具有局部搜索策略的差分进化算法

谭 跃[1,2],谭冠政[1],涂 立[3]
TAN Yue[1,2],TAN Guan-zheng[1],TU Li[3]

1.中南大学 信息科学与工程学院,长沙 410083
2.湖南城市学院 物理与电信工程系,湖南 益阳 413000
3.湖南城市学院 计算机系,湖南 益阳 413000
1.School of Information Science and Engineering,Central South University,Changsha 410083,China
2.Department of Physics and Telecom Engineering,Hunan City University,Yiyang,Hunan 413000,China
3.Department of Computer Science,Hunan City University,Yiyang,Hunan 413000,China
E-mail:tanyue7409@163.com

TAN Yue,TAN Guan-zheng,TU Li.Differential evolution algorithm with local search strategy.Computer Engineering and Applications,2009,45(7):56-58.

Abstract: At present,the hybridization of Differential Evolution (DE) with local search method confines to DE with a crossover based local search method.A novel differential evolution algorithm with best-individual based Local Search strategy (LSDE) is proposed.In LSDE,a normal distribution operator is introduced to automatically modify search length,and time-varying factors are introduced to modify the two parameters of DE.The experiment results show that except for one function,the ability of finding the optimal solutions using LSDE is better than that of using DE and differential evolution algorithm based on chaos searching (CDE),and the convergence speed of LSDE is quicker than that of DE.

Key words: Differential Evolution(DE);local search strategy;best-individual

摘 要:针对目前差分进化与局部搜索相结合仅局限于基于交叉的局部搜索的方法,提出了一种基于最佳个体局部搜索策略的差分进化算法(LSDE),并引入正态分布算子自动调整搜索步长和时变差分进化因子调整 DE 的两个参数。实验结果表明:除一个函数外,LSDE 的寻优效果比 DE 和基于混沌搜索的微分进化算法(CDE)都要好,LSDE 的收敛速度比 DE 快。

关键词:差分进化;局部搜索策略;最佳个体
DOI:10.3778/j.issn.1002-8331.2009.07.018 **文章编号**:1002-8331(2009)07-0056-03 **文献标识码**:A **中图分类号**:TP18

1 引言

局部搜索是一种用于解决优化问题的启发式算法,Kanellakis 和 Papadimitriou 于 1979 年就提出了用局部搜索的方法解决非对称的旅行商问题[1]。近年来,局部搜索和进化算法相结合的方法广泛用于解决优化问题[2-4]。

差分进化(Differential Evolution,DE)是由 Storn 和 Price 提出的一种用于连续空间全局优化的启发式算法[5],它是一种基于种群的、随机的进化算法。目前,也有 DE 和局部搜索相结合的方法,它是由 Noman 和 Iba 提出的基于交叉的局部搜索算法[4]。后来,Noman 和 Iba 对这种方法进行了详细的阐述[6],其思想是用 DE 找出每一代中的最佳个体,然后从种群中随机选择 N 个个体进行交叉操作,交叉得到的最好的个体若好于原来的最佳个体,则用交叉得到的最佳个体取代原来的最佳个体,再进行下一次搜索直到满足终止条件。

提出了一种新的局部搜索和差分进化相结合的方法,它是用 DE 找到每一代中最佳个体,然后在这个最佳个体的附近进行一定次数的局部搜索,将原最佳个体与局部搜索得到的最佳个体进行比较,若后者比前者好,则用后者随机取代种群的一个个体,否则继续直到满足终止条件,这种方法称为具有局部搜索策略的差分进化算法,简称 LSDE。在 LSDE 中,还引入了正态分布算子和时变差分进化因子自动地调整搜索步长和 DE 中的缩放因子 F 与交叉因子 CR。

2 差分进化算法

差分进化与标准的遗传算法一样,都包含有选择、交叉和变异三个操作。与遗传算法不同的是,DE 采用的是由变异到交叉再到选择的操作顺序。

变异操作:随机从种群中选择一个个体作为基向量(又称

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.50275150);高等学校博士学科点专项科研基金(No.20070533131)。

