文章编号: 1001-0920 (2012) 11-1644-05

具有人工蜂群搜索策略的差分进化算法

黄玲玲, 刘三阳, 高卫峰

(西安电子科技大学 数学科学系，西安 710071)

摘 要: 针对差分进化算法易出现早熟现象和收敛速度慢等问题, 提出一种具有人工蜂群搜索策略的差分进化算法. 利用人工蜂群搜索策略很强的探索能力, 对种群进行引导以帮助算法快速跳出局部最优点. 此外, 为了提高算法的全局收敛速度, 采用一种基于反学习的初始化方法. 通过对 12 个标准测试函数进行仿真实验并与其他算法相比较, 表明了所提出的算法具有较快的收敛速度和很强的跳出局部最优的能力.

关键词: 差分进化算法；种群初始化；搜索策略；人工蜂群算法

中图分类号: TP18 **文献标志码:** A

Differential evolution with the search strategy of artificial bee colony algorithm

HUANG Ling-ling, *LIU San-yang*, *GAO Wei-feng*

(Department of Applied Mathematics，Xidian University，Xi'an 710071，China. Correspondent：HUANG Ling-ling, E-mail：huangzxp@yahoo.cn)

Abstract: For the problems of premature convergence frequently appeared in differential evolution(DE) and its poor convergence, a differential evolution with the search strategy of artificial bee colony algorithm is proposed. The method makes full use of the exploration ability of the search strategy of artificial bee colony algorithm to guide the algorithm to jump out of the likely local optima. In addition, to enhance the global convergent speed, when producing the initial population, the opposition-based learning method is employed. Moreover, the performance of the proposed approach is testified on a suite of 12 benchmark functions and the comparisons with other algorithms are provided. Simulation results show that the proposed approach has the better convergence rate and the strong ability of preventing premature convergence.

Key words: differential evolution；population initialization；search strategy；artificial bee colony algorithm

1 引 言

Stron 等[1]于 1997 年提出的差分进化算法 (DE) 是继遗传算法、蚁群算法之后的又一种新兴的群体智能优化算法. 与其他进化算法一样, DE 是一种模拟生物进化的随机模型, 通过反复迭代, 使得那些适应环境的个体被保存下来. 由于算法具有结构简单、易于实现、无需梯度信息、参数较少等特点, 一经提出便受到众多学者的关注和研究, 并在函数优化、模式识别、系统辨识等诸多领域得到了广泛应用. 与其他智能算法类似, DE 也存在易陷入局部最优, 进化后期收敛速度慢, 对过于复杂的问题可能搜索不到最优解, 计算精度不高等问题. 这些缺点限制了 DE 在实际中的应用.

针对上述问题, 人们提出了多种改进策略[2-9]来优化 DE 的性能. 文献 [2] 引入了反学习策略, 并将其用于种群初始化和跳出局部最优点, 提出了一种新颖的加速 DE. [3] 给出了优先交叉 DE, 利用变异产生的个体的概率密度函数来分析缩放比例参数的缺陷, 并提出优先交叉准则以弥补缺陷. [4] 提出一种基于混沌和高斯局部优化的混合差分进化算法, 在进化初期利用混沌的遍历性, 可有效地避免算法陷入局部最优; 而在进化后期, 采用高斯搜索可有效地提高收敛精度. [5] 引入基于杂交的自适应局部搜索以提高算法的收敛速度. [6] 采用一个新的变异策略, 提出一种自适应的 DE (JADE). [7] 通过采用正交设计的方法加速 DE 的收敛速度, 提高了 DE 的寻优性能. 然而, 到目前为止, 诸多算法很难在提高收敛速度和避免早熟收敛

收稿日期: 2011-04-29；**修回日期:** 2011-11-02.

基金项目: 国家自然科学基金项目(60974082)；中央高校基本科研业务费专项资金项目(K5051270002).

作者简介: 黄玲玲(1983−), 女, 博士生, 从事进化计算及最优化理论与方法的研究；刘三阳(1959−), 男, 教授, 博士生导师, 从事智能信息处理、最优化方法等研究.

两方面取得平衡. 例如, 基于竞争策略和参数控制的DE (CoDE)[8]侧重于避免早熟收敛, 在一些单峰函数上优化性能略逊于JADE; 而紧凑的DE (cDE)[9]希望能在这两方面取得平衡, 虽然取得了较好的结果, 但在某些测试函数上仍不能跳出局部最优, 如Shifted Rastrigin 函数.

