计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(10):142−155 [doi: 10.15888/j.cnki.csa.008737]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于教与学策略的动态变异花授粉算法①

段艳明, 肖辉辉, 谭黔林, 赵翠芹

(河池学院 大数据与计算机学院, 河池 546300)
通信作者: 肖辉辉, E-mail: gxhcxyxzy@126.com

摘　要: 为了进一步提升花授粉算法的优化性能, 本文提出一种融入改进的教与学优化策略及动态高斯变异的新花授粉算法. 该算法先用教机制中改进的教学因子得到的最优个体与其他个体间的促进作用来提高算法的收敛速度; 同时运用种群个体之间相互学习的学机制来保持种群的多样性, 从而提升算法的优化精度; 然后, 当检测到算法陷入早熟时, 则对种群的中间个体进行动态高斯变异, 增加个体之间的差异性, 避免算法早熟, 进而提升算法的综合优化能力. 通过对 16 个标准函数的优化结果实验和非参数统计检验分析对比, 证明了该算法的有效性; 并与其他改进的花授粉算法进行比较分析, 结果显示本文算法优势较显著. 最后, 运用新算法对伸缩绳应用问题进行求解, 亦获得较好的优化结果.

关键词: 花授粉算法; 寻优性能; 教与学优化算法; 早熟; 种群多样性

引用格式: 段艳明,肖辉辉,谭黔林,赵翠芹.基于教与学策略的动态变异花授粉算法.计算机系统应用,2022,31(10):142–155. http://www.c-s-a.org.cn/1003-3254/8737.html

Dynamically Mutant Flower Pollination Algorithm Based on Teaching and Learning Strategies

DUAN Yan-Ming, XIAO Hui-Hui, TAN Qian-Lin, ZHAO Cui-Qin

(School of Big Data and Computer Science, Hechi University, Hechi 546300, China)

Abstract: This study proposes a new flower pollination algorithm by incorporating the improved teaching-learning-based optimization strategy and dynamic Gaussian mutation to enhance the optimization performance. The algorithm first speeds the convergence through the promotion effect between the optimal individual and other individuals obtained by the improved teaching factor in the teaching mechanism. At the same time, the mutual learning mechanism between individuals is adopted to maintain the diversity of the population, thereby improving the optimization accuracy. Then, when it is detected that the algorithm falls into prematurity, the dynamic Gaussian mutation is carried out on the middle individuals of the population to increase the differences between individuals. In this way, it avoids the prematurity of the algorithm and then improves the comprehensive optimization ability. The optimization results of 16 standard functions are checked by the nonparametric statistical test to prove the effectiveness of the algorithm. Compared with other improved pollination algorithms, this algorithm has significant advantages. Finally, the new algorithm is applied to solve the application problems of telescopic rope, and good optimization results are achieved.

Key words: flower pollination algorithm; optimization performance; teaching-learning-based optimization algorithm; premature convergence; population diversity

1　引言

学者 Yang 受显花植物繁殖机理的启发, 提出了一种名为花授粉算法 (flower pollination algorithm, FPA)[1] 的智能优化算法. 该算法仿生原理简单、参数较少、

① 基金项目: 国家自然科学基金 (61972184, 61562032); 河池学院高层次人才科研启动项目 (2019GCC012)
收稿时间: 2022-01-04; 修改时间: 2022-01-30, 2022-02-24; 采用时间: 2022-03-01; csa 在线出版时间: 2022-07-07

易调节, 运用转换概率参数 p 好地解决了探测和开采之间的平衡问题. 当前, FPA 算法主要在函数 (包括无约束函数和约束函数) 优化、能源、通信等应用领域得到了广泛运用, 其具体应用如下:

(1) 函数优化领域

函数优化是属于连续优化问题, 可分为无约束函数优化和带约束的函数优化. 郭庆等[2] 分析了 FPA 算法在求解多模态优化问题时存在的不足, 分别利用模拟退火和单纯形法思想对 FPA 算法的全局搜索和局部搜索进行了改进了, 设计出一种新的 FPA 算法框架; 最后, 利用改进算法对多种类型的测试函数进行求解, 并与对比算法进行了对比分析, 实验结果说明改进算法是一种具有良好竞争力的新算法.

(2) 能源领域

能源是一个国家的经济命脉, 如何提高能源的利用率是一个热点的研究话题. 为此, 众多学者运用元启发式算法来解决该问题, 并取得不错的效果. 如用 FPA 算法对能量流管理[3]、能源交易市场单价预测[4]、电力调度[5] 等问题进行优化求解, 并获得较好的结果.

(3) 通信领域

Rajeswari 等[6] 把鲍威尔法引入到花授粉算法中, 并用改进的花授粉算法解决传感网最小能量传播问题, 实验结果表明新算法与对比算法相比, 更适合解决此类优化问题.

虽然该算法在众多领域中得到了广泛应用, 但该算法与其他智能算法一样, 也存在一些不足, 如算法演化后期计算速度慢、容易早熟, 寻优精度不高, 鲁棒性差等缺陷, 导致其在解决大量复杂优化问题时获得的结果不尽人意. 为了克服这些算法存在上述的不足, 提高其收敛能力, 当前众多学者依据各种智能算法的特征和优缺点, 运用了诸多的策略和方法对其进行了改进, 构建了许多优化能力更强的新算法模型, 使得大量的工程优化问题得到更好的解决.

Zhou 等[7] 在 FPA 算法的勘探部分融入精英反向学习策略的组合探测搜索机制, 以增加个体之间的差异性, 扩展算法的探索领域; 在算法的开发部分加入贪婪搜索机制构造新的局部搜索策略, 以提高算法的开采能力; 同时对参数 p 运用了动态调整策略. 实验结果表明改进算法的寻优能力得到了一定提升. 一方面, 该算法利用精英所具有的丰富信息, 可有效提升算法的性能; 另一方面, 在该算法演化过程中, 由于精英起着重要作用, 将引导种群快速向最优解靠近, 因此, 若精英一旦陷入局部极值, 则导致整个种群停滞进化, 易造成算法陷入“早熟”问题.

Abdel-Raouf 等[8] 把 particle swarm optimization (PSO) 算法融合到 FPA 算法中, 即把 PSO 算法优化后的种群作为 FPA 算法的初始解, 同时通过混沌映射对算法的参数进行调节, 这使算法解的质量得到了改善, 但该算法由于缺乏增加种群多样性机制, 使其在演化中种群的多样性不能得到较好的保持, 易陷入局部最优. Lenin 等[9] 先对和声算法增加混沌策略进行改进, 再把其优化后的种群作为花授粉算法的初始解, 提高算法初始解的质量, 提升了算法的寻优性能, 但其同样存在文献 [8] 的不足.

Tawhid 等[10] 利用鲸鱼优化算法的搜索猎物策略和攻击猎物方法替换了 FPA 算法的全局搜索策略, 构建了一种新的混合算法 whale optimization algorithm and flower pollination algorithm (WOFPA), 其性能与对比算法相比, 得到了一定程度的提升, 但该算法增加了较多的参数, 降低了算法的灵活性, 同时新算法只在低维上验证了其性能的优越性, 没有在高维得到验证, 使其缺少了说服力.

Draa[11] 对 FPA 算法进行了定性、定量分析及其改进. 首先, 从改进的 FPA 算法文献中所利用的测试函数、实验验证方式及传统 FPA 算法参数值的设置 3 个方面进行了详细的定性分析; 其次, 依据定性分析的结果, 作者对参数 p 的值设置进行了较全面的实验对比分析, 分析结果表明当 p 的值取 0.2 时, 该算法的寻优能力获得最佳; 最后, 根据上述的定性分析和定量分析实验, 作者把反向学习方法引入到该算法中以达到提高其寻优能力, 优化结果说明该改进策略是行之有效的. 但是, 反向学习机制对算法的改善还是很有限的, 其原因在于学习机制过于简单, 不能很好地提取和利用较劣粒子所隐含的有益信息[12]; 同时在进化后期, 大部分反向点及原点都集中到小范围区域, 使反向点的适应度值并不比原值较优, 所以在演化后期试图通过反向的方式很难跳出当前的极值区域[13], 其依然不能解决算法陷入局部最优、收敛精度低的问题. 另外, 在该算法中是以一定概率采用反向学习策略, 扩大了算法的随机性, 从而减缓了其收敛速度.

综上所述, 目前的一些改进方法对 FPA 算法的优化能力起到了一定的提升作用, 但这些方法没有对种群个体之间的信息进行有效利用, 造成种群的多样性

在演化过程中不能较好地保持, 使算法易“早熟”, 影响了其全局优化能力; 同时, 当前改进的 FPA 算法在对多模态等复杂问题进行求解时, 其求解精度、稳定性及求解速度等方面还有较大的提升空间, 且有些改进措施增加了算法的时间复杂度, 降低了算法的灵活性, 另外一些改进策略对基本的 FPA 算法原有的仿生原理做了相应的修改.

为了解决在提高算法的收敛速度、解的质量的同时防止算法“早熟”的问题, 本文构建了一种把改进的教与学优化算法[14] 中的教与学思想及动态高斯变异融入到花授粉算法中的改进算法 (teaching learning Gauss flower pollination algorithm, TLGFPA) .

