带局部搜索的动态多群体自适应差分进化算法及函数优化

张雪霞,陈维荣,戴朝华
(西南交通大学电气工程学院, 四川成都 610031)

摘 要: 提出将一种改进的差分进化算法——带局部搜索的动态多群体自适应差分进化算法(DMSDELS)应用于函数优化.该算法将种群中的个体随机动态分成多个子群体,以增强个体间的信息交换;变异操作中,选择最优个体为基向量,差分向量的方向选择有利于搜索的方向,以提高收敛速度;变异尺度因子 F 与交叉概率 CR 采用自适应机制,以平衡局部搜索与全局搜索;部分优秀个体搜索达到指定代数进入局部搜索,以加快收敛.通过对 13 个 benchmark 典型复杂函数进行测试,并与其他七种优化算法进行比较,仿真结果表明:DMSDELS 算法具有较高的搜索精度和收敛性,且具有较强的跳出局部最优解能力.

关键词: 差分进化算法; 带局部搜索的动态多群体自适应差分进化算法; 优化算法
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2010) 08-1825-06

Dynamic Multi-group Self-adaptive Differential Evolution Algorithm with Local Search for Function Optimization

ZHANG Xue-xia, CHEN Wei-rong, DAI Chao-hua
(*School of Electrical Engineering, Southwest Jiaotong University. Chengdu, Sichuan* 610031, *China*)

Abstract: An improved algorithm based on differential evolution algorithms, dynamic multi-group self-adaptive differential evolution algorithm with local search (DMSDELS), is applied to optimize functions in this paper. In DMSDELS, the population is randomly and dynamically divided into multi-group individuals, which can exchange information. To speed up search, in the mutation phase the best individual is chosen as the base vector, and the selection of the direction for difference vector is benefit to search. The scaling factor F and the crossover rate CR are self-adapted in order to balance the local search and the global search. To accelerate the convergence, elitist individuals could search in local after they explored specified generations. DMSDELS is tested on thirteen complex benchmark functions. The results are compared with those of other seven algorithms. The results show that DMSDELS is better in the search precision, convergence property and has strong ability to escape from the local sub-optima.

Key words: differential evolution algorithm; dynamic multi-group self-adaptive differential evolution algorithm with local search; optimization algorithm

1 引言

全局优化问题广泛应用于计算机科学,控制工程,电力系统优化运行等科研领域,已成为科学研究人员的研究热点.近年来以遗传算法(GA)、蚁群算法(ACO)、粒子群算法(PSO)、差分进化算法(DE)等为代表的智能优化算法为全局优化问题提供了新的解决途径.

Rainer Storm 和 Kenneth Price 提出的差分进化算法[1]是一种简单而有效的全局优化算法,已在滤波器设计、电力系统无功优化、聚类分析等领域有较好的应用[2~4].该算法根据父代个体间的差分向量进行变异、交叉和选择操作,与其他进化算法一样易陷入局部最优解.目前已提出了一些 DE 改进算法.文献[5]提出自适应二次变异差分进化算法,该算法在运行过程中根据群体适应度方差的大小,增加一种新的变异算子对最优个体和部分其他个体同时进行变异操作,以提高种群多样性,增强差分进化算法跳出局部最优解的能力.文献[6,7]提出参数自适应改进差分算法,F 和 CR 分别根据差分向量幅度与目标向量收敛情况进行自适应调整.文献[8]提出基于混合优化策略的微分进化改进算法,该算

收稿日期:2008-09-03;修回日期:2010-04-26
基金项目:国家自然科学基金(No.60870004);中央高校基本科研业务费专项资金(No.SWJTU09CX026,No.SWJTU09ZT13,No.SWJTU09ZT10)

法将种群中的个体随机地分成两组,每组采用不同的优化策略.文献[9]提出基于带局部增强算子的微分进化改进算法,通过引入局部增强算子,使种群中的部分个体在当前最优个体附近寻优,以加快算法的收敛速度.文献[10,11]提出了自适应控制参数的差分进化算法.文献[12,13]提出一种基于自适应学习策略和自适应控制参数的差分进化算法.

