改进蝴蝶算法求解多维复杂函数优化问题

刘景森[1,2],马义想[2],李 煜[3]
(1. 河南大学智能网络系统研究所,河南开封 475004; 2. 河南大学软件学院,河南开封 475004;
3. 河南大学管理科学与工程研究所,河南开封 475004)

摘 要: 针对蝴蝶优化算法存在的问题,提出一种融合差分变异策略并根据进化代数自适应调整权重的蝴蝶优化算法.首先,在全局搜索阶段引入非线性惯性权重改善蝴蝶位置更新公式,自适应调节算法在不同进化时期的搜索范围和粒度,提高算法的收敛速度与寻优精度;然后通过加入F分布全局自适应随机变异对全局公式进一步改进,提升算法的全局探索遍历性,防止出现低精度早熟现象;最后在局部搜索阶段融入具有判定系数和扰动因子的双向差分变异策略,在不减损种群多样性的同时使蝴蝶个体的探索更具方向性,有利于算法摆脱局部极值点,加快收敛速度.理论分析证明了改进算法的时间复杂度与基本蝴蝶优化算法一致,6种代表性对比算法在CEC 2017基准函数上进行的多种维度测试结果表明,改进算法在求解高维复杂函数优化问题时收敛速度和寻优精度明显优于其它对比算法,维度变化对求解性能的影响更小,寻优性能更好更稳定.

关键词: 蝴蝶优化算法; 高维复杂函数; 差分变异; 非线性惯性权重; 扰动因子
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 0372-2112 (2021)06-1068-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20200148

Improved Butterfly Algorithm for Multi-dimensional Complex Function Optimization Problem

LIU Jing-sen [1,2], MA Yi-xiang [2], LI Yu [3]
(1. *Institute of Intelligent Networks System, Henan University, Kaifeng, Henan* 475004, *China*;
2. *College of Software, Henan University, Kaifeng, Henan* 475004, *China*;
3. *Institute of Management Science and Engineering, Henan University, Kaifeng, Henan* 475004, *China*)

Abstract: Aiming at the problem of butterfly optimization algorithm, this paper proposes a butterfly optimization algorithm which fuses the differential mutation strategy and adaptively adjusts the weight according to the evolutionary algebra. The nonlinear inertial weight is introduced in the global search stage to improve the update equation of butterfly position, the search range and granularity of the algorithm in different evolution periods are adjusted adaptively, and the convergence speed and optimization accuracy of the algorithm are improved. The global equation is further improved by adding F distribution global adaptive random mutation to improve the global search ergodicity of the algorithm, and prevent the occurrence of low precision precocious phenomenon. The bidirectional differential mutation strategy with decision coefficient and disturbance factor is integrated in the local search stage, which makes the exploration of butterfly individuals more directional without derogating the diversity of the population, which is beneficial to the algorithm to get rid of the local extremum points and speed up the convergence speed. The theoretical analysis proves that the time complexity of the improved algorithm is consistent with the basic butterfly optimization algorithm. The multi-dimensional test results of six representative contrast algorithms on benchmark functions of CEC 2017 show that the optimization accuracy and convergence speed of the improved algorithm is obviously better than those of other contrast algorithms in solving the optimization problem of high-dimensional complex functions, and the change of dimensions have less impact on the performance of the algorithm, and the optimization performance is better and more stable.

Key words: butterfly optimization algorithm; high-dimensional complex function; difference mutation; non-linearity inertia weight; disturbance factor

收稿日期:2020-02-04;修回日期:2020-05-27;责任编辑:梅志强
基金项目:国家自然科学基金(No. 71601071);河南省重点研发与推广专项(No. 182102310886);河南大学研究生教育创新与质量提升项目(No. SYL18060145, No. SYL19050104)

1 引言

优化是具有普遍适用性的工程数学问题,在面对大规模、复杂化问题时传统的数值优化方法难以在一定时间内给出合理解. 而现代生物元启发式算法则能够快速解决这些大规模复杂问题,并得到满意解,经典的元启发式算法主要有遗传、蚁群、粒子群等算法. 最近几年对于群智能优化算法的研究十分活跃,相继涌现多个机制各异、性能优越的新算法,如蝙蝠算法[1,2]、布谷鸟算法[3]、花朵授粉算法[4,5]、鲸鱼优化算法[6,7]、樽海鞘群算法[8]等等. 相关算法已被广泛应用于机器人路径规划[9,10]、数据聚类[11]、多阈值图像分割[12]、财务压力预测[13]、变速风力发电机[14]等诸多领域.

蝴蝶优化算法(Butterfly Optimization Algorithm, BOA)[15]是由 Sankalap Arora 和 Satvir Singh 于 2019 年正式提出的一种模拟蝴蝶觅食行为的启发式全局优化算法. BOA 算法实现简单,参数较少,思路新颖,适于求解高维函数优化问题,与最近几年提出的其它活跃优化算法相比,寻优性能较好且受维度变化影响很小,具有较大的研究潜力. 但也存在着对于 CEC 复杂函数收敛速度较慢、寻优精度不高等问题.

CEC 函数的复杂性和算法求解这些函数的困难性,吸引了大量学者的关注和研究. Chao 等[16]提出了一种基于迭代局部搜索和随机性惯性权重的蝙蝠算法,在 CEC 2005 基准测试套件上进行测试,证明了改进算法有较强的跳出局部极值能力;Wen 等[17]调整非线性控制参数和修改基于当前个体最优与全局最优位置提出了一种灰狼优化算法,且通过在 CEC 2005 基准测试套件上的测试,证明了该改进算法提高了收敛速度与寻优精度;Xu 等[18]提出了一种生物地理学优化算法与差分进化算法相结合的改进算法,在 CEC 2005 上测试了其优越的性能. Monalisa 等[19]提出了一种基于单目标进化算法集成的自适应多目标优化算法框架,在 CEC 2009 中的基准函数上测试了算法的求解性能与稳健性;Supriya 等[20]提出了一种基于当前全局最优解引导的布谷鸟搜索算法,在 CEC 2013 上的测试证明了其优越的求解性能;Tanabe 等[21]提出了 L_SHADE 算法,并在 CEC 2014 基准测试套件上证明了该算法具有较强竞争力.