TLGFPA 算法的基本思想: 首先, 根据总迭代次数与当前迭代次数的关系对教与学优化算法中的教学因子进行改进, 并把改进的教机制作用于花授粉算法中的全局寻优部分, 充分利用最优个体 (教师) 与其他个体 (学生或学员) 间的促进作用来提高算法的收敛速度; 其次, 把教与学优化算法中的学机制作用于花授粉算法中的局部寻优部分, 通过个体 (学生或学员) 之间的相互学习来保持种群的多样性, 提高其求解精度; 最后, 在教与学优化算法的教阶段的进化过程中, 由于超级个体 (教师) 不断吸引其他普通个体 (学生), 使得种群个体之间的差异性不能到较好的保持, 从而易造成算法“早熟”, 因此, 在进化过程中需要判断算法是否陷入“早熟”进行处理, 若“早熟”, 则利用动态高斯变异策略对种群的中间个体进行扰动, 增加种群个体之间的差异性, 防止算法过早收敛.

首先, 通过对 16 个经典的优化问题进行求解, 求解结果显示新算法与 FPA 算法相比, 其优化能力得到了较显著提升; 其次, 选择了 3 种 FPA 改进算法进行了对比实验, 对比分析结果显示新算法的性能总体上要好于对比算法, 显示出较好的竞争力. 最后, 运用 TLGFPA 算法对伸缩绳应用问题进行求解, 同样也得到了较好的优化结果, 进一步验证了 TLGFPA 算法的有效性和可行性.

本文的创新工作主要包括:

(1) 为了解决教学机制容易导致教与学算法“早熟”问题, 本文对该算法的教学因子进行改进, 使其随着迭代的进行而动态调整, 能够较好地解决其的勘探和开采能力平衡问题.

(2) 把改进的教机制作用于花授粉算法中的全局寻优部分, 充分运用最优个体 (教师) 与其他个体 (学生或学员) 间的促进作用来提升算法的搜索速度; 同时把教与学优化算法中的学机制作用于花授粉算法中的局部寻优部分, 通过种群个体之间的相互交流学习来维持种群的多样性, 改进其求解精度.

(3) 在算法中融入了动态高斯变异策略来提高种群个体之间的差异性, 防止算法“早熟”而导致整个种群进化陷入停滞状态.

本文第 2 节对 FPA 算法进行了简单描述; 第 3 节对教与学优化算法、改进的教与学优化算法、动态高斯变异机制、TLGFPA 算法以及该算法的时间复杂度进行了阐述; 第 4 节进行实验评测; 第 5 节对本文的研究工作进行归纳, 并对今后的研究方向进行展望.

2 FPA 算法

该算法的仿生原理是对花朵植物繁殖过程 (异花和自花授粉) 的模拟, 花朵植物的异花授粉过程是通过昆虫等方式来实现授粉繁殖, 具有莱维飞行的特征, FPA 算法的全局搜索策略模拟了该过程, 这使其具有良好的全局优化能力, 花朵植物的自花授粉是通过同株上的花朵自行传粉繁殖, 传播距离非常小, 这个过程对应于 FPA 算法的局部搜索过程. 同时, 该算法运用参数 p 来解决其的勘探和开采能力平衡问题, 这使其具有较好的全局寻优能力. 为了利用该算法更好地解决一系列实践中复杂的优化问题, 还必须对其设定 4 条约定, 其详情见文献 [1]. 依据上述思路, 学者 Yang[1] 利用以下公式对其实现了数学模型化.

FPA 算法的探索机制 (全局搜索策略) 是采用式 (1) 进行模型化:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t) \tag{1}$$

其中, x_i^t, x_i^{t+1}表示算法进化中第 t、t+1 代对应的解, g_*表示种群中的最优个体, γ 是对步长进行控制的伸缩因子, $L(\lambda)$ 表示种群个体的 Lévy 步长, 其值运用式 (2) 计算:

$$L(\lambda) \sim \frac{\lambda G(\lambda)\sin(\pi\,\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}(s >> s_0 > 0) \tag{2}$$

其中, λ 取值 1.5, $G(\lambda)$ 表示 Gamma 函数, s 的值通过式 (3) 计算得到:

$$s = \frac{\mu}{|v|/t} \tag{3}$$

其中, $\boldsymbol{\mu}$~$N\left(0,\sigma^2\right)$, V~$N(0,1)$, σ^2 的值采用式 (4) 计算:

$$\sigma^2 = \left\{\frac{G(1+\lambda)}{\lambda G[(1+\lambda)/2]}\cdot\frac{\sin(\pi\,\lambda/2)}{2^{(\lambda-1)/2}}\right\}^{1/\lambda} \tag{4}$$

FPA 算法的种群个体采用式 (5) 实现开发搜索 (局部搜索):

$$x_{\mathrm{i}}^{t+1}=x_{\mathrm{I}}^{t}+\varepsilon\left(x_{j}^{t}-x_{k}^{t}\right) \tag{5}$$

其中, $\varepsilon\in[0,1]$ 上产生的随机数, x_j^t, x_k^t 是随机选择的不同于 x_i^t 的两个不同的种群个体.

3 教与学优化策略的动态变异花授粉算法

3.1 教与学优化算法

受教与学的启发, Rao 等[14] 设计了一种名为教与学优化算法 (TLBO), 该算法模拟了教师的教学过程及学生之间的相互学习和交流的过程. 该算法的种群个体分为教师和学生两类, 教师是超级个体, 学生是普通个体, 其所学科目数对应问题的维数. TLBO 算法把进化过程分为“教学”与“学习”两个时段, 在“教学时段”是普通个体 (学生) 向最优个体 (教师) 不断学习的过程, 从而达到提升学生的成绩, 同时结合了种群均值的作用, 能有效地融合最优个体与普通个体的相互促进作用; 在“学习时段”, 随机选取两个种群个体, 并比较其优劣, 劣者向优者学习.

教与学优化算法的具体实施过程如下.

Step 1. 对算法种群 (班级) 个体的位置利用式 (6) 进行随机初始化:

$$x_i=x_i^{\mathrm{L}}+rand(0,1)\times(x_i^{\mathrm{U}}-x_i^{\mathrm{L}}),i=1,2,\cdots,D \tag{6}$$

其中, x_i^{L} 和 x_i^{U} 分别为解空间自变量 x_i 的上界和下界, D 为解空间的维数, $rand(0,1)\in(0,1)$ 之间的任意数.

Step 2. 教学时段. 每个普通个体 (学生) 通过最优个体 (教师) 和学生平均值之间的差异来向教师 (最优个体) 学习, 首先求解当前种群每一维的平均值 x_{mean}, 再利用式 (7) 计算教师与学生的差异, 最后利用式 (8) 实现位置更新, 即完成教的过程:

$$dif=s\times(x_{\mathrm{teacher}}-tf\times x_{\mathrm{mean}}) \tag{7}$$

$$x_{\mathrm{new}}^{i}=x_{\mathrm{old}}^{i}+dif \tag{8}$$

其中, x_{teacher} 是种群中的超级个体, $s=rand(0,1)$ 为学习步长, $tf=round(1+rand(0,1))$ 为教学因子.

Step 3. 对普通个体 (学生) 位置进行更新. 每个学生依据学习的效果决定对当前的位置是否更新, 若优对其当前位置进行更新, 否则保留其原有位置.

Step 4. 学习时段. 对种群中每个普通个体 (学生) x^i, 随机选取其学习对象 $x^j(i\neq j)$ 进行学习, 通过计算两者间的目标函数适应度值来进行调整位置, 即利用式 (9) 来进行学生间的学习:

$$x_{\mathrm{new}}^{i}=\begin{cases}x_{\mathrm{old}}^{i}+s\times(x^{i}-x^{j}),\ f(x^{j})<f(x^{i})\\ x_{\mathrm{old}}^{i}+s\times(x^{j}-x^{i}),\ f(x^{i})<f(x^{j})\end{cases} \tag{9}$$

其中, $f(x^i)$ 和 $f(x^j)$ 分别表示个体 x^i 和 x^j 的目标函数适应度值, $s=rand(0,1)$ 为学习步长.

Step 5. 对普通个体 (学生) 位置进行更新. 每个学生依据学习的效果决定对当前的位置是否替换, 若优, 替换, 反之保留原有位置.

Step 6. 对算法的演化条件进行判断, 如条件成立, 则优化结束, 并输出算法的最优值和其对应的最优解, 否则进化继续并转 Step 2.

3.2 动态高斯变异

高斯分布是一类重要的概率分布函数, 在众多领域得到广泛应用, 其计算公式如式 (10):

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{10}$$

其中, σ、μ 为分别表示为方差为 1 和均值为 0.

动态高斯变异的思想是给原有个体增加一个扰动项, 其具体形式如下:

$$x_i=x_i+mN(0,1)x_i \tag{11}$$

其中, $m=(Max\_iter-t)/Max\_iter$, 即 $m\in(1,0]$ 为递减变量; x_i 表示为算法在进化代数为 t 时个体 i 所对应的状态; $N(0,1)$ 是服从高斯分布的一个任意向量; Max_iter 表示算法的最大进化代数, t 表示为算法当前进化的代数.

动态高斯变异机制能在一定程度上增加了种群个体之间的差异性, 能使个体有效跳出局部极值进行全局搜索, 同时提高收敛速度.

3.3 改进的教与学优化机制

3.3.1 TLBO 算法存在的问题

根据第 3.1 节可知, TLBO 算法主要由“教学时段”和“学习时段”两个核心部分构成, 但由于该算法的搜索机制存在以下问题, 使其优化能力受到一定的影响:

问题 1. 在 TLBO 算法的教学时段, 教学因子 tf 左右着学生个体与教师个体的差异, tf 的大小决定着算法在该阶段的搜索能力, tf 越小, 则搜索能力越强. 但从基本 TLBO 算法的教学因子的计算公式可知, 其值随机地在 [1, 2] 中取值, 没有考虑到随算法的进化而相应地调整该值, 从而使教学阶段易陷入两种极端的状况, 学

生要么学得很好, 要么学得很差, 但在实际的教学过程中, 学生个体是以任何概率从教师处学到知识. 而且, 在教学前期, 学生个体与教师个体的差距是较大的, 到教学后期, 学生个体与教师个体的差距就会不断缩小.

