控制与决策
Control and Decision

求解约束优化问题的改进果蝇优化算法及其工程应用

石建平， 李培生， 刘国平， 刘鹏

引用本文:
石建平, 李培生, 刘国平, 等. 求解约束优化问题的改进果蝇优化算法及其工程应用[J]. 控制与决策, 2021, 36(2): 314−324.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.0557

求解约束优化问题的改进果蝇优化算法及其工程应用

石建平[1,2], 李培生[1†], 刘国平[1], 刘 鹏[3]

(1. 南昌大学 机电工程学院，南昌 330031；2. 贵阳学院 电子与通信工程学院，
贵阳 550005；3. 河北地质大学 宝石与材料工艺学院，石家庄 050031)

摘 要: 针对基本果蝇优化算法收敛速度慢、求解精度低、易于陷入局部极值以及算法候选解不能取负值等不足, 提出一种用于解决约束优化问题的改进果蝇优化算法. 该算法利用果蝇个体历史最佳记忆信息和种群全局历史最佳记忆信息构建多策略混合协同进化的搜索机制, 以达到有效平衡算法的全局探索与局部开发的目的, 同时也能够较好地避免算法的早熟收敛问题; 通过种群最优信息的实时动态更新和局部深度搜索策略的引入, 进一步提高该算法的收敛速度和收敛精度. 采用 13 个基准测试函数和 2 个工程优化问题来验证所提出算法的可行性与有效性, 仿真实验结果表明, 与其他典型智能优化算法相比, 所提出的优化算法具有全局搜索能力强、稳定性好、收敛速度快、收敛精度高等优势, 可有效解决复杂的约束优化问题.

关键词: 果蝇优化算法；约束优化问题；协同进化；局部搜索；测试函数；工程优化

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.0557 **开放科学(资源服务)标识码(OSID):**

引用格式: 石建平, 李培生, 刘国平, 等. 求解约束优化问题的改进果蝇优化算法及其工程应用 [J]. 控制与决策, 2021, 36(2): 314-324.

Improved fruit fly optimization algorithm for solving constrained optimization problems and engineering applications

SHI Jian-ping[1,2], *LI Pei-sheng*[1†], *LIU Guo-ping*[1], *LIU Peng*[3]

(1. School of Mechanical & Electrical Engineering，Nanchang University，Nanchang 330031，China；2. School of Electronic & Communication Engineering，Guiyang University，Guiyang 550005，China；3. School of Gems and Materials Technology，Hebei GEO University，Shijiazhuang 050031，China)

Abstract: In view of the shortcomings of the fruit fly optimization algorithm(FOA), such as slow convergence speed, low accuracy, easy to fall into local optimum, and the candidate solutions of the algorithm cannot take negative values, an improved fruit fly optimization algorithm(IFOA) for solving constrained optimization problems is proposed. Taking advantage of the best memory information of individual history and group global history, a multi-strategy hybrid co-evolutionary search mechanism is constructed, which can effectively balance the global exploration and local exploitation of the IFOA, and the premature convergence of the algorithm can also be better avoided. By introducing a real-time dynamic update mechanism and a local depth search strategy, the convergence speed and precision of the IFOA are further improved. The 13 benchmark problems and 2 engineering optimization problems are used to test the feasibility and effectiveness of the proposed method. Numerical results show that the proposed IFOA has obvious advantages such as stronger global search ability, better stability, faster convergence speed and higher convergence accuracy and so on, which can be used to effectively solve complex constrained optimization problems.

Keywords: fruit fly optimization algorithm；constrained optimization problem；co-evolutionary；local search；benchmark function；engineering optimization

0 引 言

约束优化问题(constrained optimization problem, COP)是优化领域的一个重要课题, 它广泛存在而又难以求解, 对约束优化问题求解方法的研究具有重要的理论意义和工程价值. 以最小优化问题为例, 约束优化问题可描述为

$$\min f(X), X = (x_1, x_2, \ldots, x_n).$$
$$\text{s.t. } \mathrm{g}_j(X) \leqslant 0,\ j = 1, 2, \ldots, q;$$

收稿日期: 2019-04-28；**修回日期:** 2019-09-20.
基金项目: 国家自然科学基金项目(51566012)；贵州省联合基金项目(黔科合LH字[2015]7302号).
责任编委: 刘德荣.
†通讯作者. E-mail: lps20150331@163.com.

$$h_j(X)=0,\ j=q+1,q+2,\ldots,m. \tag{1}$$

其中: $f(X)$为目标函数; X为决策向量,且$x_i\in[l_i,u_i]$ $(i=1,2,\ldots,n)$, l_i和u_i分别为决策变量x_i的下边界和上边界; $g_j(X)$为不等式约束; $h_j(X)$为等式约束. 由于目标函数与约束条件的非线性以及决策变量搜索空间不可行域的出现,使得基于梯度信息的传统优化方法难以对约束优化问题进行有效求解. 与传统优化方法相比,群智能优化算法具有原理简单、容易实现、无需函数梯度信息等优点,同时能够以较大的概率收敛于问题的全局最优解,因此成为解决约束优化问题的研究热点之一. 近年来,差分进化算法[1-3]、粒子群优化算法[4-5]、教与学优化算法[6-7]、萤火虫算法[8]、人工蜂群算法[9-11]等群智能优化算法在求解约束优化问题方面取得了显著的成效.

果蝇优化算法(fruit fly optimization algorithm, FOA)[12]是受果蝇觅食行为启发而提出的一种新型群智能优化算法. 由于具有寻优机制简单、控制参数少、收敛快、易于理解、容易编程实现等优点,FOA一经提出便得到广大研究者的关注,并成功应用于诸多领域[13-17]. 然而,FOA至少存在以下不足: 1)算法中的气味浓度判定值不能取负值; 2)算法中的气味浓度判定值在搜索空间内不属于均匀分布. 这些不足导致FOA算法在解决复杂优化问题时容易陷入早熟收敛,同时也限制了FOA算法的应用范围. 为此,广大科研工作者对FOA算法进行了卓有成效的改进研究. 如,针对FOA算法候选解非线性产生机制的不足,文献[18]提出了LGMA-FOA算法,该算法的候选解采用线性产生机制,使得算法能够在解空间均匀搜索,并通过惯性权重的引入较好平衡了算法的全局与局部搜索; 文献[19]在LGMS-FOA算法的基础上,进一步改进了其惯性权重因子敏感依赖于最大迭代次数的不足,同时采用前N个优秀候选解的平均值替换当前种群中的一个候选解,进而提出了改进的LGMS-FOA算法,即AE-LGMS-FOA算法; 文献[20]从搜索方向的优化选择、搜索步长的自适应调整、自适应交叉与变异机制和双子群进化机制4个方面对FOA进行改进,提出了IAFOA算法,但该算法的进化过程复杂而难以理解,丧失了FOA算法寻优机制简单、易于理解的优势; 文献[21]提出了带符号参数的SFOA算法,该算法通过两个符号参数的引入使得气味浓度判定值能够取负值,但气味浓度判定值仍反比于果蝇的位置距离,其候选解仍基于非线性产生机制; 基于种群的历史记忆信息,文献[22]提出了动态调整果蝇搜索范围的AFOA算法,有效提高了算法的收敛速度以及全局搜索能力.