为了克服早熟、提高收敛速度和寻优精度, 本文提出一种新的改进的差分进化算法. 首先, 采用反学习的初始化方法加快算法的全局收敛速度; 然后, 在差分进化算法中引入人工蜂群算法搜索算子对种群进行引导搜索, 从而使种群中的个体尽快跳出局部最优点. 该方法能提高算法的探索和开发能力, 避免算法因陷入局部最优而导致早熟收敛. 大量的仿真实验表明该算法能显著提高优化效率和优化性能.

2　差分进化算法

差分进化算法 (DE)[1]是一种随机的并行直接搜索算法, 其基本思想是: 从某一随机产生的初始种群开始, 按照一定的操作规则不断迭代, 并根据每一个体的适应度值, 保留优良个体, 淘汰劣质个体, 引导搜索过程向最优解逼近.

算法运行过程中保持种群规模不变, 3 种运算贯穿整个执行过程, 即变异、交叉和选择, 分别描述如下.

1) 变异. 对于个体 X_i, 按下式生成变异个体:

$$U_i = X_{r1} + F(X_{r2} - X_{r3}). \tag{1}$$

其中: X_{r1}, X_{r2}, X_{r3} 为从进化种群中随机选取的互不相同的 3 个个体; F 为缩放比例因子, 用于控制差向量的影响大小.

2) 交叉. 为增加种群多样性, 下述交叉操作被引入差分进化算法:

$$v_{i,j} = \begin{cases} u_{i,j}, \ \mathrm{rand}(0,1) \leqslant \mathrm{CR}; \\ x_{i,j}, \ \mathrm{rand}(0,1) > \mathrm{CR}. \end{cases} \tag{2}$$

其中: $j = 1, 2, \cdots, D$, D 为空间维数; $\mathrm{CR} \in [0,1]$ 为交叉概率.

3) 选择. 交叉后的个体 V_i 和父代个体 X_i 按下式进行选择操作以生成子代个体:

$$X_i = \begin{cases} V_i, \ F(V_i) < F(X_i); \\ X_i, \ F(V_i) \geqslant F(X_i). \end{cases} \tag{3}$$

3　具有人工蜂群搜索策略的差分进化算法

3.1　种群初始化方法

种群初始化在进化算法中十分重要, 因为它影响着算法的全局收敛速度和解的质量. 在没有任何先验信息可以利用的情况下, 一般采用随机初始化方法. 由于该方法不能保证产生的初始种群有效地提取解空间的信息, 在一定程度上限制了算法的效率. 为了使初始种群尽可能地具有多样性, 充分利用搜索空间的信息, 本文采用基于反学习的种群初始化[2]. 首先随机生成初始解; 然后为每个初始解产生对应的反向解; 最后对两类解排序选择, 将适应度较优的解作为初始种群, 这有助于求解效率的提高与解质量的改善. 具体步骤如下:

算法 1　反学习的初始化方法.

1) 初始化种群规模 M.

{随机初始化阶段}

2) for $i = 1$ to M do

3)　for $j = 1$ to D do

4)　$x_{i,j} = x_{\min,j} + \mathrm{rand}(0,1)(x_{\max,j} - x_{\min,j})$

5)　end for

6) end for

{反学习阶段}

7) for $i = 1$ to M do

8)　for $j = 1$ to D do

9)　$ox_{i,j} = x_{\min,j} + x_{\max,j} - x_{i,j}$

10)　end for

11) end for

12) 从 $\{X(M) \bigcup OX(M)\}$ 中选择适应值最好的 M 个个体作为初始种群.

3.2　搜索策略

所有的进化算法都试图去平衡探索能力和开发能力这一矛盾. 探索能力指算法勘探搜索空间以寻找那些可能的最优区域; 开发能力则是算法将搜索的重点放在具有较高适应值的个体上. 差分进化算法在进化中后期, 由于种群多样性的降低, 当优化复杂的多峰问题时, 如果有个体陷入局部最优跳不出去, 则它会将附近的个体向局部最优区域引导. 当很多个体陷入局部最优区域时, 很容易导致算法出现早熟现象. 即差分进化算法的探索能力较弱, 这是导致算法容易出现早熟的根源所在. 因此, 如何在探索与开发之间进行有效的权衡是提高算法性能的关键.