问题 2: 从 TLBO 算法的教学阶段可知, 通过普通个体 (学生) 之间小范围的相互交流学习来提升各自的成绩, 这使种群个体之间的差异性能够得到较好的保持, 从而保证了算法的持续全局寻优能力. 但是, 在 TLBO 算法的教学阶段, 每个学生 (个体) 都向教师 (最优个体) 学习, 虽然能快速提高搜索速度, 但更易导致种群陷入局部最优.

3.3.2 对 TLBO 算法进行改进

(1) 对 TLBO 算法的教学因子 tf 进行改进

针对第 1 个问题, 本文对教学因子 tf 做了改进, 使其随着迭代的进行而线性递减, 把教学因子 tf 的计算公式修改为:

$$tf = (tf_{\max} - tf_{\min}) \times ((N\_iter - t)/N\_iter) + tf_{\min} \quad (12)$$

其中, $tf_{\min}$ 是 tf 取的最小值, $tf_{\max}$ 是 tf 取的最大值, N_iter、t 分别表示为算法的最大进化代数和当前进化的代数.

从式 (12) 中可以看出, tf 的值在算法进化的前期取值较大, 这有利于学生的学习速度和学习能力的提高, 从而达到提升算法的勘探能力, 提高算法的寻优速度; 在算法进化的后期, 学生的学习速度和学习能力显著下降, 教学因子 tf 的值变小, 有利于提高算法的开采能力, 促进种群个体向理论解靠近, 提高解的精度.

(2) 融入动态高斯变异机制

针对问题 2, 本文把高斯变异机制融入到 TLGFPA 算法中, 从而进一步改善算法的优化能力. 这是因为 TLGFPA 算法的全局授粉部分引入了 TLBO 算法的教机制, 而 TLBO 算法的教阶段在进化过程中, 由于超级个体 (教师) 不断吸引其他普通个体 (学生), 使得种群个体在进化过程容易同化, 从而易造成算法陷入局部最优. 为此, 在进化过程中需要对算法是否陷入“早熟”进行判断, 若“早熟”, 则利用动态高斯变异策略对种群的中间个体进行扰动, 增加种群个体之间的差异性, 防止算法过早收敛.

3.4 TLGFPA 算法

从式 (1) 可知, FPA 算法采用了 Lévy 策略使其具有良好的勘探能力, 但该策略也使其求解精度较低及收敛速度慢; 而其局部搜索部分利用了差分进化 (DE) 思想, 这使其在解决高维复杂的优化问题时, 性能比较差; 同时, 跟其他群智能算法类似, 该算法在演化的后期, 随着种群多样性的减少, 算法容易早熟. 然而该算法运用转化概率参数 p 来平衡该算法的勘探和开采能力. “教与学”优化算法 (TLBO) 是最近提出的一种新群智能算法, 根据其仿生机理, 其具有极强的收敛能力且求解速度快、精度高优点, 但该算法的教阶段使算法易陷入早熟, 且其“探索”与“开采”能力之间的平衡问题没得到较好的处理. 依据两种算法的结构及特性, 把 TLBO 算法引入到花授粉算法中, 使其取长补短, 提高算法的性能. 鉴于此, 本文提出了融合教与学优化策略的动态变异花授粉算法 TLGFPA, 先根据总迭代次数与当前迭代次数的关系对教与学优化算法中的教学因子进行改进, 接着把改进的教机制作用于花授粉算法中的全局寻优部分, 充分利用最优个体与其他个体间的促进作用来提高算法的搜索速度; 同时把教与学优化算法中的学机制作用于花授粉算法中的局部寻优部分, 通过种群个体之间的相互交流学习来维系种群的多样性, 提高其计算精度; 最后, 如果适应度值连续迭代 3 次没有改进或改进很小, 则对进入下一代的中间种群个体进行动态高斯变异, 增强种群的多样性, 避免过早收敛. 通过融合多种策略, 提高算法的寻优性能.

TLGFPA 算法的伪代码如算法 1 所示, 其中, n 为种群规模, p 是算法的全局搜索和局部搜索之间的转换概率, N_iter 是算法的最大进化代数, $\varepsilon \in [0, 1]$ 的任意数, $f_{\min}$ 为种群个体中的最优值, g_*为最优解, fitness 为函数适应度值, x_i 为种群的第 i 个体.

算法 1. TLGFPA 算法伪代码

```
1. 初始化所有参数
2. for i=1: n
3.    随机产生 n 个个体 {x_i | i=1, 2, ⋯, n};
4.    计算每个个体或每个解的适应度值;
5. end
6. 计算当前种群的最优值
7. 找出当前种群最优值对应的最优解;
8. % TLGFPA 算法开始迭代
9.    for t=1: N_iter (总的迭代次数) do
10.      for i=1: n do
11.      if p>rand(0, 1) then
12.         /*全局寻优*/
13.         根据式 (10) 计算 tf 的值;
14.         计算种群个体的平均值 Sol_mean;
15.         根据式 (7) 计算 dif 值;
```

```
16.        根据式 (1) 和式 (8) 产生一个新的候选解 x_i^{t+1};
17.        对 x_i^{t+1} 进行越界判断
18.      else
19.        /* 局部寻优*/
20.        随机产生两个个体 x_j^t 和 x_k^t;
21.        根据式 (9) 产生一个新的候选解 x_i^{t+1};
22.        对 x_i^{t+1} 进行越界判断
23.      end
24.      找出当前种群的最优值和其对应的最优解
25.      if (t>=3 && abs((fun(x_i^{t+1})–fin(x_i^t)))<δ) || (t>=3 && abs((fun(x_i^{t+1})–
fun(x_i^t)))=0)
26.        高斯变异
27.        对 x_i^{t+1} 进行越界判断
28.        找出当前种群的最优值和其对应的最优解
29.      end
30.    end
31. end
```

TLGFPA 算法的实现过程如下.

步骤 1. 对 TLGFPA 算法的种群规模 n, 全局和局部搜索转换参数 p, 最大进化代数 N_iter, 问题规模 D, 教学因子的最大值 $tf_{\max}$, 教学因子的最小值 $tf_{\min}$, 分别进行赋值.

步骤 2. 对算法的种群进行初始化, 求解每个解的值, 并记录所有值中的最小值或最大值和其相应的解.

步骤 3. 若 $p>rand(0, 1)$, 利用式 (12) 计算教学因子 tf 的值.

步骤 4. 计算种群的平均位置 Sol_{mean}.

步骤 5. 运用式 (7) 计算最优花朵与种群的差异 dif.

步骤 6. 利用式 (1)、式 (8) 对花朵个体的位置进行更新, 并对新产生的解是否超出解的范围进行处理.

步骤 7. 如果 $p<rand(0, 1)$, 根据式 (9) 对花朵个体的位置进行更新, 并对新产生的解是否超出解的范围进行处理.

步骤 8. 对步骤 6 或步骤 7 产生的新个体进行求解, 并更新 $f_{\min}$ 和 g_*.

步骤 9. 如果连续 3 次, 适应度值没变或变化很小, 则执行步骤 10, 否则跳到步骤 11.

步骤 10. 对种群中间个体进行动态高斯变异.

步骤 11. 如进化代数大于阈值 N_iter, 则进化结束, 并输出所有解中的最优解和其相应的适应度值, 否则, 转步骤 3 继续进化.

3.5 TLGFPA 算法的时间复杂度分析

改进措施对算法性能的提升是否可行, 主要从两个方面来衡量: (1) 改进策略能较好地改善算法的性能; (2) 改进算法的灵活性要好, 且算法的时间复杂度值要小. 算法的时间复杂度与其进化代数成正比例. 倘若 $f(x)$ 为求解的问题, D 为求解问题的规模, 则依据算法时间复杂度的求解原则和 FPA 算法的优化机理, FPA 算法的时间复杂度表示为: $T(\text{FPA})=O(D+f(D))$. 从 TLGFPA 算法的具体实现过程可知, 步骤 3–步骤 7 和步骤 9 是新算法添加的步骤, 即包括计算教学因子 tf、计算种群的平均位置 Sol_{mean}、计算最优花朵与种群的差异 dif、更新花朵个体的位置和动态高斯变异. 依据上述公式的阐述和时间复杂度的求解规则, 则可推导出 TLGFPA 的时间复杂度表示为: $T(\text{TLGFPA})=T(\text{FPA})+T(\text{计算教学因子})+T(\text{计算种群的平均位置})+T(\text{计算最优花朵与种群的差异})+T(\text{更新花朵个体的位置})+T(\text{高斯变异})=O(D+f(D))+T(1)+T(1)+O(D+f(D))+O(D+f(D))+O(D+f(D))=O(D+f(D))$. 从上述对改进算法的时间复杂度理论分析结果可知, 改进算法与 FPA 算法相比, 其时间复杂度基本相同.

4 实验仿真及分析

4.1 测试函数仿真及分析

为了验证本文算法的可行性和有效性, 本节选用 4 类共 16 个非常具有代表性的测试函数进行实验, 其具体详情见表 1. 本节实验除了与 FPA 算法比较外, 还与人工蜂群 (artificial bee colony, ABC) 算法[15]、动态领域学习粒子群 (dynamic neighborhood learning particle swarm optimizer, DNLPSO) 算法[16]、TLBO 算法、基于教与学的花授粉算法 (teaching learning flower pollination algorithm, TLFPA) 进行比较. 实验中所有参数的具体设定为: 种群规模 n=30, 最大进化代数 $MaxN\_iter$=5000, FPA、TLFPA 和 TLGFPA 三种算法的 p 的值都取 0.2; DNLPSO 算法的其他参数值来自文献 [16]; ABC 算法中, $limit$=100.

