第 37 卷 第 1 期
2022 年 2 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol. 37 No. 1
Feb. 2022

DOI:10.13364/j.issn.1672-6510.20210064

一种基于反向学习和伯恩斯坦算子的差分进化算法

熊聪聪，杨晓艺，王 丹，李俊伟，管祥烁
（天津科技大学人工智能学院，天津 300457）

摘 要：差分进化（differential evolution，DE）算法是一种种群随机搜索算法，但其在收敛过程中存在着容易陷入局部最优、收敛精度不高等问题. 为更好地提升 DE 算法的性能，提出一种改进算法为基于反向学习和伯恩斯坦算子的差分进化算法. 反向学习策略用于增加种群的多样性，扩大种群的搜索范围，从而弥补陷入局部最优的不足，提高了收敛速度；伯恩斯坦多项式随机产生算法的结构参数值控制了进化过程中的变异和交叉阶段，改变了差分进化算法原有的进化策略，提高了收敛性能，是一种更加快速、高效的无参数方法. 通过国际标准测试函数的实验结果表明，改进后的差分进化算法具有更强的全局寻优能力，整体收敛速度和精度明显提高.

关键词：差分进化算法；伯恩斯坦多项式；反向学习
中图分类号：TP181 **文献标志码**：A **文章编号**：1672-6510（2022）01-0046-06

An Differential Evolution Algorithm Based on Opposition-Based Learning and Bernstain Operator

XIONG Congcong，YANG Xiaoyi，WANG Dan，LI Junwei，GUAN Xiangshuo
（College of Artificial Intelligence，Tianjin University of Science & Technology，Tianjin 300457，China）

Abstract：Differential evolution（DE）algorithm is a population random search algorithm. However，in the process of convergence，it is easy to fall into local optimal value and the convergence accuracy is not high. In order to improve the performance of DE algorithm，a differential evolution algorithm based on opposition-based learning and Bernstain operator（BODE）is proposed in this article. Opposition-based learning is used to increase the diversity of the population，expand the search range of the population，so as to alleviate falling into local optimal value and improve the convergence speed. Bernstain polynomial randomly generates the structural parameter values of the algorithm，which controls the mutation and crossover stages in the evolution process，changes the original evolution strategy of the differential evolution algorithm，improves the convergence performance，and is a more rapid and efficient method without parameters. The experimental results of the international standard test function show that the improved differential evolution algorithm has stronger global optimization ability. The overall convergence speed and accuracy have significantly improved.

Key words：differential evolution algorithm；Bernstain polynomial；opposition-based learning

差分进化（differential evolution，DE）算法[1]是一种模仿生物群体进化的全局优化算法. 与其他进化算法相同，DE 算法也是在一个随机生成的初始种群中开始搜索，通过变异、交叉、选择操作产生下一代的种群，不断迭代求解，直至找到最终解. 因其具有模型简单、收敛性能好、鲁棒性强等特点，被应用到模式识别、数据挖掘、人工智能等各个领域[2-3]. DE 算法与其他群体智能算法相比具有很多优点，但随着研究者的不断深入探索，发现差分进化算法在参数设置中存在着计算开销大、收敛过程中存在着易陷入局部最优和收敛精度不高的缺陷.

针对上述问题，很多研究者对算法进行改进，不断提高算法的收敛速度和精度. 比如，为了避免结构参数的试错过程中计算消耗大，Wang 等[4-5]提出的算

收稿日期：2021-03-26；**修回日期**：2021-07-04
基金项目：国家自然科学基金资助项目（11803022）
作者简介：熊聪聪（1961—），女，四川人，教授，xiongcc@tust.edu.cn

法可以使每个个体在参数池中随机选择参数作为结构参数，而基于正交算子的差分进化（orthogonal crossover differential evolution，OXDE）算法使用正交交叉概率作为结构参数. 但这些结构参数的选择方式都是人为设置的，由于缺乏经验而导致的稳定性不足还是无法避免的. 于是，研究人员提出了基于自适应的参数调节方式和随机设置参数的方法，如 Zhang 等[6]提出了基于学习的变异和交叉概率的调整方法，在学习进化过程中成功找到缩放因子和交叉概率等参数信息；Civicioglu 等[7]提出了一种非递归并且无参数的差分进化（Bernstain-search different evolution，BSD）算法，这种方法没有设置控制参数的过程，使算法更加简单、高效.

