基于混沌搜索的自适应差分进化算法

卢有麟,周建中,李英海,覃 晖
LU You-lin,ZHOU Jian-zhong,LI Ying-hai,QIN Hui

华中科技大学 水电与数字化工程学院,武汉 430074
College of Hydroelectric and Digitalization Engineering,Huazhong University of Science and Technology,Wuhan 430074,China
E-mail:youlin_lu@yahoo.cn

LU You-lin,ZHOU Jian-zhong,LI Ying-hai,et al.Adaptive differential evolution algorithm combined with chaotic search. Computer Engineering and Applications,2008,44(10):31-33.

Abstract: An adaptive differential evolution algorithm combined with chaotic search(CADE) is presented.It adjusts the cross operator adaptively according to the computation process in order to preserve the diversity of population at the initial generation as well as to improve the global convergence ability.Chaotic search which behaves well in local search is adopted to enhance the precision of solution and the probability of obtaining global optimal solution.Several typical benchmark functions are tested and experimental results show that the presented algorithm has remarkable global convergence ability,and it can avoid premature convergence effectively.

Key words: differential evolution algorithm;adaptive;chaotic search;global optimization

摘 要:提出一种基于混沌搜索的自适应差分进化算法(CADE),该算法在计算过程中自适应地调整交叉率,在搜索初期保持种群多样性的同时增强算法的全局收敛性。具有较强局部遍历搜索性能的混沌搜索的引入使得算法具有较好的求解精度,增加搜索到全局最优解的概率。对几种典型的测试函数对 CADE 进行了测试,实验结果表明,该算法能有效地避免早熟收敛,具有良好的全局收敛性。

关键词:差分进化算法;自适应;混沌搜索;全局优化
文章编号:1002-8331(2008)10-0031-03 **文献标识码**:A **中图分类号**:TP301

1 引言

由 Storn R 和 Price K 于 1995 年提出的差分进化(Differential Evolution,DE)算法[1,2]是一种采用浮点矢量编码的并行随机搜索算法,其原理简单,受控参数少,可对在连续空间内进行随机、直接、并行的全局搜索。在 1996 年举行的第一届国际进化优化竞赛上,DE 表现优异, 被证明是最快的进化算法之一。近年来,DE 以其鲁棒性、稳健性和在实数域上强大的全局搜索能力在多个领域得到广泛的应用,并取得较好的结果。

差分进化算法与实数编码的遗传算法相似,但在产生子代的方式上有所不同,DE 在父代个体间的差分矢量的基础上进行变异和交叉操作。同时,算法采用"贪婪"选择策略产生子代个体。这种策略能使算法的收敛速度加快,但算法早熟收敛的概率也随之增加。为提高差分进化算法的性能,许多学者提出了改进方法:文献[3]在算法中加入了迁移操作和加速操作以提高 DE 的种群多样性和收敛速度;文献[4]提出了一种双群体伪并行差分进化算法,该算法采用双种群结构提高了 DE 的全局搜索能力和收敛速率。

差分进化算法的性能与其交叉操作密切相关,较小的交叉率可提高算法局部搜索能力,而较大的交叉率可加强种群的多样性和算法的全局收敛性。本文提出的基于混沌搜索的自适应差分进化算法(CADE)在计算过程中自适应的调整交叉算子,使算法在初期能保持种群的多样性的同时提高后期算法的局部搜索能力,并在此基础上引入局部遍历搜索能力较强的混沌搜索,进一步提高了 CADE 的搜索精度。

2 基本差分进化算法

DE 同遗传算法(GA)一样,也包括交叉、变异和选择等进化算子,但是与其他进化算法不同的是,DE 在随机选择的父代个体间差分矢量的基础上生成变异个体;接着其按一定的概率对父代个体与生成的变异个体进行交叉操作, 生成试验个体;最后采用贪婪策略在父代个体与试验个体之间进行选择操作,选择具有更佳适应度的个体作为子代个体。

基金项目:国家自然科学基金重点项目(the Key Project of National Natural Science Foundation of China No.50539140);国家自然科学基金(the National Natural Science Foundation of China under Grant No.50579022)。

作者简介:卢有麟(1985-),男,博士生,主要研究方向:现代优化算法及水电站优化调度;周建中(1959-),男,教授,博士生导师,通讯作者,主要研究方向:人工智能算法及复杂系统分析与仿真;李英海(1981-),男,博士生,主要研究方向:现代优化算法及水资源优化调度;覃晖(1983-),男,博士生,主要研究方向:人工智能算法及水电站经济运行。