4.1.1 固定迭代次数的性能比较

为了验证本文算法的有效性, 佐证其收敛精度 (解的质量) 的优越性, 且能较好地防止陷入局部最优, 所有算法在每个求解问题上都独自执行 30 次, 求解出其对应的 Mean error (寻优均值偏差)、St.dev (标准方差) 值, 并为了判定 TLGFPA 算法与其余 5 种对比算法寻优性能是否存在显著性差异, 利用非参数统计检验 (α=0.05 的 Wilcoxon 检验) 分别对 16 个函数的实验

结果进行统计分析, 其分析结果见表 2, 其中符号“‡”, “≈”, “†”分别说明 TLGFPA 算法的寻优精度逊于、相当于和高于其余算法, “*w*/*t*/*l*”符号分别说明 TLGFPA 算法与对比算法对比, 在求解问题上, 有 *w* 个的优化结果好于比较算法, 有 *t* 个的寻优结果与比较算法相当, 有 *l* 个的优化效果劣于比较算法.

表 1 本文使用的实验测试函数

funID	函数名	搜索空间	理论最优值	维数	迭代次数
F01	Sphere	[−100, 100]	0	30, 100	500, 1 000
F02	Schwefel 2.22	[−10, 10]	0	30, 100	500, 1 000
F03	Schwefel 1.2	[−100, 100]	0	30, 100	1 000, 2 000
F04	Schwefel 2.21	[−100, 100]	0	30, 100	1 000, 2 000
F05	Rosenbrock	[−30, 30]	0	30, 100	5 000, 5 000
F06	Step	[−100, 100]	0	30, 100	500, 500
F07	Quartic with noise	[−1.28, 1.28]	0	30, 100	3 000, 4 000
F08	Rastrigin	[−5.12, 5.12]	0	30, 100	500, 1 000
F09	Ackley	[−32, 32]	0	30, 100	1 500, 2 000
F10	Griewank	[−600, 600]	0	30, 100	500, 1 000
F11	Generalized Penalized 1	[−50, 50]	0	30, 100	1 500, 2 000
F12	Generalized Penalized 2	[−50, 50]	0	30, 100	5 000, 5 000
F13	Kowalik	[−5, 5]	0.000 307 5	4	500
F14	Goldsteim−Price	[−2, 2]	3	2	150
F15	Rotated Rastrigin’s	[−5.12, 5.12]	0	30, 100	500, 1 000
F16	Rotated Griewank’s	[−600, 600]	0	30, 100	500, 1 000

表 2 在固定进化代数下算法的寻优能力对比 (*D*=30, 4, 2)

funID	Mean error (St.dev)					
	ABC	DNLPSO	TLBO	FPA	TLFPA	TLGFPA
F01	2.03E+01†(2.76E+01)	2.53E+02†(8.38E+02)	6.49E−02†(5.70E−02)	1.22E+03†(3.14E+02)	1.39E−48†(3.22E−48)	**0.00E+00(0.00E+00)**
F02	1.41E+00†(6.73E−01)	7.24E+00†(7.96E+00)	2.70E+00†(9.64E−01)	2.34E+01†(4.04E+00)	4.15E−25†(7.06E−25)	**0.00E+00(0.00E+00)**
F03	2.07E+04†(4.20E+03)	8.69E+02†(1.17E+03)	7.97E−02†(1.05E−01)	5.27E+02†(1.22E+02)	2.05E−102†(6.40E−102)	**0.00E+00(0.00E+00)**
F04	5.58E+01†(7.66E+00)	7.20E+00†(4.80E+00)	3.82E−02†(3.25E−02)	1.03E+01†(1.49E+00)	1.49E−50†(2.05E−50)	**0.00E+00(0.00E+00)**
F05	**6.06E−01‡**(7.31E−01)	1.19E+02†(4.24E+02)	2.89E+01†(2.89E−02)	7.39E+03†(6.28E+03)	2.89E+01†(**2.23E−02**)	2.64E+01+(1.03E+00)
F06	3.07E+01†(2.53E+01)	3.20E+02†(4.87E+02)	**0.00E+00≈(0.00E+00)**	1.23E+03†(3.60E+02)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F07	1.78E−01‡(3.58E−02)	1.64E−01‡(1.34E−01)	2.49E−04‡(1.26E−04)	3.28E−02‡(1.31E−02)	**6.19E−05‡(3.39E−05)**	5.02E−01+(3.05E−01)
F08	3.48E+01†(5.74E+00)	5.85E+01†(1.97E+01)	1.43E+02†(1.73E+01)	2.21E+02†(1.41E+01)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F09	1.11E−02†(6.17E−03)	3.59E+00†(1.97E+00)	1.14E+00†(6.70E−01)	6.08E+00†(9.58E−01)	4.20E−15†(**9.01E−16**)	**2.55E−15**(1.80E−15)
F10	9.40E−01†(5.42E−01)	4.18E+00†(1.04E+01)	1.74E−01†(1.91E−01)	1.20E+01†(3.45E+00)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F11	**6.21E−08‡(1.17E−07)**	3.93E+00†(1.06E+01)	5.15E−01†(1.74E−01)	3.82E+00†(1.51E+00)	1.73E−01†(5.64E−02)	9.01E−03(1.03E−02)
F12	**1.31E−14‡(5.16E−14)**	4.53E+04†(2.48E+05)	2.65E+00†(4.86E−01)	1.83E+01†(8.39E+00)	2.86E+00†(2.51E−01)	7.28E−01(3.19E−01)
F13	7.01E−04†(4.50E−04)	5.55E−04†(2.60E−04)	9.06E−04†(2.56E−04)	1.94E−04†(1.18E−04)	1.36E−04†(1.73E−04)	**1.09E−08(4.73E−09)**
F14	1.38E−01†(3.60E−01)	3.87E−04†(1.88E−03)	3.02E−04†(7.86E−04)	1.30E−02†(1.75E−02)	6.41E−06†(7.42E−06)	**7.99E−14(2.38E−15)**
F15	3.23E+02†(2.47E+01)	2.49E+02†(5.23E+01)	1.69E+02†(1.31E+01)	2.22E+02†(1.67E+01)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F16	9.09E−01†(4.96E−02)	2.40E+00†(3.37E+00)	1.91E−01†(2.23E−01)	1.18E+01†(2.64E+00)	**0.00E+00≈(0.00E+00)**	7.40E−18(2.82E−17)
w/*t*/*l*	12/0/4	15/0/1	14/1/1	15/0/1	10/5/1	—

对表 2 分析获得, TLGFPA 算法与 ABC, DNLPSO, TLBO, FPA 及 TLFPA 算法相比, 有 8 个函数能找到理论最优值, 3 个函数找的值接近理论值. 与 TLFPA 算法相比, 只有在函数 F07 上, 寻优均值偏差要劣于对比算法, 说明加入“动态高斯变异”策略, 能使算法避免陷入局部最优, 提高了算法的收敛精度, 同时也提升了算法的鲁棒性. 与 ABC, DNLPSO, TLBO 及 FPA 算法比较, 在 16 个函数上, 分别有 12、15、14、15 个函数的寻优均值偏差要好于对比算法, 这说明 TLGFPA 算法的优化精度明显优于对比的算法, 验证了本文提出的新

算法是可行有效的, 达到了改进的目的.

为了佐证新算法在高维函数上的优化能力同样表现出色, 表 3 列出了 6 种算法在 14 个函数上 D=100 的优化结果, 并对实验结果从数学角度进行统计分析. 从表 3 可以获得, TLGFPA 算法有 7 个函数能找到理论最优值, 2 个函数找的值无限接近理论值, 与 TLFPA 算法相比, 只有函数 F03 和 F07 的寻优均值偏差要逊于该算法, 与其余 4 种算法对比, 本文算法的求解精度优势较显著.

综上所述, 本文提出的新算法在 4 类不同求解问题上的寻优能力与对比算法相比, 其优化能力要优于对比算法, 也说明本文算法能较好地解决“早熟”问题.