为克服早熟现象,跳出局部最优解,本文将一种带局部搜索的动态多群体自适应差分进化算法(DMSDELS)[14]应用于函数优化,通过对几种典型的 benchmark 函数测试,结果表明:DMSDELS 算法具有较高的搜索精度和收敛性,且具有较强的跳出局部最优解能力.

2 带局部搜索的动态多群体自适应差分进化算法(DMSDELS)

带局部搜索的动态多群体自适应差分进化算法基于差分算法[6,7,10,11,15].该算法将种群中的个体随机动态分成多个子群体,以增强个体间的信息交换;将三个随机选择的用于变异的个体中的最优个体作为变异操作中的基向量,差分向量的方向选择有利于搜索的方向,以加快搜索;变异尺度因子 F 与交叉概率 CR 采用自适应机制,以平衡局部搜索与全局搜索;当部分优秀个体搜索到指定代数时进入局部搜索.以加快收敛.算法描述如下。

2.1 动态多群体

DMSDELS 将种群个体随机动态分成多个子群体,以使个体间动态交换信息.在迭代中,种群个体随机分成三个子群体,每个子群体中的个体按适应度大小从好到差降序排列,则可得到三个子群体中各自的最优个体.在每次评价前,用得到的这三个最优个体随机更新每个子群体中倒数三个相对差的个体,然后进行评价,即完成了一次个体间的信息交换.被分隔的三个子群体重新合并为一个种群,在迭代周期后将重新随机划分为三个子群体,再次得到每个子群体最优个体,将得到的三个最优个体与上一迭代周期中得到的进行比较,保留优秀个体,以达到不同群体间动态交换信息的目的.

2.2 变异操作中的基向量与差分向量

在随机产生的三个不同个体中选择最优个体为基向量,差分向量的方向朝适应度更优的个体方向[6,7].将三个随机选择的用于变异的个体按适应度大小进行排序,适应度最优的个体记为 $\boldsymbol{x}_{r1}$,次优的个体记为 $\boldsymbol{x}_{r2}$,最差的个体记为 $\boldsymbol{x}_{r3}$,变异操作方程应以 $\boldsymbol{x}_{r1}$为变异基向量,差分向量应指向 $\boldsymbol{x}_{r2}$,即以 $\boldsymbol{x}_{r2}-\boldsymbol{x}_{r3}$为差分向量[6,7].则变异向量 $\boldsymbol{v}$ 可表示为:

$$\boldsymbol{v}=\boldsymbol{x}_{r1}+F*(\boldsymbol{x}_{r2}-\boldsymbol{x}_{r3}) \tag{1}$$

2.3 自适应变异尺度因子 F 和自适应交叉概率 CR

自适应的变异尺度因子 F 和交叉概率 CR,会促进全局搜索与局部搜索达到自适应平衡[6,7].由于种群的进化是在三个子群体中完成的,因此种群的变异、交叉均是在三个子群体中进行.在同一代的进化中,每个子群体的个体有各自的自适应变异尺度因子 F 和交叉概率 CR.

2.3.1 自适应变异尺度因子 F

自适应变异尺度因子 F 是根据差分向量变化的尺度来自适应调整 F 值.F 自适应策略可表示为:

$$F_{gi}^{t}=F_l+(F_u-F_l)\frac{f_{g\,\mathrm{middle}}^{t}-f_{g\,\mathrm{best}}^{t}}{f_{g\,\mathrm{worst}}^{t}-f_{g\,\mathrm{best}}^{t}} \tag{2}$$

式(2)中 F_{gi}^{t}为当前代 t 子群体 g 中第 i 个个体变异尺度因子,F_u 和 F_l 分别为变异尺度因子的上下限,$f_{g\,\mathrm{best}}^{t}$,$f_{g\,\mathrm{middle}}^{t}$和 $f_{g\,\mathrm{worst}}^{t}$分别为当前代 t 子群体 g 中更新第 i 个个体 F 值时随机选择的三个个体中最优、次优和最差的个体适应度.