FOA及其相关改进算法主要是针对解决无约束优化问题而提出的,针对解决约束优化问题的FOA改进研究鲜有报道. 本文提出一种用于求解约束优化问题的改进果蝇优化算法(improved fruit fly optimization algorithm, IFOA),该算法省略了果蝇个体距离的计算,解决了FOA不能搜索负值空间以及在解空间不能按均匀分布搜索的缺陷; 其次,在果蝇嗅觉搜索环节引入二次搜索机制,较好平衡了算法的全局探索与局部开发; 最后,采取动态实时更新的视觉搜索策略,进一步加速了算法的收敛速度. 用13个基准测试函数验证本文算法的可行性与有效性,最后将该算法应用于解决2个实际工程问题.

1　基本果蝇优化算法

果蝇的嗅觉器官能够嗅到40千米内食物发出的气味,果蝇正是凭借其强大的嗅觉能力向食物源靠近; 当接近食物源时,果蝇利用其敏锐的视觉能力发现食物或同伴的聚集位置,并直接向该位置飞去. 受果蝇觅食行为的启发,Pan[12]提出了果蝇优化算法,其基本优化步骤如下.

step 1: 随机初始化果蝇种群位置$X_{\rm axis}$和$Y_{\rm axis}$.

step 2: 按照下式更新果蝇个体i的位置:

$$\begin{cases} X_i=X_{\rm axis}+{\rm RandomValue};\\ Y_i=Y_{\rm axis}+{\rm RandomValue}. \end{cases} \tag{2}$$

其中RandomValue为固定步长区间$[-L,L]$内随机生成的步长值.

step 3: 计算果蝇个体与坐标原点的距离Dist_i和气味浓度判定值S_i:

$$\mathrm{Dist}_i=\sqrt{X_i^2+Y_i^2}, \tag{3}$$

$$S_i=1/\mathrm{Dist}_i. \tag{4}$$

step 4: 气味浓度判定值S_i对应了优化问题中的决策变量,将其代入气味浓度判定函数(适应度函数),得到每个果蝇位置对应的气味浓度值

$$\mathrm{Smell}_i=\mathrm{Function}(S_i). \tag{5}$$

step 5: 根据下式找出果蝇种群中气味浓度最高的果蝇个体(针对最小优化问题):

$$[\mathrm{bestSmell},\mathrm{bestIndex}]=\min(\mathrm{Smell}). \tag{6}$$

其中: bestSmel为最高气味浓度值, bestIndex为气味浓度最高果蝇个体的序号.

step 6: 按下式保存最高气味浓度值及其对应的位置坐标,果蝇种群利用视觉飞向该位置:

$$\begin{cases} \mathrm{Smellbest}=\mathrm{bestSmell},\\ X_{\rm axis}=X(\mathrm{bestIndex}),\\ Y_{\rm axis}=Y(\mathrm{bestIndex}). \end{cases} \tag{7}$$

step 7: 重复循环step 2 ~ step 5,如果当前的最高气味浓度值优于上一代的最高气味浓度值,并且当前迭代次数小于最大迭代次数,则执行step 6;否则,算法退出循环并输出寻优结果.

2 改进的果蝇优化算法

FOA中的气味浓度判定值S_i为距离Dist_i的倒数,导致S_i无法取负值且不能均匀分布于搜索空间,使得FOA的应用受到极大的限制. 设果蝇个体i的位置向量为$X_i=(x_i^1,x_i^2,\dots,x_i^n)$,本文直接按$S_i=X_i$得到气味浓度判定值,即放弃了果蝇距离$\mathrm{Dist}_i$的计算,使算法的计算环节得到简化,同时解决了$S_i$不能取负值以及在解空间中不是均匀分布的不足.

2.1 算法的嗅觉搜索

受粒子群优化算法(particle swarm optimization, PSO)[23]由粒子个体历史最佳位置和种群全局历史最佳位置共同引导算法寻优搜索的启发,本文将果蝇个体i按下式进行第1次搜索:

$$x_i^j(t+1)=\begin{cases}\mathrm{Pbest}_i^j(t)+\omega\times(\mathrm{Pbest}_{r_2}^j(t)-\mathrm{Pbest}_{r_1}^j(t)),\\ \qquad\qquad \mathrm{rand}_i^j<P;\\ \mathrm{Gbest}^j(t)+\omega\times|\mathrm{Gbest}^j(t)-\mathrm{Pbest}_i^j(t)|\times\\ \exp(r_1)\times\cos(2\pi\times r_2),\ \mathrm{rand}_i^j\geqslant P.\end{cases}\tag{8}$$

$$\omega=\frac{1}{2}\times\left(1+\cos\left(\pi\times\sqrt{\frac{t-1}{t_{\max}}}\right)\right).\tag{9}$$

其中:x_i^j为位置向量X_i的第j维分量;Pbest_i^j、$\mathrm{Pbest}_{r_1}^j$及$\mathrm{Pbest}_{r_2}^j$分别为果蝇个体i、r_1及r_2的历史最佳位置向量Pbest_i、Pbest_{r_1}及Pbest_{r_2}的第j维分量;r_1、r_2为$[1,m]$内满足$r_1\neq r_2\neq i\neq\mathrm{bestIndex}$的随机整数;$\mathrm{Gbest}^j$为种群全局历史最佳位置向量Gbest的第$j$维分量;$\mathrm{rand}_i^j$、$r_1$、$r_2$为$[0,1]$内服从均匀分布的随机数;$P$为选择概率;$t$为当前迭代次数,$t_{\max}$为最大迭代次数;$\omega$为扰动缩放系数.

可见,果蝇个体i对应的第j维位置分量x_i^j是以概率P和$1-P$分别选择$\mathrm{Pbest}_i^j(t)$或$\mathrm{Gbest}^j(t)$为扰动中心进行扰动而得到. 其中,扰动中心$\mathrm{Pbest}_i^j(t)$对应的扰动差分分量为$\omega\times(\mathrm{Pbest}_{r_2}^j(t)-\mathrm{Pbest}_{r_1}^j(t))$,扰动中心$\mathrm{Gbest}^j(t)$对应的扰动差分分量为$\omega\times|\mathrm{Gbest}^j(t)-\mathrm{Pbest}_i^j(t)|\times\exp(r_1)\times\cos(2\pi\times r_2)$. 前一种扰动方案兼顾了果蝇种群的多样性,而后一种扰动方案则加速了种群的收敛性. 扰动缩放系数ω采用式(9)非线性递减的取值策略,确保算法在进化的前期阶段以较大的搜索半径进行搜索,避免过早收敛;随着算法迭代次数的增加,搜索半径逐渐减小,有助于算法在局部邻域挖掘质量更优的候选解.

为了进一步提高算法的收敛精度及收敛速度,若果蝇种群的最优位置信息Gbest连续age代没有得到更新,则按下式进行第2次局部深度搜索:

$$\mathrm{Gb}=(\mathrm{Gbest}(t)+\mathrm{Pbest}_{r_3}(t))/2+\omega\times(\mathrm{Pbest}_{r_4}(t)-\mathrm{Pbest}_{r_5}(t)).\tag{10}$$

其中:Gb是局部深度搜索得到的中间候选解;r_3、r_4、r_5是$[1,m]$内随机选择且满足$r_3\neq r_4\neq r_5\neq\mathrm{bestIndex}$的整数. 可见,局部搜索是以$\mathrm{Gbest}(t)+\mathrm{Pbest}_{r_3}(t)/2$为中心位置,以$\omega\times(\mathrm{Pbest}_{r_4}(t)-\mathrm{Pbest}_{r_5}(t))$为差分扰动向量对其进行扰动得到中间候选解.

