*

自适应局部增强微分进化改进算法

黄仁全[1]，　靳　聪[2]，　贺筱军[1]，　吕文平[1]
(1. 空军工程大学导弹学院,陕西　三原　713800;2. 空军工程大学电讯工程学院,陕西　西安　710011)

摘要　在分析微分进化算法基本原理基础上,为加快算法收敛速度,对其交叉概率和交叉因子进行自适应调整改进;为增强算法局部搜索能力,引入局部增强算子和扰动因子改进算法,即自适应局部增强微分进化算法。选取 5 个典型测试函数,将改进后算法与 PSO 算法、微分进化算法和局部增强微分进化算法仿真比较。仿真结果表明:自适应局部增强微分进化算法为收敛时间最短、迭代次数最少的优化算法,验证了算法改进的有效性。

关键词　微分进化算法;交叉概率;自适应调整;增强算子
DOI　10.3969/j.issn.1009-3516.2011.03.018
中图分类号　TP24　　**文献标识码**　A　　**文章编号**　1009-3516(2011)03-0084-06

优化是人们在科学研究、工程技术和经济管理等诸多领域中经常碰到的问题,它所研究的就是在众多方案中寻找最优方案,因此求解全局最优化成为一个重要课题。在寻求全局最优解方面,随机算法表现出较好的性能,目前几种知名随机算法包括遗传算法、模拟退火算法、粒子群算法、演化策略和随机微分方程法等。1995 年 Rainer Storn 和 Kenneth Price 提出了微分进化算法(Differential Evolution Algorithm,DE)[1-3],该算法具有收敛速度快、鲁棒性强、控制参数少等优点。

微分进化算法是一种实数编码的进化算法,在收敛速度和稳定性方面都超过了当前较为流行的几种随机算法。为进一步挖掘微分进化算法潜能,目前对微分进化算法改进作了一些研究,如 Hui Yuan 和 Jouni Lampinen 通过改进变异操作方式而提出一种三角变异差分进化算法[4-5],Anyong Qing 提出的动态微分进化算法[6-7]等。上述微分进化算法的改进方法不同程度地提高了算法收敛速度,但算法依然具有较大的提升空间。本文提出自适应局部增强微分进化算法,有效提高了算法收敛性能。

1　基本微分进化算法

DE 法继承了"优胜劣汰"的思想,属于进化的一种。其基本思想是:对种群中的每个个体 i,从当前种群中随机选择 3 个点,以其中的一个点为基础、另 2 个点为参照作一个扰动,所得点与个体 i 交叉后进行"自然选择",保留较优者,实现种群的进化。微分进化算法的核心思想是利用随机偏差扰动产生新的中间个体,因此微分进化算法具有较强的收敛性。假设待优化问题为$\min\limits_{x\in R^n} f(x)$[2],算法主要步骤如下:

1)初始化。设种群规模为 N,变量个数为 m,交叉概率 P_c,交叉因子 P_m;进化代数 $t=0$,自变量的下界 l_b 和 u_b,随机生成初始种群 $\boldsymbol{X}(0)=\{\boldsymbol{X}_1(0),\boldsymbol{X}_2(0),\cdots,\boldsymbol{X}_N(0)\}$,其中 $\boldsymbol{X}_i(0)=\{x_1^i(0),x_2^i(0),\cdots,x_m^i(0)\}$。

2)个体评价。计算每个个体 $\boldsymbol{X}_i(t)$ 的目标适应度函数值 $f(\boldsymbol{X}_i(t))$,并记录。

3)繁殖。对于种群中的每个个体 $\boldsymbol{X}_i(t)$,随机生成 3 个互不相同的随机整数,r_1、r_2、$r_3\in\{1,2,\cdots,N\}$ 和

* 收稿日期:2010-07-14
基金项目:国防科技重点实验室基金资助项目
作者简介:黄仁全(1983-),男,湖南郴州人,博士生,主要从事防空作战建模与仿真研究.
E-mail:huangrenquan@126.com