但是，上述算法只是对结构参数的改进，对算法性能的提升有限. 为了解决差分进化算法在进化过程中陷入局部最优导致收敛速度过慢的问题，Zhu 等[8]提出了一种混合交叉算法，该多目标进化算法通过差分变异策略构造出自适应混合交叉算子，使多目标进化算法的全局与局部的搜索能力得到均衡；童旅杨等[9]通过使用多个差分变异算子增加种群的多样性，从而避免了差分进化算法在进化过程中易于陷入局部最优的问题；Zhang 等[10]提出了一种具有排序选择的差分进化（differential evolution with a rank-up selection，RUSDE）算法；钱峥远等[11]提出了一种精英化岛屿的种群差分进化算法，实现了全局搜索和局部搜索能力并重，使算法具有较强的搜索能力与稳定性.

虽然上述文献中提出的改进算法在进化时相对于传统 DE 算法的性能有了一定的提升，但对算法的改进比较单一，存在着对算法性能提升不足的问题. 基于此，本文提出了一种结合参数改进与进化策略改进的算法，即基于反向学习和伯恩斯坦算子的差分进化算法（differential evolution algorithm based on opposition-based learning and Bernstain operator，BODE）. 通过对国际标准测试函数进行测试实验，将 BODE 与已有的优化算法进行了性能比较，结果表明本算法在搜索速度以及搜索最优值的精度上更具有优势.

1　预备知识

1.1　伯恩斯坦多项式

伯恩斯坦多项式（Bernstain polynomial）[12]是逼近连续函数的一系列多项式，它们可以自然地组合成近似于[0，1]上的任何连续函数. 二阶伯恩斯坦多项式定义为

$$B_{s,n}(t)=\binom{n}{s}t^s(1-t) \tag{1}$$

其中：n 为迭代次数；t 为成功的概率，取值范围为[0，1]；s 为迭代 n 次后实验成功的概率；$\binom{n}{s}=\frac{n!}{s!(n-s)!}$. 通过这个式子可以将固定概率和可变成功次数转移到固定成功次数和可变概率中. 通过式（2）可生成（$n+1$）大小的 n 次伯恩斯坦多项式. 其中 $s<0$ 并且 $s>n$，$B_{s,n}=0$.

$$\begin{aligned}&B_{0,2}(t)=(1-t)^2\\&B_{1,2}(t)=2t(1-t)\\&B_{2,2}(t)=t^2\end{aligned} \tag{2}$$

图 1 为 $0\leqslant t\leqslant 1$ 时二阶伯恩斯坦多项式曲线图.

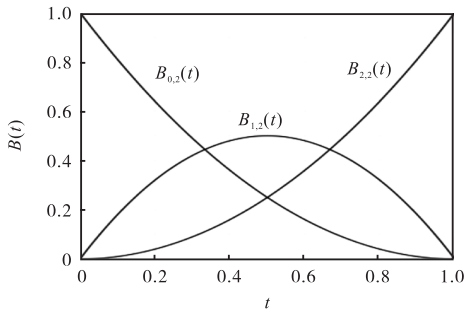

图 1　$0\leqslant t\leqslant 1$ 时二阶伯恩斯坦多项式曲线图

Fig. 1　2nd degree Bernstain polynomials when $0\leqslant t\leqslant 1$

1.2　反向学习策略

反向学习（opposition-based learning，OBL）策略[13]用反向解与当前解相比，有高出 50% 的概率更靠近全局最优值. 反向学习策略的主要思想是对问题进行求解 x 时，初始值 x 是通过经验积累或者是单纯的随机猜想，可以同时使用 x 的相反值尝试得到更好的解 x^*，使得下一代 x 能够更快逼近最优解. 在种群的进化过程中，如果该个体对应的反向个体的适应值优于该个体的适应值，则反向个体替代该个体，并进入下一代种群. 反向学习策略有助于更快地找到最优解，并加快优化算法的收敛速度.

