一种高效动态自适应差分进化算法

肖 鹏 邹德旋 张 强
(江苏师范大学电气工程及自动化学院 江苏 徐州 221116)

摘 要 针对差分进化算法易早熟收敛、收敛精度低等特点,文中提出一种高效动态自适应差分进化(EDSDE)算法。该算法从变异因子、变异策略以及交叉因子方面入手,将变异因子设置成线性递减函数,在基向量前加入幅值系数以平衡全局搜索和局部搜索,将交叉因子设置成在[0,1]内不断震荡且每隔50代更新一次的动态自适应函数。仿真实验结果表明,EDSDE能获得更好的优化结果,且比其他算法具有更好的优化性能。

关键词 早熟收敛,动态自适应差分进化算法,线性递减函数,幅值系数,动态自适应函数
中图法分类号 TP301 **文献标识码** A

Efficient Dynamic Self-adaptive Differential Evolution Algorithm

XIAO Peng ZOU De-xuan ZHANG Qiang
(School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China)

Abstract This paper proposed an efficient dynamic self-adaptive differential evolution (EDSDE) algorithm based on the characteristics of premature convergence, low convergence accuracy. The algorithm starts with mutation factor, mutation strategy and crossover factor. It sets the mutation factor to a linear decreasing function, incorporates an amplitude coefficient into the base vector to balance the global search and the local search, and sets the crossover factor to a dynamic self-adaptive function that is constantly oscillated within [0,1] and updated every 50 generations. The simulation results show that EDSDE can obtain better optimization results and exhibit more desirable performance then the other algorithms.

Keywords Premature convergence, Dynamic self-adaptive differential evolution algorithm, Linear decreasing function, Amplitude coefficient, Dynamic self-adaptive function

1 概述

差分进化(Differential Evolution, DE)算法[1]是美国学者Storn和Price为求解切比雪夫(Chebyshev)不等式[2]而提出的一种群体智能优化算法,和其他智能算法一样,DE算法也是一种模拟生物进化过程的随机模型,通过不断循环迭代,使那些适应环境的优等个体被保存下来。相较于其他智能算法,DE算法保留了基于种群的全局搜索模式[3],采用变异、交叉、选择[4]的进化策略,降低了进化操作的复杂性。目前,DE算法在很多领域得到了广泛应用,如电力系统调度[5-6]、边坡滑面搜索[7]、车间调度[8-9]、协同干扰资源分配[10]、虹膜定位[11]、神经网络优化[12]等。但是DE算法在进化的后期,由于种群个体之间差异化信息程度的降低,容易出现早熟收敛以及搜索停滞等方面的缺陷[13]。

针对这些缺陷,很多学者提出了大量的改进策略,刘龙龙等[14]提出了一种新的改进差分进化算法(NIDE),通过改进变异因子、变异策略以及交叉因子的思想提高算法收敛速度;欧阳海滨等[15]提出了随机变异差分进化算法(RMDE),采用随机选择的方式进行变异和扰动,增加种群的多样性,以此来平衡算法的局部搜索和全局搜索;Brest等[16]提出了自适应控制参数的差分进化算法—数值基准问题的比较研究(JDE);Zhang等[17]提出了可选外部存档的自适应差分进化算法(JADE);Islam等[18]提出了一种适用于全局数值优化的新型变异和交叉策略的自适应差分进化算法(MDEpBX)。

本文在以上几种改进算法的基础上,提出了一种高效动态自适应差分进化算法(EDSDE),并选取几种典型的函数进行测试。测试结果表明,EDSDE算法具有很强的收敛性能,且不易陷入局部最优。

2 标准的差分进化算法

标准差分进化(DE)算法的基本原理是从任意一个随机产生的初始种群开始,通过变异、交叉、选择操作,不断进行迭代更新,淘汰劣等个体,保存优等个体,并引导搜索结果最终靠近全局最优解。标准DE算法的基本流程如下。

本文受国家自然科学基金青年基金(61403174)资助。

肖 鹏(1989—),男,硕士生,主要研究方向为群智能算法,E-mail:876601620@qq.com;邹德旋(1982—),男,博士生,副教授,主要研究方向为群智能算法,E-mail:472101976@qq.com;张 强(1980—),男,硕士生,主要研究方向为群智能算法。

(1)初始化种群

初始化阶段,首先确定一系列基本参数,包括空间维数 N、种群大小 NP、迭代次数 NI、变异因子 F、交叉因子 CR、搜索空间的下限 $x_{j,i}^{\min}$ 以及搜索空间的上限 $x_{j,i}^{\max}$。随后初始种群在 $[x_{j,i}^{\min},x_{j,i}^{\max}]$($i=1,2,\cdots,NP$;$j=1,2,\cdots,N$)中随机产生,具体表达式为:

$$x_{j,i}^{0}=x_{j,i}^{\min}+rand(0,1)\cdot(x_{j,i}^{\max}-x_{j,i}^{\min}) \tag{1}$$

其中,$x_{j,i}^{0}$ 表示第 0 代的第 i 个体的第 j 维,$rand(0,1)$ 表示(0,1)区间内服从均匀分布的随机数。

(2)变异操作

变异操作的基本原理是将差分向量加权后与基向量求和从而产生变异向量,在变异操作中最常用的策略是 $DE/rand/1/bin$,具体表达式为:

$$v_i^{g+1}=x_{r_3}^{g}+F\cdot(x_{r_1}^{g}-x_{r_2}^{g}),i\neq r_1\neq r_2\neq r_3 \tag{2}$$

其中,i,r_1,r_2,r_3 为两两互不相同的随机整数,$x_{r_1}^{g}$ 表示第 r_1 个个体的第 g 代。

(3)交叉操作

交叉操作的基本原理是变异向量与某个预先决定的目标向量进行参数混合后生成试验向量,具体表达式为:

$$u_{j,i}^{g+1}=\begin{cases}v_{j,i}^{g+1}, & \text{if } rand(0,1)\leqslant CR \text{ or } j=j_{rand}\\ x_{j,i}^{g}, & \text{otherwise}\end{cases} \tag{3}$$

其中,$j_{rand}\in[1,2,\cdots,N]$,且是一个随机整数。

(4)选择操作

选择操作的基本原理是如果试验向量的适应度值优于目标向量的适应度值,则在下一代中用试验向量取代目标向量,否则目标向量仍保存下来,即采用贪婪策略选择适应度较好的进入下一代,具体表达式为:

$$x_i^{g+1}=\begin{cases}u_i^{g+1}, & \text{if } f(u_i^{g+1})\leqslant f(x_i^{g})\\ x_i^{g}, & \text{otherwise}\end{cases} \tag{4}$$