随机整数 $j_r \in \{1,2,\cdots,m\}$。其中:$x_j^{(i)'}(t)=\begin{cases} x_j^{(r_1)}(t)+P_m(x_j^{(r_2)}(t)-x_j^{(r_3)}(t)) & ,\ \text{若 rang}<P_c \text{ 或 } j=j_r \\ x_j^{(i)}(t) & ,\ \text{否则} \end{cases}$。

4)选择。$\boldsymbol{X}_i(t+1)=\begin{cases} \boldsymbol{X}'_i(t), \text{若} f(\boldsymbol{X}'_i(t))<f(\boldsymbol{X}_i(t)) \\ \boldsymbol{X}_i(t), \text{否则} \end{cases}$。

5)终止检验。如果种群 $\boldsymbol{X}_i(t+1)$ 满足终止准则,则输出 $\boldsymbol{X}_i(t+1)$ 中有最小目标值的个体作为最优解,否则,转 2)。算法计算流程见图 1。微分进化算法主要有 3 个控制参数:种群规模 N,交叉概率 P_c 和交叉因子 P_m。依据文献[8],N 通常取 $5m-10m$,其中 m 为变量个数,P_c 通常取 0.5,P_m 通常取 0.1,当变量个数较大时,N 取 m 甚至 $m/4$ 即可。

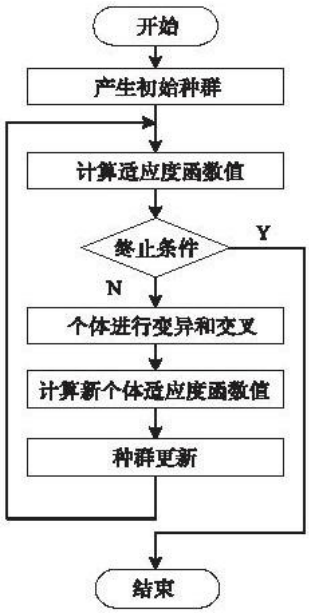

图 1　DE 算法流程

Fig. 1　The flow chart of DE algorithm

2　微分进化算法改进

本节在基本微分进化算法分析基础上,为实现交叉概率 P_c 和交叉因子 P_m 的动态调整及增强算法局部搜索能力,分别对算法进行自适应和局部增强改进。

2.1　算法自适应改进

微分进化算法作为进化算法的一种,由算法寻优的基本原理可知:①若交叉概率取值 P_c 越大,新个体产生的速度就越快,同时个体被破坏的可能性就会增加,使得具有高适应度值的个体结构很快被破坏;但是如果 P_c 过小,会使搜索过程缓慢,以致停滞不前。②若交叉因子 P_m 取值越大,算法产生新的种群越快,随机性越强,若 P_m 过大,那么微分进化算法就变成了纯粹的随机搜索算法;如果 P_m 过小,则算法不易产生新的个体。

在基本微分进化算法中,交叉概率 P_c 和交叉因子 P_m 为恒定值,用于求解复杂多变量优化问题时,其求解效率不高。为提高算法求解效率,引入自适应调整的思想。交叉概率 P_c 和交叉因子 P_m 依据个体的适应度值自适应进行调整,当群体有陷入局部最优解的趋势时,就相应地提高 P_c 和 P_m;当群体在解空间发散时,就降低 P_c 和 P_m。同时,对于适应值高于群体平均适应值的个体,对应于较低的 P_c 和 P_m,使该解得以保护进入下一代;而低于平均适应值的个体,相对应于较高的 P_c 和 P_m,使该解被淘汰掉。因此,自适应微分进化算法在保持群体多样性的同时,保证算法的收敛能力,有效地提高了微分进化算法寻优能力。