针对 BOA 算法存在的不足,本文提出一种融合差分变异策略并根据进化代数自适应调整权重的蝴蝶优化算法(Butterfly Optimization Algorithm based on Differential Mutation and Adaptive weight, DMABOA). 该算法首先在全局搜索阶段引入非线性惯性权重自适应调节算法在不同进化时期的探测范围和搜索粒度;同时通过加入 F 分布自适应随机变异对全局公式进一步改进,提高种群的活跃性并扩大了算法的全局搜索范围. 在算法局部搜索阶段,融合具有判定系数和扰动因子的定向变异策略,扰动因子增强了跳离局部极值的能力,而定向变异策略则加快了算法的收敛速度. 理论分析证明了本文算法并没有降低算法执行效率,而 6 种代表性对比算法在 CEC 2017 基准函数上进行的多种维度函数极值优化结果表明,DMABOA 算法在高维复杂函数上的求解精度和收敛速度都有显著提高.

2 基本蝴蝶优化算法

在 BOA 算法中,每只蝴蝶都会散发一定浓度的香味,同时也都能感知到其它蝴蝶的香味,并朝着散发更浓香味的蝴蝶移动. 蝴蝶的香味浓度公式如下:

$$f = c * I^a \tag{1}$$

其中,f 表示香味浓度大小,I 为刺激强度,与适应度有关;a 为幂指数,c 为感知形态. a 和 c 的范围通常是[0, 1],在 BOA 算法中 $a = 0.1$,c 的初值为 0.01[15].

基本 BOA 算法的流程如以下步骤所示:

Step1 初始化参数:种群大小 N、转换概率 p、维度 dim、蝴蝶香味的感知形态 c 和幂指数 a、迭代次数 N_{iter} 和蝴蝶的初始位置 $\boldsymbol{x} = (x_1, x_2, \cdots, x_{\text{dim}})$.

Step2 计算每只蝴蝶的目标函数适应度值并找到当前最优值,由适应度值决定每只蝴蝶的刺激强度,并由此计算蝴蝶散发的香味浓度.

Step3 产生均匀分布随机数 rand,用于决策蝴蝶是进行全局搜索还是局部搜索.

Step4 当 rand < p 时,进行全局搜索,蝴蝶个体向全局最优解移动,全局搜索公式为:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + (r^2 * g^* - \boldsymbol{X}_i^t) * f \tag{2}$$

式中,$\boldsymbol{X}_i^t$ 表示第 i 只蝴蝶在第 t 次迭代时的解向量,r 为[0,1]之间的均匀分布随机数,g^* 表示当前代全局最优解.

Step5 当 rand ≥ p 时,进行局部搜索,蝴蝶进行局部游走,局部搜索公式为:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + (r^2 * \boldsymbol{X}_j^t - \boldsymbol{X}_k^t) * f \tag{3}$$

式中,$\boldsymbol{X}_i^t, \boldsymbol{X}_j^t, \boldsymbol{X}_k^t$ 分别表示第 i, j, k 只蝴蝶在第 t 代的解向量,其中 $\boldsymbol{X}_j^t$ 和 $\boldsymbol{X}_k^t$ 是在第 t 代种群中随机选择的两个个体的位置向量.

Step6 判断算法运行是否符合结束条件,若符合则记录当前最优解及其目标值,若不符合则根据式(4)更新香味浓度的感觉形态 c 并转回 Step2,继续下一轮迭代.

$$c(t+1) = c(t) + \frac{0.025}{c(t) \cdot N_{\text{iter}}} \tag{4}$$

式中,t 为当前迭代次数,N_{iter} 为最大迭代次数.

3 DMABOA 改进算法

3.1 引入非线性惯性权重

在迭代进化类算法中,惯性权重可以用来调节与控制算法的全局勘测和局部开采能力. 针对基本蝴蝶算法对于复杂函数收敛速度慢、寻优精度低的缺点,本文在全局搜索阶段引入自适应惯性权重,该权重随着进化代数的增加非线性递减,提出的惯性权重函数如下:

$$w=\frac{1}{(t+1)\hat{}(\alpha+\beta)\hat{}3}*\left(\frac{\alpha*t\hat{}\alpha}{N_{\text{iter}}\hat{}\alpha}+\frac{\beta*t\hat{}\beta}{N_{\text{iter}}\hat{}\beta}\right) \tag{5}$$

其中,α 和 β 为权重系数,且 $\alpha=3,\beta=5$. 分析上式可以得出,惯性权重 w 的值在[0,1]之间,取值随着迭代次数的增加呈减速递减趋势,即在进化前期,权重值较大,算法的全局搜索能力较强、探测范围较大,有利于算法跳出局部最优,而随着迭代次数的增加权重值减小,此时蝴蝶在最优值附近进行精细挖掘,使得算法的寻优精度提升和收敛速度加快. 而后期不断减速的递减趋势,也在提升挖掘能力的同时,平衡了一定的搜索能力,维持了适当的种群活跃性和跳离局部极值点的能力. 改进后的全局位置更新公式为:

$$\boldsymbol{X}_i^{t+1}=w*\boldsymbol{X}_i^t+(r^2*g^*-\boldsymbol{X}_i^t)*f \tag{6}$$

3.2 加入具有全局自适应特征的 F 分布随机变异

在基本蝴蝶优化算法中,种群活跃性是决定算法全局寻优遍历性的关键因素,本文为提高蝴蝶种群的活跃性,将自适应随机变异引入全局公式中,使算法在全局搜索时,下一代蝴蝶位置的更新不仅由当前蝴蝶位置和全局最优蝴蝶位置决定,也有当代种群中随机蝴蝶位置的参与,从而增加了蝴蝶种群的多样性,提高了算法在全局阶段的搜索能力. 自适应随机变异公式如下:

$$\text{Ra}=\varepsilon*(\boldsymbol{X}_j^t-\boldsymbol{X}_k^t) \tag{7}$$

$$\varepsilon=\varepsilon_0+\text{fpdf}()*(1-\varepsilon_0)*\left(\frac{N_{\text{iter}}-t}{\pi*N_{\text{iter}}}\right) \tag{8}$$