表 3 6 种算法在固定迭代次数下的寻优性能比较 (D=100)

funID	Mean error (St.dev)					
	ABC	DNLPSO	TLBO	FPA	TLFPA	TLGFPA
F01	7.04E+03†(3.60E+03)	7.54E+03†(9.34E+03)	6.07E–03†(8.75E–03)	4.80E+03†(7.89E+02)	1.20E–100†(2.67E–100)	**0.00E+00(0.00E+00)**
F02	5.03E+01†(5.04E+00)	1.19E+02†(4.99E+01)	2.49E+00†(1.19E+00)	5.68E+01†(5.55E+00)	3.90E–51†(1.04E–50)	**0.00E+00(0.00E+00)**
F03	2.04E+05†(1.74E+04)	5.68E+04†(2.78E+04)	8.71E–04†(1.70E–03)	5.20E+03†(1.28E+03)	**1.24E–208‡(0.00E+00)**	2.00E–206(**0.00E+00**)
F04	9.08E+01†(1.84E+00)	3.81E+01†(8.45E+00)	1.77E–03†(1.39E–03)	1.52E+01†(1.35E+00)	5.17E–103†(1.30E–102)	**0.00E+00(0.00E+00)**
F05	1.09E+02≈(2.74E+01)	9.14E+04†(2.44E+05)	9.89E+01†(4.11E–02)	3.58E+05†(1.83E+05)	9.89E+01†(**3.96E–02**)	**9.74E+01**(9.58E–01)
F06	4.57E+04†(1.86E+04)	1.53E+04†(6.96E+03)	**0.00E+00≈(0.00E+00)**	7.65E+03†(1.54E+03)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F07	1.44E+00†(1.63E–01)	2.32E+00†(2.39E+00)	2.39E–04‡(1.10E–04)	7.98E–01†(3.19E–01)	**4.29E–05‡(2.85E–05)**	5.34E–01+(2.96E–01)
F08	3.03E+02†(2.66E+01)	4.10E+02†(1.18E+02)	3.16E+02†(2.60E+02)	8.02E+02†(3.01E+01)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F09	4.78E+00†(3.76E–01)	1.12E+01†(2.25E+00)	6.28E–01†(3.60E–01)	8.71E+00†(4.04E–01)	**4.32E–15≈(6.49E–16)**	**4.32E–15(6.49E–16)**
F10	5.98E+01†(3.04E+01)	8.43E+01†(9.21E+01)	1.31E–02†(1.52E–02)	4.64E+01†(6.21E+00)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F11	3.97E–01†(2.61E–01)	4.44E+04†(2.30E+05)	9.85E–01†(9.47E–02)	1.13E+01†(2.67E+00)	4.37E–01†(7.40E–02)	**9.52E–02+(2.94E–02)**
F12	**1.22E–05‡(9.63E–06)**	9.64E+04†(5.00E+05)	1.07E+01†(9.13E–01)	1.48E+04†(2.12E+04)	9.99E+00†(3.62E–03)	7.25E+00(4.48E–01)
F15	1.48E+03†(5.55E+01)	1.05E+03†(1.25E+02)	8.12E+01†(5.21E+01)	8.30E+02†(2.52E+01)	**0.00E+00≈(0.00E+00)**	**0.00E+00(0.00E+00)**
F16	1.03E+00†(1.58E–02)	5.68E+01†(5.74E+01)	1.13E–02†(1.28E–02)	4.91E+01†(1.14E+01)	**0.00E+00≈(0.00E+00)**	7.40E–18(2.82E–17)
w/*t*/*l*	12/1/1	14/0/0	12/1/1	14/0/0	6/6/2	—

4.1.2 固定寻优精度的性能对比

为了比较 TLGFPA 算法的稳定性和寻优速度, 在寻优精度固定下, 检验其收敛速度和鲁棒性的优越性, 其实验结果如表 4 所示. 本节所有算法在每一个求解问题上都设置了一个固定的寻优精度边界值, 若算法收敛到上述设定的阈值 (边界值) 且进化代数已大于一定的代数 (本文设置为 3 000), 则断定该算法对该求解问题寻优失败; 其他参数设置同上, 对比算法在每个求解问题上独立执行 30 次; 计算出 *SR* (执行成功率, 执行成功率=算法找到求解精度边界值所需要的代数/算法总的执行代数) 值和 *mean_iter* (平均进化代数) 值, 对比算法的 *SR* 越大且 *mean_iter* 越小, 其性能越优, 表 4 中符号“NA”表示对比算法寻优受挫, 最好结果用加粗突出显示. 从表 4 中可获得, FPA 算法只有在 2 个求解问题上的寻优成功率达到 100%, 其余都是 0, 对于函数 F05, 所有算法都寻优失败, 这是由于该函数的特殊属性, 使得很多群智能算法很难找到其全局极小值. TLGFPA 算法和 TLFPA 算法在 16 个函数上, 都有 11 个函数的寻优成功率是 100%, 但是 TLGFPA 算法在这 11 个函数上的平均收敛迭代次数都小于 TLFPA 算法, 尤其在函数 F01、F15、F16 上, TLGFPA 算法的平均收敛迭代次数比 TLFPA 算法小 1 个数量级, 这说明加入“动态高斯变异”机制, 能有效提高算法的收敛速度. 与其余 4 种算法相比, TLGFPA 算法的 *mean_iter* 和 *SR* 值明显好于对比算法, 说明 TLGFPA 算法的求解速度和稳定性要优于比较算法.

由表 4 可知, FPA 算法的总的平均寻优成功率只有 12.5%, 通过在 FPA 算法中引入新的策略, TLFPA 和 TLGFPA 算法的寻优成功率得到大幅度提升, 其中 TLGFPA 算法的寻优成功率为 65%, 是 6 种算法中最高的, 说明 TLGFPA 算法的稳定性最强.

4.1.3 Friedman 与 Wilcoxon 检验

为了更客观评价对比算法的综合优化能力, 本节从数学统计的角度 (非参数 Friedman 统计检验) 进行对比实验, 其对比结果见表 5 和表 6 所示, 分别是所有算法在 16 个函数 (维数 D=30, 函数 F13、F14 的维数

分别为 4 和 2) 和 14 个函数 (维数 D=100) 上的 Friedman 的检验结果, 从表 5 和表 6 可以看出, TLGFPA 算法不管在低维还是在高维上, 总体性能最好, 则进一步说明改进的算法是有效的.

表 4　6 种算法的寻优成功率、平均迭代次数比较

funID	Threshold	*SR* (*mean_iter*)					
		ABC	DNLPSO	TLBO	FPA	TLFPA	TLGFPA
F01	1.00E−10	100% (2.01E+03)	53.33% (1.18E+03)	13.33% (2.72E+03)	0 (NA)	100% (1.18E+02)	100% (7.48E+01)
F02	1.00E−10	3.33% (3.00E+03)	0 (NA)	0 (NA)	0 (NA)	100% (1.92E+02)	100% (1.05E+02)
F03	1.00E−10	0 (NA)	0 (NA)	3.33% (2.78E+03)	0 (NA)	100% (1.22E+02)	100% (1.00E+02)
F04	1.00E−10	0 (NA)	0 (NA)	0 (NA)	0 (NA)	100% (1.96E+02)	100% (1.06E+02)
F05	1.00E−04	0 (NA)	0 (NA)	0 (NA)	0 (NA)	0 (NA)	0 (NA)
F06	1.00E−10	100% (9.64E+02)	66.67% (1.01E+03)	100% (2.51E+02)	0 (NA)	100% (3.69E+01)	100% (3.84E+01)
F07	1.00E−03	0 (NA)	0 (NA)	100% (1.04E+03)	0 (NA)	100% (2.00E+02)	96.67% (7.78E+02)
F08	1.00E−10	6.67% (2.87E+03)	0 (NA)	0 (NA)	0 (NA)	100% (1.40E+02)	100% (8.72E+01)
F09	1.00E−10	0 (NA)	23.33% (1.84E+03)	0 (NA)	0 (NA)	100% (1.87E+02)	100% (9.50E+01)
F10	1.00E−10	60% (2.59E+03)	23.33% (1.45E+03)	13.33% (2.63E+03)	0 (NA)	100% (1.22E+02)	100% (8.00E+01)
F11	1.00E−02	100% (5.41E+02)	53.33% (9.80E+02)	0 (NA)	0 (NA)	0 (NA)	3.33% (2.87E+03)
F12	1.00E−01	100% (4.31E+02)	60% (1.04E+03)	0 (NA)	0 (NA)	0 (NA)	0 (NA)
F13	1.00E−04	6.67% (1.79E+03)	50% (1.12E+03)	0 (NA)	100% (6.16E+02)	76.67% (3.38E+02)	100% (3.73E+01)
F14	1.00E−04	96.67% (2.24E+02)	100% (4.32E+02)	93.33% (1.15E+02)	100% (1.90E+02)	100% (6.89E+01)	100% (2.64E+01)
F15	1.00E−10	0 (NA)	0 (NA)	0 (NA)	0 (NA)	100% (1.49E+02)	100% (7.72E+01)
F16	1.00E−10	0 (NA)	0 (NA)	0 (NA)	0 (NA)	100% (1.28E+02)	100% (8.07E+01)
Total average (%)		35.83	21.50	16.17	12.50	63.83	65

表 5　对比算法在 16 个函数上的 Friedman's test (D=30, 4, 2)

Algorithms	Rankings
TLGFPA	**1.82**
TLFPA	2.34
TLBO	3.61
ABC	3.94
DNLPSO	4.03
FPA	5.26

表 6　对比算法在 14 个函数上的 Friedman's test (D=100)

Algorithms	Rankings
TLGFPA	**1.59**
TLFPA	1.97
TLBO	2.98
ABC	4.39
FPA	4.87
DNLPSO	5.20

为了更深入地对算法的优化能力进行综合评价比较, 运用 a=0.05 的 Wilcoxon 检验分析, 其对比分析结果见表 7 和表 8, 从两个表中获得, TLGFPA 算法与 ABC, DNLPSO, TLBO 和 FPA 算法之间的优化性能的差于性比较显著, 与 TLFPA 显著性差异要弱一些, 但也小于 0.05. 这表明改进策略对提升 FPA 算法的寻优性能是可行有效的.

4.1.4　寻优性能对比分析图

为了更直观地表明 TLGFPA 算法的整体寻优性能优于其余对比算法, 本文画出了部分求解问题上的算法求解过程曲线图, 具体见图 1. 从图 1 可知, TLGFPA 算法与其他 5 种对比算法相比, 其优化能力明显好于其他算法, 其经过少量的进化代数就能找到理论最优值或接近理论上的最优值; 在 F06 和 F16 求解问题上, TLGFPA 算法和 TLFPA 算法的优化能力旗鼓相当, 但明显好于其余对比算法, 在 F05 和 F11 函数上, TLGFPA 算法的求解精度要劣于 ABC 算法, 但比其他对比算法好; 对于函数 F07, TLGFPA 算法要劣于 TLBO, FPA 和 TLFPA 算法, 但略好于其他 2 种对比算法. 这说明 TLGFPA 的优化过程得到提速, 寻优精度得到提高.