2.3.2 自适应交叉概率 CR

自适应交叉概率 CR 根据当前代所处子群体中个体适应度与该子群体平均适应度的相对值来自适应调整 CR 值,其更新策略分两种情况考虑:一种情况是当前代所处子群体中个体适应度小于该个体所在子群体的平均适应度,即个体适应度相对较好的情况,CR 采用式(3)(a)更新策略;另一种情况是当前代所处子群体中个体适应度不小于该个体所在子群体的平均适应度,即个体适应度相对较差的情况,CR 采用式(3)(b)更新策略.

$$CR_{gi}^{t}=\begin{cases}CR_{gi}^{t-1}, & f_{gi}^{t}<\overline{f_g^{t}} \quad (a)\\ CR_l+(CR_u-CR_l)\dfrac{f_{gi}^{t}-f_{g\min}^{t}}{f_{g\max}^{t}-f_{g\min}^{t}}, & f_{gi}^{t}\geqslant\overline{f_g^{t}} \quad (b)\end{cases} \tag{3}$$

式(3)中 CR_{gi}^{t}为当前代 t 子群体 g 中第 i 个个体交叉概率,CR_u 和 CR_l 分别为交叉概率上下限,f_{gi}^{t}为当前代 t 子群体 g 中第 i 个个体的适应度,$\overline{f_g^{t}}$,$f_{g\min}^{t}$,$f_{g\max}^{t}$分别为当前代 t 子群体 g 的平均适应度、最小适应度和最大适应度.CR_{gi}^{t-1}是随着进化代数 t 的不同而逐代更新的,其更新策略为:

$$\begin{cases}CR_{gi}^{t}=CR_l+(CR_u-CR_l)*rand_i(0,1),\text{if } t=0\\ CR_{gi}^{t}=CR_{gi}^{t-1},\text{if } 0<t\leqslant t_{\max}\end{cases} \tag{4}$$

在 $f_{gi}^{t}<\overline{f_g^{t}}$ 情况下,个体适应度相对较好,该个体的交叉概率应继承上一代 CR 的调整策略,即采用式(3)(a)更新策略.第一代的 CR_{gi}^{t-1}初始值是在其约束范围内随机选取的,从第二代开始直到最大搜索代数 $t_{\max}$,CR_{gi}^{t-1}的选取均用上一代得到的 CR_{gi}^{t-2}进行更新.之所以

能够得到当前代较好适应度的个体是由于上一代对该个体交叉概率做了较好的调整,因此该个体在当前代可继续朝着此方向搜索,即该个体当前代的交叉概率继续采用上一代的交叉概率.这样继承了上一代较好的交叉概率,使交叉不再是随机性操作,而是朝着较好适应度方向的确定性随机操作.

在 $f_{gi}^{t} \geqslant \overline{f_g^t}$ 情况下,个体适应度相对较差,该个体的交叉概率应根据该个体适应度的大小进行调整,即采用式(3)(*b*)更新策略.式(3)(*b*)策略可以调整相对较差的个体朝较好适应度方向搜索.

如此继承好的交叉概率,调整差的交叉概率的方法,使交叉概率的自适应性具有了确定性的随机性质,更有利于算法的收敛.

2.4 局部搜索

为了加速算法的收敛,当达到指定代数时,对优秀个体进行局部搜索.优秀个体是从当前代三个子群体的最好个体中选出两个,局部搜索算法采用二次规划法(SQP).

3 仿真及分析

本文对 13 个经典 benchmark 函数做了仿真测试,为了使 DMSDELS 算法的测试结果更具对比性,本文将 DMSDELS 与 DE[1], PSO-w[16], PSO-cf[17], CLPSO[18], SACPDE[11], SADE[12]和 SACPMDE[6,7]进行比较.DE 采用 rand/1/bin, F = 0.5, CR = 0.9; PSO-w 的学习因子 $c_1 = c_2 = 2$,惯性权重 $w = 0.9 \sim 0.4$ 且随时间线性递减,粒子最大速 $v_{\max}$取动态变化范围的 20%; PSO-cf 的学习因子 $c_1 = c_2 = 2.01$,收缩因子 $\chi = 0.729844$; CLPSO 的学习因子 $c = 1.49445$,惯性权重 $w = 0.9 \sim 0.4$ 且随时间线性递减; SACPDE 结果参见文献[11]; SADE 参数设置参见文献[12]; SACPMDE 中 $F_l = 0.1$, $F_u = 0.9$, $CR_l = 0.1$, $CR_u = 0.6$; DMSDELS 中 $F_l = 0.1$, $F_u = 0.9$, $CR_l = 0.1$, $CR_u = 0.6$,三个子群体,两个优秀个体达到指定代数时进入局部搜索.开发环境为 MATLAB7.0,运行于 Pentium(*R*) 4 CPU 2.94GHz,512MB 内存的联想台式电脑.