定义 1　普通反向学习（generalized opposition-based learning，GOBL）. 假设当前个体 $x_i(t)$ 在其反向个体 x_i^* 第 j 维上，可定义为

$$x_i^*(t)=k\left(a_j(t)+b_j(t)\right)-x_{ij}(t) \tag{3}$$

定义 2　改进的反向学习（improved opposition-based learning，IOBL）[14]. 假设当前个体 $x_i(t)=$

$(x_{i1},x_{i2},\cdots,x_{iD})$ 是当前种群中的第 i 个个体，则对应的反向个体 $x_i^*(t)=\left(x_{i1}^*,x_{i2}^*,\cdots,x_{iD}^*\right)$ 可定义为

$$x_{ij}^*(t)=k_1\left(a_j(t)+k_2b_j(t)\right)-x_{ij}(t) \tag{4}$$

其中：x_{ij} 属于 $\left[a_j,b_j\right]$，k、k_1 和 k_2 为 $(0,1)$ 之间的随机数，$\left[a_j,b_j\right]$ 表示个体 x_{ij} 在第 j 维的区间，具体表示为

$$\begin{cases}a_j(t)=\min\left(x_{ij}(t)\right)\\ b_j(t)=\max\left(x_{ij}(t)\right)\end{cases} \tag{5}$$

假如反向个体 x_{ij}^* 不属于 $\left[a_j,b_j\right]$，则更新方式为

$$\begin{cases}x_{ij}^*=2.0\cdot x_{\max}-x_{ij}^*,\text{ if } x_{ij}^*>x_{\max}\\ x_{ij}^*=2.0\cdot x_{\min}-x_{ij}^*,\text{ if } x_{ij}^*<x_{\min}\end{cases} \tag{6}$$

本文对算法改进所选择的反向学习策略为普通反向学习策略.

1.3　差分进化算法

差分进化算法是一种基于种群和定向搜索的优化算法，采用浮点矢量编码生成种群，主要步骤为初始化、变异、交叉和选择. 通过个体间的差异获得变异个体，将变异个体按照确定的交叉策略与初始个体进行基因互换后得到交叉个体，选择适应值更好的个体作为新一代种群中的个体，从而在种群的进化过程中随机搜索，直至找到最优值. DE 有几种最基本的形式，本文所用的是 DE 算法中使用最为广泛的一种“DE/rand/1/bin”.

初始化：在 D 维空间中，假设种群 P 由 N 个个体组成，初始的种群在 $\left[U_j,L_j\right]$ 中产生，即

$$P_{i,j}=L_j+\text{rand}(0,1)\cdot\left(U_j-L_j\right) \tag{7}$$

其中：U_j 为搜索空间的最大值，L_j 为搜索空间的最小值，$\text{rand}(0,1)$ 是在 $(0,1)$ 之间产生的一个服从均匀分布的随机数.

变异：对当前种群中的每个个体 $P_i(G)$ 产生与之对应的变异个体 $V_i(G)$. F 是缩放因子，决定了种群个体差分的步长. F 值较大会增强算法的全局搜索能力，但会降低算法的收敛速度；F 值较小会导致个体间差异较小，从而使得算法易于陷入局部最优. 变异操作可表示为

$$V_i(G)=P_{r_1}(G)+F\cdot\left(P_{r_2}(G)-P_{r_3}(G)\right) \tag{8}$$

其中：G 为迭代次数，r_1、r_2 和 r_3 为集合 $\{1,2,\cdots,N\}$ 中任意互不相等的随机数.

交叉：将种群中个体的部分基因与变异个体的部分基因按照规则交换，以提高种群的多样性. C 为交叉概率，C 值越大，交叉个体 $U_i(G)$ 从变异个体 $V_i(G)$ 中获得的基因越多，从而更接近变异个体，全局收敛的能力越强，但是会降低收敛速度；C 值越小，全局收敛能力会越弱，容易陷入局部最优. 交叉过程可表示为

$$U_i(G)=\begin{cases}V_i(G),\text{if}\left(\text{rand}_j(0,1)\leqslant C\right)\text{or}\left(j=j_{\text{rand}}\right)\\ X_i(G),\text{otherwise}\end{cases} \tag{9}$$

其中：$\text{rand}_j(0,1)$ 为 $(0,1)$ 之间的任意随机数，j_{rand} 为 $[1,D]$ 内的随机整数.