表 7　对比算法在 16 个函数上的 Wilcoxon's test (D=30, 4, 2)

TLGFPA vs.	p-values
ABC	3.02E−75
DNLPSO	5.79E−93
TLBO	1.89E−56
FPA	4.09E−121
TLFPA	3.17-05

表 8　对比算法在 14 个函数上的 Wilcoxon's test (D=100)

TLGFPA vs.	p-values
ABC	1.07E−97
DNLPSO	6.51E−112
TLBO	4.61E−47
FPA	2.99E−119
TLFPA	3.95E−05

同时, 为了进一步深入地说明 TLGFPA 算法在求解问题过程中的鲁棒性和解的质量优于 FPA 等对比

算法, 本文也画出了对比算法在部分求解问题上执行 30 次的优化结果变化对比图和所有算法在求解问题上的方差对比分析图, 具体分别见图 2 和图 3. 可见在这 4 类函数上, TLGFPA 算法的求解精度、稳定性要好于 DNLPSO, ABC, TLBO 和 FPA 对比算法, 而略好于 TLFPA 算法. 这进一步显示改进算法在求解精度、稳定性上都得到了较好的提升.

综上所述, TLGFPA 算法的优化能力表现比较突出, 验证了本文改进算法的可行性和有效性, 也证明了加入动态高斯变异的有效性.

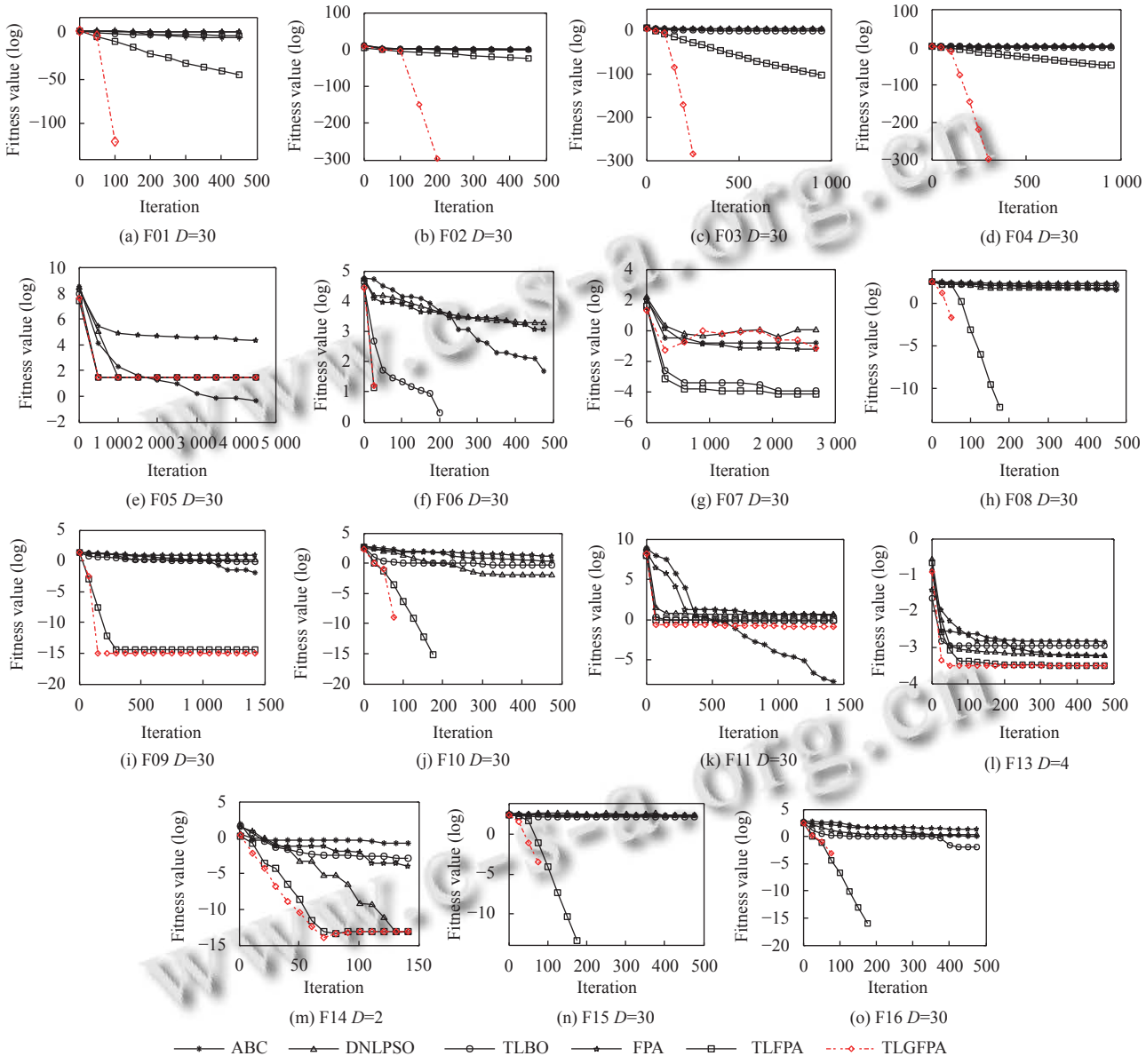

图 1 6 种算法的适应度值收敛曲线

4.2 与现有改进的 FPA 算法比较

为了测试 TLGFPA 算法与现有改进的 FPA 算法在性能上是否具有不错的竞争力, 本节选用 3 种改进的 FPA 算法进行对比, 分别是改进的花授粉算法 (modified flower pollination algorithm, MFPA)[11]、基于广义反向学习的改进花授粉算法 (modified genera-lised oppsition-based flower pollination algorithm, MGOFPA)[11] 及基于精英反向学习的花授粉算法 (elite opposition-based flower pollination algorithm, EOFPA)[7], 为了更好突出本文算法的优势, 在迭代次数较少 (最大迭代次数为 500) 的情况下比较 4 种算法的寻优性能. 其中, 对于 TLGFPA 算法除迭代次数外, 其余实验参数和求解问题

同第 4.1 节; 对于 MFPA、MGOFPA 和 EOFPA 算法的参数 p 取相同的值 0.8、种群规模 n=30 和进化代数不同于相关文献外, 其他参数取自相关文献中, 为了避免实验的偶然性误差, 每种对比算法在其每个求解问题上都单独执行 30 次, 各种对比算法的优化结果如表 9 所示.