附表为 13 个典型函数,变量维数、函数变量取值范围 *S*、函数的收敛精度、理论上目标函数最优值 $f_{\min}$分别给出.表 1、表 2 和表 3 分别为单峰值函数,具有多个局部极小值点的多峰值函数及具有少量局部极小值点的多峰值函数优化结果.13 个函数的最大迭代次数分别给出,搜索的最好值为 Best,搜索的平均值为 Mean,搜索的精度为 Std. Dev.. 表 4 比较了 DMSDELS 与 SACPMDE 达到收敛精度时 CPU 运行平均时间 Mean,达到收敛精度时函数评价次数 *n* 及成功率 SR.其中时间单位为秒(s),函数评价次数为平均函数评价次数,成功率指达到收敛精度的运行次数占总运行次数的比率(%).

表 1　单峰值函数算法性能比较

函数	算法	Best	Mean	Std. Dev.
f_1 (最大迭代次数 = 1500)	DE	5.2e-014	3.7e-013	3.9e-013
	PSO-w	1.8e-015	1.7e-013	4.6e-013
	PSO-cf	4.5e-045	2.3e-041	4.5e-041
	CLPSO	3.2e-013	2.7e-012	1.7e-012
	SACPDE	—	1.1e-028	1.0e-028
	SADE	2.7e-037	1.9e-035	2.3e-035
	SACPMDE	1.0e-024	6.4e-024	7.6e-024
	DMSDELS	1.4e-045	8.4e-045	7.7e-045
f_2 (最大迭代次数 = 2000)	DE	6.2e-010	3.7e-009	2.2e-009
	PSO-w	4.0e-012	6.7e-011	8.0e-011
	PSO-cf	3.3e-029	1.6e-000	4.2e-000
	CLPSO	1.6e-009	3.8e-009	1.7e-009
	SACPDE	—	1.0e-023	9.7e-024
	SADE	2.0e-014	5.8e-014	3.2e-014
	SACPMDE	3.4e-015	1.1e-014	5.3e-015
	DMSDELS	1.6e-023	6.1e-023	2.5e-023
f_3 (最大迭代次数 = 5000)	DE	1.1e-011	1.8e-010	1.5e-010
	PSO-w	2.3e-002	2.4e-001	2.2e-001
	PSO-cf	3.0e-019	3.3e + 002	1.8e + 003
	CLPSO	3.4e-002	4.2e-001	3.6e-001
	SACPDE	—	3.1e-014	5.9e-014
	SADE	3.7e-040	3.6e-037	1.1e-036
	SACPMDE	2.7e + 002	8.3e + 002	3.1e + 002
	DMSDELS	2.5e-002	7.4e-001	1.3e + 000
f_4 (最大迭代次数 = 5000)	DE	6.8e-013	3.1e-002	8.7e-002
	PSO-w	1.2e-002	7.0e-002	4.7e-002
	PSO-cf	1.5e-016	7.1e-013	2.2e-012
	CLPSO	6.9e-004	2.1e-003	1.3e-003
	SACPDE	—	0	0
	SADE	6.2e-011	2.6e-010	1.8e-010
	SACPMDE	6.3e-013	4.1e-012	2.4e-012
	DMSDELS	3.5e-024	1.7e-022	2.8e-022
f_5 (最大迭代次数 = 20000)	DE	0	3.5e-031	2.5e-030
	PSO-w	1.1e-002	1.8e + 003	1.3e + 004
	PSO-cf	1.9e-012	7.3e + 003	2.5e + 004
	CLPSO	1.7e-001	3.6e + 001	3.1e + 001
	SACPDE	—	0	0
	SADE	0	4.5e-030	7.5e-030
	SACPMDE	1.3e + 001	1.8e + 001	1.4e + 000
	DMSDELS	4.4e-011	1.3e-001	7.3e-001
f_6 (最大迭代次数 = 3000)	DE	2.0e-003	4.7e-003	1.3e-003
	PSO-w	3.0e-003	6.3e-003	2.2e-003
	PSO-cf	9.9e-004	2.5e-003	1.4e-003
	CLPSO	1.0e-003	3.0e-003	9.7e-004
	SACPDE	—	3.2e-003	7.5e-004
	SADE	1.5e-003	3.2e-003	8.4e-004
	SACPMDE	2.8e-003	4.8e-003	1.0e-003
	DMSDELS	8.1e-004	2.3e-003	5.1e-004