2006 年, Karaboga[10]提出了一种新的智能优化算法 —— 人工蜂群算法 (ABC), 模拟蜜蜂群的智能采蜜行为; 同时, 文献 [10] 也表明, ABC 在优化复杂的多峰问题时显示出了很好的优化性能. 这主要得力于ABC 搜索策略具有很强的探索能力, 即

$$z_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}). \tag{4}$$

其中: $k \in \{1, 2, \cdots, M\}$ 和 $j \in \{1, 2, \cdots, D\}$ 为随机选取的指标, 并且 k 不等于 i; $\phi_{i,j}$ 为 $[-1, 1]$ 之间的随机数. 根据人工蜂群算法的搜索策略, 新的候选解向种群中随机选择的个体移动. 由于选择的随机性, 适应值好的个体和适应值差的个体被选择的概率是相

同的, 可见, 人工蜂群算法具有很强的探索能力.

然而, ABC 的搜索策略侧重于提高算法的探索能力, 却以牺牲开发能力为代价. 为此, Zhu 等[11]受粒子群优化算法的启发, 给出了一种新的搜索策略, 即

$$z_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) + \psi_{i,j}(p_{g,j} - x_{i,j}). \quad (5)$$

其中: k, j 和 $\phi_{i,j}$ 的选取同式 (4); $\psi_{i,j}$ 为 [0, 1.5] 中的随机数. 可见, 新的搜索策略由于最优位置 P_g 的引导, 在保证探索能力的同时也能提高开发能力.

基于上述分析, 本文利用 ABC 搜索策略很强的探索能力来弥补差分进化算法探索能力弱的缺点, 进而达到避免算法早熟的目的.

3.3 算法的基本思想和步骤

差分进化算法在寻优过程中存在收敛速度慢、容易出现早熟的缺点, 严重影响了算法的性能. 对此, 本文提出一种具有人工蜂群搜索策略的差分进化算法, 其基本思想是: 在 DE 的迭代中引入人工蜂群搜索策略, 对整个种群进行搜索, 使其快速跳出局部最优, 达到避免早熟的目的; 同时, 为了提高算法的收敛速度和全局寻优能力, 采用反学习的初始化方法. 这些操作可使算法在提高收敛速度和避免早熟两方面取得平衡, 从而其性能得到大幅提高. 具体步骤如下:

算法 2 具有人工蜂群搜索策略的差分进化算法.

1) 初始化收缩因子 F 和交叉因子 CR, 最大函数评价次数 MFE.

2) 在定义空间中, 用反学习的初始化方法产生 M 个个体并组成初始种群 $X_i, i = 1, 2, \cdots, M$.

3) while 算法的停止条件不满足 do

4) for $i = 1$ to M do

5) 按式 (1) 进行变异操作;

6) 按式 (2) 进行杂交操作;

7) 按式 (3) 进行选择操作.

8) for $k = 1$ to K do

9) 用式 (4) 或 (5) 在 X_i 周围搜索候选解 Z_i.

10) if $F(Z_i) < f(X_i)$ do

11) $X_i = Z_i$.

12) end if

13) end for

14) end for

15) end while

16) 输出最优解及最优值.

注 1 本文将所提出的具有人工蜂群搜索策略的差分进化算法简记如下: 采用搜索算子 (4) 记为 SSDE1, 采用搜索算子 (5) 记为 SSDE2.

4 仿真实验

为了测试本文提出的改进的差分进化算法的效果, 对文献 [7] 中的 12 个标准测试函数进行仿真实验. 其中: f_1 和 f_2 为单峰函数, $f_3 \sim f_{12}$ 为多峰函数. 表 1 给出了 12 个测试函数的名称、维数、搜索空间范围和最优值, $K = 5$. 种群规模 M、缩放比例因子 F 和交叉概率 CR 均采用文献 [5] 的参数设置方案, 即 $M = 30$, $F = 0.9$, $\mathrm{CR} = 0.9$.