3　高效动态自适应差分进化算法

高效动态自适应差分进化算法(EDSDE)主要从变异因子 F 的取值、交叉因子 CR 的取值以及变异策略的改进上对 DE 算法进行优化。

3.1　变异因子 F 的优化

由式(2)可知,变异因子 F 的大小控制着差分向量的幅值大小,它的取值影响着收敛性与收敛速度。当 F 较小时,收敛速度较快,但容易收敛到局部最优解;当 F 较大时,收敛速度较慢,但有利于收敛到全局最优解。基于以上原理,本文在 EDSDE 算法中提出了一种新的动态自适应变异因子,具体表达式为:

$$F=F_{\max}-(F_{\max}-F_{\min})\cdot(ni/NI) \tag{5}$$

其中,ni 表示第 ni 代,$F_{\max}$ 表示变异因子的最大值,$F_{\min}$ 表示变异因子的最小值,本文取 $F_{\max}=0.99$,$F_{\min}=0.2$。由式(5)可知,F 呈线性递减趋势。

3.2　变异策略的优化

变异策略是 DE 算法的关键步骤,它是决定全局寻优与局部寻优的重要因素,不同的变异策略往往对搜索结果产生巨大的影响。式(2)为最基本也是最常用的变异策略,但在搜索后期,它往往会陷入局部最优,使得收敛速度大幅降低,容易出现早熟收敛的现象。EDSDE 算法在式(2)的基础上,提出一种新的变异策略,具体表达式为:

$$v_i^{g+1}=(1-F)\cdot x_{r_3}^{g}+F\cdot(x_{r_1}^{g}-x_{r_2}^{g}),i\neq r_1\neq r_2\neq r_3 \tag{6}$$

由于变异因子 F 呈线性递减变化趋势,故 $1-F$ 呈线性递增变化趋势。根据式(6)可知,在寻优初期,基向量 $x_{r_3}^{g}$ 的幅值系数较小,差分向量 $x_{r_1}^{g}-x_{r_2}^{g}$ 的幅值系数较大,而到了寻优后期,基向量 $x_{r_3}^{g}$ 的幅值系数较大,差分向量 $x_{r_1}^{g}-x_{r_2}^{g}$ 的幅值系数较小,这样可以兼顾全局搜索能力与局部开发能力。

3.3　交叉因子 CR 的优化

由式(3)可知,交叉因子 CR 的大小决定着新个体从变异个体或父代个体那里继承基因的概率。当 CR 较大时,新个体继承变异个体基因的概率更大;当 CR 较小时,新个体继承父代个体基因的概率更大。DE 算法中往往将 CR 的值取为定值,这在很大程度上会使算法陷入局部最优,影响寻优的性能。针对这一特点,本文在 EDSDE 算法中提出了一种新的动态自适应交叉因子,其产生的具体流程图如图 1 所示。

```
a=(1+cos(1))/2
for ni=1:NI
  if mod(ni,50)==0
    CR(ni)=(1+cos(ni))/2
    a=CR+(ni)
  else
    CR(ni)=a
  end
end
```

图 1　CR 产生的流程

由图 1 可知,EDSDE 算法每迭代 50 次更新一次 CR 的值,且 CR 在[0,1]区间内震荡分布,CR 取值的不断变化使得新个体能够随机继承变异个体或父代个体的基因,跳出局部最优解。

交叉因子 CR 的仿真曲线如图 2 所示。

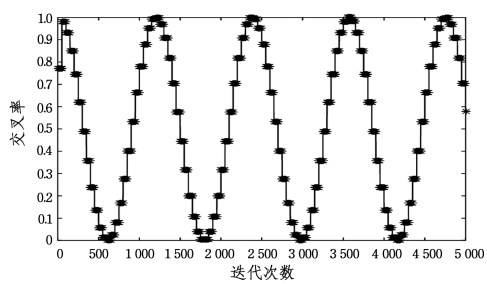

图 2　CR 变化曲线图

4　实验仿真与结果分析

为了验证 EDSDE 算法的有效性,本文从文献[19-20]中选取了 10 个标准测试函数进行测试,并与 DE/rand/1/bin、NIDE 算法、RMDE 算法、JDE 算法、JADE 算法、MDEpBX 算法进行比较,10 个标准测试函数如表 1 所列。

表 1 标准测试函数

函数名	表达式	取值范围
Sphere	$f_1(x)=\sum_{i=1}^{N}x_i^2$	$-100\leqslant x_i\leqslant 100$
Schwefel's problem 2.22	$f_2(x)=\sum_{i=1}^{N}\lvert x_i\rvert+\prod_{i=1}^{N}\lvert x_i\rvert$	$-10\leqslant x_i\leqslant 10$
Step	$f_3(x)=\sum_{i=1}^{N}(\lfloor x_i+0.5\rfloor)^2$	$-10\leqslant x_i\leqslant 10$
Griewangk's	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}(\frac{x_i}{\sqrt{i}})+1$	$-600\leqslant x_i\leqslant 600$
Rastrigrin	$f_5(x)=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i)+10)$	$-5.12\leqslant x_i\leqslant 5.12$
Alpine	$f_6(x)=\sum_{i=1}^{N}\lvert x_i\sin x_i+0.1x_i\rvert$	$-10\leqslant x_i\leqslant 10$
Quartic	$f_7(x)=\sum_{i=1}^{N}ix_i^4+random[0,1]$	$-1.28\leqslant x_i\leqslant 1.28$
Ackley's problem	$f_8(x)=-20e^{-0.02\sqrt{N^{-1}\sum_{i=1}^{N}x_i^2}}-e^{N^{-1}\sum_{i=1}^{N}\cos(2\pi x_i)}+20+e$	$-30\leqslant x_i\leqslant 30$
Salomon problem	$f_9(x)=1-\cos(2\pi\sqrt{\sum_{i=1}^{N}x_i^2})+0.1\sqrt{\sum_{i=1}^{N}x_i^2}$	$-100\leqslant x_i\leqslant 100$
Schaffer's	$f_{10}=0.5+\frac{\sin^2\sqrt{(x_1^2+x_2^2)^2}-0.5}{1+0.01(x_1^2+x_2^2)^2}$	$-10\leqslant x_i\leqslant 10$

