求解大规模优化问题的改进鲸鱼优化算法[J]. 系统工程理论与实践, 2017, 37(11): 2983-2994.
(Long W, Cai S H, Jiao J J, et al. Improved whale optimization algorithm for large scale optimization problems[J]. Systems Engineering — Theroy & Practice, 2017, 37(11): 2983-2994.)

[12] 姜天华. 混合灰狼优化算法求解柔性作业车间调度问题[J]. 控制与决策, 2018, 33(3): 503-508.
(Jiang T H. Mixed grey wolf optimization algorithm for flexible job shop scheduling problem[J]. Control and Decision, 2018, 33(3): 503-508.)

[13] 罗家祥, 倪晓晔, 胡跃明. 融合多种搜索策略的差分进化大规模优化算法[J]. 华南理工大学学报: 自然科学版, 2017, 45(3): 97-103.
(Luo J X, Ni X Y, Hu Y M. The integration of a variety of search strategy differential evolution optimization algorithm[J]. Journal of South China University of Technology: Natural Science Edition, 2017, 45(3): 97-103.)

[14] Mirjalili S. Moth-flame optimization algorithm: A novel nature- inspired heuristic paradigm[J]. Knowledge-Based Systems, 2015, 89(1): 228-249.

[15] Yamany W, Fawzy M, Tharwat A, et al. Moth-flame optimization for training multi-layer perceptrons[C]. The 11th International Computer Engineering Conference (ICENCO), Cairo: IEEE, 2015: 267-272.

[16] Jangir P, Trivedi I N, Parmar S A, et al. Optimal active and reactive power dispatch problem solution using moth-flame optimizer[C]. 2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS). Nagercoil: IEEE, 2016: 491-496.

[17] 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 & Management, 2016, 123: 535-548.

[18] 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.
(Cui D W. Application of moth-flame optimization algorithm in parameter inversion of confined aquifer[J]. Journal of Yangtze River Scientific Research Institute, 2016, 33(7): 28-33.)

[19] 王子琪, 陈金富, 张国芳, 等. 基于飞蛾扑火优化算法的电力系统最优潮流计算[J]. 电网技术, 2017, 41(11): 3642-3647.
(Wang Z Q, Chen J F, Zhang G F, et al. Optimal power flow calculation of power system based on moth fire suppression optimization algorithm[J]. Grid Technology, 2017, 41(11): 3642-3647.)

[20] Storn R, Price K V. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[21] Mahdavi S, Rahnamayan S, Shirim E. Muctilevel framework for large-scale global optimization[J]. Soft Computing, 2017, 21(14): 4111-4140.

[22] Bolufe R A, Fiol G S, Chen S. A minimum population search hybrid for large scale global optimization[C]. IEEE Congress on Evolutionary Computation. Sendai: IEEE, 2015: 1958-1965.

作者简介

刘小龙(1977−), 男, 讲师, 博士, 从事仿生优化、智能优化算法等研究, E-mail: xlliu@scut.edu.cn.

(责任编辑：孙艺红)