从表 1 ~ 3 可以看出,DMSDELS 与 DE 比较,对于 13 个函数,除 DE 求解函数 f_3, f_5, f_{12}性能较好,DMSDELS 求解其他函数均比 DE 性能优.DMSDELS 与 PSO-w, PSO-cf 和 CLPSO 比较,对于 13 个函数除 CLPSO 求解函数 f_7 性

能较好,求解函数 f_3 性能稍好,PSO-w 求解函数 f_3 性能稍好外,DMSDELS 求解其他函数均比三种粒子群优化算法性能优.DMSDELS 与 SACPDE 比较,在这组结果中,两种算法各有优势.对于 13 个函数(具体参见文献[11]),对于单峰值函数 f_3,f_4,f_5 的求解,SACPDE 性能优;对于多个局部极小值点的多峰值函数 f_7 的求解,SACPDE 性能优;对于少量局部极小值点的多峰值函数的求解,SACPDE 只对求解函数 f_{12}性能优.由此可见,DMSDELS 与 SACPDE 在求解单峰值函数和多个局部极小值点的多峰值函数各具优势,而在处理少量局部极小值点的多峰值函数时 DMSDELS 性能更优.DMSDELS 与 SADE 比较,对于 13 个函数,除 SADE 对求解函数 f_3,f_5,f_7,f_{12}性能较好外,对其他函数的求解 DMSDELS 均比 SADE 性能优.DMSDELS 与 SACPMDE 比较,对于 13 个函数,除 SACPMDE 对求解函数 f_7 性能较好,函数 f_{12}性能稍好外,对其他函数的求解 DMSDELS 均比 SACPMDE 性能优.

表 2 多个局部极小值点的多峰值函数算法性能比较

函数	算法	Best	Mean	Std.Dev.
f_7 (最大迭代次数 = 5000)	DE	1.0e+001	8.1e+001	3.2e+001
	PSO-w	8.0e+000	2.1e+001	8.0e+000
	PSO-cf	2.7e+001	6.2e+001	1.8e+001
	CLPSO	2.9e-010	1.3e-009	8.6e-010
	SACPDE	—	0	0
	SADE	0	3.3e-002	1.8e-001
	SACPMDE	0	0	0
	DMSDELS	6.0e+000	1.3e+001	3.7e+000
f_8 (最大迭代次数 = 1500)	DE	3.4e-015	3.7e-014	4.1e-014
	PSO-w	8.9e-015	2.2e+000	5.5e+000
	PSO-cf	1.6e-032	1.7e+001	1.8e+001
	CLPSO	8.8e-012	4.8e-011	4.0e-011
	SACPDE	—	6.6 e-030	7.9 e-030
	SADE	1.6e-032	3.5e-003	1.9e-002
	SACPMDE	1.6e-032	6.0e-032	4.6e-032
	DMSDELS	1.6e-032	1.6e-032	5.6e-048
f_9 (最大迭代次数 = 1500)	DE	4.1e-014	2.9e-013	2.9e-013
	PSO-w	8.2e-007	5.7e+002	3.6e+002
	PSO-cf	1.3e-032	2.4e+002	2.4e+002
	CLPSO	1.2e-010	6.4e-010	4.5e-010
	SACPDE	—	5.0e-029	3.9e-029
	SADE	8.1e-032	9.0e-030	2.0e-029
	SACPMDE	1.0e-031	4.1e-031	2.6e-031
	DMSDELS	1.3e-032	1.3e-032	5.6e-048