选择：将所需要求解的问题换成适应度函数，越接近目标个体，适应度越小. 将当前种群中个体的适应度 $f\left(X_i(G)\right)$ 与相应的实验个体 $f\left(U_i(G)\right)$ 的适应度相比较，如果实验个体的适应度小于或等于当前个体的适应度，则选择实验个体进入下一代种群. 选择操作可表示为

$$X_i(G+1)=\begin{cases}U_i(G),\text{if } f\left(U_i(G)\right)\leqslant f\left(X_i(G)\right)\\ X_i(G),\text{otherwise}\end{cases} \tag{10}$$

其中 $f(.)$ 为适应度函数，函数的具体形式是由求解的问题决定的.

2　改进算法

差分进化算法的性能与结构参数值 F 和 C 的选择有很大程度上的依赖性，在进化过程中也存在陷入局部最优而导致收敛速度降低的问题. 本文提出一种改进算法为基于反向学习和伯恩斯坦算子的差分进化算法（differential evolution algorithm based on opposition-based learning and Bernstain operator，BODE）. 利用反向学习产生反向群体，扩大了种群的搜索空间，为找到潜在的最优个体提供更多机会，避免算法进化过程中容易陷入局部最优的问题；通过伯恩斯坦多项式随机产生结构参数并控制变异和交叉的过程，避免参数确定导致的试错过程，增强了个体的寻优能力，从而节省了搜索时间.

算法 1：BODE 算法

1. 输入：随机向量 $\boldsymbol{X}_i$.
2. 输出：个体的位置信息.
3. BEGIN.
4. 随机初始化种群.
5. 按照反向学习策略生成反向种群.
6. 计算种群中个体的适应值，选择适应值更好

的个体作为下一代种群.

7. WHILE(终止条件是否满足).

8. 由伯恩斯坦算子控制生成突变控制矩阵 $\boldsymbol{M}$.

9. 由伯恩斯坦算子控制生成进化步长 F.

10. 生成实验个体.

11. 判断实验个体边界.

12. 由反向学习策略实现反向跳转.

13. 更新个体.

14. END WHILE.

15. 得到最优个体.

16. END.

基于反向学习和伯恩斯坦算子的差分进化算法具体流程如下：

(1)在初始化阶段，按照式(7)产生种群 P，同时按照式(11)产生反向种群 P'，在种群 P 和 P'中选择适应值更好的个体组成初始种群.

$$P'_{i,j}=a_j+b_j-P_{i,j} \tag{11}$$

(2)通过式(12)和式(13)生成突变控制矩阵 $\boldsymbol{M}$. $\boldsymbol{M}$ 的初始值由式(12)定义.

$$\boldsymbol{M}_{(i=1:N,\ j=1:D)}=0 \tag{12}$$

$$\boldsymbol{M}_{(i,\,\boldsymbol{u}(1[\rho,D]))}=1 \tag{13}$$

其中：随机数 ρ 的取值由式(2)产生，$\boldsymbol{u}$ 向量为

$$\boldsymbol{u}=\text{permute}(1:D) \tag{14}$$

$\text{permute}(1:D)$ 函数可以随机交换 $(.)$ 中元素的序列.

(3)在 BODE 中的进化步长 F 定义为

$$F=\begin{cases}\left[\eta^3_{(1,1:D)}.*\left|\lambda^3_{(1,1:D)}\right|\right]'\cdot\boldsymbol{Q}(1,1:N),\ \text{if } k_2<k_3\\ \lambda^3_{(N,1)}\cdot\boldsymbol{Q}_{(1,D)},\ \text{otherwise}\end{cases} \tag{15}$$

其中：k_2、k_3、η 和 λ 是在每次调用时产生的随机值，矩阵 $\boldsymbol{Q}$ 是全 1 矩阵.