式(7)中,$\boldsymbol{X}_j^t$ 和 $\boldsymbol{X}_k^t$ 是当代种群内部随机产生的随机解,ε 为变异算子,其计算公式如表达式(8)所示,其中 ε_0 是初始变异算子,经大量测试,ε_0 为 0.1 时,算法改进效果最好. fpdf( ) 为服从 F 分布产生的随机数,F 分布函数呈现峰值不对称形态,先迅速升高而后缓慢下降,有助于蝴蝶个体随着迭代次数的增加而自适应地收敛到全局最优解位置,结合迭代次数与 F 分布的公式如下:

$$F=\frac{\Gamma\frac{\mu+\lambda}{2}}{\Gamma\frac{\mu}{2}\Gamma\frac{\lambda}{2}}*\left(\frac{\mu}{\lambda}\right)*\left(\frac{\mu}{\lambda}\cdot\left(1+\frac{t}{N_{\text{iter}}}\right)\right)\hat{}\left(\frac{\mu+1}{2}-1\right)$$

$$*\left(1+\frac{\mu}{\lambda}\cdot\left(1+\frac{t}{N_{\text{iter}}}\right)\right)\hat{}\left(-\frac{\mu+\lambda}{2}\right) \tag{9}$$

式中,λ 和 μ 是位置不可互换的自由度,经反复测试,$\mu=3,\lambda=4$ 时,算法在全局搜索时能力最强,收敛速度最快. 此时全局搜索的公式如下所示:

$$\boldsymbol{X}_i^{t+1}=w*\boldsymbol{X}_i^t+(r^2*g^*-\boldsymbol{X}_i^t+\text{Ra})*f \tag{10}$$

3.3 融入差分定向变异策略的局部搜索

定向变异[22](Targeted Mutation,TM)策略是对差分变异策略的一种改进,将定向变异策略融入到蝴蝶优化算法的局部搜索过程中,可以很好地调节差分向量随机性和确定性之间的平衡,从而避免了在局部搜索阶段因为较大随机性而导致的搜索效率较低和收敛速度较慢等问题.

蝴蝶优化算法求得高质量全局最优解的关键是算法能否摆脱局部极值,若只是单一加入定向变异策略则在加快收敛速度的同时也会增加算法落入局部极值的可能. 为了降低这一风险,DMABOA 算法在定向差分策略的基础上,利用判定系数 ξ 在当前蝴蝶位置与当前代全局最优蝴蝶位置二者之间进行选择,如果决策随机数大于判定系数,就基于当前蝴蝶位置进行随机游走;若决策随机数小于等于该系数,则基于当前代全局最优蝴蝶位置进行随机游走. 定向变异策略表达式如下:

if rand $>\xi$

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_i^t+(r^2*\boldsymbol{X}_j^t-\boldsymbol{X}_k^t)*f \tag{11}$$

else

$$\boldsymbol{X}_i^{t+1}=\eta\cdot g^*+\theta*(\boldsymbol{X}_j^t-\boldsymbol{X}_k^t)*f \tag{12}$$

其中,θ 为黄金比例系数,r 是[0,1]之间的均匀分布随机数,$g*$ 是当前全局最优解,可以使新解朝着最优解方向移动,加快收敛速度. 经大量实验反复测试,$\xi=0.9$ 时,算法局部搜索能力最强,且更容易跳出局部极值;η 为扰动因子,是为了防止因当前最优值的引入而导致在局部搜索时算法陷入局部极值,进一步提高算法的寻优性能,找到全局最优解,η 的计算公式如下:

$$\eta=1+\text{gamrnd}()*\tan\left(\pi*\left(\text{rand}-\frac{1}{2}\right)\right) \tag{13}$$

式中 gamrnd( ) 是伽马随机数,gamrnd( ) 随机数取值的跳跃性有助于算法跳出局部最优值,并使得扰动因子 η 的取值更加灵活多样.

4 DMABOA 算法流程与时间复杂度分析

4.1 DMABOA 算法流程

DMABOA 算法描述如下:

适应度函数 $f(x)$,$\boldsymbol{x}=(x_1,x_2,\cdots,x_{\text{dim}})$

初始化蝴蝶种群 $\boldsymbol{x}_i(i=1,2,\cdots,n)$

初始化各参数 $c,a,p,\varepsilon_0,N_{\text{iter}}$
计算所有个体适应度值,并进行排序,找到当前最优值
WHILE($t <= N_{\text{iter}}$)
　　由式(5)计算非线性惯性权重
　　由式(8)计算随机变异算子
　FOR $i=1:n$
　　由式(1)计算蝴蝶的香味浓度 f
　　IF rand $<p$
　　由式(7)计算全局随机变异
　　按式(10)进行全局搜索
　　ELSE
　　IF rand $>\xi$
　　根据式(11)进行基于当前位置的随机游走
　　ELSE
　　由式(13)计算扰动因子
　　根据式(12) 进行基于最优位置的随机游走
　　END IF
　　END IF
　计算新的适应度值,确定当前最优值和最优值位置
　END FOR
　根据式(4)更新香味浓度的感觉形态 c
END WHILE
输出结果

4.2 DMABOA 算法的时间复杂度分析

在基本蝴蝶优化算法中,假设:位置自变量的维数为 n,种群大小为 N,各参数初始值设置、生成一个均匀分布随机数、求目标函数值、将所有个体的适应度值排序并得出当代最优个体适应度值的时间分别为 t_1、t_2、$f(n)$、t_3,则算法初始化阶段的总体时间复杂度为:

$$T_1 = O(t_1 + N(t_2 n + f(n)) + t_3) = O(n + f(n))$$

进入迭代后,迭代次数为 N_{iter}. 在计算香味浓度时,设:蝴蝶个体由式(1)产生香味浓度的时间为 t_4,则该阶段的时间复杂度为:

$$T_2 = O(N(t_4)) = O(t_4)$$

在全局搜索阶段,假设:种群中有 $m(0 \leqslant m \leqslant N)$ 只蝴蝶进行全局搜索,新蝴蝶适应度值的计算时间为 $f(n)$,根据式(2)中蝴蝶个体每一维进行位置更新的时间为 t_5,新、旧蝴蝶个体适应度值比较替换的时间为 t_6,则该阶段的时间复杂度为:

$$T_3 = O(m(f(n) + t_5 n + t_6)) = O(n + f(n))$$

在局部搜索阶段,种群中有 $(N-m)$ 只蝴蝶进行局部随机游走,假设:由式(3)生成新个体中每一维元素的时间为 t_7,计算新适应度值的时间 $f(n)$ 和比较替换新旧解适应度值的时间 t_6 都与全局搜索阶段相同,故该阶段的时间复杂度为:

$$T_4 = O((N-m)(t_7 n + f(n) + t_6)) = O(n + f(n))$$

在记录最优解阶段,设:每个个体适应度值与当前最优解进行比较替换的时间为 t_8,则这一阶段的时间复杂度为:$T_5 = O(N(t_8)) = O(t_8)$

每轮迭代的最后,设:由式(4)对感觉形态 c 进行更新的时间为 t_9,则该阶段的时间复杂度为:$T_6 = O(t_9)$

综上所述,BOA 算法的总时间复杂度为:

$$\begin{aligned} T(n) &= T_1 + N_{\text{iter}}(T_2 + T_3 + T_4 + T_5 + T_6) \\ &= O(n + f(n)) \end{aligned}$$