综上所述,两次搜索的最大区别在于:第1次搜索是基于位置向量的“单维”层面展开,对于保持种群多样性的能力更强;第2次搜索则是在位置向量层面上进行,对于加速算法收敛速度的贡献更大. 尽管IFOA算法的嗅觉搜索借鉴了PSO算法向种群历史经验学习的思想,但两者的进化机制有本质区别:PSO算法中的粒子位置是以速度为步长进行更新,而粒子飞行速度的更新由粒子的动量部分、自我认知部分和社会认知部分共同决定;IFOA算法的嗅觉搜索是利用不同果蝇个体之间的差异信息在扰动中心位置附近进行随机扰动搜索,同时采用扰动缩放系数ω来平衡算法的全局探索与局部开发. IFOA算法的嗅觉搜索机制使得其比PSO算法能够更好地保持种群的多样性,进而具有更强的全局搜索能力.

2.2 算法的视觉搜索

FOA算法的搜索过程按果蝇种群“宏观”层面划分为嗅觉搜索和视觉搜索两个环节,即所有果蝇个体按式(2)完成嗅觉搜索后,才找出果蝇种群中气味浓度值最高的果蝇个体,然后整个果蝇种群利用视觉飞向该果蝇个体对应的位置. 这种只向最优个体学习的搜索方式存在以下不足:首先,如果最佳果蝇个体对应的位置为局部极值,将容易导致算法早熟收敛现象的发生;其次,这种搜索方式忽略了非最优果蝇个体对种群搜索的贡献,过于强调果蝇个体间的竞争,而弱化了个体间的协作.

在IFOA中,将从果蝇个体的“微观”层面来展开算法的嗅觉搜索与视觉搜索,使得所有果蝇个体的成功搜索经验得到充分利用. IFOA中的视觉搜索是指式(8)及(10)中搜索中心位置向量的动态实时更新,它是通过下式实时更新果蝇个体的历史极值和种群历史极值间接实现的:

$$\mathrm{Pbest}_i=\begin{cases}X_i,\ F(X_i)\leqslant F(\mathrm{Pbest}_i);\\ \mathrm{Pbest}_i,\ F(X_i)>F(\mathrm{Pbest}_i).\end{cases}\tag{11}$$

$$\mathrm{Gbest}=\begin{cases}X_i,\ F(X_i)\leqslant F(\mathrm{Gbest});\\ \mathrm{Gbest},\ F(X_i)>F(\mathrm{Gbest}).\end{cases}\tag{12}$$

$$\mathrm{Gbest}=\begin{cases}\mathrm{Gb},\ F(\mathrm{Gb})\leqslant F(\mathrm{Gbest});\\ \mathrm{Gbest},\ F(\mathrm{Gb})>F(\mathrm{Gbest}).\end{cases}\tag{13}$$

其中$F(\cdot)$为经过约束处理后的惩罚适应度函数.

综上,IFOA算法采用实时更新的视觉搜索策略能够充分利用每个果蝇个体的成功搜索经验,有效提高种群的搜索效率.

2.3　边界处理

在算法寻优过程中,会出现个体位置超出搜索空间范围的现象. 对于越界现象,通常的做法是将其设置为边界值. 为增强种群多样性,本文将越界分量按下式进行处理:

$$x_i^j=\begin{cases}u_j-\min(x_i^j-u_j,u_j-l_j)\times r,\ x_i^j>u_j;\\ l_j+\min(l_j-x_i^j,u_j-l_j)\times r,\ x_i^j<l_j.\end{cases}\tag{14}$$

其中:r为[0,1]内均匀分布的随机数,u_j、l_j分别为x_i^j的上边界值和下边界值,$\min(\cdot)$为取最小值函数.

2.4　约束处理

对于约束优化问题,除了要考虑目标函数外,还要考虑约束条件. 研究者提出了多种约束处理技术,其中罚函数法是最常用的约束处理技术之一. 罚函数法的思想是:基于个体的约束违反程度构造惩罚项,通过对原目标函数增加惩罚项来构造惩罚适应度函数,将约束优化问题转换为无约束优化问题进行处理[24]. 本文具体采用的约束处理方法如下:

$$\min\ F(X)=f(X)+\phi(X);\tag{15}$$

$$\phi'(X)=\sum_{j=1}^{q}G_j+\sum_{j=q+1}^{m}H_j;\tag{16}$$

$$\phi(X)=\begin{cases}\phi'(X)\times\mathrm{cn}\times\delta_1,\ |f(X)|<\mathrm{fit};\\ \phi'(X)\times\delta_2,\ |f(X)|\geqslant\mathrm{fit};\end{cases}\tag{17}$$

$$G_j=(\max(0,g_j(X)))^{\alpha};\tag{18}$$

$$H_j=|h_j(X)|^{\beta}.\tag{19}$$

其中:$f(X)$为原目标函数;$F(X)$为使用罚函数修正后的目标函数,即惩罚适应度函数;$\phi(X)$为罚函数;$\phi'(X)$为约束的违反程度,简称约束违反度;cn为违反约束条件的个数,简称约束违反数;δ_1、δ_2为正的惩罚系数,一般前者取值较小,而后者取值较大;fit为较小的正常数;$g_j(X)$为不等式约束;$h_j(X)$为等式约束;G_j为不等式约束$g_j(X)$的约束违反度;H_j为等式约束$h_j(X)$的约束违反度;α和β通常设置为1或2,对于不等式约束本文设置为1(即$\alpha=1$),对于等式约束本文设置为2(即$\beta=2$).

2.5　算法步骤

IFOA算法的具体步骤如下.

step 1: 设置个体的搜索范围、种群规模、最大迭代次数等相关参数,在搜索空间内对果蝇个体位置进行随机初始化.

step 2: 计算每个果蝇个体的适应值,根据式(15)～(19)进行约束处理,个体的初始位置即为该个体的个体历史最佳位置,种群最佳位置即为种群全局历史最佳位置.

step 3: 按照式(8)和(9)更新果蝇个体i位置并按式(14)进行越界处理,计算果蝇个体i的适应值,并按式(15)～(19)进行约束处理,按式(11)实时更新个体i的历史最佳位置Pbest_i,按式(12)实时更新种群的历史最佳位置Gbest.

step 4: 若种群的所有个体更新完毕,则执行step 5,否则返回step 3更新种群中的下一个果蝇个体.

step 5: 如果种群最优位置Gbest连续age代没有得到更新,则按照式(9)、(10)进行$k_{\max}$次局部深度搜索,并参照式(14)进行越界处理,按式(15)～(19)进行约束处理,按式(13)实时更新种群的历史最佳位置Gbest.

step 6: 若种群局部搜索完毕,则执行step 7,否则返回step 5.

step 7: 若算法达到最大迭代次数,则寻优过程结束并输出寻优结果,否则返回step 3继续执行下一代迭代操作.