(4)在 BODE 中，实验个体的生成阶段是一个随机交叉的过程，按照式(16)生成实验个体.

$$T=P+F.*\boldsymbol{M}.*\left(\left(\boldsymbol{\omega}^*\right)^3.*\boldsymbol{E}+\left(1-\left(\boldsymbol{\omega}^*\right)^3.*P_{\text{best}}-P\right)\right) \tag{16}$$

其中：P_{best} 为最佳种群；$\boldsymbol{E}=\boldsymbol{\omega}\cdot P_{\boldsymbol{L}_1}+(1-\boldsymbol{\omega})\cdot P_{\boldsymbol{L}_2}$；$\boldsymbol{\omega}$ 是服从正态分布的 $N\times D$ 矩阵；$\boldsymbol{L}_1=\text{permute}(1:N)$，$\boldsymbol{L}_2=\text{permute}(1:N)$，且 $\boldsymbol{L}_1\neq\boldsymbol{L}_2$.

(5)如果实验个体超过了搜索范围，即 $T_{i,j}<L_j$ 或者 $T_{i,j}>U_j$，则个体采用式(17)进行更新，其中 δ 为 $(0,1)$ 之间的随机数.

$$T_{i,j}=L_j+\delta\cdot\left(U_j-L_j\right) \tag{17}$$

(6)按照式(18)生成反向跳转的个体，$W_{\min}$ 和 $W_{\max}$ 分别为当前种群中个体的最小值和最大值.

$$P'_{i,j}=W_{\min}+W_{\max}-P_{i,j} \tag{18}$$

(7)分别求出实验个体、反向实验个体和原个体的适应度，按照式(10)选择适应度更好的个体组成新一代种群.

算法 2：反向学习策略

1. BEGIN.

2. 生成初始种群 P.

3. for(int $i=0$; $i<N$; i++).

4. for(int $j=0$; $j<n$; j++).

5. $P'_{i,j}=a_j+b_j-P_{i,j}$.

6. 计算 $P'_{i,j}$ 的适应度.

7. 在$\{P,P'\}$选择更优值输出.

8. END.

算法 3：伯恩斯坦算子

1. BEGIN.

2. 根据式(12)—式(14)生成突变控制矩阵 $\boldsymbol{M}$.

3. 根据式(15)生成进化步长 F.

4. 根据式(16)生成实验个体 T.

5. 由式(17)判断实验个体 T 的边界.

6. END.

3　实验仿真与结果分析

为了验证改进后算法的性能，选择了 6 种比较经典的测试函数，主要分为两大类，其中 f_1—f_3 为单峰函数，f_4—f_6 为多峰函数，见表 1.

3.1　实验环境与参数设置

实验环境：Windows 10 操作系统，运行内存为 8 GB，运行软件为 MATLAB 2017a.

基本参数设置：种群大小 $N=48$，种群个体的维度 $D=30$，最大迭代次数为 5 000 次，各算法独立运行次数为 20 次，缩放因子 $F=0.5$，交叉概率 $C=0.9$.

3.2　仿真实验结果

BODE 算法与 DE 算法、粒子群优化(particle swarm optimization，PSO)算法及 BSD 算法的比较结果见表 2，其中 PSO 算法的实验结果来源于文献[15].

表 1　6种测试函数

Tab. 1　Six benchmark functions

函数	测试函数	D	S	$f_{\min}$
f_1	$f(x)=\sum_{i=1}^{D}x_i^2$	30	$[-100,100]^D$	0
f_2	$f(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]^D$	0
f_3	$f(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^D$	0
f_4	$f(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	$[-5.12,5.12]^D$	0
f_5	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+\mathrm{e}$	30	$[-32,32]^D$	0
f_6	$f(x)=1/4000\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$	30	$[-600,600]^D$	0

表 2　4种算法在 6 个测试函数的实验结果

Tab. 2　Experimental results of four algorithms on six benchmark functions

函数	DE		PSO		BSD		BODE	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
f_1	5.69×10^{-17}	8.83×10^{-18}	3.69×10^{-37}	3.69×10^{-37}	2.35×10^{-42}	1.01×10^{-41}	2.23×10^{-43}	3.27×10^{-43}
f_2	7.26×10^{-19}	9.56×10^{-19}	2.92×10^{-24}	1.14×10^{-23}	4.02×10^{-41}	8.13×10^{-43}	3.53×10^{-45}	5.35×10^{-45}
f_3	5.62×10^{-11}	8.55×10^{-11}	1.20×10^{-3}	2.11×10^{-3}	3.80×10^{-3}	3.20×10^{-3}	6.31×10^{-14}	2.41×10^{-14}
f_4	42.90	12.40	20.80	5.94	1.67×10^{-2}	7.26×10^{-2}	3.93×10^{-4}	3.25×10^{-4}
f_5	6.45×10^{-6}	3.31×10^{-6}	1.34×10^{-3}	4.24×10^{-2}	1.15×10^{-14}	7.90×10^{-14}	9.75×10^{-15}	2.40×10^{-15}
f_6	4.02×10^{-2}	2.53×10^{-2}	2.32×10^{-1}	4.43×10^{-1}	2.61×10^{-2}	8.65×10^{-4}	0.00	0.00