表 1 测试函数的维数、搜索空间和最优值

函数	名称	维数	搜索空间	最优值
f_1	Sphere	30	$[-100, 100]$	0
f_2	Rosebrock	30	$[-30, 30]$	0
f_3	Schwefel	30	$[-500, 500]$	0
f_4	Rastrigin	30	$[-5.12, 5.12]$	0
f_5	NCRastrigin	30	$[-5.12, 5.12]$	0
f_6	Griewank	30	$[-600, 600]$	0
f_7	Ackley	30	$[-32, 32]$	0
f_8	Penalized	30	$[-50, 50]$	0
f_9	Penalized2	30	$[-50, 50]$	0
f_{10}	Levy	30	$[-10, 10]$	0
f_{11}	Shifted Rastrigin	30	$[-5.12, 5.12]$	0
f_{12}	Shifted Griewank	30	$[-600, 600]$	0

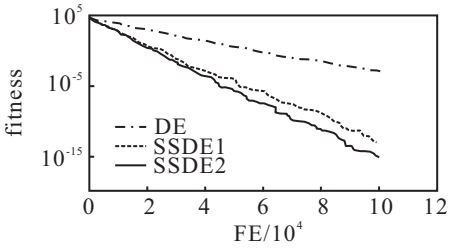
将本文算法与 DE 进行比较, 后者的参数设置见文献 [5]. 最大函数评价次数为 100 000. 采用 Matlab 7.0 编程工具进行仿真. Best, Median, Worst, Mean 和 Std 分别为算法独立实验 20 次的最好值、中间值、最差值、平均值和标准差. Best, Median, Worst 反映了解的质量; Mean 显示了在给定的函数评价次数下算法所能达到的精度, 反映了算法的收敛速度; Std 反映了算法的稳定性和鲁棒性. 结果如表 2 所示.

由表 2 数据对比可以看出, 在所有的标准测试函数中, 无论是解的质量, 还是算法的收敛精度和稳定性, SSDE1 和 SSDE2 算法都比 DE 算法有了很大的提高. 特别地, 在这 3 种算法中, SSDE2 算法几乎在所有测试函数上都取得了最好的优化结果, 除了在 Rosebrock 函数上, SSDE1 算法求解的结果最好.

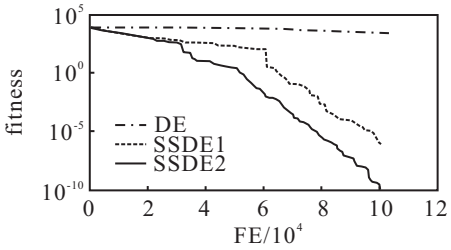
为了更直观地反映算法的寻优效果, 将 SSDE1 算法、SSDE2 算法和 DE 算法进行比较. 3 种算法对相关测试函数的收敛曲线如图 1 和图 2 所示. 由图可知, 本文提出的差分进化算法由于引入了搜索策略和反学习的初始化方法, 使得算法在处理多峰函数时能很快跳出局部最优, 收敛到全局最优解; 在处理单峰函数时具有较快的收敛速度. 而 SSDE1 算法的搜索算子因没有最优位置 $\boldsymbol{p}_g$ 的引导, 降低了算法的开发能力, 在处理多峰函数时虽能跳出局部最优, 但其收敛速度比 SSDE2 算法慢, 如函数 f_3, f_5 和 f_{10} 所示. 因此, 可以得出结论: 本文所提出的 SSDE2 算法的优化能力整体上强于 DE 和 SSDE1 算法.