为相应地提高群体中表现优良的个体的交叉概率和交叉因子,使得无论在何种情况下都不会处于一种近似停滞不变的状态,个体的交叉概率和交叉因子按式(1)-(2)调整[9]:

$$P_c=\begin{cases} P_{c1}-\dfrac{(P_{c1}-P_{c2})(f'-f_{\text{avg}})}{f_{\max}-f_{\text{avg}}} & ,\ f'\geqslant f_{\text{avg}} \\ P_{c1} & ,\ f'<f_{\text{avg}} \end{cases} \tag{1}$$

$$P_m=\begin{cases} P_{m1}-\dfrac{(P_{m1}-P_{m2})(f_{\max}-f)}{f_{\max}-f_{\text{avg}}} & ,\ f\geqslant f_{\text{avg}} \\ P_{m1} & ,\ f<f_{\text{avg}} \end{cases} \tag{2}$$

式中:P_{c1} 为交叉概率较大值,取 0.7-0.9;P_{c2} 为交叉概率较小值,取 0.4-0.6;P_{m1} 为交叉因子较大值,取 0.08-0.1;P_{m2} 为交叉因子较小值,取 0.01-0.05;$f_{\max}$ 为群体中的最大适应度值;f_{avg} 为每代群体适应度平均值;f' 为 $\boldsymbol{X}_{r_2}(t)$、$\boldsymbol{X}_{r_3}(t)$ 中较大的适应度值;f 为 $\boldsymbol{X}_{r_1}(t)$ 的适应度值。

通过式(1)、式(2)可依据群体适应度值,分别对交叉概率 P_c 和交叉因子 P_m 自适应调整,从而加快算法寻优速度。

2.2 算法局部增强改进

微分进化算法通过随机偏差扰动产生新的中间个体,因此算法的局部搜索能力较弱,在逼近全局最优解时,仍需要经过多次迭代才能获得最优值,从而影响了算法的寻优速度。

在对微分进化算法特性进行分析的基础上,文献[10]提出了一种加速收敛的微分进化改进算法——局部增强算子的微分进化算法(MPDE)。其基本思想是:在按基本微分进化算法得到新种群后,以 M_P($0 < M_P < 1$)概率对新种群中的部分个体(不含当前最优个体)重新赋值,并使这部分个体分布在当前种群中的最优个体附近,引入局部增强算子式(3),以增强这部分个体的贪婪性,加快算法收敛速度[11]:

$$\boldsymbol{X}_{i,t+1} = \boldsymbol{X}_{\text{best},t+1} + P_l(\boldsymbol{X}_{r_1,t+1} - \boldsymbol{X}_{r_2,t+1}) \tag{3}$$

式中 $\boldsymbol{X}_{i,t+1}$ 为增强后个体,替换旧个体 $\boldsymbol{X}_{r_1,t+1}$ 和 $\boldsymbol{X}_{r_2,t+1}$,$\boldsymbol{X}_{\text{best},t+1}$ 为当前最优个体,r_1,r_2 满足 $r_1 = r_2 \neq i$,P_l 为扰动因子,通常取 0.5 左右[10]。

当一定数量的个体被选中并按局部增强算子更新时,其效果相当于使这部分个体在当前种群中的最优个体附近随机扰动。为了控制进行局部增强个体的数量,MPDE 算法引入了新参数——局部增强算子 M_P。当 $M_P = 0$ 时,没有个体被选中进行局部增强,算法与基本微分进化算法一致;当 $M_P = 1$ 时,算法每迭代一次,种群中除最优个体外其它个体都按式(3)更新一次。引入局部增强算子以后,算法每迭代一次,就会使部分个体变得更贪婪,并在当前最优解附近寻优,同时,并不是所有个体都变得更贪婪,从而使得算法的贪婪性又受到一定限制,以避免影响算法的良好收敛性。

对微分进化算法局部增强的实质是:使种群中的部分个体在当前最优个体附近扰动寻优。在保证种群多样性的同时,增加优良个体的贪婪性,以保证算法又快又好地找到全局最优解。通过引入扰动因子 P_l,可以增强算法的局部搜索能力,加快算法的收敛速度,尤其是在逼近全局最优解时,可以减少收敛所需迭代次数。