3　实验结果及分析

为了检验提出算法求解约束优化问题的性能,采用13个典型的约束测试函数对算法进行测试,关于测试函数的详细描述说明见参考文献[25].

3.1　不同改进FOA算法的对比

为了验证IFOA算法处理约束优化问题的有效性,将其对上述13个约束优化问题的优化结果与FOA、LGMS-FOA[18]、AE-LGMS-FOA[19]、SFOA[21]及AFOA[22]的优化结果进行对比. 所有算法采用Matlab R2013a进行编程,计算机配置为Intel Core (TM) i7-7700、3.6 GHz、16 GB内存、Windows 10操作系统.

所有算法的种群规模为100,最大迭代次数为5 000. IFOA算法的其余参数设置为$P=0.8,\mathrm{fit}=0.1,\alpha=1,\beta=2,\mathrm{age}=10,k_{\max}=100$;其余算

法的相关参数按对应参考文献进行设置. 各测试函数分别独立运行50次,记录各算法寻优搜索的平均值mean、 最大值max、 最小值min、 标准差std及函数的平均有效评估次数ANVE(average number of valid evaluations)如表1所示. 其中,平均有效评估次数是指算法按指定精度收敛于全局最优值时的最小评估次数的平均值(对于未能收敛于函数全局最优的情形,则按算法完成最大迭代次数时对函数的总评估次数进行统计). 对于函数G2、G3、G8、G11、G12及G13的收敛精度要求为10^{-4},其余函数的收敛精度要求为10^{-3}. 各算法对部分函数的平均寻优收敛曲线如图1所示.

表1 FOA及其改进算法对约束优化问题的寻优统计结果

函数/最优值	算法	max	mean	min	std	ANVE
G1/−15	FOA	−4.499 83	−7.288 99	−11.191 43	1.47	500 000
	LGMS-FOA	−0.045 60	−0.733 73	−1.729 41	0.33	500 000
	AE-LGMS-FOA	0.035 24	−4.017 00	−14.985 17	3.94	505 000
	AFOA	-4.880 19	−7.674 08	−10.013 94	1.19	500 000
	SFOA	−1.726 69	−3.924 94	−5.739 29	0.96	500 000
	IFOA	−15	−15	−15	0	164 111.58
G2/−0.803 619	FOA	−2.093 615e+03	−2.104 852e+03	−2.117 903e+03	5.41e+00	500 000
	LGMS-FOA	−0.158 223	−0.178 253	−0.214 417	1.26e-02	500 000
	AE-LGMS-FOA	−0.182 550	−0.196 231	−0.229 199	8.85e-03	505 000
	AFOA	−0.405 287	−0.488 676	−0.610 051	5.00e-02	500 000
	SFOA	−1.507 066	−6.738 172	−12.305 756	2.42e+00	539 199
	IFOA	−0.803 603	−0.803 617	−0.803 619	2.39e-06	553 916.66
G3/−1	FOA	−0.940 69	−1.320 66	−6.658 63	1.26e+00	500 000
	LGMS-FOA	−2.290 18e-04	−0.040 13	−0.125 73	3.49e-02	500 000
	AE-LGMS-FOA	−0.001 51	−0.063 00	−0.377 66	5.99e-02	505 000
	AFOA	0	−0.800 55	−0.947 51	2.16e-01	500 000
	SFOA	0.024 84	−0.005 71	−0.164 68	2.91e-02	500 000
	IFOA	−1.000 52	−1.000 52	−1.000 52	9.94e-16	434 916.78
G4/−30 665.539	FOA	−40 771.614	−40 786.234	−40 789.641	3.36e+00	500 000
	LGMS-FOA	−29 656.027	−30 002.558	−30 348.929	1.50e+02	500 000
	AE-LGMS-FOA	−30 060.773	−30 286.509	−30 478.594	7.07e+01	505 000
	AFOA	−30 310.682	−30 501.229	−30 827.315	1.02e+02	500 000
	SFOA	−40 790.497	−40 792.150	−40 794.403	6.96e-01	500 000
	IFOA	−30 665.538	−30 665.538	−30 665.538	2.18e-11	134 756.46
G5/5 126.498 1	FOA	9 510.499 3	1 167.258 9	370.621 9	1.22e+03	500 000
	LGMS-FOA	6 160.389 5	5 583.083 6	5 282.350 1	2.345e+02	500 000
	AE-LGMS-FOA	6 108.197 6	5 410.873 6	5 173.883 9	2.08e+02	505 000
	AFOA	6 043.043 7	4 839.521 4	0.050 0	1.63e+03	500 000
	SFOA	0.025 0	0.003 7	−0.078 8	1.29e-02	500 000
	IFOA	5 126.495 4	5 126.495 4	5 126.495 4	2.47e-08	786 696.56
G6/−6 961.813 88	FOA	−7 643.706 96	−7 965.501 01	−8 837.683 68	2.60e+02	500 000
	LGMS-FOA	0.098 26	−2 279.578 41	−4 231.285 15	1.35e+03	500 000
	AE-LGMS-FOA	0.099 13	−211.285 97	−3 787.200 93	7.62e+02	505 000
	AFOA	−6 846.156 37	−6 916.083 71	−6 965.715 04	2.86e+01	500 000
	SFOA	−8 871.215 23	−9 001.479 03	−9 051.188 33	2.78e+01	500 000
	IFOA	−6 961.813 88	−6 961.813 88	−6 961.813 88	1.82e-12	129 950.76
G7/24.306 209 1	FOA	8 958.235 376 5	1 391.891 270 7	995.205 517 8	1.23e+03	500 000
	LGMS-FOA	31.474 013 86	26.321 770 9	24.622 172 1	1.46e+00	500 000
	AE-LGMS-FOA	25.429 286 1	24.604 033 6	24.349 029 6	1.87e-01	505 000
	AFOA	1.290 103 8e+06	4.370 601 5e+04	34.095 652 0	1.87e+05	500 000
	SFOA	1 683.660 722 1	1 324.196 588 0	1 118.874 243 1	8.54e+01	500 000
	IFOA	24.306 778 6	24.306 435 4	24.306 224 4	1.29e-04	695 875.24
G8/−0.095 825	FOA	0.095 521	−0.067 493	−0.095 820	5.19e-02	499 591.84
	LGMS-FOA	−0.007 271	−0.022 513	−0.095 822	2.53e-02	470 752.34
	AE-LGMS-FOA	−0.002 132	−0.040 029	−0.094 542	2.54e-02	505 000
	AFOA	−0.095 735	−0.095 797	−0.095 824	2.35e-05	499 659.06
	SFOA	79.718 701	−409.687 560	−947.029 380	2.85e+02	500 000
	IFOA	−0.095 825	−0.095 825	−0.095 825	8.33e-17	10 331.56

表1(续)