仿真实验使用 Matlab R2016a 软件来编程实现，且实验配置为 Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz。算法的基本参数设置如下：维数 $N=50$ 维、10 维以及 2 维，种群大小 $NP=50$；DE/rand/1/bin 中的变异因子 $F=0.5$，交叉因子 $CR=0.9$；NIDE 算法中的参数设置见文献[14]；RMDE 算法中的参数设置见文献[15]；JDE 算法中的参数设置见文献[16]；JADE 算法中的参数设置见文献[17]；MDEpBX 算法中的参数设置见文献[18]；EDSDE 算法中的 $F_{max}=0.99$，$F_{min}=0.2$，CR 取值在[0,1]之间，且每迭代 50 次更新一次，其余参数值均与上述 3 种算法一致。表 2、表 3、表 4 分别列出 50 维、100 维与 2 维时的实验仿真数据。表中主要参数包括：最优值 best、平均值 mean、标准差 std、中值 median、平均时间以及胜率，这些参数均通过运行 30 次取平均值得出。

表 2 测试函数的优化结果(50 维)

函数	最大迭代次数	算法	最小值 min	平均值 mean	标准差 std	中值 median	平均时间/s	胜率/%
f_1	2500	DE/rand/1/bin	5.24×10^{-20}	5.24×10^{-20}	0.00	5.24×10^{-20}	1.48	100
		NIDE	1.87×10^{-20}	4.05×10^{-20}	2.12×10^{-20}	4.20×10^{-20}	1.55	100
		RMDE	4.21×10^{-5}	5.48×10^{-4}	6.08×10^{-4}	3.80×10^{-4}	1.37	100
		JDE	2.30×10^{-21}	2.72×10^{-20}	1.05×10^{-20}	2.88×10^{-21}	1.47	100
		JADE	5.81×10^{-60}	1.04×10^{-47}	1.81×10^{-47}	9.64×10^{-60}	7.09	100
		MDEpBX	1.86×10^{-13}	1.86×10^{-13}	1.01×10^{-12}	1.86×10^{-13}	2.87	100
		EDSDE	0.00	0.00	0.00	0.00	1.35	—
f_2	2500	DE/rand/1/bin	2.11×10^{-12}	2.11×10^{-12}	0.00	2.11×10^{-12}	1.77	100
		NIDE	6.57×10^{-13}	9.19×10^{-13}	2.31×10^{-13}	1.00×10^{-12}	1.85	100
		RMDE	2.60×10^{-5}	0.82×10^{-2}	1.39×10^{-2}	2.63×10^{-4}	1.84	100
		JDE	3.41×10^{-14}	1.17×10^{-10}	1.95×10^{-10}	9.43×10^{-12}	1.48	100
		JADE	3.24×10^{-32}	2.08×10^{-28}	3.60×10^{-28}	1.23×10^{-31}	6.72	100
		MDEpBX	6.04×10^{-18}	6.04×10^{-18}	2.30×10^{-17}	6.04×10^{-18}	3.07	100
		EDSDE	0.00	0.00	0.00	0.00	1.61	—
f_3	5000	DE/rand/1/bin	0.00	0.00	0.00	0.00	3.54	100
		NIDE	0.00	0.00	0.00	0.00	3.65	100
		RMDE	0.00	0.00	0.00	0.00	3.29	100
		JDE	0.00	0.00	0.00	0.00	3.10	100
		JADE	0.00	0.00	0.00	0.00	15.08	100
		MDEpBX	5.23	5.23	25.3	5.23	6.32	100
		EDSDE	0.00	0.00	0.00	0.00	3.34	—
f_4	5000	DE/rand/1/bin	3.19×10^{-2}	3.19×10^{-2}	0.00	3.19×10^{-2}	3.72	100
		NIDE	0.00	0.00	0.00	0.00	3.93	100
		RMDE	7.12×10^{-9}	2.12×10^{-2}	3.05×10^{-2}	0.74×10^{-2}	3.50	100
		JDE	0.00	0.00	0.00	0.00	3.38	100
		JADE	0.00	0.00	0.43×10^{-2}	0.00	15.92	100
		MDEpBX	1.06×10^{-2}	1.06×10^{-2}	4.38×10^{-2}	1.06×10^{-2}	6.61	100
		EDSDE	0.00	0.00	0.00	0.00	3.42	—
f_5	5000	DE/rand/1/bin	41.79	41.79	0.00	41.79	3.63	100
		NIDE	2.00	4.47	0.81	4.47	3.87	100
		RMDE	3.06×10^{-13}	5.92×10^{-11}	5.39×10^{-12}	5.92×10^{-11}	3.31	100
		JDE	19.86	32.35	15.18	27.94	3.84	100
		JADE	0.00	0.00	0.00	0.00	14.92	100
		MDEpBX	3.88	3.88	11.95	3.88	6.87	100
		EDSDE	0.00	0.00	0.00	0.00	3.26	—

(续表)

函数	最大迭代次数	算法	最小值 min	平均值 mean	标准差 std	中值 median	平均时间/s	胜率/%
		DE/rand/1/bin	0.00	0.00	0.00	0.00	1.08	100
		NIDE	1.13×10^{-10}	4.37×10^{-10}	4.57×10^{-10}	4.37×10^{-10}	1.19	100
		RMDE	0.00	0.00	0.00	0.00	1.03	100
f_6	1500	JDE	0.34×10^{-2}	0.95×10^{-2}	0.60×10^{-2}	0.97×10^{-2}	1.17	100
		JADE	9.22×10^{-4}	10.56×10^{-4}	8.37×10^{-4}	10.38×10^{-4}	4.32	100
		MDEpBX	4.58×10^{-7}	4.58×10^{-7}	2.44×10^{-6}	4.58×10^{-7}	2.04	100
		EDSDE	0.00	0.00	0.00	0.00	0.95	—
		DE/rand/1/bin	1.17×10^{-2}	1.17×10^{-2}	0.00	1.17×10^{-2}	4.78	100
		NIDE	0.36×10^{-2}	0.40×10^{-2}	5.65×10^{-4}	0.40×10^{-2}	5.15	100
		RMDE	0.31×10^{-2}	0.34×10^{-2}	4.18×10^{-4}	0.34×10^{-2}	4.87	100
f_7	5000	JDE	0.85×10^{-2}	0.85×10^{-2}	1.04×10^{-4}	0.85×10^{-2}	5.17	100
		JADE	0.76×10^{-1}	0.76×10^{-1}	0.56×10^{-2}	0.76×10^{-1}	18.30	100
		MDEpBX	0.26×10^{-2}	0.26×10^{-2}	0.86×10^{-2}	0.26×10^{-2}	9.72	100
		EDSDE	1.12×10^{-6}	2.11×10^{-6}	1.40×10^{-6}	2.11×10^{-6}	4.82	—
		DE/rand/1/bin	5.98	5.98	0.00	5.98	0.12	100
		NIDE	3.91	4.60	0.67	4.62	0.09	100
		RMDE	1.22	2.04	0.73	2.28	0.08	100
f_8	100	JDE	2.63	3.18	0.57	3.18	0.08	100
		JADE	1.83	1.98	0.15	1.98	0.31	100
		MDEpBX	0.20	0.20	0.58	0.20	0.17	100
		EDSDE	8.88×10^{-16}	8.88×10^{-16}	0.00	8.88×10^{-16}	0.07	—
		DE/rand/1/bin	0.60	0.60	0.00	0.60	0.64	100
		NIDE	0.60	0.63	0.06	0.60	0.64	100
		RMDE	0.40	0.53	0.15	0.50	0.60	100
f_9	1000	JDE	0.50	0.73	0.80	0.61	0.53	100
		JADE	0.52	0.68	0.72	0.69	3.06	100
		MDEpBX	0.57	0.62	0.61	0.65	1.33	100
		EDSDE	0.00	0.00	0.00	0.00	0.57	—