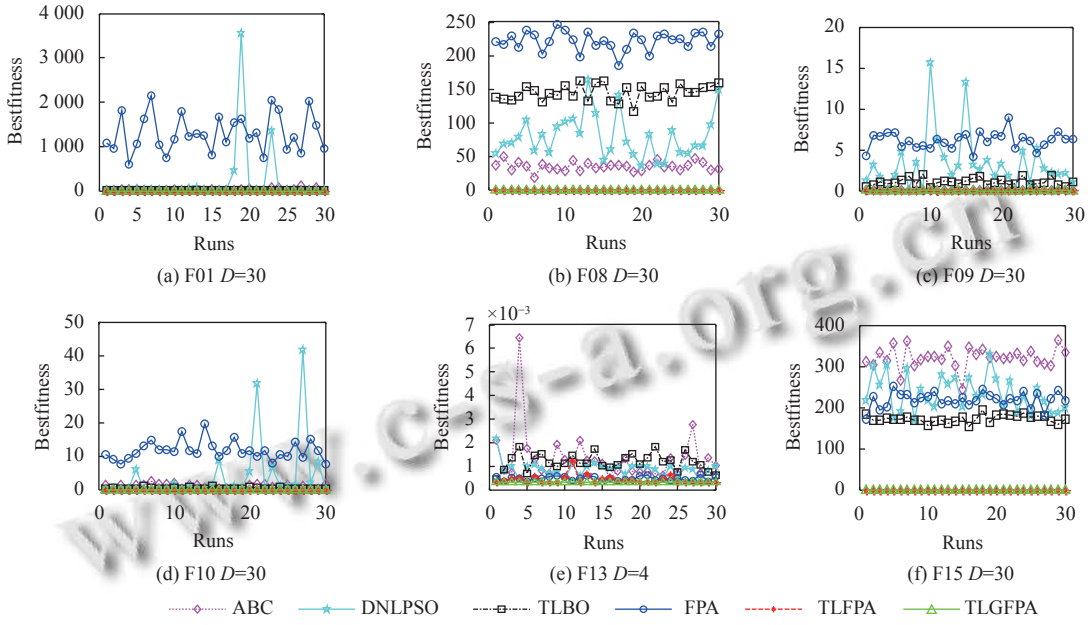

图 2　6 种算法在部分函数上的最优适应度值比较图

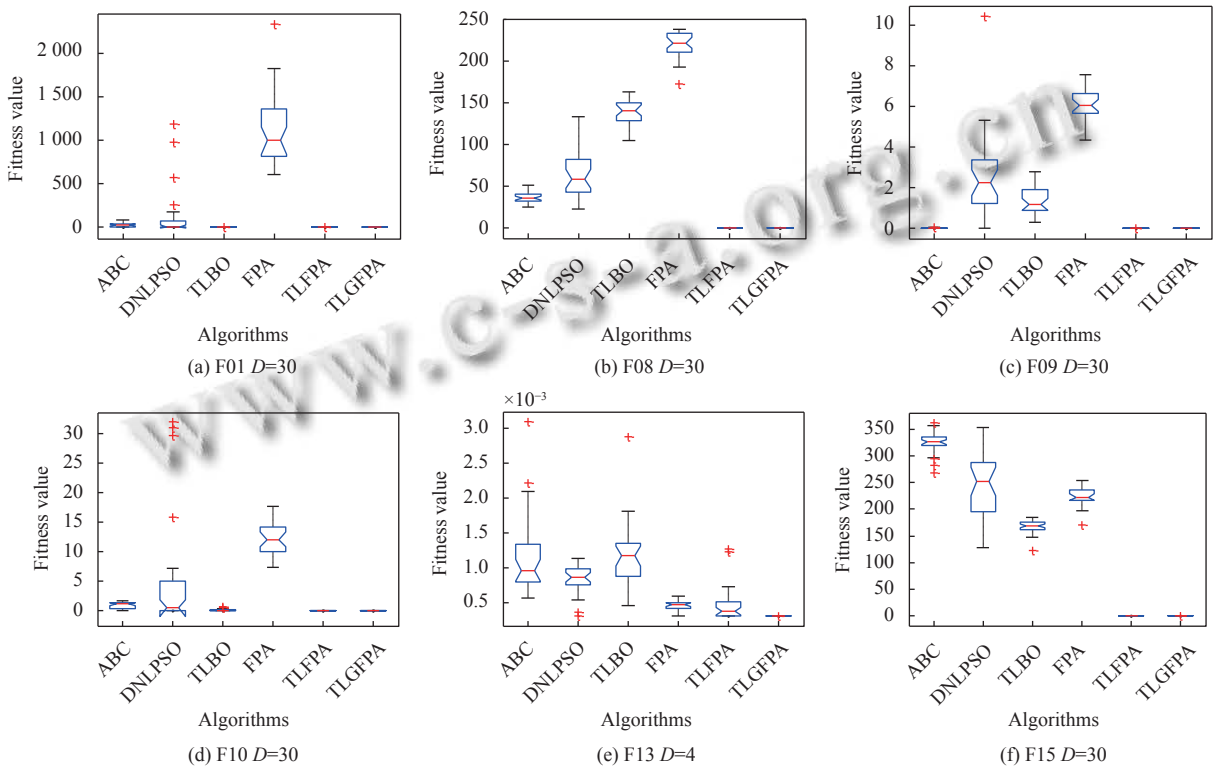

图 3　6 种算法的全局最优值方差分析

从表 9 可知, 改进算法在所有求解问题上的优化能力明显好于 MFPA 算法; 与 MGOFPA算法相比, TLGFPA 算法的寻优能力在 93.75% 的求解问题上要优; EOFPA 和 TLGFPA 算法相比, TLGFPA算法在 7 个求解问题上的优化能力要比 EOFPA 算法优, 尤其是在 F02 求解问题上, TLGFPA算法能找到理论上的最优解, 两种算法在 4 个求解问题 (F06、F08、F14 及 F15) 上的优化能力旗鼓相当, 而 TLGFPA 算法在剩余 5 个求解问题上的寻优能力要逊色于 EOFPA算法.

为了更直观地说明 TLGFPA算法的优化性能好于其余对比算法, 本文画出了对比算法在部分求解问题上的收敛过程曲线图, 具体见图 4. 从图 4 可看出, 本文算法的求解速度在 7 个函数上是 4 种算法中最快的. 借助表 9 中“†”的数学统计结果和图 4 的收敛曲线可以看出, TLGFPA 算法的性能总体上要优于其他改进算法, 显示出较好的竞争力.

表 9 4 种改进算法的寻优性能对比

funID	Mean error (St.dev)			
	MFPA	MGOFPA	EOFPA	TLGFPA
F01	7.29E+03†(1.92E+03)	4.75E–01†(8.08E–01)	4.91E–44†(2.69E–43)	5.09E–96(2.79E–95)
F02	2.29E+01†(4.87E+00)	1.20E–07†(1.22E–07)	1.89E–198†(0.00E+00)	0.00E+00(0.00E+00)
F03	2.82E+03†(9.66E+02)	3.69E–01†(6.44E–01)	4.28E–216† (0.00E+00)	7.16E–226(0.00E+00)
F04	1.71E+01†(2.56E+00)	6.75E–08†(9.08E–08)	1.63E–193†(0.00E+00)	9.28E–294(0.00E+00)
F05	3.60E+06†(2.36E+06)	3.17E+01†(3.04E+00)	2.67E+01‡(4.17E–01)	2.80E+01(4.86E–01)
F06	7.06E+03†(2.03E+03)	2.67E–01†(9.80E–01)	0.00E+00≈(0.00E+00)	0.00E+00(0.00E+00)
F07	1.93E+00†(8.79E–01)	1.09E–02‡(5.17E–03)	2.67E–03‡(1.81E–03)	5.27E–01(2.60E–01)
F08	2.55E+02†(1.88E+01)	7.49E+01†(4.83E+01)	0.00E+00≈(0.00E+00)	0.00E+00(0.00E+00)
F09	1.39E+01†(9.38E–01)	1.89E–01†(2.08E–01)	8.88E–16≈0.00E+00)	7.28E–15(1.45E–15)
F10	1.23E+02†(2.77E+01)	2.95E+00†(2.08E+00)	5.34E–01† (2.72E–01)	6.75E–02(6.31E–02)
F11	5.80E+05†(1.01E+06)	6.68E–01†(1.21E–01)	1.12E–02‡(3.20E–02)	8.08E–02(3.34E–02)
F12	4.54E+06†(3.02E+06)	3.09E+00†(3.01E–01)	5.28E–01‡(3.39E–01)	9.03E–01(2.33E–01)
F13	8.36E–04†(3.55E–04)	4.87E–04†(1.85E–04)	2.65E–04†(4.26E–04)	1.34E–08(2.14E–08)
F14	1.37E–03†(2.20E–03)	2.37E–03†(3.48E–03)	7.99E–14≈(2.90E–15)	7.99E–14(2.38E–15)
F15	2.60E+02†(1.70E+01)	5.41E+01†(4.25E+01)	0.00E+00≈(0.00E+00)	0.00E+00(0.00E+00)
F16	1.23E+02†(3.39E+01)	3.08E+00†(2.61E+00)	6.99E–01† (3.06E–01)	5.87E–02(4.41E–02)
w/t/l	16/0/0	15/0/1	7/4/5	—

4.3 利用改进算法对伸缩绳应用问题进行求解

为了验证新算法在应用优化问题上的寻优性能, 检验其可行性和有效性, 本文选用伸缩绳应用问题进行测试.

4.3.1 伸缩绳问题的描述

在满足 $x_1(d)$、$x_2(D)$ 和 $x_3(P)$ 三个约束条件下使设计的绳最轻, 其中 d、D 及 P 分别表示为“线的直径”“平均卷的直径”及“卷数”. 该问题利用式 (13) 对其进行数学模型化, 其对应的结构如图 5 所示.

$$\min F_{17}(x)=(x_3+2)x_2{x_1}^2$$

$$\text{s.t}\begin{cases} g_1(x)=1-\dfrac{{x_2}^3x_3}{71785{x_1}^4}\leqslant 0\\ g_2(x)=\dfrac{4{x_2}^2-x_2x_1}{12566({x_2}{x_1}^3-{x_1}^4)}+\dfrac{1}{5108{x_1}^2}-1\leqslant 0\\ g_3(x)=1-\dfrac{140.45x_1}{{x_2}^2x_3}\leqslant 0\\ g_4(x)=\dfrac{x_1+x_2}{1.5}-1\leqslant 0\\ 0.05\leqslant x_1\leqslant 2,0.25\leqslant x_2\leqslant 1.3,2\leqslant x_3\leqslant 5 \end{cases} \tag{13}$$

4.3.2 实验结果及对比分析

为了进一步佐证本文算法是否行之有效, 本节利用 TLGFPA 算法对上述伸缩绳设计问题进行求解, 并与文献协同进化差分进化算法 (co-evolutionary differential evolution, CDE)[17], 一种基于可行性规则的混合粒子群约束优化 (hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, HPSO)[18], 加速自适应权重模型 (accelerating adaptive trade-off model, AATM)[19], 社会和文化算法 (society and civilization algorithm, SCA)[20], 精英教学优化算法 (elitist teaching-learning-based optimization, ETLBO)[21], 反馈精英教学优化算法 (feedback elitist teaching-learning-based optimization, FETLBO)[22], 基于精英反向学习的花授粉算法 (elite opposition-based flower pollination algorithm, EOFPA)[7]求解结果进行比较. 本节的实验参数分别设置为: 种群规模都为 25, 进化代数为

2 000, 其余参数设置同第 4.1 节, 实验结果如表 10 所示. 从表 10 可以看出 TLGFPA 算法对伸缩绳问题的优化效果要优于其他 5 种算法, 这表明本文的改进策略是可行和有效的.