收稿日期:2007-11-28 **修回日期**:2008-01-04

2.1 变异操作

DE 是在父代个体之间的差分矢量的基础上进行变异操作的,每个差分矢量包括父代(第 g 代)的两个不同个体(x_{r1}^{g},x_{r2}^{g})。实际应用中 DE 有多种变异方案[5,6],其变异个体的生成方法各不相同。本文中算法采用的变异操作的方程如下:

$$x_m=x_{best}^{g}+F*[(x_{r1}^{g}-x_{r2}^{g})+(x_{r3}^{g}-x_{r4}^{g})] \tag{1}$$

其中,x_m 为变异个体,x_{best}^{g} 为当代种群中的最优个体,x_{r1}^{g}、x_{r2}^{g}、x_{r3}^{g} 和 x_{r4}^{g} 是从种群中随机挑选的互不相同的个体,$F\in(0,1.2]$为变异率,用来控制两个差分矢量($x_{r1}^{g}-x_{r2}^{g}$)以及($x_{r3}^{g}-x_{r4}^{g}$)对最优个体 x_{best}^{g} 的影响。由式(1)可知,此变异方案在 4 个不同个体产生的两个差分矢量的基础上对当代最优个体进行变异操作,可保持一定程度的种群多样性。

2.2 交叉操作

DE 使用交叉操作来提高种群的多样性。算法对变异操作生成的变异个体 x_m 和当代种群中的第 i 个个体 x_i^{g} 进行交叉操作,产生试验个体 x_c。其操作的方程如下:

$$x_{cj}=\begin{cases}x_{mj} & \text{if } rnd()\leqslant CR \text{ or } j=rndr(i)\\ x_{ij} & \text{if } rnd()>CR \text{ and } j\neq rndr(i)\end{cases}\quad j=0,1,2,\cdots,D \tag{2}$$

式(2)中,$rnd()$是[0,1]之间随机数产生函数;$rndr(i)\in\{1,2,\cdots,D\}$为随机产生的数,用来确保 x_c 中至少有一位是由 x_m 贡献;D 为优化变量的维数;$CR\in(0,1)$为交叉率,其决定着除了第 $rndr(i)$位以外 x_c 中哪位由 x_m 贡献,哪位由 x_i^{g} 贡献。

2.3 选择操作

DE 采用“贪婪”选择策略产生子代个体。经过交叉操作后的试验个体 x_c 与 x_i^{g} 进行竞争,两者中适应度更优的个体被选为子代个体,如果目标函数要被最小化,其操作的方程如下:

$$x_i^{g+1}=\begin{cases}x_c & \text{if } f(x_c)\leqslant f(x_i^{g})\\ x_i^{g} & \text{else}\end{cases} \tag{3}$$

式(3)中,x_i^{g+1} 为子代中第 i 个个体,$f(x)$为目标函数。

3 基于混沌搜索的自适应差分进化算法

3.1 交叉率自适应调整策略

在基本差分进化算法中,交叉算子 CR 通常是事先取定的一个 0 到 1 之间的实数,用来控制 x_m 和 x_i^{g} 对 x_c 的贡献,其大小在计算过程中保持不变。但是在实际搜索进程当中,若 CR 值较大,则 x_m 对 x_c 贡献较多,此时算法的局部搜索能力较强,收敛速度较快;若 CR 值较小,则 x_i^{g} 对 x_c 贡献较多,此时种群的多样性较好,算法全局搜索能力较强[7]。良好的 CR 应在搜索的初始阶段较小以保持种群的多样性,此时算法进行全局搜索,可有效避免早熟现象发生;而在搜索的后期其值应随着搜索进程逐步增大,提高算法的局部搜索能力。基于这种思想,本文在 CADE 中将 CR 视为一个与当前进化代数 g_{now} 与最大进化代数 $g_{\max}$ 相关的变量,构造如下函数进行计算:

$$CR=CR_0\cdot 2^{e^{\left(1-\frac{g_{now}}{g_{now}+1}\right)}} \tag{4}$$

其中,$CR_0\in(0,0.5)$为事先选定的初始交叉率,其值较小;g_{now} 为当前进化代数,$g_{\max}$ 为最大进化代数。基于以上思路设计的交叉率 CR 可以根据搜索进程自适应的调整自身大小,从而优化的算法性能。