在 DMABOA 算法中,种群大小、位置自变量维度与基本 BOA 完全一致,目标函数适应度值求解、参数设置的时间也与 BOA 相同,因此 DMABOA 算法在初始化种群、计算香味浓度、记录最优解和更新感觉形态 c 阶段的时间复杂度都与基本蝴蝶优化算法相同,其总的时间复杂度为:

$$\begin{aligned} T_1' &= T_1 + N_{\text{iter}}(T_2 + T_5 + T_6) \\ &= O(n + f(n)) + N_{\text{iter}}(O(t_4 + t_8 + t_9)) \\ &= O(n + f(n)) \end{aligned}$$

在全局搜索阶段,DMABOA 算法在位置更新公式中引入了自适应动态惯性权重和随机变异,由式(5)可知,惯性权重 w 随迭代次数发生改变,但在同一代种群中惯性权重相同. 同样,由式(8)在算法中的流程可知,同一代种群中变异算子也相同. 设:种群中有 m 只蝴蝶进行全局搜索,由式(5)产生惯性权重 w 的时间为 g_1,由式(8)计算变异算子的时间为 g_2,由式(7)计算全局随机变异的时间为 g_3,则该阶段的时间复杂度为:

$$\begin{aligned} T_2' &= O(g_1 + g_2 + m(g_3 + t_5 n + f(n) + t_6)) \\ &= O(n + f(n)) \end{aligned}$$

在局部搜索阶段,蝴蝶分两种方式进行局部随机游走,设:由式(11)基于当前解进行随机游走的蝴蝶个数为 $q(0 \leqslant q \leqslant N-m)$,则由式(12)基于当前最优解进行随机游走的蝴蝶个数为 $N-m-q$. 于是,当 rand $>\xi$ 时,该阶段的时间复杂度为:

$$T_3' = O(q(t_7 n + f(n) + t_6)) = O(n + f(n))$$

当 rand $<\xi$ 时,假设:由式(13)计算扰动因子的时间为 g_4,由式(12)生成新个体中每一维元素的时间为 g_5,则该阶段的时间复杂度为:

$$\begin{aligned} T_4' &= O((N-m-q)(g_4 + g_5 n + f(n) + t_6)) \\ &= O(n + f(n)) \end{aligned}$$

故而,局部搜索阶段的时间复杂度为:

$$T_5' = T_3' + T_4' = 2O(n + f(n)) = O(n + f(n))$$

基于上述分析,DMABOA 算法的总时间复杂度为:

$$T(n) = T_1' + N_{\text{iter}}(T_2' + T_5') = O(n + f(n))$$

由此可知,与基本 BOA 相比,本文算法 DMABOA 并没有额外增加时间复杂度,两者完全相同,执行效率没有下降.

5 复杂函数优化问题实验结果分析

5.1 测试函数与仿真环境

为了检验 DMABOA 在面对多维复杂函数极值优化

问题时的求解性能和收敛情况,选取本文算法 DMABOA、基本蝴蝶算法 BOA[15]及 4 种测试 CEC 的有效改进算法 ILSSIWBA[16]、IGWO[21]、GCS[23]、CMMDEBBO[18]共 6 种算法,通过 10 个 CEC 2017 基准测试函数进行仿真对比测试. 为了保证算法比较的客观性和公平性,6 种对比算法采用相同软、硬件平台,且运行次数、种群规模、空间维度和最大迭代次数都保持一致,即 $N=30$, $\dim=10/50/100$, $N_{\text{iter}}=500$,每种算法分别独立运行 50 次.

参数设置方面,DMABOA 算法的各参数取值均与基本算法 BOA 相同. 4 种对比改进算法的参数设置与其各自原文献中的设置值相同. 在 ILSSIWBA 算法中, $A_0=0.9$, $r_0=0.1$, $\mu_{\min}=0.4$, $\mu_{\max}=0.9$;在 IGWO 算法中, $\omega_{\min}=0.1$ 和 $\omega_{\max}=0.9$;在 GCS 算法中, $\delta=0.01$, pa $=0.25$;在 CMMDEBBO 算法中, CR $=0.9$, Pe $=0.5$.

具体测试函数如表 1 所示. 表 1 中的所有函数均为 CEC 2017 基准测试函数,为了方便测试并提高测试的空间搜索范围,CEC 2017 中将所有基准测试函数的自变量各维取值范围都设定为[-100,100]. 这些函数中, $f_1(x)$ ~$f_3(x)$和$f_6(x)$是复杂单峰函数,主要为了验证算法的收敛速度. $f_4(x)$ ~$f_5(x)$和$f_7(x)$ ~$f_{10}(x)$为复杂多峰函数,有较多局部极值点,主要是为了测试算法跳出局部极值的能力. 这些测试函数的求解难度和各自不同的极值特征很适合测试算法的寻优性能.

表 1 CEC 测试函数

序号	函数名	最优值
f_1	Bent Cigar	0
f_2	Sum of Different Power	0
f_3	Zakharov	0
f_4	Rosenbrock′s	0
f_5	Rastigin′s	0
f_6	High Conditioned Elliptic	0
f_7	Expanded Schaffer′s F6	0
f_8	ACKley′s	0
f_9	Griewank′s	0
f_{10}	Schaffer′s F7	0

5.2 寻优精度分析

表 2 统计了 6 种对比算法在空间维度为 10 维、50 维和 100 维时,各自独立运行 50 次的最差解、最优解和平均值.