表 3 少量局部极小值点的多峰值函数算法性能比较

函数	算法	Best	Mean	Std.Dev.
f_{10} (最大迭代次数 = 100)	DE	0.39789	0.39789	1.1e-008
	PSO-w	0.39789	0.39789	2.3e-012
	PSO-cf	0.39789	0.39789	5.3e-012
	CLPSO	0.39789	0.39789	1.1e-013
	SACPDE	—	0.39789	2.3e-008
	SADE	0.39789	0.39789	4.9e-007
	SACPMDE	0.39790	0.39790	1.3e-006
	DMSDELS	0.39789	0.39789	0
f_{11} (最大迭代次数 = 100)	DE	3	3	3.3e-015
	PSO-w	3	3	2.5e-011
	PSO-cf	3	3	2.0e-011
	CLPSO	3	3	5.5e-013
	SACPDE	—	3	1.7e-015
	SADE	3	3	5.7e-015
	SACPMDE	3	3	6.0e-014
	DMSDELS	3	3	2.2e-015
f_{12} (最大迭代次数 = 100)	DE	−10.403	−10.403	2.1e-007
	PSO-w	−4.607	−2.137	8.3e-001
	PSO-cf	−10.403	−6.474	3.6e+000
	CLPSO	−10.339	−9.403	1.1e+000
	SACPDE	—	−10.403	4.9e-007
	SADE	−10.403	−10.403	4.9e-005
	SACPMDE	−10.403	−10.399	1.6e-002
	DMSDELS	−10.403	−10.180	1.2e+000
f_{13} (最大迭代次数 = 100)	DE	−10.536	−10.536	3.2e-006
	PSO-w	−6.632	−2.200	1.0e+000
	PSO-cf	−10.536	−8.112	3.5e+000
	CLPSO	−10.457	−9.473	1.3e+000
	SACPDE	—	−10.536	5.8e-006
	SADE	−10.536	−10.536	3.9e-003
	SACPMDE	−10.536	−10.536	2.4e-004
	DMSDELS	−10.536	−10.536	6.5e-011

表 4 算法的运行时间(s)、函数评价次数 n 以及成功率 SR(%)比较

DMSDELS (SACPMDE)	$f_1^{\uparrow}$	$f_2^{\uparrow}$	$f_3^{\uparrow}$	$f_4^{\uparrow}$	$f_5^{\uparrow}$	$f_6^{\uparrow}$	$f_7^{\downarrow}$
Mean	7.23 (11.95)	28.40 (16.27)	— (—)	269.41 (63.38)	2435.61 (—)	— (—)	— (20.49)
n	19992 (56913)	48487 (74904)	— (—)	157039 (292490)	490383 (—)	— (—)	— (93093)
SR	100 (100)	100 (100)	0 (0)	100 (100)	96.67 (0)	0 (0)	0 (100)
DMSDELS (SACPMDE)	$f_8^{\uparrow}$	$f_9^{\uparrow}$	$f_{10}^{\uparrow}$	$f_{11}^{\uparrow}$	f_{12}	$f_{13}^{\uparrow}$	
Mean	8.57 (6.35)	12.61 (8.38)	0.58 (0.94)	0.61 (0.75)	1.09 (1.41)	1.07 (1.48)	
n	21387 (26922)	28577 (35342)	2250 (4735)	2243 (3735)	3412 (5251)	3207 (5183)	
SR	100 (100)	100 (100)	100 (100)	100 (100)	96.67 (100)	100 (100)	

从表 4 可以看出,对于大部分函数的求解,DMSDELS 相较 SACPMDE 的 CPU 运行时间稍长;但函数评价次数相较 SACPMDE 均较少.对于没有成功收敛的函数(成功率 SR 为 0),本文没有讨论其运行时间与函数评价次数.由于编写程序本身的优化程度存在差异,因此,可从运行时间与函数评价次数综合评测算法的运行速度.另外,DMSDELS 与 SACPMDE 算法性能比较的 t_test 指标(详见 Matlab)用表 4 中的"↑/↓"分别表示 DMSDELS 优/劣于 SACPMDE.无标识的表示两算法性能相当.结合函数的 Mean,Best,Std.Dev 以及求解的成功

率,可以看出,DMSDELS 总体优于 SACPMDE.