作者简介:谭跃(1974-),男,博士生,主要研究方向:进化计算、机器人;谭冠政(1962-),男,教授,博士生导师,主要研究方向:智能机器人系统与控制、人工智能与认知系统、先进控制理论与先进算法;涂立(1974-),男,讲师,主要研究方向:网络技术。

收稿日期:2008-08-29 **修回日期**:2008-10-30

扰动向量)和另外两个不同的个体作为差分向量,通过下面的公式得到变异向量:

$$\boldsymbol{v}_i^{G+1}=x_{r1}^{G}+F(x_{r2}^{G}-x_{r3}^{G}) \tag{1}$$

其中,$r_1,r_2,r_3\in\{1,2\cdots,N_P\}$,且 $r_1\neq r_2\neq r_3$,N_P 为种群的规模,F 为缩放因子,G 为当前种群的代数,$G+1$ 表示下一代。

交叉操作:交叉操作是在变异产生的第 i 个个体 v_i^{G+1} 和种群中的第 i 个体 x_i^G 之间进行,交叉操作得到实验向量:

$$\boldsymbol{u}_{i,j}^{G+1}=\begin{cases}\boldsymbol{v}_{i,j}^{G+1} & \text{if}(randb(j)\leqslant CR)\ \text{or}\ j=rnbr(j)\\ \boldsymbol{x}_{i,j}^{G} & \text{if}(randb(j)>CR)\ \text{or}\ j\neq rnbr(j)\end{cases} \tag{2}$$

其中,$j\in\{1,2\cdots,D\}$,D 为问题的维数,CR 为交叉因子,$randb(j)$ 为第 j 次评价时产生均匀分布的一个随机数,其范围为(0,1),$rnbr(j)$为{1,2,…,D}中随机选择的一个整数。

选择操作:DE 的选择是一对一选择方式,它是在实验向量 $\boldsymbol{u}_i^{G+1}$ 与原种群的个体 x_i^G 之间进行。选择的原则是适应度更优的个体进入到下一代,用公式表示为:

$$x_i^{G+1}=\begin{cases}x_i^{G+1} & \text{if}\ f(u_i^{G+1})\leqslant f(x_i^{G})\\ x_i^{G} & \text{otherwise}\end{cases} \tag{3}$$

式中,x_i^{G+1} 为下一代的第 i 个个体,f 为适应度函数或目标函数。

3　具有局部搜索策略的差分进化算法

3.1　局部搜索算法

局部搜索算法是从一个初始解 X 出发,然后不断地在 X 的邻域内搜索比 X 更好的解 X'。如果找到比 X 更好的解,就用 X'代替 X,继续局部搜索;否则搜索结束。

3.2　时变差分进化因子

根据文献[7],引入时变交叉概率因子 CR 使算法在初期较小的 CR 能提高全局搜索能力,在后期较大的 CR 能提高局部搜索能力,为此引入新的交叉因子:

$$CR(g)=CR_0(1-\sqrt{g_{\max}^2-g^2}/2g_{\max}) \tag{4}$$

式中,CR_0 为初始交叉因子,g 为当前进化代数,$g_{\max}$ 为最大进化代数。

对于缩放因子 F,采用同样的方式加以动态调整:

$$F(g)=F_0(1-\sqrt{g_{\max}^2-g^2}/2g_{\max}) \tag{5}$$

式中,F_0 为初始缩放因子,g 为当前进化代数,$g_{\max}$ 为最大进化代数。

3.3　具有局部搜索策略的差分进化算法的描述

具有局部搜索策略的差分进化算法(LSDE)的基本原理是在每一代中用 DE 搜索到最佳个体 X_{best},在 X_{best} 的附近再进行 k 次局部搜索,得到 k 个个体,找出 k 个个体中适应度值最佳的个体 X_{kbest},若 X_{kbest} 适应度值比 X_{best} 好,用 X_{kbest} 的适应度取代 X_{best} 的适应度,并将 X_{kbest} 随机取代种群中的某个个体再返回,否则直接返回。在 X_{best} 附近进行局部搜索用公式表示如下:

$$X_k=X_{best}+\eta_k \tag{5}$$

式中,η_k 为服从正态分布的均值为 0、方差为 1 的正态分布随机数。

根据前面的分析,LSDE 的流程表示为:

步骤 1　初始化:种群规模 N_P,初始缩放因子 F_0,初始交叉因子 CR_0,种群最大进化代数 $G_{\max}$,输入变量的上界 $H_{i,j}$ 和下界 $L_{i,j}$,$i\in\{1,2,\cdots,N_P\}$,$j\in\{1,2,\cdots,D\}$,局部搜索次数 k。

步骤 2:根据方程式(6)产生初始种群 P^0:

$$P_{i,j}^0=L_{i,j}+rand(1)(H_{i,j}-L_{i,j}) \tag{6}$$

其中,$i\in\{1,2,\cdots,N_P\}$,$j\in\{1,2,\cdots,D\}$,$rand(1)$为(0,1)的均匀分布的一个随机数。

步骤 3　对初始种群的 N_P 个体进行评价,找出适应度值最佳的个体 X_{best},记录 X_{best} 的适应度值 F_{best} 及其在种群中的索引 $index(X_{best})$,并置进化代数 G=1;

步骤 4　执行变异操作,用式(1)和式(5)得到变异向量 $\boldsymbol{v}_i^{G+1}$;

步骤 5　执行交叉操作,用式(2)和式(4)得到实验向量 $\boldsymbol{u}_i^{G+1}$;

步骤 6　执行选择操作,用式(3)得到下一代个体 x_i^{G+1};

步骤 7　执行判断操作,若 $x_{i,j}^{G+1}<L_{i,j}$ 或 $x_{i,j}^{G+1}>H_{i,j}$,则 $x_{i,j}^{G+1}=L_{i,j}+rand(1)(H_{i,j}-L_{i,j})$,go to 步骤 8,否则继续;

步骤 8　执行比较操作,比较 $f(x_i^{G+1})$ 和 F_{best} 大小,若 $f(x_i^{G+1})$ 比 F_{best} 好,则 $X_{best}=x_i^{G+1}$,$F_{best}=f(x_i^{G+1})$,$index(X_{best})=i$,go to 步骤 9,否则继续;

步骤 9　重复执行步骤 4~步骤 8 N_P 次;

步骤 10　对式(5)进行 k 次计算得到 X_k;

步骤 11　找出 X_k 中适应度值最好的个体 X_{kbest},若 X_{kbest} 的适应度值 $f(X_{kbest})$比 F_{best} 好,则:$X_{best}=X_{kbest}$,$F_{best}=f(X_{kbest})$,并且产生一随机整数 $j\in[1,N_P]$,$index(X_{best})=j$,$x_j^{G+1}=X_{kbest}$,go to 步骤 12,否则继续;

步骤 12　$G=G+1$,如果 $G<G_{\max}$,转到步骤 4,否则继续;

步骤 13　输出 X_{best} 和 F_{best}。

在步骤 7 中执行判断操作是因为下一代个体中可能存在某些变量超出了取值范围,出现这种情况是因为用式(1)产生变异向量时,导致变异向量中的某些变量超出了取值范围,如果选择操作将这些变量选入到下一代个体中,则下一代个体中就存在超出取值范围的变量,解决的办法重新产生一些变量取代这些超出范围的变量。

由于在每一代的进化之后就加入局部搜索,因此算法的收敛速度很快,这可能导致陷入局部最优,为此对算法加以简单改进,若在开始连续 100 代内误差限(误差限定义为算法找到的最优值与全局最优值之差的绝对值)达不到 1/10,则停止局部搜索,重新启用 DE 搜索,并在 DE 的最后 50 代加入局部搜索直至停止。