表 2 6 种算法在固定迭代次数下的寻优性能比较

函数	算法	dim = 10			dim = 50			dim = 100		
		最差解	最优解	平均值	最差解	最优解	平均值	最差解	最优解	平均值
f_1	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	1.27E-21	9.20E-28	7.24E-23	2.27E-14	1.83E-14	2.03E-14	2.27E-14	1.84E-14	2.03E-14
	ILSSIWBA	4.9E-2155	0	1.0E-216	3.4E-213	1.8E-303	6.7E-215	1.1E-226	1.0E-307	2.4E-228
	IGWO	5.7E-274	6.5E-282	1.8E-275	1.1E-240	6.1E-244	9.3E-242	2.9E-233	7.0E-237	2.5E-234
	GCS	1.4E+09	1.1E+07	2.8E+08	1.0E+10	1.0E+10	1.0E+10	1.0E+10	1.0E+10	1.0E+10
	CMMDEBBO	3.7E-19	2.1E-21	5.4E-20	2.1E+07	1.0E+03	6.8E+05	4.0E+08	2.9E+07	1.2E+08
f_2	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	2.00E-14	1.25E-14	1.70E-14	9.74E-22	2.28E-27	3.14E-23	1.15E-16	7.63E-27	2.30E-18
	ILSSIWBA	3.7E-236	1.3E-301	7.7E-238	3.0E-225	0	6.0E-227	8.4E-239	8.0E-305	2.6E-240
	IGWO	0	0	0	0	0	0	0	0	0
	GCS	7.0E+04	1.2E+03	1.3E+04	1.0E+07	3.1E+06	6.6E+06	9.1E+07	4.8E+07	7.2E+07
	CMMDEBBO	4.2E-11	3.0E-53	8.4E-13	6.0E+40	4.1E+14	1.7E+39	2.7E+110	5.1E+70	5.4E+108
f_3	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	1.90E-14	1.37E-14	1.65E-14	1.99E-14	1.49E-14	1.75E-14	1.97E-14	1.43E-14	1.73E-14
	ILSSIWBA	2.0E-207	4.6E-305	4.2E-209	2.0E-224	0	4.0E-226	2.2E-230	9.9E-303	4.5E-232
	IGWO	6.5E-256	3.7E-265	1.5E-257	8.5E-200	1.3E-210	2.7E-201	4.8E-172	3.3E-191	1.0E-173
	GCS	9.8E+03	2.2E+02	2.3E+03	2.3E+04	9.3E+03	1.4E+04	7.3E+04	3.5E+04	5.5E+04
	CMMDEBBO	1.4E-21	4.5E-24	1.2E-22	3.3E+02	1.2E+00	5.4E+01	3.9E+03	9.4E+02	2.2E+03
f_4	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	1.98E-14	3.04E-22	1.58E-14	2.13E-14	1.78E-14	1.93E-14	2.17E-14	3.73E-24	1.91E-14
	ILSSIWBA	4.4E-29	0	2.0E-30	3.1E-29	0	1.5E-30	2.0E-29	0	1.1E-30
	IGWO	0	0	0	0	0	0	0	0	0
	GCS	5.4E+01	0	7.2E+00	1.1E+02	5.5E-04	1.2E+01	6.5E+01	1.8E-24	9.2E+00
	CMMDEBBO	1.4E+02	3.2E+00	9.6E+00	1.6E+07	3.1E+02	7.6E+05	1.1E+07	3.2E+05	3.8E+06

续表

函数	算法	dim = 10			dim = 50			dim = 100		
		最差解	最优解	平均值	最差解	最优解	平均值	最差解	最优解	平均值
f_5	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	5.04E+01	0	2.61E+01	3.72E+02	0	2.89E+01	2.11E+01	1.75E+01	1.95E+01
	ILSSIWBA	1.1E-13	0	8.5E-15	5.1E-13	0	2.2E-14	1.7E-12	0	8.6E-14
	IGWO	2.8E+01	0	6.1E+00	2.9E+01	0	3.7E+00	2.0E+01	0	4.0E+00
	GCS	2.2E+03	5.8E+01	7.7E+02	3.0E+04	1.2E+04	2.1E+04	6.9E+04	3.3E+04	5.1E+04
	CMMDEBBO	2.1E+00	1.3E-11	3.3E-01	3.0E+02	1.8E+02	2.3E+02	1.8E+03	8.1E+02	1.1E+03
f_6	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	2.13E-14	1.257E-14	1.85E-14	2.21E-14	1.72E-14	1.97E-14	2.26E-14	1.63E-14	1.98E-14
	ILSSIWBA	1.9E-236	7.1E-308	7.1E-238	1.5E-213	0	3.0E-215	1.8E-232	0	4.1E-234
	IGWO	2.0E-276	2.6E-284	1.6E-277	1.0E-242	4.1E-246	8.0E-244	5.9E-236	3.4E-239	5.0E-237
	GCS	3.9E+07	5.2E+05	8.5E+06	8.9E+08	1.6E+08	4.1E+08	1.9E+09	3.0E+08	9.7E+08
	CMMDEBBO	4.1E-22	7.2E-25	5.2E-23	3.5E+04	1.7E+00	1.2E+03	8.7E+05	3.6E+04	2.3E+05
f_7	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	3.11E+00	1.70E+00	2.52E+00	2.11E+01	1.75E+01	1.95E+01	4.47E+01	3.96E+01	4.21E+01
	ILSSIWBA	4.4E-16	0	3.3E-17	1.3E-15	0	1.3E-16	6.4E-15	0	2.1E-16
	IGWO	2.2E+00	1.3E-05	1.3E+00	1.9E+01	0	1.4E+01	4.2E+01	0	2.5E+01
	GCS	2.2E+00	4.0E-01	1.3E+00	1.9E+01	1.4E+01	1.7E+01	4.1E+01	3.3E+01	3.8E+01
	CMMDEBBOO	1.5E+00	4.1E-01	8.9E-01	1.9E+01	1.7E+01	1.8E+01	4.3E+01	3.9E+01	4.1E+01
f_8	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	2.24E-09	0	4.49E-11	4.263E-14	0	9.308E-15	4.263E-14	0	3.197E-15
	ILSSIWBA	1.4E-14	0	1.3E-15	1.4E-14	0	1.1E-15	7.1E-15	0	1.6E-15
	IGWO	2.1E+01	0	2.4E+00	2.1E+01	0	2.5E+00	2.1E+01	0	3.0E+00
	GCS	2.0E+01	8.4E+00	1.7E+01	2.0E+01	2.0E+01	2.0E+01	2.1 E+01	2.0E+01	2.1E+01
	CMMDEBBOO	2.0E+01	1.7E-06	1.8E+01	2.1E+01	2.0E+01	2.0E+01	2.1E+01	2.1E+01	2.1E+01
f_9	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	1.81E-01	0.012293	3.62E-03	2.00E-14	2.22E-16	5.18E-15	2.05E-14	2.22E-15	1.27E-14
	ILSSIWBA	7.8E-16	0	5.8E-17	8.9E-16	0	6.7E-17	6.0E-15	0	3.3E-16
	IGWO	3.2E-01	0	5.6E-02	1.8E-02	0	2.1E-03	2.3E-02	0	1.5E-03
	GCS	1.6E+00	1.9E-01	7.5E-01	8.4E+00	4.1E+00	6.0E+00	2.0E+01	9.1E+00	1.4E+01
	CMMDEBBO	9.9E-02	4.4E-15	2.3E-02	8.2E-02	7.9E-06	1.1E-02	1.1E+00	2.9E-01	6.9E-01
f_{10}	DMABOA	0	0	0	0	0	0	0	0	0
	BOA	6.19E-02	1.385E-07	1.39E-02	3.76E-09	2.19E-13	7.97E-11	5.20E-12	3.29E-13	1.43E-12
	ILSSIWBA	1.5E-155	5.2E-218	2.9E-157	2.8E-155	2.0E-223	5.6E-157	1.1E-147	6.7E-223	2.3E-149
	IGWO	7.5E-139	1.5E-142	4.9E-140	4.7E-125	5.6E-127	9.2E-126	7.5E-122	9.1E-124	1.3E-122
	GCS	1.6E-02	1.7E-04	5.1E-03	1.1E-01	2.9E-02	7.0E-02	9.9E-02	3.5E-02	6.9E-02
	CMMDEBBOO	5.2E-02	1.3E-03	1.8E-02	3.8E+00	9.1E-01	2.4E+00	1.1E+01	4.0E+00	6.3E+00