表 3　测试函数的优化结果(100 维)

函数	最大迭代次数	算法	最小值 min	平均值 mean	标准差 std	中值 median	平均时间/s	胜率/%
		DE/rand/1/bin	0.21×10^{-2}	0.21×10^{-2}	0.00	0.21×10^{-2}	2.16	100
		NIDE	1.94×10^{-12}	3.80×10^{-20}	2.94×10^{-12}	2.26×10^{-12}	2.45	100
		RMDE	0.36×10^{-2}	3.66×10^{-2}	4.03×10^{-2}	2.46×10^{-2}	2.15	100
f_1	2500	JDE	5.18×10^{-9}	1.70×10^{-8}	1.78×10^{-8}	8.40×10^{-9}	2.30	100
		JADE	9.38×10^{-24}	4.15×10^{-23}	5.06×10^{-23}	1.52×10^{-23}	10.25	100
		MDEpBX	2.27×10^{-4}	2.27×10^{-4}	0.10×10^{-2}	2.27×10^{-4}	4.13	100
		EDSDE	0.00	0.00	0.00	0.00	2.19	—
		DE/rand/1/bin	7.70×10^{-8}	8.24×10^{-4}	0.00	8.24×10^{-4}	2.47	100
		NIDE	7.49×10^{-2}	8.01×10^{-8}	3.13×10^{-9}	7.98×10^{-8}	2.71	100
		RMDE	4.90×10^{-7}	17.95×10^{-2}	9.11×10^{-2}	22.20×10^{-2}	2.50	100
f_2	2500	JDE	1.27×10^{-12}	1.81×10^{-6}	1.42×10^{-6}	1.63×10^{-6}	2.46	100
		JADE	0.02	1.03×10^{-9}	1.75×10^{-9}	4.04×10^{-11}	10.03	100
		MDEpBX	0.00	0.02	0.11	0.02	4.25	100
		EDSDE	0.00	0.00	0.00	0.00	2.49	—
		DE/rand/1/bin	0.00	0.00	0.00	0.00	4.85	100
		NIDE	0.00	0.00	0.00	0.00	5.25	100
		RMDE	0.00	0.00	0.00	0.00	4.79	100
f_3	5000	JDE	0.00	0.00	0.00	0.00	4.95	100
		JADE	0.00	0.00	0.00	0.00	26.35	100
		MDEpBX	37.23	37.23	147.34	37.23	8.68	100
		EDSDE	0.00	0.00	0.00	0.00	5.23	—
		DE/rand/1/bin	0.99×10^{-2}	0.99×10^{-2}	0.00	0.99×10^{-2}	5.87	100
		NIDE	0.00	0.00	0.00	0.00	7.02	100
		RMDE	6.42×10^{-4}	3.51×10^{-2}	5.95×10^{-2}	9.08×10^{-4}	5.70	100
f_4	5000	JDE	0.00	0.00	0.00	0.00	6.04	100
		JADE	1.32×10^{-16}	1.06×10^{-15}	1.84×10^{-15}	1.22×10^{-15}	21.76	100
		MDEpBX	1.26×10^{-2}	1.26×10^{-2}	4.34×10^{-2}	1.26×10^{-2}	9.73	100
		EDSDE	0.00	0.00	0.00	0.00	5.77	—
		DE/rand/1/bin	112.43	112.43	0.00	112.43	6.13	100
		NIDE	60.90	61.37	0.65	61.37	6.56	100
		RMDE	4.00×10^{-4}	1.37×10^{-2}	0.02	1.37×10^{-2}	5.52	100
f_5	5000	JDE	226.28	316.89	84.81	330.04	6.45	100
		JADE	0.00	0.00	0.00	0.00	22.51	100
		MDEpBX	11.87	11.87	36.32	11.87	9.94	100
		EDSDE	0.00	0.00	0.00	0.00	5.53	—