表 2　3种算法对12个函数的计算结果比较

Fun		Best	Median	Worst	Mean	Std
	DE	4.63e-04	0.001 315	0.004 687	0.001 258	0.001 273
f_1	SSDE1	3.68e-15	2.12e-14	2.81e-13	7.67e-14	8.46e-14
	SSDE2	3.95e-16	5.08e-16	4.15e-15	1.53e-15	1.00e-15
	DE	22.047 785	30.204 121	98.256 663	46.462 372	30.265 107
f_2	SSDE1	0.105 000	1.853 945	3.670 690	1.154 334	1.094 901
	SSDE2	0.003 462	0.608 520	20.677 177	4.919 787	7.249 512
	DE	5.79e+02	3.14e+03	7.41e+03	3.92e+03	1.79e+03
f_3	SSDE1	8.76e-10	8.76e-10	1.07e-06	2.84e-07	4.22e-07
	SSDE2	5.45e-12	3.27e-11	8.56e-10	1.73e-10	2.61e-10
	DE	14.325 150	55.900 817	88.540 473	43.781 406	18.417 313
f_4	SSDE1	2.14e-09	2.22e-07	9.20e-05	1.21e-05	2.73e-05
	SSDE2	3.33e-10	1.02e-09	4.22e-08	1.07e-08	1.49e-08
	DE	25.021 520	67.821 497	94.755 745	52.359 627	22.936 123
f_5	SSDE1	6.017e-07	1.45e-05	0.010 816	0.001 296	0.003 204
	SSDE2	4.42e-08	2.46e-06	3.81e-06	5.91e-07	7.57e-07
	DE	0.001 303	0.020 814	0.020 814	0.007 567	0.005 677
f_6	SSDE1	1.12e-13	2.21e-13	1.85e-11	4.00e-12	6.09e-12
	SSDE2	2.22e-16	5.10e-15	2.80e-13	3.57e-14	8.21e-14
	DE	0.002 405	0.006 595	0.013 884	0.007 148	0.002 979
f_7	SSDE1	4.38e-08	1.76e-07	3.69e-07	1.87e-07	1.01e-07
	SSDE2	1.11e-08	2.27e-08	3.92e-08	2.25e-08	9.25e-09
	DE	2.43e-04	0.001 681	0.040 318	0.005 593	0.011 679
f_8	SSDE1	5.82e-16	1.76e-15	8.30e-15	2.62e-15	2.42e-15
	SSDE2	3.03e-18	5.11e-18	4.42e-17	1.91e-17	1.35e-17
	DE	0.006 386	0.017 020	0.379 713	0.082 814	0.124 446
f_9	SSDE1	6.73e-15	9.27e-15	1.16e-13	5.57e-14	3.56e-14
	SSDE2	8.89e-17	2.46e-16	1.62e-15	5.45e-16	4.40e-16
	DE	1.30e-05	7.94e-05	2.42e-04	8.89e-05	7.64e-05
f_{10}	SSDE1	4.09e-15	1.66e-14	8.53e-14	2.89e-14	2.49e-14
	SSDE2	8.96e-17	1.60e-15	1.74e-15	5.27e-16	5.94e-16
	DE	24.148 452	66.638 523	1.14e+02	61.353 934	28.198 135
f_{11}	SSDE1	4.57e-07	0.001 822	0.095 701	0.011 527	0.028 367
	SSDE2	1.30e-10	5.40e-10	1.83e-08	6.62e-09	7.03e-09
	DE	0.005 111	0.010 679	0.045 527	0.023 427	0.012 854
f_{12}	SSDE1	3.83e-13	4.18e-12	0.007 396	7.39e-04	0.002 218
	SSDE2	3.77e-15	3.25e-14	5.02e-08	5.17e-09	1.50e-08