2.3 自适应局部增强微分进化算法流程

自适应局部增强微分进化算法(ADMPDE)就是在基本微分进化算法基础上,对其交叉概率 P_c 和交叉因子 P_m 分别按式(1)和式(2)自适应调整;为增强算法对优良个体的贪婪性,按式(3)对算法进行局部增强。因此,自适应局部增强微分进化算法的流程,就是在基本算法基础上增加交叉概率和交叉因子的自适应操作,同时增加局部增强功能,基本流程见图 2。因此,ADMPDE 算法既具有自适应调整的能力,同时又具有 MPDE 算法的优点,加快了寻优速度。

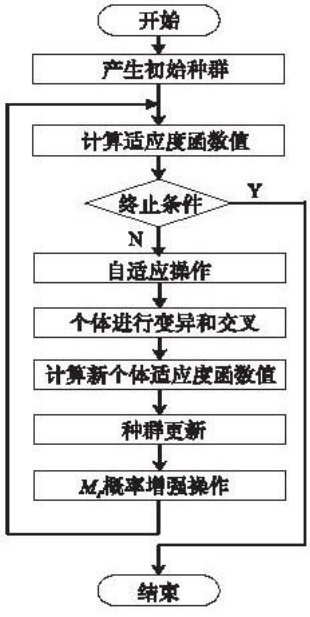

图 2 ADMPDE 算法流程

Fig. 2 The flow chart of ADMPDE algorithm

3 仿真测试

为测试自适应局部增强微分进化算法的性能,选择典型寻优算法测试函数,分别与 PSO 算法、DE 算法和带局部增强算子的微分进化算法(MPDE)比较。

3.1 测试函数选择

在算法测试中,常用的测试函数有[11-13]:Sphere 函数(f_1)、Rosenbrock 函数(f_2)、Rastrigin 函数(f_3)、Griewank 函数(f_4)和 Schaffer 函数(f_5)等,测试函数相关描述见表 1,求解函数在定义域内的最小值。其中,Sphere 函数和 Rosenbrock 函数为单峰值函数,而 Rastrigin 函数、Griewank 函数和 Schaffer 函数为多峰值函数,具有多个局部最优解。选择以上测试函数,能从不同方面测试算法的收敛性。

表 1　测试函数

Tab. 1　The test functions

函数	表达式	维数	定义域	最优值	终止条件
f_1	$\sum_{i=1}^{n} x_i^2$	30	$[-100,100]^n$	0	0.01
f_2	$\sum_{i=1}^{n} (100(x_{i+1} - x_i^2))^2 + (x_i - 10^2)$	30	$[-30,30]^n$	0	100
f_3	$\sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12,5.12]^n$	0	100
f_4	$\frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600,600]^n$	0	0.1
f_5	$0.5 + \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	2	$[-100,100]^n$	0	10^{-5}

3.2　仿真试验分析

为测试 ADMPDE 算法的收敛性能,将它与 PSO 算法、MPDE 算法和 DE 算法进行比较。4 种算法取相同的种群规模,PSO 算法参数设置依据文献[10]取值。在仿真测试中,选取以下参数作为评价标准:搜索成功率(ps)、最小进化代数(mingen)、最大进化代数(maxgen)、平均进化代数(avggen)和平均进化时间(avgtime)。算法通过 Matlab 测试,分别编写 PSO 算法、DE 算法、MPDE 算法和 ADMPDE 算法程序:PSO(c_1, c_2, w, l_b,u_b, nofv, popsize)、DE(nofv, l_b, u_b, popsize, pm, pc)、MPDE (nofv, l_b, u_b, popsize, scalfac, pc, mp)和 ADMPDE (nofv, l_b, u_b, popsize, pl, mp, p_{c1}, p_{c2}, p_{m1}, p_{m2})。算法函数中相关参数含义及取值见表 2。