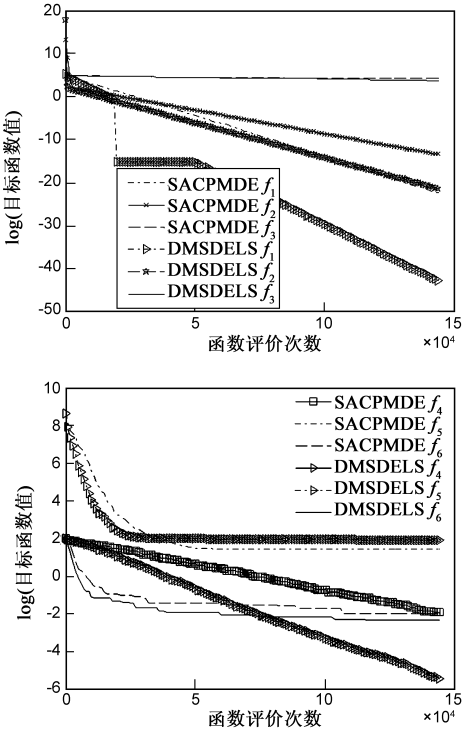

图1　算法的收敛曲线

图 1 给出 DMSDELS 与 SACPMDE 的运行一次(最接近平均最优值的那一次)的收敛情况.横坐标为函数评价次数,纵坐标为函数值的对数值.

综上,从综合指标评测算法性能,DMSDELS 具有较好的全局探索能力和避免陷入局部最优解的能力,该算法在搜索全局最优解与保持种群多样性之间能够做到很好的平衡.

4　结论

本文将带局部搜索的动态多群体自适应差分进化算法用于函数优化,通过对 13 个典型 benchmark 函数进行测试,结果表明:该算法比 DE、PSO-w、PSO-cf、CLPSO、SACPMDE,SADE 算法总体性能优,SACPDE 与该算法各具优势.可见,带局部搜索的动态多群体自适应差分进化算法有较好的收敛性与收敛精度,有较强的跳出局部最优解的能力.

附表

13 个 benchmark 函数

函数	n	S	收敛精度	$f_{\min}$
$f_1(\boldsymbol{x})=\sum_{i=1}^{n}x_i^2$	30	$[-5.12,5.12]^n$	1e-6	0
$f_2(\boldsymbol{x})=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	1e-6	0
$f_3(\boldsymbol{x})=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	$[-100,100]^n$	1e-6	0
$f_4(\boldsymbol{x})=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	30	$[-100,100]^n$	1e-6	0
$f_5(\boldsymbol{x})=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	30	$[-30,30]^n$	1e-6	0
$f_6(\boldsymbol{x})=\sum_{i=1}^{n}ix_i^4+rand[0,1)$	30	$[-1.28,1.28]^n$	1e-6	0
$f_7(\boldsymbol{x})=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	30	$[-5.12,5.12]^n$	1e-3	0
$f_8(\boldsymbol{x})=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\cdot[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	30	$[-50,50]^n$	1e-3	0
$f_9(\boldsymbol{x})=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\cdot[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2\cdot[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	1e-3	0
$f_{10}(\boldsymbol{x})=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos(x_1)+10$	2	$[-5,15]^n$	0.398+1e-4	0.398
$f_{11}(\boldsymbol{x})=[1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2,2]^n$	3+1e-4	3
$f_{12}(\boldsymbol{x})=-\sum_{i=1}^{7}[(\boldsymbol{x}-a_i)\cdot(\boldsymbol{x}-a_i)^{\mathrm{T}}+c_i]^{-1}$	4	$[0,10]^n$	−10	−10.4029
$f_{13}(\boldsymbol{x})=-\sum_{i=1}^{10}[(\boldsymbol{x}-a_i)\cdot(\boldsymbol{x}-a_i)^{\mathrm{T}}+c_i]^{-1}$	4	$[0,10]^n$	−10	−10.5364

参考文献:

[1] Rainer Storn, Kenneth Price. Differential evolution—a simple

and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341 – 359.

[2] Rainer Storn. Designing nonstandard filters with differential evolution[J]. IEEE Signal Processing Magazine, 2005, 22(1): 103 – 106.

[3] Varadarajan M, Swarup KS. Network loss minimization with voltage security using differential evolution[J]. Electric Power Systems Research, 2008, 78(5): 815 – 823.