4　仿真实验

为了检验 LSDE 的有效性,选择文献[8]中典型的测试函数作为测试对象,这些测试函数包含了单峰独立、多峰独立、多峰非独立三种类型。另外,为使测试对象更为全面,再选择一个单峰非独立函数作为测试对象,这些测试函数见表 1。

实验参数设置为:种群规模 N_P=25;对 f_1、f_3、f_4、f_5、f_6 初始缩放因子 F_0 和初始交叉因子 CR_0 分别为 0.5 和 0.1,f_2 均为 0.9;f_1 和 f_3 的最大进化代数均为 $G_{\max}$=800,f_2 的最大进化代数 $G_{\max}$=2 000,f_4 的最大进化代数 $G_{\max}$=200,f_5 和 f_6 的最大进化代数 $G_{\max}$=1 000;局部搜索次数 k=60。在以上这些参数设置下,DE 和 LSDE 单独运行 20 次,得到的结果如表 2 所示。

表 1 典型测试函数

函数名称	函数类型	测试函数	输入变量取值范围	全局最优值
Sphere	单峰独立	$f_1(x)=\sum_{i=1}^{30}x_i^2$	[-100,100]	0
Rosenbrock	多峰非独立	$f_2(x)=\sum_{i=1}^{9}[100(x_{i+1}-x_i^2)^2+(1-x_i)^2]$	[-30,30]	0
Griewank	多峰非独立	$f_3(x)=1/4\,000\sum_{i=1}^{10}x_i^2-\prod_{i=1}^{10}\cos(x_i/\sqrt{i})+1$	[-600,600]	0
Schaffer	多峰非独立	$f_4(x)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}-0.5$	[-100,100]	-1
Rastrigin	多峰独立	$f_5(x)=\sum_{i=1}^{30}[x_i^2-10\cos(2\pi x_i)+10]$	[-5.12,5.12]	0
Schwefel	单峰非独立	$f_6(x)=\sum_{i=1}^{30}(\sum_{j}^{i}x_j)^2$	[-100,100]	0

从表 2 可知,LSDE 能够找到三个函数(f_3、f_4、f_5)的全局最优值,而 DE 连一个函数的全局最优值也找不到;对于不能找到全局最优值的函数(f_1、f_2、f_6),LSDE 找到的最优值与全局最优值的误差也很小,而 DE 找不到全局最优值的函数中只有一半的误差较小(f_1、f_2、f_3),另一半的误差较大(f_4、f_5、f_6)。总的来说,LSDE 和 DE 对 f_2 的寻优效果相当,而对其他函数 LSDE 的寻优效果大大好于 DE。

表 2 DE 和 LSDE 的寻优结果

函数	算法	最大最优值	最小最优值	平均值
f_1	DE	3.926 3e-08	5.365 3e-09	1.766 0e-08
	LSDE	2.082 9e-16	1.694 6e-44	1.135 1e-17
f_2	DE	4.727 7e-08	8.307 4e-11	9.010 3e-09
	LSDE	6.456 8e-07	7.143 0e-11	4.056 4e-08
f_3	DE	4.281 3e-05	1.812 5e-06	7.446 4e-06
	LSDE	0	0	0
f_4	DE	-0.979 2	-0.993 7	-0.989 2
	LSDE	-1	-1	-1
f_5	DE	12.200 5	0.103 9	3.015 4
	LSDE	0	0	0
f_6	DE	1.806 7e+04	6.994 5e+03	1.211 1e+04
	LSDE	1.968 1e-08	9.481 9e-16	2.637 5e-09