由表 2 可知，BODE 算法不仅在单峰函数 f_1—f_3 中有绝对优势，在多峰函数 f_4—f_6 中也有明显优势. 在单峰函数 f_1 和 f_2 中，BODE 算法的收敛精度明显高于 DE 算法和 PSO 算法的；在函数 f_3 中，BODE 算法的收敛精度远高于 PSO 算法和 BSD 算法的. BODE 算法在多峰函数 f_6 中迭代到 5 000 次时，已经达到了收敛的最优值 0. 由此可知，加入反向学习策略和伯恩斯坦算子对增强算法的寻优能力以及提高算法的收敛精度有着重要作用.

测试函数结果对比如图 2 所示.

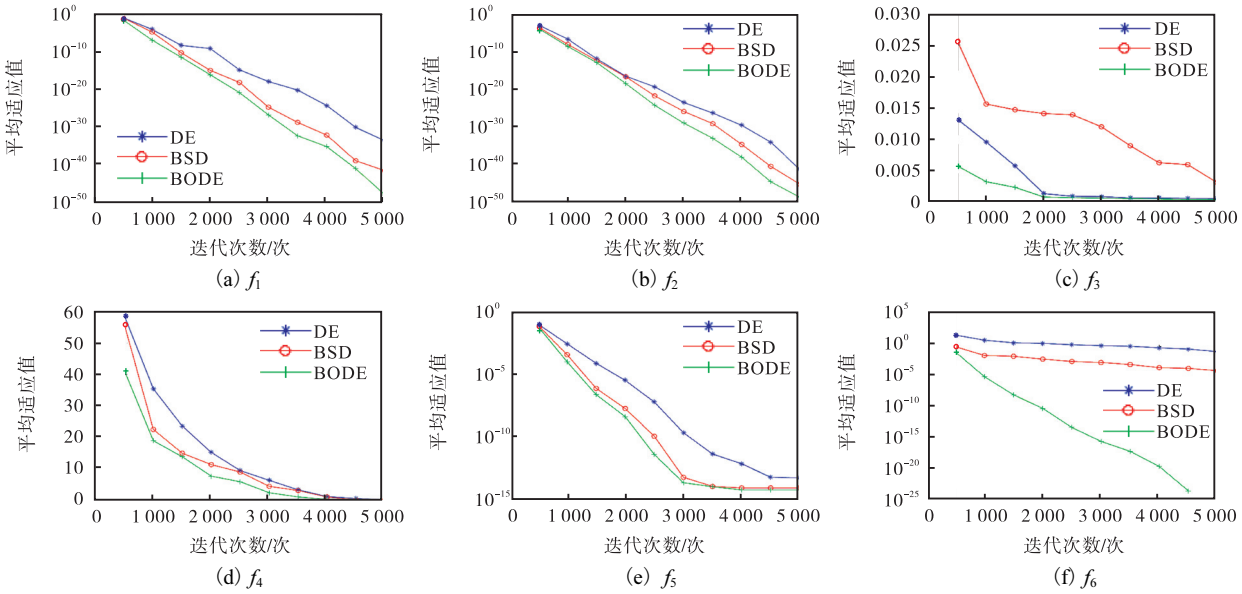

图 2　测试函数结果对比

Fig. 2　Comparison of benchmark functions

由图 2 可知，本文的 BODE 算法在收敛进度上与其他算法大致相同，但 BODE 算法的反向学习策略和伯恩斯坦算子可以加快收敛速度并提高收敛精度，有效地防止了算法的早熟现象，能够更快找到最优值. BODE 算法在处理单峰测试函数时，收敛精度和收敛速度优势明显. 虽然 BODE 算法在函数 f_3 中没有表现优秀，但在整体上对算法的收敛性能有一定的帮助. BODE 算法在测试多峰函数时，能够解决算法在多峰函数中易于陷入局部最优的问题，帮助其快速找到全局最优值，且在收敛速度和精度上也表现突出. 对易于使算法陷入局部最优的函数 f_6 中，BODE 算法表现出了较强的搜索能力.