由表2中的数据可以看出,随着维数增加,除了ILSSIWBA和本文算法DMABOA,其它4种算法的寻优性能都会有不同程度的降低,有的算法甚至从一开始就陷入局部极值. 对于上述基准测试函数,DMABOA算法在不同维度下找到的最差解、最优解和平均值都是理论最优值,其寻优性能远远超过其余5种算法,显示了DMABOA算法在面对复杂函数时优越的求解能力和维度变化适应性.

由表2观察10维下的寻优结果,虽然ILSSIWBA在$f_1(x)$、$f_4(x)$ ~$f_5(x)$、$f_7(x)$ ~$f_9(x)$这6个函数上找到的最优解也是理论最优值,但是从这些函数的最差解和平均值数据可以看出,ILSSIWBA的求解并不稳定,虽然有找到理论最优值的可能,但跳出局部极值的能力却仍有不足;IGWO算法的情况几乎和ILSSIWBA算法一样,也具有找到全局理论最优的可能,但跳出局部极值的能力却远远低于几乎摆脱了所有函数局部极值的DMABOA算法;而BOA、GCS和CMMDEBBO算法则是在几乎所有函数上都没有达到全局最优值. 这说明本文改进算法在低维下的寻优效果更好,跳出局部极值的能力更强.

观察50维和100维下的寻优结果,GCS和CMMDEBBO算法在10个函数上都没有找到过理论最优值,而BOA、ILSSIWBA和IGWO虽然在一部分函数上找到理论最优,但在大部分函数上都陷入了局部极值无法跳出,

体现出这些算法在高维函数求解时的困难和劣势.

以上寻优结果表明,在 10 维、50 维、100 维条件下,DMABOA 算法表现出优越的求解性能,对低维和高维条件下几乎所有复杂函数都找到了理论最优值. 这些结果说明本文算法 DMABOA 很好解决了蝴蝶优化算法在高维复杂函数上寻优精度不高、易陷入局部极值和求解不稳定的问题.

5.3 收敛曲线分析

算法寻优性能的好坏,可以通过收敛曲线图直观地反映出来. 图 1 ~ 图 6 给出了 DMABOA、BOA、ILSSIWBA、IGWO、GCS 和 CMMDEBBO 这 6 种算法在 100 维时对于 6 个多峰函数的收敛曲线对比图,单峰函数的收敛曲线结果相对简单且与多峰函数类似,不再冗余列出.

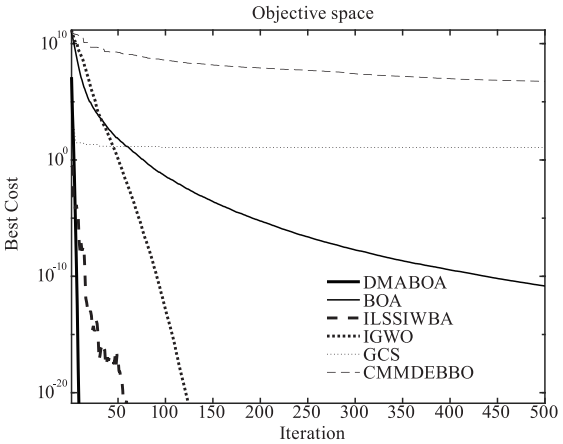


图1 $f_4(x)$的收敛曲线

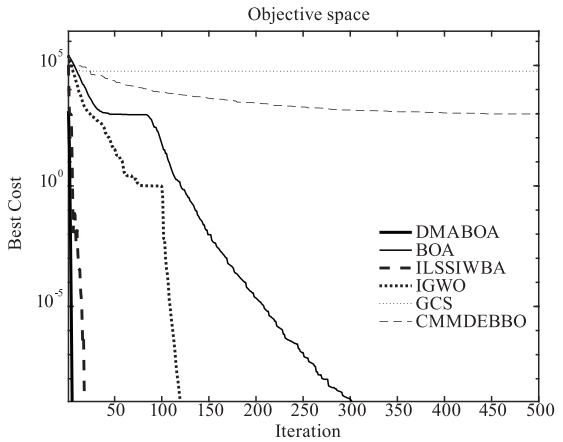


图2 $f_5(x)$的收敛曲线

以上 6 个测试函数的收敛曲线对比图清晰地展现了 DMABOA、BOA、ILSSIWBA、IGWO、GCS 和 CMMDEBBO 算法在迭代次数相同、维度为 100 情况下的寻优结果和收敛曲线变化趋势. 从图中可以看出,相对于另外 5 种算法在大多数情况下无法收敛到全局最优、经常陷入局部极值、收敛速度缓慢,DMABOA 算法没有陷入局部极值的情况,不仅找到了全局最优值,而且其收敛曲线没有拐点,说明改进后算法跳出局部极值能力更强.