函数/最优值	算法	max	mean	min	std	ANVE
G9/680.630 057 3	FOA	859.041 909 3	739.063 747 3	690.213 583 7	5.19e+01	500 000
	LGMS-FOA	681.936 427 6	680.955 506 3	680.640 905 8	2.79e-01	500 000
	AE-LGMS-FOA	680.931 670 0	680.676 189 6	680.633 205 4	6.50e-02	505 000
	AFOA	717.696 006 4	687.704 340 3	681.677 494 8	7.30e+00	500 000
	SFOA	17 981.365 186 6	1 427.368 600 3	897.763 219 2	2.37e+03	500 000
	IFOA	680.630 057 5	680.630 057 4	680.630 057 4	2.02e-08	559 717.38
G10/7 049.330 7	FOA	1.603 2	0.497 3	0.250 7	0.245	500 000
	LGMS-FOA	20 484.182 4	13 051.728 2	7 932.904 7	2.77e+03	500 000
	AE-LGMS-FOA	19 608.203 5	12 327.398 3	5 238.579 6	2.58e+03	505 000
	AFOA	9 814.891 0	1 987.755 1	−22 233.762 0	7.50e+03	500 000
	SFOA	6.652 7e-04	−8.118 4e-05	−1.314 6e-03	3.61e-04	500 000
	IFOA	7 049.284 4	7 049.260 7	7 049.249 4	6.40e-03	883 825.38
G11/0.75	FOA	53.22	2.10	0.75	7.34e+00	498 916.66
	LGMS-FOA	0.75	0.75	0.75	3.15e-11	12 328.58
	AE-LGMS-FOA	0.75	0.75	0.75	2.83e-11	235 029.70
	AFOA	0.75	0.75	0.75	4.00e-05	499 015.48
	SFOA	11.48	1.57	0.09	1.99e+00	500 000
	IFOA	0.75	0.75	0.75	2.70e-11	372 647.46
G12/−1	FOA	−0.278 22	−0.301 32	−0.360 52	2.19e-02	500 000
	LGMS-FOA	−0.986 86	−0.994 09	−0.999 73	3.48e-03	470 699.18
	AE-LGMS-FOA	−0.997 27	−0.998 84	−0.999 96	7.04e-04	386 976.52
	AFOA	−0.999 95	−0.999 98	−1.000 00	1.09e-05	499 655.18
	SFOA	−0.194 41	−0.254 29	−0.478 28	4.34e-02	500 000
	IFOA	−1	−1	−1	0	4 505.80
G13/0.053 949 8	FOA	1.208 449 0	1.009 437 6	1.000 003 4	3.69e-02	500 000
	LGMS-FOA	0.676 351 7	0.097 145 7	0.054 145 2	1.12e-01	480 349.96
	AE-LGMS-FOA	1.009 929 2	0.114 258 9	0.053 976 0	2.12e-01	474 935.70
	AFOA	3.683 427 6	0.823 785 4	1.901 662 2	6.33e-01	500 000
	SFOA	1.818 661 7	1.006 407 4	0.825 839 7	1.31e-01	500 000
	IFOA	0.054 300 3	0.053 958 8	0.053 924 1	6.98e-05	904 975.02

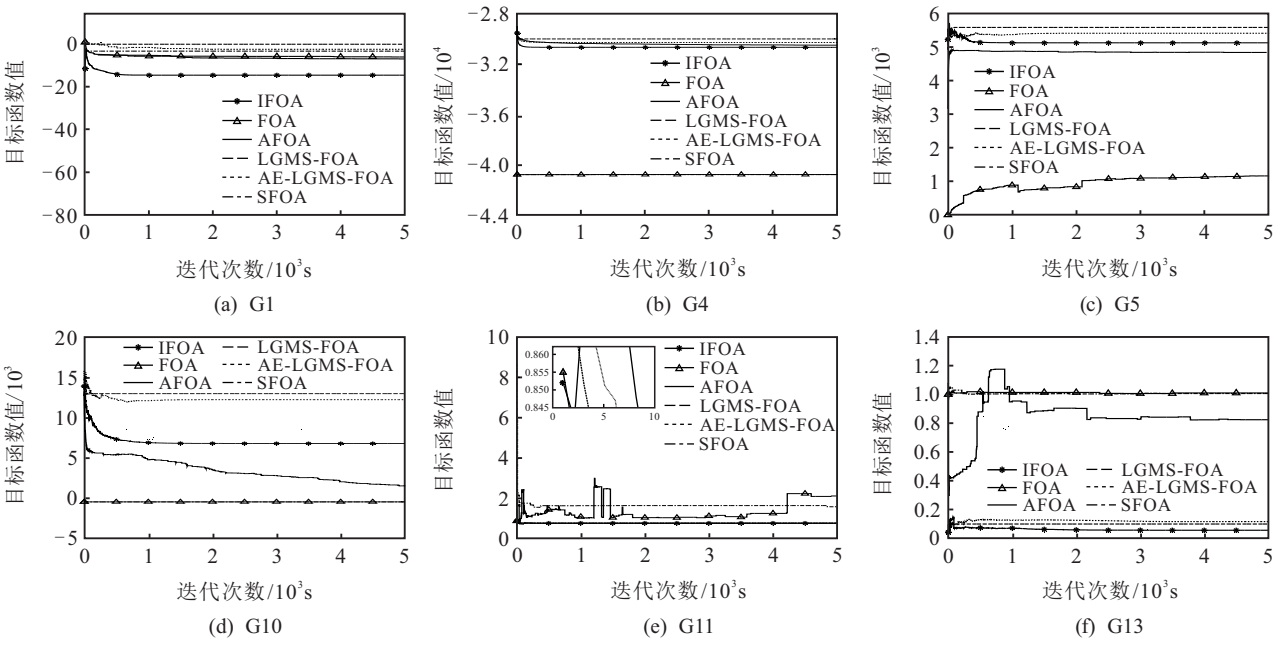

图1　IFOA对部分函数的平均寻优收敛曲线

根据表1的统计结果可知: 对于函数G11, LGMS-FOA、AE-LGMS-FOA、AFOA及IFOA算法能够以非常高的精度收敛于最优值,它们获得了相近的收敛质量,而FOA及SFOA算法对该函数寻优效果不理想;对于除G11外的其余约束优化问题,FOA、LGMS-FOA、AE-LGMS-FOA、AFOA及SFOA算法都无法对问题的最优解进行有效搜索;对于函数G1、G6、G8、G11及G12, IFOA算法能够收敛至其最优值,同时算法的稳定性好,对于其余函数, IFOA也获得了较高的收敛精度,且同样保持了较好的寻优稳定性;对于函数G1、G3、G4、G6、G8及G12, IFOA算法按指定精度收敛于全局最优时所用的平均函数评估次数低于其他算法的平均评估次数,说明了其高效的全局搜索能力有效降低了算法的寻优时间,使得在相同函数评估次数的前提下,IFOA算法具有更高的收敛精度和更强的寻优稳定性. 图1的平均寻优收敛曲线验证了上述结论,进一步说明IFOA算法能够用于有效解决约束优化问题.

3.2 IFOA与其他优化算法的对比

为进一步验证IFOA处理约束优化问题的能力,将IFOA的优化结果与文献[2]中的ε-DE算法及文献[26]中的ABC、PSO及HCS-LSAL算法的优化结果进行比较,各算法寻优的平均值mean、最优值best、最差值worst及标准差std如表2所示.