表 2　函数参数

Tab. 2　The parameters of the programs

参数	含义	取值
c_1、c_2	学习因子	$c_1 = c_2 = 2$
w	加权系数	0.715 6
nofv	变量个数	函数维数 30 或 2
opsize	种群规模	90
l_b、u_b	变量下界和上界	参考函数定义域
pm	交叉因子	0.1
pc	交叉概率	0.5
mp	增强因子	0.01
P_1	扰动因子	0.5
P_{c1}、P_{c2}	交叉概率	$P_{c1} = 0.8$, $P_{c2} = 0.5$
P_{m1}, P_{m2}	交叉因子	$P_{m1} = 0.09$, $P_{m2} = 0.03$

若算法迭代 1 500 代还未收敛,认为算法收敛失败。每种算法进行 20 次仿真,其结果分别见表 3 – 7。

表 3　f_1 函数仿真结果

Tab. 3　The simulation result of the function f_1

算法	ps/(%)	mingen	maxgen	avggen	avgtime/s
PSO	100	443	612	521	2.434 6
DE	100	411	440	432	0.897 1
MPDE	100	236	260	248	0.470 9
ADMPDE	100	145	289	182	0.357 2

表 4 f_2 函数仿真结果

Tab. 4 The simulation result of the function f_2

算法	ps/(%)	mingen	maxgen	avggen	avgtime/s
PSO	100	482	1 225	783	3. 651 4
DE	100	451	917	608	0. 987 9
MPDE	100	59	336	93	0. 493 7
ADMPDE	100	42	127	66	0. 413 6

表 5 f_3 函数仿真结果

Tab. 5 The simulation result of the function f_3

算法	ps/(%)	mingen	maxgen	avggen	avgtime/s
PSO	100	327	573	484	3. 286 1
DE	100	274	477	395	0. 944 6
MPDE	100	130	193	168	0. 731 5
ADMPDE	100	75	143	97	0. 504 4

表 6 f_4 函数仿真结果

Tab. 6 The simulation result of the function f_4

算法	ps/(%)	mingen	maxgen	avggen	avgtime/s
PSO	100	418	561	519	4. 352 4
DE	100	391	421	406	1. 852 0
MPDE	100	206	258	226	1. 424 4
ADMPDE	100	127	213	149	1. 352 1

表 7 f_5 函数仿真结果

Tab. 7 The simulation result of the function f_5

算法	ps/(%)	mingen	maxgen	avggen	avgtime/s
PSO	100	214	465	376	1. 145 9
DE	100	189	658	462	0. 866 8
MPDE	100	104	191	158	0. 519 4
ADMPDE	100	67	93	78	0. 435 3

通过仿真分析,ADMPDE 算法收敛最快,迭代时间最少。ADMPDE 算法迭代次数比 MPDE 算法减少 36%(按平均收敛次数计算),而其收敛时间减少 15%。图 3 为 PSO、DEMPDE、ADMPDE 4 种算法对 f_1 函数仿真结果,从图形中可比较直观看出 ADMPDE 算法收敛最快,且算法迭代次数分别为:509、424、252 和 164。因此,无论从算法迭代次数或算法收敛时间比较,4 种算法收敛优劣顺序为:ADMPDE 算法、MPDE 算法、DE 算法和 PSO 算法。

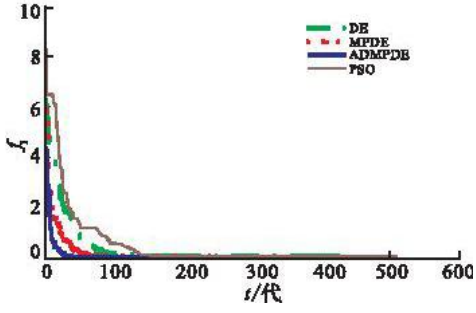

图 3 收敛性比较

Fig. 3 Comparison of different algorithms

4 结束语

微分进化算法为收敛性较好的优化算法,论文通过分析微分进化算法的特点,对算法进行自适应和局部增强改进,提出了自适应局部增强微分进化算法。通过与 PSO 算法、DE 算法、MPDE 算法仿真比较,得出 ADMPDE 算法为收敛速度最快,时间最短的算法。算法的改进丰富了微分进化算法理论,算法的改进研究有待于进一步深入。

参考文献:

[1] 阳明盛,罗长童. 最优化原理、方法及其求解软件[M]. 北京:科学出版社,2006.