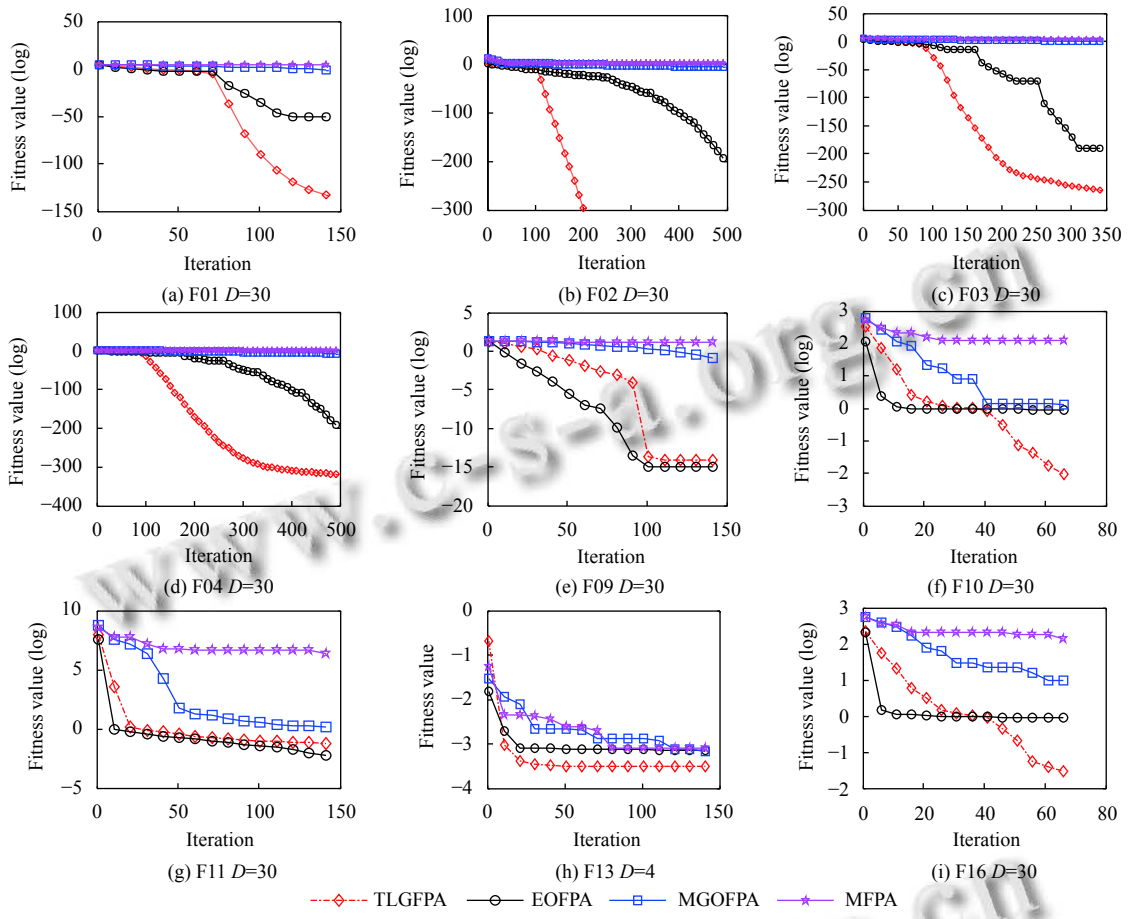

图 4 4 种改进的 FPA 算法收敛曲线图

表 10 8 种算法对伸缩绳优化问题的求解结果

Algorithm	$x_1(d)$	$x_2(D)$	$x_3(P)$	$f(x)$
CDE	0.051 609	0.354 714	11.410 831	0.126 702
HPSO	0.051 706	0.357 126	11.265 083	0.012 665 2
AATM	0.051 813	0.359 690	11.119 252	0.012 668 2
SCA	0.052 160	0.368 158	10.648 442	0.012 669 2
ETLBO	0.051 565	0.353 713	11.468 954	0.012 667 38
FETLBO	0.051 691	0.356 758	11.286 578	0.012 665 23
EOFPA	0.051 689	0.356 726	11.288 454	0.012 665 23
TLGFPA	0.051 038	0.341 244	12.257 819	0.012 665 23

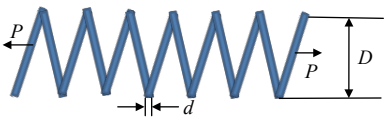

图 5 伸缩绳的结构图

5 结论与展望

为了进一步解决花授粉算法的优化能力还不太理想的问题, 本文提出了先对教与学优化算法中的教学因子进行改进, 再把改进的教与学优化机制和动态高斯变异融入到花授粉算法中的混合算法 (TLGFPA). 首先, 通过对 16 个无约束函数的仿真实验显示新算法能够较好地解决“早熟”问题, 有效地提升了 FPA 算法的求解精度和求解速度等性能. 其次, 选择了 3 种 FPA 改进算法进行了对比实验, 对比分析结果显示 TLGFPA 算法的性能总体上要优于其他改进算法, 显示出较好的竞争力. 最后, 用 TLGFPA 算法求解伸缩绳应用优化问题, 实验结果进一步验证了本文算法在解决应用问题时, 也比 FPA 算法和其他对比算法具有更好的优化能力.

虽然本文算法性能还不错, 但在解决大规模优化问题时, 其优化能力还有待提高, 后续将针对此问题进一步研究. 在今后的研究工作中, 可以尝试利用改进差分进化算法的思路来对其局部搜索部分进行改进, 增强其开发能力. 因为其局部搜索部分是采用了传统差

分进化算法的思想, 在优化复杂多维的优化问题时, 性能较差. 同时, 将其应用领域进一步地进行推广, 如解决动态组合优化问题. 其次, 目前“二进制的花授粉算法”研究很少, 如何将二进制的花授粉算法运用到软件工程、大数据、量子计算等领域将是值得研究的问题.

参考文献

1 Yang XS. Flower pollination algorithm for global optimization. 11th International Conference on Unconventional Computation and Natural Computation. Orléan: Springer, 2012. 240–249. [doi: 10.1007/978-3-642-32894-7_27]

2 郭庆, 惠晓滨, 张贾奎, 等. 多模函数优化的改进花朵授粉算法. 北京航空航天大学学报, 2018, 44(4): 828–840. [doi: 10.13700/j.bh.1001-5965.2017.0240]

3 De M, Das G, Mandal KK. An effective energy flow management in grid-connected solar—Wind-microgrid system incorporating economic and environmental generation scheduling using a meta-dynamic approach-based multiobjective flower pollination algorithm. Energy Reports, 2021, 7: 2711–2726. [doi: 10.1016/j.egyr.2021.04.006]

4 Sahoo S, Swain S, Dash R, *et al*. Novel Gaussian flower pollination algorithm with IoT for unit price prediction in peer-to-peer energy trading market. Energy Reports, 2021, 7: 8265–8276. [doi: 10.1016/j.egyr.2021.08.170]

5 沈艳军, 杨鑫, 刘允刚. 考虑需求响应的水火电优化调度改进型花朵授粉算法. 控制与决策, 2019, 34(8): 1645–1653. [doi: 10.13195/j.kzyjc.2018.0356]

6 Rajeswari M, Thirugnanasambandam K, Raghav RS, *et al*. Flower pollination algorithm with powell's method for the minimum energy broadcast problem in wireless sensor network. Wireless Personal Communications, 2021, 119(2): 1111–1135. [doi: 10.1007/s11277-021-08253-1]

7 Zhou YQ, Wang R, Luo QF. Elite opposition-based flower pollination algorithm. Neurocomputing, 2016, 188: 294–310. [doi: 10.1016/j.neucom.2015.01.110]

8 Abdel-Raouf O, Abdel-Baset M, El-Henawy I. A new hybrid flower pollination algorithm for solving constrained global optimization problems. International Journal of Applied Operational Research, 2014, 4(2): 1–13.

9 Lenin K, Reddy BR, Kalavathi MS. Shrinkage of active power loss by hybridization of flower pollination algorithm with chaotic harmony search algorithm. Control Theory and Informatics, 2014, 4(8): 31–38.

10 Tawhid MA, Ibrahim AM. Solving nonlinear systems and unconstrained optimization problems by hybridizing whale optimization algorithm and flower pollination algorithm. Mathematics and Computers in Simulation, 2021, 190: 1342–1369. [doi: 10.1016/j.matcom.2021.07.010]

11 Draa A. On the performances of the flower pollination algorithm—Qualitative and quantitative analyses. Applied Soft Computing, 2015, 34: 349–371. [doi: 10.1016/j.asoc.2015.05.015]

12 夏学文, 刘经南, 高柯夫, 等. 具备反向学习和局部学习能力的粒子群算法. 计算机学报, 2015, 38(7): 1397–1407. [doi: 10.11897/SP.J.1016.2015.01397]

13 喻飞, 李元香, 魏波, 等. 透镜成像反学习策略在粒子群算法中的应用. 电子学报, 2014, 42(2): 230–235. [doi: 10.3969/j.issn.0372-2112.2014.02.004]

14 Rao RV, Savsani VJ, Vakharia DP. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design, 2011, 43(3): 303–315. [doi: 10.1016/j.cad.2010.12.015]

15 Karaboga D. An idea based on honey bee swarm for numerical optimization. Kayseri: Erciyes University, 2005.

16 Nasir M, Das S, Maity D, *et al*. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Information Sciences, 2012, 209: 16–36. [doi: 10.1016/j.ins.2012.04.028]

17 Huang FZ, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation, 2007, 186(1): 340–356. [doi: 10.1016/j.amc.2006.07.105]

18 He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation, 2007, 186(2): 1407–1422. [doi: 10.1016/j.amc.2006.07.134]

19 Wang Y, Cai ZX, Zhou YR. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. International Journal for Numerical Methods in Engineering, 2009, 77(11): 1501–1534. [doi: 10.1002/nme.2451]

20 Ray T, Liew KM. Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation, 2003, 7(4): 386–396. [doi: 10.1109/TEVC.2003.814902]

21 Rao RV, Patel V. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. International Journal of Industrial Engineering Computations, 2012, 3(4): 535–560. [doi: 10.5267/j.ijiec.2012.03.007]

22 于坤杰, 王昕, 王振雷. 基于反馈的精英教学优化算法. 自动化学报, 2014, 40(9): 1976–1983. [doi: 10.3724/SP.J.1004.2014.01976]

(校对责编: 牛欣悦)