表2 IFOA与其他智能优化算法对约束优化问题的寻优统计结果

函数/最优值	算法	best	mean	worst	std
G1/−15	ε-DE	−15.000	−15.000	−15.000	0
	ABC	−15	−15	−15	0
	PSO	−15	−14.715 1	−12.453 1	7.40e-01
	HCS-LSAL	−15.000 0	−15.000 0	−15.000 0	1.21e-08
	IFOA	−15	−15	−15	0
G2 / −0.803 619	ε-DE	−0.803 619	−0.803 004	−0.792 608	2.47e-03
	ABC	−0.803 615	−0.799 336	−0.777 438	6.84e-03
	PSO	−0.803 443	−0.740 577	−0.631 598	4.20e-02
	HCS-LSAL	−0.803 619	−0.762 088	−0.640 249	4.10e-02
	IFOA	−0.803 619	−0.803 617	−0.803 603	2.39e-06
G3 / −1	ε-DE	−1	−1	−1	3.9e-06
	ABC	−1	−1	−1	4.68e-05
	PSO	−1.004 8	−1.003 4	−0.997 6	1.70e-02
	HCS-LSAL	−1.000 000	−1.000 000	−1.000 000	1.00e-10
	IFOA	−1.000 52	−1.000 52	−1.000 52	9.94e-16
G4 / −30 665.539	ε-DE	−30 665.539	−30 665.539	−30 665.539	2.1e-05
	ABC	−30 665.539	−30 665.539	−30 665.539	2.22e-11
	PSO	−30 665.539	−30 665.539	−30 665.539	0
	HCS-LSAL	−30 665.539	−30 665.539	−30 665.539	7.20e-06
	IFOA	−30 665.538	−30 665.538	−30 665.538	2.18e-11
G5 / 5 126.498 1	ε-DE	5 126.498	5 126.498	5 126.498	1.7e-05
	ABC	5 126.736	5 178.139	5 317.196	5.60e+01
	PSO	5 126.484 2	5 202.362 7	5 520.146 7	1.10e+02
	HCS-LSAL	5 126.498 1	5 126.498 1	5 126.498 1	6.36e-06
	IFOA	5 126.495 4	5 126.495 4	5 126.495 4	2.47e-08
G6 / −6 961.813 88	ε-DE	−6 961.814	−6 961.814	−6 961.814	2.3e-08
	ABC	−6 961.814	−6 961.814	−6 961.814	0
	PSO	−6 961.814	−6 961.814	−6 961.814	0
	HCS-LSAL	−6 961.814	−6 961.814	−6 961.814	8.36e-06
	IFOA	−6 961.813 88	−6 961.813 88	−6 961.813 88	1.82e-12
G7 / 24.306 209 1	ε-DE	24.306	24.306	24.306	6.3e-06
	ABC	24.315	24.415	24.854	1.24e-01
	PSO	24.319	24.989	26.194	5.51e-01
	HCS-LSAL	24.306 2	24.306 2	24.306 2	4.93e-08
	IFOA	24.306 224 4	24.306 435 4	24.306 778 6	1.29e-04
G8 / −0.095 825	ε-DE	−0.095 825	−0.095 825	−0.095 825	8.4e-17
	ABC	−0.095 825	−0.095 825	−0.095 825	4.23e-17
	PSO	−0.095 825	−0.095 825	−0.095 825	0
	HCS-LSAL	−0.095 825	−0.095 825	−0.095 825	0
	IFOA	−0.095 825	−0.095 825	−0.095 825	8.33e-17

表2(续)

函数/最优值	算法	best	mean	worst	std
G9/680.630 057 3	ε-DE	680.63	680.63	680.63	2.2e-07
	ABC	680.632	680.647	680.691	1.55e-02
	PSO	680.632	680.653	680.699	1.80e-02
	HCS-LSAL	680.630 1	680.630 1	680.630 1	1.43e-05
	IFOA	680.630 057 4	680.630 057 4	680.630 057 5	2.02e-08
G10/7 049.330 7	ε-DE	7 049.248	7 049.248	7 049.248	9.0e-06
	ABC	7 051.706	7 233.882	7 473.109	1.10e+02
	PSO	7 054.125 6	7 173.266 1	7 335.247 7	8.40e+02
	HCS-LSAL	7 049.237	7 099.668	7 250.957	8.65e+01
	IFOA	7 049.284 4	7 049.260 7	7 049.249 4	6.40e-03
G11/0.75	ε-DE	0.75	0.75	0.75	6.9e-14
	ABC	0.75	0.75	0.75	2.30e-05
	PSO	0.749	0.749	0.749	3.90e-07
	HCS-LSAL	0.749 999	0.750 000	0.750 000	3.70e-09
	IFOA	0.75	0.75	0.75	2.70e-11
G12/−1	ε-DE	−1.000	−1.000	−1.000	0
	ABC	−1.000	−1.000	−1.000	0
	PSO	−1	−1	−1	0
	HCS-LSAL	−1.000 00	−1.000 00	−1.000 00	0
	IFOA	−1	−1	−1	0
G13/0.053 949 8	ε-DE	0.053 949	0.069 631	0.438 846	7.6e-02
	ABC	0.053 985	0.158 552	0.442 905	1.72e-01
	PSO	0.053 866 6	0.552 753	1.856 102	4.20e-01
	HCS-LSAL	0.053 949 8	0.053 949 8	0.053 949 8	1.00e-10
	IFOA	0.053 924 1	0.053 958 8	0.054 300 3	6.98e-05

从表2可知: 与ε-DE相比, 对于函数G1、G7、G8、G9、G11及G12, IFOA获得了相似的收敛性能; 对于函数G6及G13, IFOA的整体收敛性能优于ε-DE; 对于函数G2, IFOA获得了相似的最优值和较好的平均值、最差值和标准差; 对于函数G3, IFOA获得了相似的最优值、平均值及最差值和较好的标准差; 对于G4和G5, IFOA的最优值、平均值及最差值略差于ε-DE, 但其寻优稳定性远优于ε-DE; 对于G10, IFOA的标准差略差于ε-DE, 但获得了较好的最优值、平均值及最差值. 与ABC相比, 对于函数G1、G6、G8及G12, IFOA获得了相似的收敛性能; 对于函数G2、G5、G7、G9、G10及G13, IFOA获得了较好的整体收敛性能; 对于函数G3和G11, IFOA获得了相似的最优值、平均值及最差值, 但其算法稳定性远优于ABC; 对于G4, IFOA的寻优稳定性略优于ABC, 但其最优值、平均值及最差值略差于ABC. 与PSO相比, 对于函数G1、G2、G3、G5、G7、G9、G10、G11及G13, IFOA的整体收敛性能明显优于PSO; 对于其余函数, IFOA获得了近似的收敛结果. 与HCS-LSAL相比, 对于函数G1、G3、G6、G9及G11, IFOA获得了相似的最优值、平均值和最差值, 但其算法稳定性优于HCS-LSAL; 对于函数G2, IFOA获得了相似的最优值和较好的平均值、最差值及标准差; 对于函数G4和G5, IFOA的最优值、平均值及最差值略差于HCS-LSAL, 但其寻优稳定性远优于HCS-LSAL; 对于函数G7和G13, IFOA的整体收敛性能略低于HCS-LSAL; 对于函数G10, IFOA获得了较好的整体收敛结果; 对于G8和G12, 两者的整体收敛性能相当.

综上所述, 相对于其他智能优化算法, IFOA算法在处理复杂约束优化问题上具备相当的竞争力.

4　IFOA在机械优化设计中的应用

为检验IFOA解决实际工程问题的有效性, 将其用于解决机械设计中的两个工程优化问题, 即焊接梁最小费用问题和最小化张力弦质量问题.