[4] Swagatam Das, Ajith Abraham, Amit Konar. Automatic clustering using an improved differential evolution algorithm[J]. IEEE Transaction on Systems, Man and Cybernetics, 2008, 38(1): 218 – 236.

[5] 吴亮红,王耀南,袁小芳,周少武.自适应二次变异差分进化算法[J].控制与决策,2006,21(8):898 – 901.
Wu Lianghong, Wang Yaonan, Yuan Xiaofang, Zhou Shaowu. Differential evolution algorithm with adaptive second mutation [J]. Control and Decision, 2006, 21(8): 898 – 901. (in Chinese)

[6] Wu Lianghong, Wang Yaonan. Self-adapting control parameters modified differential evolution for trajectory planning manipulator[J]. Control Theory Application, 2007, 5(4): 365 – 373.

[7] 吴亮红.差分进化算法及应用研究[D].湖南:湖南大学电气与信息工程学院,2007.65 – 71.
Wu Lianghong. The Research and Applications of Differential Evolution Algorithm[D]. Hunan: Control Theory and Control Engineering, Hunan University, 2007, 65 – 71. (in Chinese)

[8] 赵光权,彭喜元,孙宁.基于混合优化策略的微分进化改进算[J].电子学报,2006,34(12A):2402 – 2405.
Zhao Guangquan, Peng Xiyuan, Sun Ning. A modified differential evolution algorithm with hybrid optimization strategy[J]. Acta Electronic Sinica, 2006, 34(12A): 2402 – 2405. (in Chinese)

[9] 赵光权,彭喜元,孙宁.带局部增强算子的微分进化改进算法[J].电子学报,2007,35(5):849 – 853.
Zhao Guangquan, Peng Xiyuan, Sun Ning. A modified differential evolution algorithm with local enhanced operator[J]. Acta Electronic Sinica, 2007, 35(5): 849 – 853. (in Chinese)

[10] Janez Brest, Viljem Žumer, Marjan Sepesy Maučec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization[A]. IEEE Congress on Evolutionary Computation[C]. Vancouver: IEEE, 2006. 215 – 222.

[11] Janez Brest, Sašo Greiner, Borko Boškovic, Viljem Žumer. 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.

[12] Qin A K, Suganthan P N. Self-adaptive differential evolution algorithm for numerical optimization[A]. IEEE Congress on Evolutionary Computation[C]. Edinburgh: IEEE, 2005. 1785 – 1791.

[13] Huang V L, Qin A K, Suganthan P N. Self-adaptive differential evolution algorithm in constrained real-parameter optimization[A]. IEEE Congress on Evolutionary Computation[C]. Vancouver: IEEE, 2006. 17 – 24.

[14] Zhang Xuexia, Chen Weirong, Cai Wenzhao, Dai Chaohua. Dynamic multi-groups self-adaptive differential evolution algorithm with local search for reactive power optimization [A]. The 2009 Asia-Pacific Power and Energy Engineering Conference[C]. Wuhan: IEEE, 2009. 2911 – 2914.

[15] Zhang Xuexia, Chen Weirong, Dai Chaohua, Cai Wenzhao. Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization[J]. International Journal of Electrical Power & Energy Systems, 2010, 32(5): 351 – 357.

[16] Shi Y, Eberhart R. Empirical study of particle swarm optimization[A]. Proceedings of the 1999 Congress on Evolutionary Computation[C]. Washington: IEEE, 1999. 1945 – 1950.

[17] Clerc M, Kennedy J. The particle swarm – explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transaction on Evolutionary Computation, 2002, 6(1): 58 – 73.

[18] Liang JJ, Qin A K, Suganthan P N, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transaction on Evolutionary Computation, 2006, 10(3): 7 – 82.

作者简介:

张雪霞 女,1979 年生于天津.西南交通大学电气工程学院博士研究生,研究方向为计算智能、电力系统及其自动化.
E-mail: zxxswjtu@gmail.com

陈维荣 男,1965 年出生于四川内江.西南交通大学电气工程学院教授,博士,博士生导师,研究方向为电力系统及其自动化、智能信息处理等.

戴朝华 男,1973 年出生于湖南新化.西南交通大学电气工程学院,博士,研究方向为智能信息处理、模式识别.