（续表）

函数	最大迭代次数	算法	最小值 min	平均值 mean	标准差 std	中值 median	平均时间/s	胜率/%
f_6	1500	DE/rand/1/bin	0.03	0.03	0.00	0.03	1.91	100
		NIDE	0.03	3.35×10^{-2}	0.43×10^{-2}	3.35×10^{-2}	1.91	100
		RMDE	0.16×10^{-2}	2.74×10^{-2}	3.65×10^{-2}	2.74×10^{-2}	1.70	100
		JDE	9.39×10^{-2}	0.13	2.83×10^{-2}	0.14	1.77	100
		JADE	3.69×10^{-2}	5.07×10^{-2}	1.58×10^{-2}	4.72×10^{-2}	5.87	100
		MDEpBX	8.71×10^{-4}	8.71×10^{-4}	0.32×10^{-2}	8.71×10^{-4}	2.87	100
		EDSDE	0.00	0.00	0.00	0.00	1.85	—
f_7	5000	DE/rand/1/bin	8.05×10^{-2}	8.05×10^{-2}	0.00	8.05×10^{-2}	8.72	100
		NIDE	0.96×10^{-2}	0.98×10^{-2}	2.55×10^{-4}	0.98×10^{-2}	9.81	100
		RMDE	0.12×10^{-2}	0.37×10^{-2}	0.36×10^{-2}	0.37×10^{-2}	8.81	100
		JDE	3.36×10^{-2}	5.08×10^{-2}	2.90×10^{-2}	3.45×10^{-2}	8.96	100
		JADE	0.26×10^{-1}	0.26×10^{-1}	3.57×10^{-2}	0.26×10^{-1}	24.79	100
		MDEpBX	3.26×10^{-1}	3.26×10^{-2}	0.10	3.26×10^{-1}	13.73	100
		EDSDE	3.64×10^{-6}	3.94×10^{-6}	4.28×10^{-7}	3.94×10^{-6}	9.03	—
f_8	100	DE/rand/1/bin	8.11	8.11	0.00	8.11	0.14	100
		NIDE	6.39	6.56	0.16	6.57	0.14	100
		RMDE	1.05	1.44	0.60	1.15	0.15	100
		JDE	4.25	4.87	1.03	4.30	0.13	100
		JADE	2.66	2.81	0.14	2.82	0.41	100
		MDEpBX	0.25	0.25	0.76	0.25	0.23	100
		EDSDE	8.88×10^{-16}	8.88×10^{-16}	0.00	8.88×10^{-16}	0.12	—
f_9	1000	DE/rand/1/bin	4.25	4.25	0.00	4.25	0.95	100
		NIDE	1.70	1.73	0.06	1.70	0.98	100
		RMDE	0.30	0.73	0.59	0.50	0.96	100
		JDE	2.80	3.19	0.52	2.99	0.81	100
		JADE	0.47	0.51	0.63	0.49	4.22	100
		MDEpBX	0.53	0.53	1.67	0.53	1.66	100
		EDSDE	0.00	0.00	0.00	0.00	0.90	—

表 4 测试函数的优化结果(2 维)

函数	最大迭代次数	算法	最小值 min	平均值 mean	标准差 std	中值 median	平均时间/s	胜率/%
f_{10}	500	DE/rand/1/bin	0.00	0.00	0.00	0.00	0.11	100
		NIDE	0.00	0.00	0.00	0.00	0.12	100
		RMDE	0.00	0.00	0.00	0.00	0.10	100
		JDE	0.00	0.00	0.00	0.00	0.11	100
		JADE	0.00	0.00	0.00	0.00	0.77	100
		MDEpBX	0.00	0.00	0.00	0.00	0.36	100
		EDSDE	0.00	0.00	0.00	0.00	0.10	—

通过分析表 2 可知，在 $N=50$ 维时，对 f_1-f_6，f_9 而言，EDSDE 算法均可以找到最好的最小值、平均值、方差以及中值；对 f_6 而言，虽然 DE/rand/1/bin 以及 RMDE 算法均能找到最小值，但是其运行时间均长于 EDSDE 算法；对 f_7，f_8 而言，虽然 EDSDE 算法没能找到最优解，但是其收敛精度与寻优结果均优于其余 3 种算法。

为了验证 EDSDE 算法在高维度搜索空间下的寻优能力，将 N 设置为 100 维再次进行仿真实验，同时通过分析表 3 可知，对 f_1-f_6，f_9 而言，EDSDE 算法均可以找到最好的最小值、平均值、方差以及中值；对 f_7，f_8 而言，虽然 EDSDE 算法没能找到最优解，但是其收敛精度与寻优结果均优于其余 6 种算法。

通过分析表 4 可知，在 $N=2$ 维时，对 f_{10} 而言，EDSDE 算法均可以找到最好的最小值、平均值、方差以及中值。

纵观表 2 至表 4，EDSDE 算法不仅寻优能力更好，而且运行的平均时间相较于其余 6 种算法并没有增加，且胜率均为 100%，这表明该算法具有极强的稳定性，不是偶然或随机形成的。很明显，EDSDE 算法相较于 DE/rand/bin、NIDE 算法、RMDE 算法、JDE 算法、JADE 算法以及 MDEpBX 算法的收敛精度都更好，DE/rand/bin、NIDE 算法、RMDE 算法、JDE 算法、JADE 算法以及 MDEpBX 算法在不同函数上都会陷入不同程度的局部最优，这表明 EDSDE 算法具有杰出的性能，能在收敛精度与收敛速度之间保持良好的平衡。

图 3－图 11 分别是 $N=50$ 维时 f_1-f_9 的收敛曲线图，图 12－图 20 分别是 $N=100$ 维时 f_1-f_9 的收敛曲线图，图 21 是 $N=2$ 维时 f_{10} 的收敛曲线图，图 22 是 f_{10} 的三维仿真图。这些收敛曲线均为平均优化曲线，其横坐标代表迭代次数，纵坐标代表平均函数值。

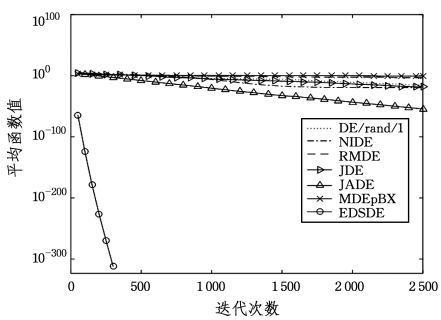

图 3 f_1 收敛曲线(50 维)

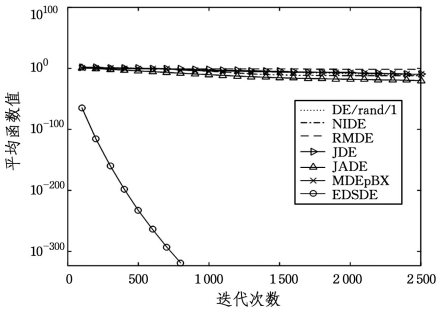

图 4　f_2 收敛曲线(50 维)

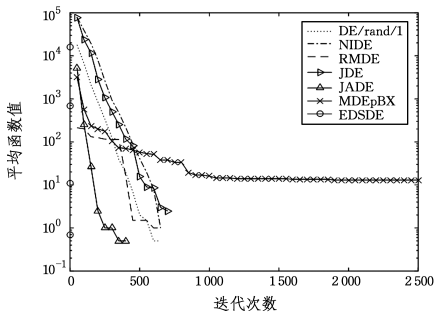

图 5　f_3 收敛曲线(50 维)

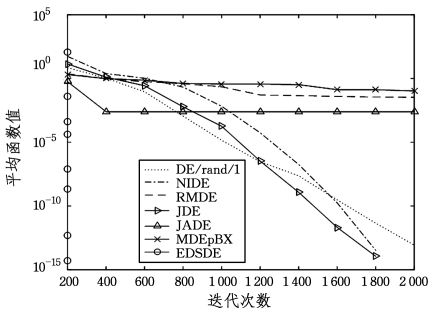

图 6　f_4 收敛曲线(50 维)