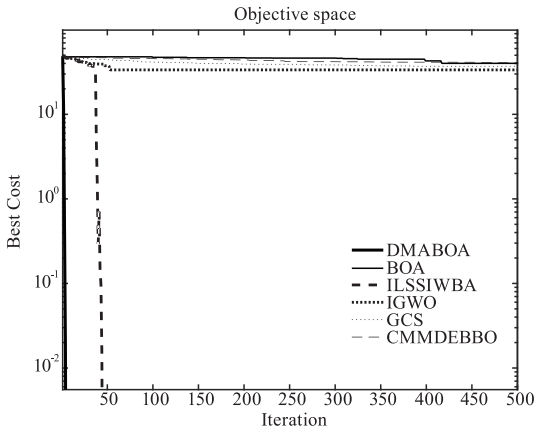


图3 $f_7(x)$的收敛曲线

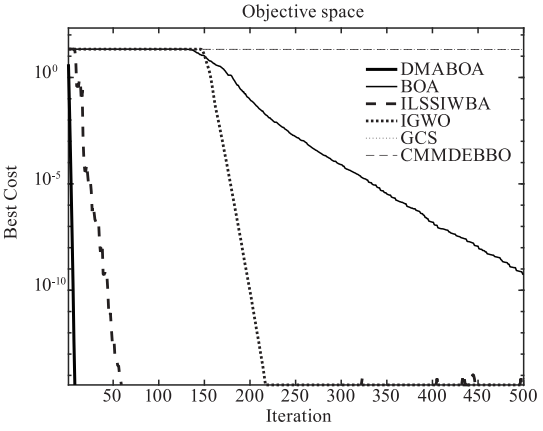


图4 $f_8(x)$的收敛曲线

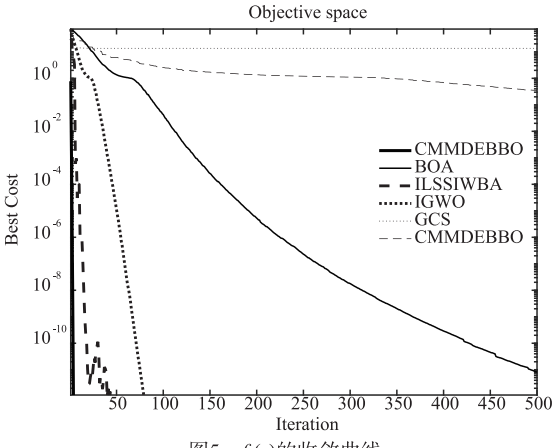


图5 $f_9(x)$的收敛曲线

由表 2 数据可知,ILSSIWBA 算法在函数 $f_4(x)$ ~ $f_5(x)$、$f_7(x)$ 和 $f_9(x)$ 上,IGWO 算法在函数 $f_4(x)$ ~ $f_5(x)$ 和 $f_7(x)$ ~ $f_9(x)$ 上找到的最优解是理论最优,但由图 1 ~ 图 5 可以看出,这 2 种算法的收敛速度都没有 DMABOA 算法快,若进化代数较少,就无法收敛到理论

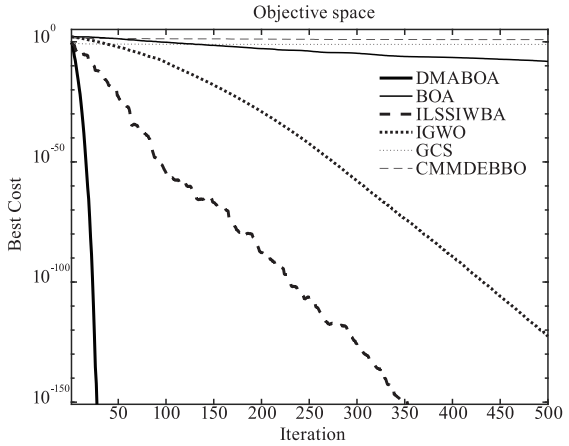


图6　$f_{10}(x)$的收敛曲线

最优值附近. BOA、GCS 和 CMMDEBBO 算法的收敛性较差,在很多函数上前期就已经陷入了局部极值无法跳出,完全无法在 500 代内收敛到理论最优值附近,表现出对于求解高维复杂函数的无能为力.

由 5.2 节的寻优精度和 5.3 节的收敛曲线图分析可知,在 10 维、50 维、100 维条件下,DMABOA 算法表现出非常优越的寻优性能,在几乎所有 CEC 复杂函数各个维度上都找到了理论最优值,且 DMABOA 算法在高维条件下的收敛速度也明显要比 BOA、ILSSIWBA、IGWO、GCS 和 CMMDEBBO 算法更快. 这主要是因为 DMABOA 算法在全局公式中引入了非线性惯性权重自适应调节算法在不同进化时期的搜索范围和粒度,使得算法在前期搜索时范围扩大,增强跳出局部极值的能力,在后期进行精细挖掘,提高寻优精度,加快收敛速度;同时加入 F 分布全局自适应随机变异,增强蝴蝶在整个空间区域搜索的随机性和跳跃性,不仅增加了种群的多样性,提高算法的全局搜索能力,也有利于算法在高维情况下跳出局部极值,找到全局最优解. 而在进行局部游走时,则是融入了具有判定系数和扰动因子的定向差分变异策略,其中扰动因子起到了维系蝴蝶种群多样性的作用,这种多样性使算法更容易跳出局部极值,提高寻优精度;而定向差分变异策略则使蝴蝶个体的探索更具方向性,加快了算法的收敛速度,使其能够在较少进化代数内找到理论最优值. 上述实验说明 DMABOA 算法较好解决了蝴蝶优化算法在求解高维复杂函数极值优化问题上寻优精度不高、易陷入局部极值、收敛速度较慢的缺点,具有很好的寻优能力和维度适应性,本文对算法的改进是成功和有效的.