问题1　焊接梁最小费用问题.

该问题的决策变量为$h(x_1)$、$l(x_2)$、$t(x_3)$及$b(x_4)$, 问题的数学描述如下:

$$\min f(x) = 1.104\,71x_1^2x_2 + 0.048\,11x_3x_4(14.0 + x_2);$$

$$\text{s.t. } g_1(x) = \tau(x) - \tau_{\max} \leqslant 0,$$

$$g_2(x) = \sigma(x) - \sigma_{\max} \leqslant 0,$$

$$g_3(x) = x_1 - x_4 \leqslant 0,$$

$$g_4(x) = 0.104\,71x_1^2 + 0.048\,11x_3x_4(14.0 + x_2) - 5 \leqslant 0,$$

$$g_5(x) = 0.125 - x_1 \leqslant 0,$$

$$g_6(x) = \delta(x) - \delta_{\max} \leqslant 0,$$

$$g_7(x) = P - P_c(x)1 \leqslant 0.$$

其中

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},$$

$$M = P\Big(L + \frac{x_2}{2}\Big),\ \sigma(x) = \frac{6PL}{x_4x_3^2},$$

$$L = 14, R = \sqrt{\frac{x_2^2}{4} + \Big(\frac{x_1 + x_3}{2}\Big)^2},$$

$$\delta(x) = \frac{4PL^3}{Ex_3^3x_4},\ P = 6\,000,$$

$$J = 2\Big\{\sqrt{2}x_1x_2\Big[\frac{x_2^2}{12} + \Big(\frac{x_1 + x_3}{2}\Big)^2\Big]\Big\},$$

$$G = 12 \times 10^6,\ E = 30 \times 10^6,$$

$$P_c(x) = \frac{4.013\mathrm{E}\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\Big(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\Big),$$

$$\tau_{\max} = 13\,600,\ \sigma_{\max} = 30\,000,\ \delta_{\max} = 0.25,$$

$$0.1 \leqslant x_1 \leqslant 2,\ 0.1 \leqslant x_2 \leqslant 10,$$

$$0.1 \leqslant x_3 \leqslant 10,\ 0.1 \leqslant x_4 \leqslant 2.$$

问题2 最小化张力弦质量问题.

该问题的决策变量为$d(x_1)$、$D(x_2)$及$p(x_3)$,问题的数学描述如下:

$$\min f(x) = (x_3 + 2)x_2x_1^2;$$

$$\text{s.t. } g_1(x) = 1 - \frac{x_2^3x_3}{71\,785x_1^4} \leqslant 0,$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12\,566(x_2x_1^3 - x_1^4)} + \frac{1}{5\,108x_1^2} - 1 \leqslant 0,$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0,$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0.$$

其中

$$0.05 \leqslant x_1 \leqslant 2.0,$$

$$0.25 \leqslant x_2 \leqslant 1.3,$$

$$2.0 \leqslant x_3 \leqslant 15.0.$$

对于上述两个工程问题,用IFOA分别独立运行30次(算法的参数设置与前述相同),将其结果与CDE[1]、CTLBO[7]、IFA[8]及UABC[9]的结果进行对比,其结果统计如表3～表6所示(NA表示原文没有提供具体数据).

表3 不同算法求解问题1的最优结果比较

变量	CDE	CTLBO	IFA	UABC	IFOA
$x_1(h)$	0.203 137	0.205 730	0.205 730	0.205 730	0.205 730
$x_2(l)$	3.542 998	3.470 489	3.470 489	3.470 489	3.470 489
$x_3(t)$	9.033 498	9.036 626	9.036 624	9.036 624	9.036 624
$x_4(b)$	0.206 480	0.205 730	0.205 730	0.205 730	0.205 730
$g_1(x)$	−44.578 568	−0.000 002	−1.70e+04	−0.000 028	−0.000 000
$g_2(x)$	−44.663 534	−0.009 189	−0.510 316	−0.000 025	−0.000 000
$g_3(x)$	−0.003 042	−0.000 000	−1.21e-06	−0.000 000	−0.000 000
$g_4(x)$	−3.423 726	−3.432 984	−3.432 950	−3.432 984	−3.432 984
$g_5(x)$	−0.078 137	−0.080 730	−0.080 729	−0.080 730	−0.080 730
$g_6(x)$	−0.235 557	−0.235 540	−0.235 541	−0.235 540	−0.235 540
$g_7(x)$	−38.028 268	−0.000 004	−0.042 917	−0.000 050	−3.64e-12
$f(x)$	1.733 462	1.724 852	1.724 894	1.724 852	1.724 852

表4 不同算法求解问题1的寻优统计比较

算法	最优解	均值	最差解	标准差
CDE	1.733 461	1.768 158	1.824 105	2.2e-02
CTLBO	1.724 852	1.724 852	1.724 853	0
IFA	1.724 894	1.724 978	1.725 321	1.33e-05
UABC	1.724 852	1.724 853	NA	1.70e-06
IFOA	1.724 852	1.724 852	1.724 852	1.11e-15

表5 不同算法求解问题2的最优结果比较

变量	CDE	CTLBO	IFA	UABC	IFOA
$x_1(d)$	0.051 609	0.051 664	0.051 632 1	0.051 691	0.051 689 0
$x_2(D)$	0.354 714	0.356 112	0.355 341 7	0.356 769	0.356 716 9
$x_3(p)$	11.410 831	11.313 513	11.370 456	11.285 988	11.289 013 7
$g_1(x)$	−3.90e-05	−1.50e-08	−4.89e-06	−0.000 000	−5.37e-12
$g_2(x)$	−1.83e-04	−5.60e-08	−1.89e-05	−0.000 000	−4.62e-12
$g_3(x)$	−4.048 627	−4.057 519	−4.050 933	−4.053 886	−4.053 784
$g_4(x)$	−0.729 118	−0.728 149	−0.728 684	−0.727 694	−0.727 729
$f(x)$	0.012 670 2	0.012 654 7	0.012 665 8	0.012 665	0.012 665 2

表 6　不同算法求解问题2的寻优统计比较

算法	最优解	均值	最差解	标准差
CDE	0.012 670 2	0.012 670 3	0.012 679 0	2.7e-05
CTLBO	0.012 654 7	0.012 654 93	0.012 656 93	5.50e-07
IFA	0.012 665 8	0.012 706 0	0.012 812 0	4.37e-05
UABC	0.012 665	0.012 683	NA	3.31e-05
IFOA	0.012 665 2	0.012 665 2	0.012 665 2	1.13e-11

对于问题1, 由表3和表4可知: 与CDE和IFA相比, IFOA获得了较好的最优解、平均解、最差解及标准差; 与CTLBO相比, IFOA获得了相似的整体求解结果; 与UABC相比, IFOA获得了相似的最优解, 但其平均解、最差解及标准差优于UABC.

对于问题2, 由表5和表6可知: 与CDE和IFA相比, IFOA获得了较好的最优解、平均解、最差解及标准差; IFOA的最优解、平均解及最差解略差于CTLBO, 但IFOA的求解稳定性远优于CTLBO; 与UABC相比, IFOA获得了相似的最优解, 但其平均解、最差解及标准差优于UABC.