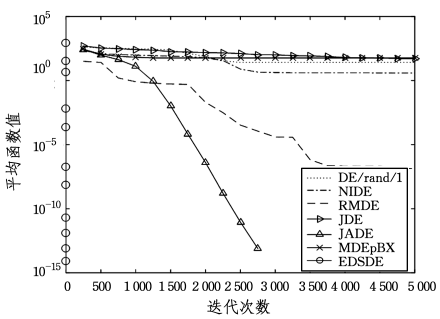

图 7　f_5 收敛曲线(50 维)

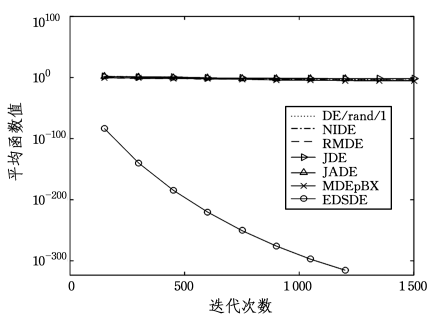

图 8　f_6 收敛曲线(50 维)

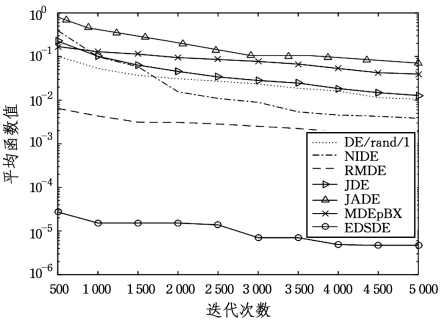

图 9　f_7 收敛曲线(50 维)

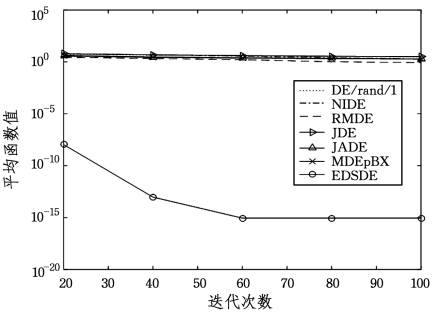

图 10　f_8 收敛曲线(50 维)

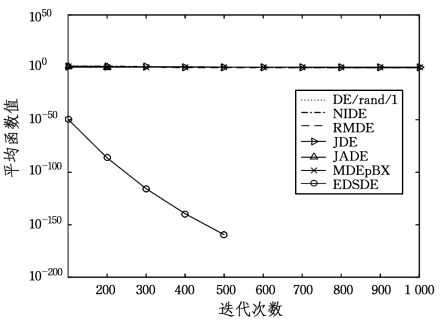

图 11　f_9 收敛曲线(50 维)

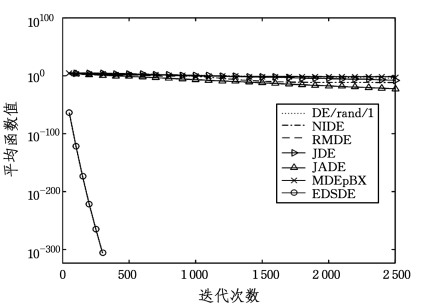

图 12　f_1 收敛曲线(100 维)

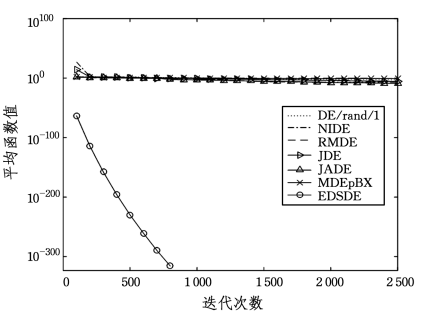

图 13　f_2 收敛曲线(100 维)

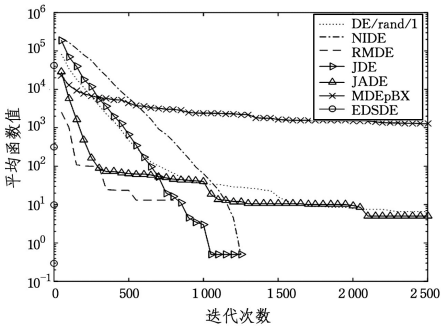

图 14 f_3 收敛曲线(100 维)

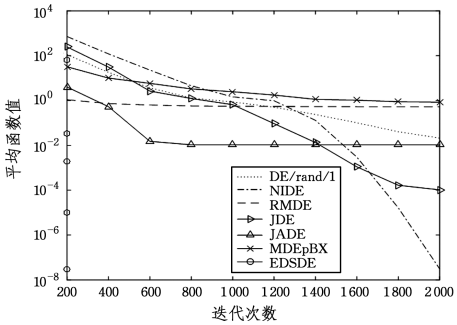

图 15 f_4 收敛曲线(100 维)

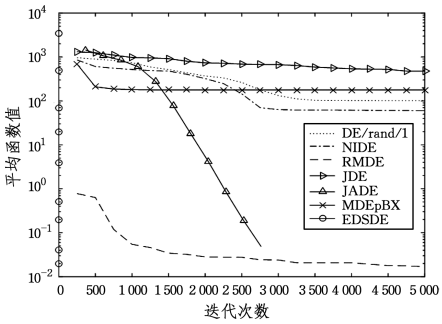

图 16 f_5 收敛曲线(100 维)

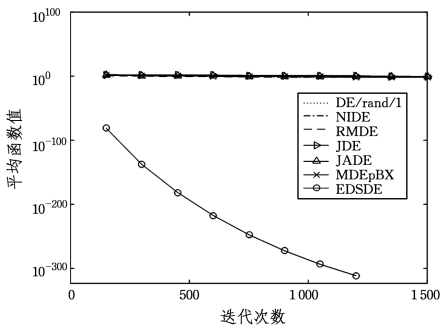

图 17 f_6 收敛曲线(100 维)

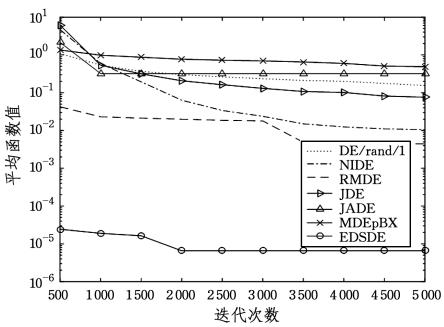

图 18 f_7 收敛曲线(100 维)