6　结束语

本文提出了一种融合差分变异策略并根据进化代数自适应调整权重的蝴蝶优化算法. 在全局搜索阶段,首先引入非线性惯性权重自适应调节算法在不同进化时期的搜索范围和粒度,提高算法的收敛速度和寻优精度,而后通过加入 F 分布全局自适应随机变异对全局公式进一步改进,提高了种群的活跃性,扩大了算法的搜索范围,防止算法陷入早熟过早的低精度收敛. 在局部搜索阶段,融入含有判定系数和扰动因子的定向差分变异策略,增强了算法跳出局部极值的能力,并在维系种群多样性的同时使蝴蝶个体的探索更具方向性,加快了算法的收敛速度. 理论分析证明了改进算法的时间复杂度与基本蝴蝶优化算法相同,6 种代表性算法在 CEC 2017 复杂函数上进行的不同维度测试结果表明,本文改进算法增强了蝴蝶个体跳出局部极值的能力、加快了收敛速度、提高了寻优精度,在维度变化时具有很好的寻优稳定性,较好解决了基本蝴蝶算法面对复杂函数存在的问题.

参考文献

[1] Yang X S. Bat algorithm for multi-objectiveoptimisation. [J]. International Journal of Bio-Inspired Computation, 2011,3(5):267 – 274.

[2] Yang X S, Deb S. Cukoo search via levy flight[A]. Proceedings of World Congress on Nature & Biologically Inspired Computing[C]. Coimbatore, India: IEEE, 2009. 210 – 214.

[3] 戚远航,蔡延光,蔡颢,等. 带容量约束的供应链物流运输调度问题的双层变邻域蝙蝠算法[J]. 电子学报, 2019,47(07):1434 – 1442.
QI Y H, CAI Y G, CAI H, et al. Two-level bat algorithm with variable neighborhood search for capacitated vehicle routing problem in supply chain[J]. Acta Electronica Sinica, 2019,47(07):1434 – 1442. (in Chinese)

[4] Yang X S. Firefly algorithms for multimodal optimization. international symposium on stochastic algorithms[A]. Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications[C]. Berlin, Heidelberg: Springer, 2009. 169 – 178.

[5] Liu J S, Liu L, Li Y. A differential evolution flower pollination algorithm with dynamic switch probability[J]. Chinese Journal of Electronics, 2019,28(04):737 – 747.

[6] Seyedali M, Andrew L. The whale optimization algorithm [J]. Advances in Engineering Software, 2016,95(5):51 – 67.

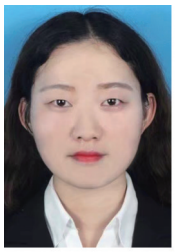
[7] 肖子雅,刘升. 精英反向黄金正弦鲸鱼算法及其工程优化研究[J]. 电子学报, 2019,47(10):2177 – 2186.
XIAO Z Y, LIU S. Study on elite opposition-based golden-sine whale optimization algorithm and its application of project optimization[J]. Acta Electronica Sinica, 2019,47(10):1434 – 1442. (in Chinese)

[8] Seyedali M, Amir H. G, Seyedeh Z M, Shahrzad S, Hossam

F, Seyed M M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems [J]. Advances in Engineering Software, 2017(114): 163 – 191.
[9] 王晓燕,杨乐,张宇,孟帅. 基于改进势场蚁群算法的机器人路径规划[J]. 控制与决策,2018,33(10),1775 – 1781.
Wang X Y, Yang L, Zhang Y, Meng S. Robot path planning based on improved potential field ant colony algorithm[J]. Control and Decision, 2018, 33(10), 1775 – 1781. (in Chinese)
[10] Patricia S, Andrés I, Akemi G. Make robots be bats: specializing robotic swarms to the bat algorithm[J]. Swarm and Evolutionary Computation, 2019, 44: 113 – 129.
[11] Saida I B, Nadjet K, Omar B. A new quantum chaotic cuckoo search algorithm for data clustering[J]. Expert Systems with Applications, 2018, 96(15): 358 – 372.
[12] Khairuzzaman A K M, Chaudhury S. Multilevel thresholding using grey wolf optimizer for image segmentation [J]. Expert Systems With Applications, 2017, 86(15): 64 – 76.
[13] Luo J, Chen H L, Zhang Q, et al. An improved grasshopper optimization algorithm with application to financial stress prediction[J]. Applied Mathematical Modelling, 2018, 64: 654 – 668.
[14] Mohammed H Q, Hany M H, Saad A. Enhanced salp swarm algorithm: application to variable speed wind generators[J]. Engineering Applications of Artificial Intelligence, 2019, 80: 82 – 96.
[15] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization[J]. Soft Computing, 2019, 23(3): 715 – 734.
[16] Gan C, Cao W H. A new bat algorithm based on iterative local search and stochastic inertia weight[J]. Expert Systems with Applications, 2018, 104(15): 202 – 212.
[17] Long W, Jiao JJ, Liang X M, Tang M Z. Inspired grey wolf optimizer for solving large-scale function optimization problems [J]. Applied Mathematical Modelling, 2018, 60: 112 – 126.
[18] Chen X, Tianfield H, Du W L, Liu G H. Biogeography-based optimization with covariance matrix based migration [J]. Applied Soft Computing, 2016, 45: 71 – 85.
[19] Pal M, Bandyopadhyay S. ESOEA: Ensemble of single objective evolutionary algorithms for many-objective optimization[J]. Swarm and Evolutionary Computation, 2019, 50: 10051.
[20] Dhabal S, Venkateswaran P. An efficient gbest-guided cuckoo search algorithm for higher order two channel filter bank design[J]. Swarm and Evolutionary Computation, 2017, 33: 68 – 84.
[21] Tanabe R, Fukunaga A S. Improving the search performance of SHADE using linear population size reduction [A]. 2014 IEEE Congress on Evolutionary Computation (CEC)[C]. Beijing, China: IEEE, 2014. 1658 – 1665
[22] Zheng W J, Fu H H, Yang G W. Targeted mutation: A novel mutation strategy for differential evolution[A]. 2015 IEEE 27th International Conference on Tools with Artificial Intelligence[C]. Vietri sul Mare, Italy: IEEE, 2015. 286 – 293.
[23] Supriya Dhabal, Palaniandavar Venkateswaran. An efficient gbest-guided Cuckoo search algorithm for higher order two channel filter bank design[J]. Swarm and Evolutionary Computation, 2017, 33: 68 – 84.

作者简介

刘景森 男,1968 年生,河南开封人,博士,教授,研究方向:智能算法、优化控制和网络安全等.
E-mail: ljs@ henu. edu. cn.



马义想 女,1994 年生,河南驻马店人,硕士研究生,研究方向:智能算法.
E-mail: myxhenu@ 163. com

李 煜(通讯作者) 女,1969 年生,河南开封人,博士,教授,研究方向:智能算法和电子商务等.
E-mail: leey@ henu. edu. cn.