YANG Mingsheng, LUO Changtong. The theories, methods and dissolving software of optimization [M]. Beijing: Science press, 2006. (in Chinese)

[2] 黄仁全,李为民,周晓光,等.基于微分进化算法的防空导弹火力分配研究[J].空军工程大学学报:自然科学版,2009,10(5):41 -44.
HUANG Renquan, LI Weimin, ZHOU Xiaoguang, et al. Research on firepower distribution model of surface to air missile based on differential evolution algorithm [J]. Journal of air force engineering university: natural science edition, 2009, 10(5):41 -44. (in Chinese)

[3] 黄仁全,李为民,雷中原.多静态红外传感器反导预警部署优化研究[J].传感器与微系统,2010,29(5):31 -33.
HUANG Renquan, LI Weimin, LEI Zhongyuan. Research on deployment optimization to multi - static infrared sensors in early warning of ballistic missile[J]. Transducer and microsystem technologies, 2010, 29(5): 31 -33. (in Chinese)

[4] Goldberg D E. Genetic algorithms: in search optimization and machine learning[M]. New York: Addison wesely press, 1989.

[5] Fan Huiyuan , Jouni Lampinen. A trigonometric mutation operation to differential evolution [J]. Journal of global optimization, 2003, 27(10): 105 -129.

[6] 周晓光,李为民,陈刚.一种近正交试验设计方法[J].空军工程大学学报:自然科学版,2010,11(3):84 -88.
ZHOU Xiaoguang, LI Weimin, CHEN Gang. A method of nearly orthogonal exprimental design [J]. Journal of air force engineering university: natural science edition, 2010, 11(3):84 -88. (in Chinese)

[7] Qing Anyong . Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems [J]. IEEE transactions on geoscience and remote sensing, 2006, 44(1): 116 -125.

[8] Storn R, Price K. DE:a simple and efficient adaptive scheme for global optimization over continuous space[J]. Technical report, 1995, 25(6):95 -102.

[9] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithm[J]. IEEE transactions on SMC, 1994, 24(4):413 -419.

[10] 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 electronica sinica, 2007, 35(5): 849 -853. (in Chinese)

[11] Adleman L M. Molecular computation of solutions to combinatorial problems [J]. Science, 1994, 266:1021 -1024.

[12] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection[J]. Information processing letters, 2003, 85(6): 317 -325.

[13] Adlelson Velsky G M, Landis E M. An algorithm for organization of information [M]. New York: Soviet mathematics doklady press, 1962.

(编辑:田新华)

A Modified Differential Evolution Algorithm with Adaptive and Local Enhanced Operator

HUANG Ren - quan[1], JIN Cong[2], HE Xiao - jun[1], LÜ Wen - ping[1]

(1. Missile Institute, Air Force Engineering University , Sanyuan 713800, Shaanxi, China; 2. Telecommunication Engineering Institute, Air Force Engineering University, Xi'an 710077, China)

Abstract: The differential evolution algorithm is robust, easy to use, and requires few control parameters. However, as to the local optimizing ability, it is limited. Based on the principium analysis of the algorithm, the adaptive modification of the cross rate and the cross operator is proposed to improve the efficiency of the algorithm. To enhance the local optimizing, the local enhanced operator and the disturbed operator are proposed. Numerical study is carried out using five benchmark functions. Compared with the PSO algorithm, DE and MPDE, the ADMPDE is the most efficient algorithm of all, which verifies that the modification is effective.

Key words: differential evolution algorithm; cross rate; adaption; local enhanced operator