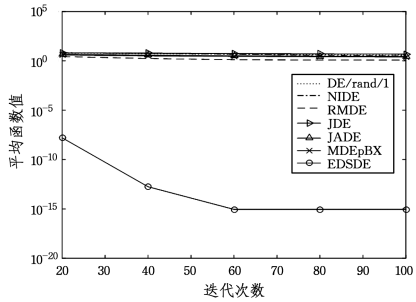

图 19 f_8 收敛曲线(100 维)

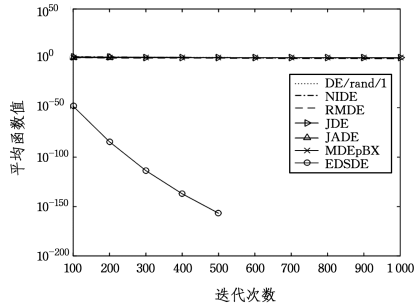

图 20 f_9 收敛曲线(100 维)

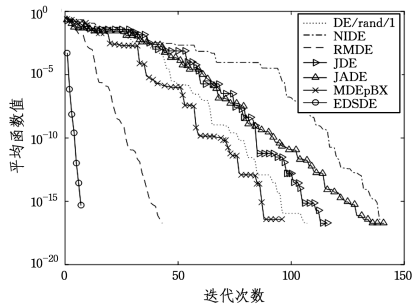

图 21 f_{10}收敛曲线(2 维)

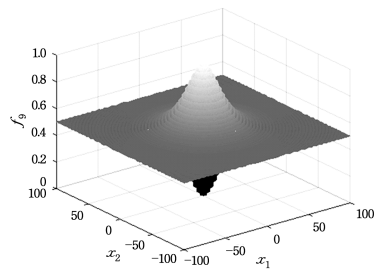

图 22 f_{10}三维图

从图 3－图 11 可以进一步看出,在 $N=50$ 维时,对 f_3 与 f_6 而言,虽然 DE/rand/1/bin 以及 RMDE 算法都能找到全局最优解,但都需要到搜索的中后期才能找到全局最优解,而 EDSDE 算法的收敛速度要快得多;对 f_4 而言,虽然 NIDE 算法、JDE 算法以及 JADE 算法也能找到最优解,但迭代次数需要达到 2000 代左右才能找到全局最优解,而 EDSDE 算法几乎瞬间就收敛到全局最优解;对 f_3 而言,虽然 DE/rand/bin、NIDE 算法、RMDE 算法、JDE 算法以及 JADE 算法都能找到全局最优解,但 EDSDE 算法在迭代初期就表现出极快的收敛速度。

从图 12－图 20 可以进一步看出,虽然空间维数从 50 维上升到 100 维,但 EDSDE 算法的寻优能力并没有明显的降低,反观其余 6 种算法的寻优能力均有不同程度的下降,这证

明了 EDSDE 算法不仅适应低维度的搜索空间，同时也满足高维度搜索空间的要求。

从图 21、图 22 可以进一步看出，f_{10} 在整个定义域内具有无数个局部最优解，一般算法很容易陷入这些局部最优解，但 EDSDE 算法具有很强的跳出局部最优的能力，能在迭代 20 次左右就找到全局最优解，而其余 6 种算法虽然也能找到全局最优解，但迭代次数均有不同程度的增加，收敛速度相对更慢。

另外，EDSDE 算法选取的 F_{max} 及 F_{min} 数值时是通过多组数据对比得出，并通过仿真曲线效果图确定最终值。本文选出 4 组 F_{max}、F_{min} 数据，具体如表 5 所列。

表 5　F_{max} 与 F_{min} 参数节选

序号	F_{max}	F_{min}
1	0.99	0.1
2	0.9	0.2
3	0.8	0.4
4	0.99	0.2

图 23－图 32 分别是 1－4 组数据在 $N=50$ 维时 f_1-f_{10} 的收敛曲线图。

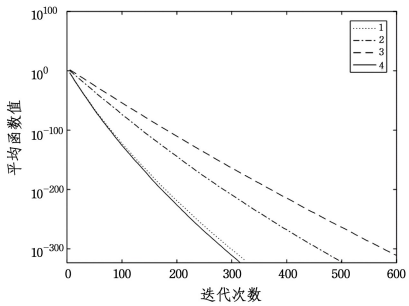

图 23　f_1 收敛曲线(50 维)

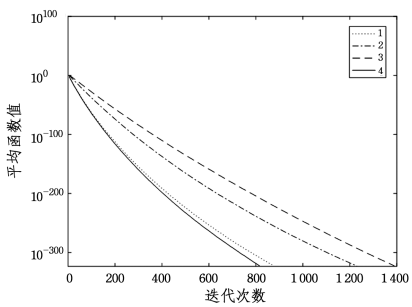

图 24　f_2 收敛曲线(50 维)

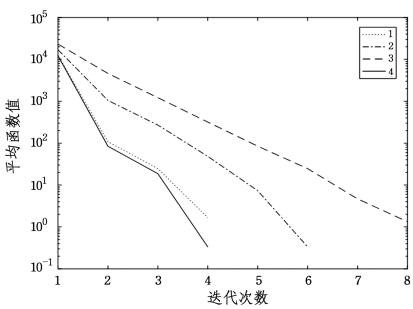

图 25　f_3 收敛曲线(50 维)

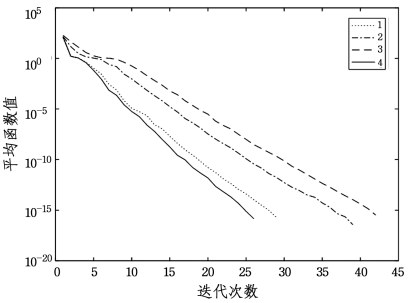

图 26　f_4 收敛曲线(50 维)

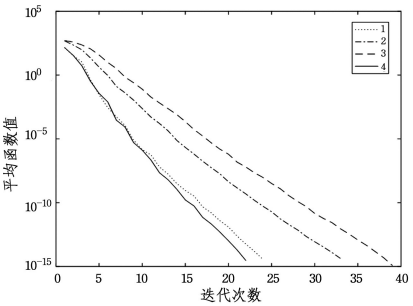

图 27　f_5 收敛曲线(50 维)