4　结　语

为了提升 DE 算法的性能，本文对算法中结构参数以及搜索策略进行改进，解决算法过早陷入局部最优的问题，以达到提高算法的收敛精度和速度、减少计算消耗的效果. 通过采用反向学习策略增加种群的多样性，扩大搜索范围；而伯恩斯坦算子控制进化过程中的变异和交叉阶段，避免了参数的设定，增强了算法的搜索能力，提高了算法的收敛精度，使得算法更加高效. 通过对国际标准测试函数的对比实验结果表明：BODE 算法与其他算法相比有着更好的整体收敛性能，在进化阶段可以跳出局部最优，收敛速度更快，精度更高. 测试函数对比图进一步说明了本文算法在收敛速度和收敛精度方面有更好的表现. 后续工作可以在进化过程中加入更加高效的优化策略以及验证和分析改进后算法的性能这两方面进行，进一步提高算法的可靠性.

参考文献：

[1] STORN R，PRICE K. Differential evolution：a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization，1997，11(4)：341–359.

[2] STORN R. System design by constraint adaptation and differential evolution[J]. IEEE Transactions on evolutionary computation，1999，3(1)：22–34.

[3] MAGALH C S D，DOS C H，ALMEIDA D M，et al. Improving differential evolution accuracy for flexible ligand docking using a multi-solution strategy[C]// Proceedings of the 13th International Conference on Intelligent Data Engineering and Automated Learning. Berlin：Springer，2012.

[4] WANG Y，CAI Z，ZHANG Q. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Transactions on evolutionary computation，2011，15(1)：55–66.

[5] WANG Y，CAI Z，ZHANG Q. Enhancing the search ability of differential evolution through orthogonal crossover[J]. Information sciences，2012，185(1)：153–177.

[6] ZHANG J，SANDERSON A C. JADE：adaptive differential evolution with optional external archive[J]. IEEE Transactions on evolutionary computation，2009，13(5)：945–958.

[7] CIVICIOGLU P，BESDOK E. Bernstain-search differential evolution algorithm for numerical function optimization[J]. Expert systems with applications，2019，138：112831.

[8] ZHU Q，LIN Q，DU Z，et al. A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm[J]. Information sciences，2016，345：177–198.

[9] 童旅杨，董明刚，敬超. 基于分解和多策略变异的多目标差分进化算法[J]. 计算机应用研究，2019，36(7)：1955–1959.

[10] ZHANG K，YU Y. An enhancing differential evolution algorithm with a rank-up selection：RUSDE[J]. Mathematics，2021，9(5)：569.

[11] 钱峥远，曾国荪. 精英化岛屿种群引导的差分进化算法[J/OL]. 计算机工程与应用：1–11. [2021–03–25]. http://kns.cnki.net/kcms/detail/11.2127.TP.20210223.1333.012.html.

[12] FATEMEH A，AMIN H，ZHAO X L. On the use of Bernstain-bézier functions for modelling the post-fire stress-strain relationship of ultra-high strength steel (grade 1200)[J]. Engineering structures，2018，175：605–616.

[13] TIZHOOSH H R. Opposition-based learning：a new scheme for machine intelligence[C]//International Conference on Computational Intelligence for Modelling，Control and Automation and International Conference on Intelligent Agents，Web Technologies and Internet Commerce(CIMCA-IAWTIC'06). New York：IEEE，2005.

[14] 朱德刚，洪建，张洁. 基于反向学习和高斯扰动的粒子群优化算法[J]. 计算机与数字工程，2019，47(12)：2993–2998.

[15] HE S，WU Q H，SAUNDERS J R. Group search optimizer：an optimization algorithm inspired by animal searching behavior[J]. IEEE Transactions on evolutionary computation，2009，13(5)：973–990.

责任编辑：郎婧