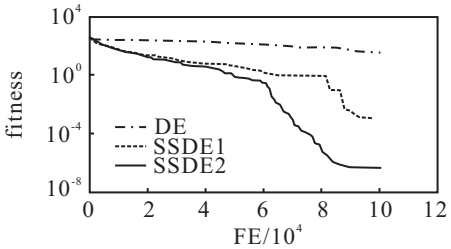


(a) f_1 函数

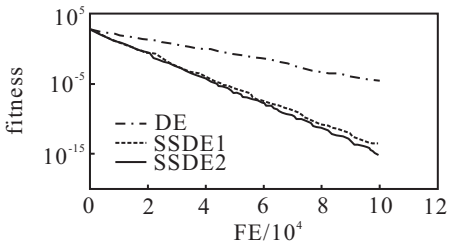


(b) f_3 函数

图 1　f_1 和 f_3 函数的收敛性能比较



(a) f_5 函数



(b) f_{10} 函数

图 2　f_5 和 f_{10} 函数的收敛性能比较

表 3　6种算法对8个函数的计算结果比较

Function	Index	DE	DEahcSPX	SaDE	Orth-DE	ODE	SSDE2
f_1	Mean	5.73e-17	1.75e-31	1.08e-130	4.50e-72	2.53e-58	2.83e-54
	Std	2.03e-16	4.99e-31	4.55e-130	3.12e-71	4.07e-58	4.29e-54
f_2	Mean	5.20e+01	4.52e+00	3.12e+01	1.20e+00	2.88e+01	0.398 714
	Std	8.56e+01	1.55e+01	2.78e+01	1.85e+00	1.32e+01	1.195 969
f_3	Mean	4.90e+02	4.70e+02	5.21e-03	1.77e+02	1.98e+01	1.73e-10
	Std	2.34e+02	2.96e+02	8.32e-03	1.86e+02	4.25e+01	2.61e-10
f_4	Mean	2.55e+01	2.14e+01	0	3.63e+01	3.76e+01	0
	Std	8.14e+00	1.23e+01	0	9.94e+00	1.82e+01	0
f_6	Mean	2.66e-03	2.07e-03	0	7.17e-03	0	0
	Std	5.73e-03	5.89e-03	0	1.29e-02	0	0
f_7	Mean	1.37e-09	1.66e-14	1.53e-01	5.55e-01	2.66e-14	1.15e-14
	Std	1.32e-09	3.41e-014	3.56e-01	6.89e-01	2.41e-14	3.17e-15
f_8	Mean	4.56e-02	2.07e-02	1.03e-02	2.00e-01	1.57e-32	1.57e-32
	Std	1.31e-01	8.46e-02	3.14e-02	5.14e-01	2.73e-48	2.73e-48
f_9	Mean	1.44e-01	1.71e-31	2.19e-04	3.33e-03	1.35e-32	1.35e-32
	Std	7.19e-01	5.35e-31	1.55e-03	9.23e-03	1.10e-47	1.10e-47

为进一步展示SSDE2算法的先进性, 将SSDE2算法与DE, DEahcSPX[5], SaDE[12], Orth-DE[7], ODE[2]和APSO[9]算法的结果进行比较, 如表3所示. 为公平起见, 算法的最大函数评价次数与文献[5]相同, 即300 000. 从表3中可以看出, 对于除 f_1 外的6个函数, SSDE2算法的性能都有较大的改善, 优化结果和理论值的误差最小.

当然, 新算法也有自身的局限性. 由于在DE中引入了探索能力很强的搜索算子, 在一定程度上忽视了算法的开发能力. 从表3可以观察到SSDE1算法的开发能力不及SSDE2算法, 导致前者全局收敛速度慢于后者; 而在优化Sphere函数时SSDE2算法不及文献[12]的SaDE算法. 这是由于在处理多峰优化问题时, 如果算法探索能力强, 则能快速跳出局部最优, 这样既节省了计算量, 又能找到较好的解, 在一定程度上提高了算法的收敛速度. 然而, 由于过于重视探索而轻视开采, 使算法放弃了有可能找到更好解的区域而去探索另一片区域. 这样一味地提高算法的探索能力虽然可使算法快速跳出局部最优, 但却降低了算法的收敛速度(在优化Sphere函数时可以看到). 从算法的整体性能而言, 本文提出的算法通过搜索算子与新的初始化方法的相互作用, 使算法的性能得到了改善, 取得了令人满意的结果.

5 结 论

为了解决标准差分进化算法早熟收敛和收敛速度慢等问题, 本文提出了一种具有人工蜂群搜索策略的差分进化算法. 对12个函数寻优的实验结果表明, 本文算法在优化效率、优化性能和鲁棒性方面均比标准DE及其他改进的差分进化算法有了较大的改善.

如何在保证算法快速跳出局部最优的同时, 又具有高速的收敛速度, 是下一步的研究内容.

参考文献(References)

[1] Stron R, Price K. Differential evolution − A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[2] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution[J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 64-79.

[3] Ali M M. Differential evolution with preferential crossover[J]. European J of Operational Research, 2007, 181(3): 1137-1147.

[4] 贾东立, 郑国莘. 基于混沌和高斯局部优化的混合差分进化算法[J]. 控制与决策, 2010, 25(6): 899-902.
(Jia D L, Zheng G X. Hybrid differential evolution combined with chaos and Gaussian local optimization[J]. Control and Decision, 2010, 25(6): 899-902.)

[5] Nasimul N, Hitoshi I. Accelerating differential evolution using an adaptive local search[J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 101-125.

[6] Zhang J Q, Sanderson A C. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Trans on Evolutionary Computation, 2009, 13(5): 945-958.

[7] Gong W Y, Cai Z H, Jiang L X. Enhancing the performance of differential evolution using orthogonal design method[J]. Applied Mathematics and Computation, 2008, 206(1): 56-69.

[8] Wang Y, Cai Z X, Zhang Q F. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 55-66.

[9] Ernesto M, Ferrante N, Li Y, et al. Compact differential evolution[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 32-54.

[10] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony(ABC) algorithm[J]. J of Global Optimization, 2007, 39(3): 459-471.

[11] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[12] Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2009, 13(2): 398-417.