为比较 DE 和 LSDE 的收敛速度,对 20 次最小最优值进行降序排列,选择第 10 位最小最优值对应的那次运行结果作为最终结果,得到的寻优曲线如图 1 所示。图中,纵坐标为误差函数(误差函数定义为$|f(x)-f(x^*)|$,$f(x^*)$为全局最优值,$f(x)$为算法找到的最优值)的常用对数值,横坐标为进化代数,红色实线和蓝色实线分别表示 DE 和 LSDE 的寻优曲线。

图 1 中有些曲线在没有到达最大进化代数时就消失了,这种情况表示在某个时刻算法已经找到全局最优值,如 LSDE 对 f_3、f_5 的寻优曲线,而 LSDE 对 f_4 的寻优曲线中,没有蓝色实线,这种情况表示在第 1 代后,算法就找到了全局最优值。此外,LSDE 对 f_2 的寻优曲线中,出现了寻优结果的正跳变,这种情况表示算法在开始连续 100 代内误差限达不到 0.1,停止局部搜索,重新启用 DE 搜索,在 DE 的最后 50 代加入局部搜索直至停止。从总的收敛速度来看,LSDE 和 DE 对 f_2 的收敛速度相当,而对其他函数 LSDE 的收敛速度都比 DE 快。

从以上的实验结果分析可以看出,LSDE 的寻优能力和收敛速度比 DE 更好。为了比较 LSDE 与其它算法的优劣,选择文献[8]中基于混沌搜索的微分进化算法(对于 Differential Evolution,国内有差分进化、微分进化、差异演化三种翻译形式)进行了比较,比较结果如表 3 所示。表中,CDE 是基于混沌搜索的微分进化算法的简称,CDE 的所有结果摘抄于文献[8]。这样比较是很公平的,因为 LSDE 的种群规模、初始的 F 和 CR 值、进化代数与文献[8]完全相同,并且 LSDE 局部搜索的次数与 CDE 混沌搜索的次数也相同。

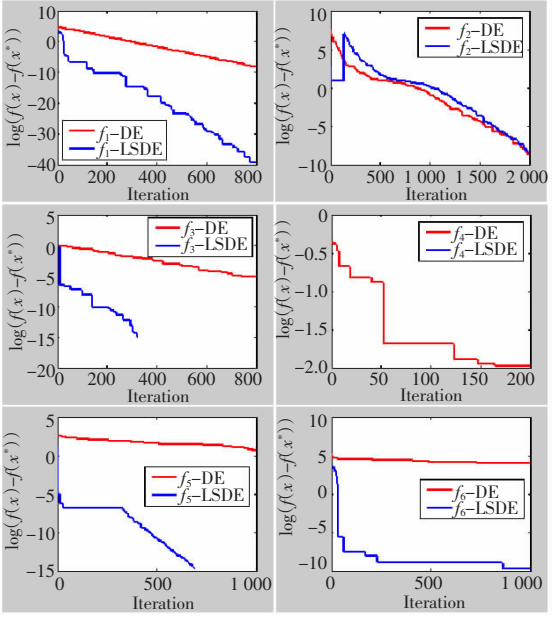

图 1 f_1~f_6 寻优曲线

表 3 LSDE 和 CDE 的比较结果

函数	算法	最大最优值	最小最优值	平均值
f_1	CDE	2.506 4e-08	5.113 1e-09	1.466 3e-08
	LSDE	2.082 9e-16	1.694 6e-44	1.135 1e-17
f_2	CDE	1.024 3e-12	7.827 7e-26	6.636 7e-14
	LSDE	6.456 8e-07	7.143 0e-11	4.056 4e-08
f_3	CDE	1.576 3e-09	0	8.044 1e-11
	LSDE	0	0	0
f_4	CDE	-0.993 296 9	-1	-0.999 539 8
	LSDE	-1	-1	-1
f_5	CDE	7.505 1e-11	3.234 4e-11	5.108 9e-11
	LSDE	0	0	0

从表 3 可以看出,CDE 对每个函数的 20 次运行结果中,有些函数能找到全局最优值(f_3、f_4),但并不是每次都能找到函

(下转 92 页)