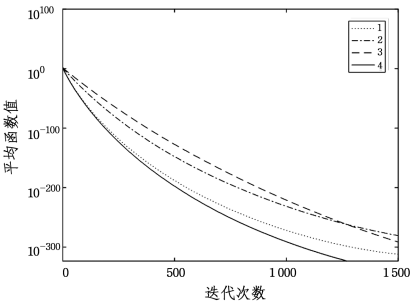

图 28　f_6 收敛曲线(50 维)

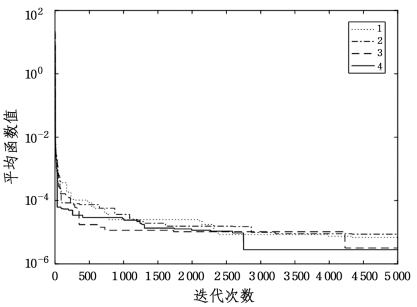

图 29　f_7 收敛曲线(50 维)

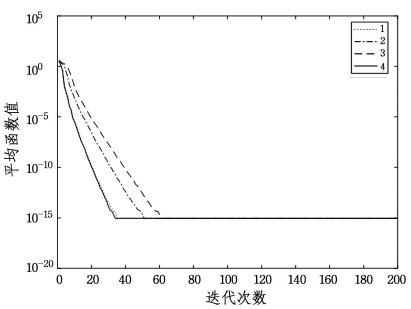

图 30　f_8 收敛曲线(50 维)

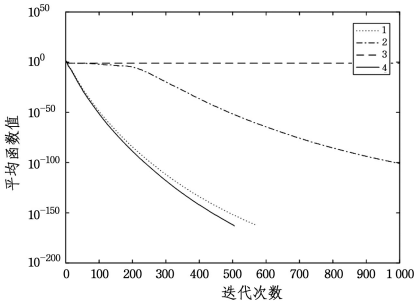

图 31 f_9 收敛曲线(50 维)

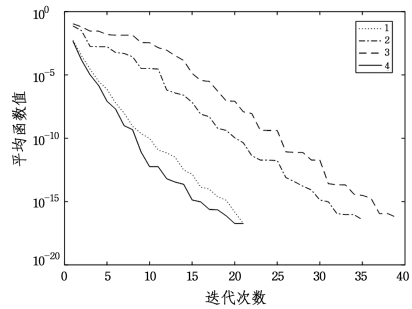

图 32 f_{10} 收敛曲线(50 维)

从图 23－图 32 可以看出,基于第 4 组数据的所有函数的收敛速度均好于 1－3 组,只有第 1 组的收敛速度接近第 4 组,相比之下取 $F_{max}=0.99$,$F_{min}=0.2$。

结束语 本文在 DE/rand/bin、NIDE 算法、RMDE 算法、JDE 算法、JADE 算法以及 MDEpBX 算法的基础之上,通过线性递减策略的变异因子解决了收敛速度较慢的问题,通过不断震荡更新的交叉因子实现了跳出局部最优的缺陷,以及通过提出新的变异策略兼顾了全局搜索与局部开发的能力,并采用文献[19-20]中推荐的 10 种测试函数进行了优化计算比较。实验结果表明,EDSDE 算法不论在高维度还是低维度时的收敛精度以及综合寻优能力均强于其他 3 种算法,说明本文提出的改进差分进化算法具有高效性、适应性、稳定性,为以后的实际工程应用研究提供了一条可行的路径。

参 考 文 献

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

[2] 霍玉洪. 切比雪夫不等式及其应用[J]. 长春工业大学学报:自然科学版,2012,33(6):713-714.

[3] 曲良东,何登旭,吴尽昭. 一种群模式全局搜索算法[J]. 模式识别与人工智能,2013,26(6):592-597.

[4] ZOU D X,GAO L Q. An Efficient Improved Differential Evolution Algorithm[C]// 第三十一届中国控制会议. Hefei,China,2012:2385-2390.

[5] 孙成富,赵建洋,高磊. 基于变权重因子差分进化算法的梯级水火电力系统调度[J]. 数据采集与处理,2017,32(1):95-103.

[6] 卢有麟,周建中,覃晖,等. 差分进化算法在电力系统环境经济调度中的应用[J]. 华中科技大学学报(自然科学版),2010,38(8):121-124.

[7] 张锟,徐青,王一凡,等. 自适应差分进化算法在边坡滑面搜索中的应用[J]. 岩土力学,2017,38(5):1503-1509.

[8] 翁志远,方杰,孔敏,等. 改进差分进化算法的作业车间调度优化策略[J]. 控制工程,2017,24(6):1282-1285.

[9] 郭丽. 自适应差分进化算法解决多目标有限缓冲车间调度问题研究[D]. 郑州:郑州大学,2016.

[10] 吴娜,车蕾. 基于改进 DDE 算法的协同干扰资源分配[J]. 电光与控制,2018,25(2):107-110.

[11] 邹德旋,王鑫,段纳. 一种基于修正差分进化的虹膜定位算法[J]. 控制理论与应用,2013,30(9):1194-1200.

[12] 李目,何怡刚,周少武,等. 一种差分进化算法优化小波神经网络及其在弱信号检测中的应用[J]. 计算机应用与软件,2010,27(3):29-31,39.

[13] 陈亮. 改进自适应差分进化算法及其应用研究[D]. 上海:东华大学,2012.

[14] 刘龙龙,颜七笙. 一种新的改进差分进化算法[J]. 江西科学,2017,35(4):485-489.

[15] 欧阳海滨,高立群,孔祥勇. 随机变异差分进化算法[J]. 东北大学学报:自然科学版,2013,34(3):330-334.

[16] BREST J,GREINER S,BOSKOVIC B,et al. Self-adapting control parameters in differential evolution:A comparative study on numerical benchmark problems[J]. IEEE Transactions on Evolutionary Computation,2006,10(6):646-657.

[17] ZHANG J,SANDERSON A C. JADE:adaptive differential evolution with optional external archive[J]. IEEE Transactions on Evolutionary Computation,2009,13(5):945-958.

[18] ISLAM S M,DAS S,GHOSH S,et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization[J]. IEEE Transactions on Systems,Man,and Cybernetics,Part B(Cybernetics),2012,42(2):482-500.

[19] RAHNAMAYAN S,TIZHOOSH H R,SALAMA M M A. Opposition-Based Differential Evolution[M]. Springer Berlin Heidelberg,2008.

[20] ALI M M,KHOMPATRAPORN C,ZABINSKY Z B. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems[J]. Journal of Global Optimization,2005,31(4):635-672.