5　结　论

本文提出了一种改进的果蝇优化算法, 用于解决约束优化问题. 该算法克服了FOA算法的候选解不取负值以及FOA算法不能在解空间均匀搜索的不足, 拓宽了FOA算法的应用领域. 改进算法利用果蝇个体历史最佳位置信息和种群全局历史最佳位置信息来共同引导果蝇个体的嗅觉搜索, 较好保持了果蝇种群的多样性, 有效避免了算法早熟收敛现象的发生; 在果蝇视觉搜索阶段采取了实时动态更新机制, 加速了算法的收敛速度; 引入的局部邻域搜索策略, 进一步提高了算法的收敛精度. 利用13个基准测试函数验证本文算法的可行性与有效性, 实验结果表明本文算法具有较快的收敛速度、较高的收敛精度及较强的寻优稳定性. 最后, 将该算法用于解决2个工程优化问题, 取得了良好的优化效果.

参考文献(References)

[1] Huang F Z, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization[J]. Applied Mathematics and Computation, 2007, 186(1): 340-356.

[2] 郑建国, 王翔, 刘荣辉. 求解约束优化问题的ε-DE算法[J]. 软件学报, 2012, 23(9): 2374-2387.
(Zheng J G, Wang X, Liu R H. ε-Differential evolution algorithm for constrained optimization problems[J]. Journal of Software, 2012, 23(9): 2374-2387.)

[3] Gong W Y, Cai Z H, Liang D W. Adaptive ranking mutation operator based differential evolution for constrained optimization[J]. IEEE Transactions on Cybernetics, 2015, 45(4): 716-727.

[4] Elsayed S M, Sarker R A, Mezura-Montes E. Self-adaptive mix of particle swarm methodologies for constrained optimization[J]. Information Sciences, 2014, 277: 216-233.

[5] 张永强, 徐宗昌, 呼凯凯, 等. 基于私有云和改进粒子群算法的约束优化求解[J]. 系统工程与电子技术, 2016, 38(5): 1086-1092.
(Zhang Y Q, Xu Z C, Hu K K, et al. Constrained optimization problems solving based on private cloud and improved particle swarm optimization[J]. Systems Engineering and Electronics, 2016, 38(5): 1086-1092.)

[6] Yu K J, Wang X L, Wang Z. Constrained optimization based on improved teaching-learning-based optimization algorithm[J]. Information Sciences, 2016: 352/353: 61-78.

[7] 刘三阳, 靳安钊. 求解约束优化问题的协同进化教与学优化算法[J]. 自动化学报, 2018, 44(9): 1690-1697.
(Liu S Y, Jin A Z. A co-evolutionary teaching-learning-based optimization algorithm for constrained optimization problems[J]. Acta Automatica Sinica, 2018, 44(9): 1690-1697.)

[8] 龙文, 蔡绍洪, 焦建军, 等. 求解约束优化问题的萤火虫算法及其工程应用[J]. 中南大学学报: 自然科学版, 2015, 46(4): 1260-1267.
(Long W, Cai S H, Jiao J J, et al. Firefly algorithm for solving constrained optimization problems and engineering applications[J]. Journal of Central South University: Science and Technology, 2015, 46(4): 1260-1267.)

[9] Brajevic I, Tuba M L. An upgraded articial bee colony (ABC) algorithm for constrained optimization problems[J]. Journal of Intelligent Manufacturing, 2013, 24(4): 729-740.

[10] Liang Y S, Wan Z P, Fang D B. An improved artificial bee colony algorithm for solving constrained optimization

problems[J]. International Journal of Machine Learning and Cybernetics, 2017, 8(3): 739-754.

[11] Bansal J C, Joshi S K, Sharma H. Modified global best artificial bee colony for constrained optimization problems[J]. Computers & Electrical Engineering, 2018, 67: 365-382.

[12] Pan W T. A new fruit fly optimization algorithm: taking the financial distress model as an example[J]. Knowledge-Based Systems, 2012, 26: 69-74.

[13] Pan W T. Mixed modified fruit fly optimization algorithm with general regression neural network to build oil and gold prices forecasting model[J]. Kybernetes, 2014, 43(7): 1053-1063.

[14] Wang L, Shi Y L, Liu S. An improved fruit fly optimization algorithm and its application to joint replenishment problems[J]. Expert Systems with Applications, 2015, 42(9): 4310-4323.

[15] Lv S X, Zeng Y R, Wang L. An effective fruit fly optimization algorithm with hybrid information exchange and its applications[J]. International Journal of Machine Learning and Cybernetics, 20178, 9(10): 1623-1648.

[16] Han X M, Liu Q M, Wang H Z, et al. Novel fruit fly optimization algorithm with trend search and co-evolution[J]. Knowledge-Based Systems, 2018, 141: 1-17.

[17] Hu R, Wen S P, Zeng Z G, et al. A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm[J]. Neurocomputing, 2017, 221: 24-31.

[18] Shan D, Cao G H, Dong H J. LGMS-FOA: An improved fruit fly optimization algorithm for solving optimization problems[J]. Mathematical Problems in Engineering, 2013, 2013: 1-9.

[19] Darvish A, Ebrahimzadeh A. Improved fruit-fly optimization algorithm and its applications in antenna arrays synthesis[J]. IEEE Transactions on Antennas and Propagation, 2018, 66(4): 1756-1766.

[20] Wu L, Liu Q, Tian X, et al. A new improved fruit fly optimization algorithm IAFOA and its application to solve engineering optimization problems[J]. Knowledge-Based Systems, 2018, 144: 153-173.

[21] Babalik A, İscan H, Babaoğluİ, et al. An improvement in fruit fly optimization algorithm by using sign parameters[J]. Soft Computing, 2018, 22(22): 7587-7603.

[22] Zhang Y W, Cui G M, Zhu E Z, et al. AFOA: An adaptive fruit fly optimization algorithm with global optimizing ability[J]. International Journal on Artificial Intelligence Tools, 2016, 25(6): 1650032.

[23] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proceedings of ICNN'95 International Conference on Neural Networks. Perth: IEEE, 1995: 1942-1948.

[24] 李智勇, 黄滔, 陈少淼, 等. 约束优化进化算法综述[J]. 软件学报, 2017, 28(6): 1529-1546.
(Li Z Y, Huang T, Chen S M, et al. Overview of constrained optimization evolutionary algorithms[J]. Journal of Software, 2017, 28(6): 1529-1546.)

[25] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation, 2000, 4(3): 284-294.

[26] Long W, Liang X, Huang Y, et al. An effective hybrid cuckoo search algorithm for constrained global optimization[J]. Neural Computing and Applications, 2014, 25(3/4): 911-926.

作者简介

石建平(1981−), 男, 博士生, 从事机器人技术与智能机电系统的研究, E-mail: sjp6565@126.com;

李培生(1969−), 男, 教授, 博士生导师, 从事机器人技术与智能机电系统、制造系统可靠性等研究, E-mail: lps20150331@163.com;

刘国平(1964−), 男, 教授, 博士生导师, 从事机器人技术与智能机电系统等研究, E-mail: liuguoping.ncu@163.com;

刘鹏(1983−), 男, 副教授, 博士, 从事优化设计的研究, E-mail: llp080@126.com.

(责任编辑：齐 霁)