3.2 混沌搜索

混沌是非线性动力学系统中特有的一种现象,具有内在随机性和遍历性等特点。本文采用人们熟悉的一维 Logistic 映射来设计混沌变异操作,其迭代方程为:

$$r_j^{k+1}=\lambda r_j^{k}(1-r_j^{k}),r_j\in[0,1] \tag{5}$$

其中,r_j^{k} 是混沌变量,当 λ 取值为 4.0 时,系统陷入混沌状态,r_j^{k} 在[0,1]内遍历。试验证明,利用混沌的遍历性,可以较好的实现局部搜索[8]。在寻优过程中,对最优个体 $x_{best}^{g}=[x_1,x_2,\cdots,x_j,\cdots,x_D]^{\mathrm{T}}$ 进行混沌变异操作,步骤如下:

步骤 1 令混沌搜索次数 $k=0$,随机生成 D 个不同的混沌变量 $r^{k}=[r_1^{k},r_2^{k},\cdots,r_j^{k},\cdots,r_D^{k}]^{\mathrm{T}}$,其中 r_j^{k}($k=1,2,\cdots,D$)的值不能取 0,0.25,0.5,0.75,1 这 5 个混沌迭代方程的不动点。

步骤 2 将混沌变量 r_j^{k} 映射到优化变量 x_j 的取值区间[$x_{j\min}$,$x_{j\max}$]得到搜索尺度 P_m。其中 $x_{j\max}$,$x_{j\min}$ 为 x_j 的取值范围。

$$P_m=x_{j\min}+r_j^{k}(x_{j\max}-x_{j\min}) \tag{6}$$

步骤 3 由下式计算 x_j 的新值:

$$x_j'=(1-u_g)x_j+u_g p_m \tag{7}$$

步骤 4 通过 Logistic 迭代方程得到 $r^{k+1}=[r_1^{k+1},r_2^{k+1},\cdots,r_j^{k+1},\cdots,r_D^{k+1}]^{\mathrm{T}}$;

步骤 5 计算 $x_{best}^{g}{}'=[x_1',x_2',\cdots,x_j',\cdots,x_D']^{\mathrm{T}}$ 的适应度,如果大于 x_{best}^{g} 的适应度或混沌迭代达到一定步数,则停止混沌搜索操作;否则转步骤 2。

式(7)中的 u_g 为缩放因子,其决定变异操作对最优个体的影响。为了提高算法搜索的性能,本文通过下式计算计算 u_g:

$$u_g=e^{-m*g_{now}/g_{\max}} \tag{8}$$

其中,m 为控制缩放的参数。由式(8)计算出的 u_g 随着进化代数的增加逐渐变小,即混沌变搜索的搜索范围围绕着最优个体逐渐变小。这样,在进化初期该操作对最优个体的影响大,算法可在一个较大的范围内进行全局遍历搜索,有利于保持种群的多样性;在进化后期 u_g 较小,算法可在最优个体附近的一个很小范围内进行局部遍历搜索,有利于提高解的精度。

3.3 算法流程

本文提出的 CADE 在 DE 算法的基础上根据求解进程自适应的调整交叉率 CR,有效避免算法早熟收敛以及求解精度不高等情况的发生。若最优解在给定代数内没有发生变化,则可判断搜索陷入局部最优,此时该算法对最优个体实施给定步数的混沌变异操作,以提高算法的求解精度。

CADE 算法的具体流程如下:

步骤 1 初始化参数:种群大小 Pop,变异算子 F,交叉率 CR_0,最大进化代数 $g_{\max}$ 以及缩放参数 m;

步骤 2 初始化种群;

步骤 3 用适应度函数评价种群中每个个体,找出其中的最优个体;

步骤 4 判断最优解是否经过指定迭代次数后依然没有发

生变化,若是,则对最优个体进行一定代数的混沌搜索操作;

步骤 5　按式(4)计算当前代的 CR;

步骤 6　对种群中所有个体进行交叉、变异和选择操作;

步骤 7　判断最优解的精度是否满足要求或是否达到最大进化代数,若是则算法停止输出最优个体作为最优解,否则进化代数加 1,转入步骤 3 继续进行。

4　试验分析

为了验证本文提出的 CADE 的有效性,下面通过 3 个典型的测试函数对 CADE 进行测试,并选用 DE,遗传算法 GA 作为对比算法与 CADE 进行比较。其中函数 f_1 为多峰二次的 Rastrigin 函数,在 $S=\{x_i \in (-5.12,+5.12), i=1,2,\cdots,n\}$范围内有大约 $10n$ 个局部极小点;f_2 为非凸,病态的 Rosenbrock 函数;f_3 是多峰的 Griewank 函数,具有大量的局部极小点。

$$f_1=\sum_{i=1}^{30}(x_i^2-10\cos(2\pi x_i)+10), -5.12\leqslant x_i\leqslant 5.12 \quad (9)$$

$$f_2=\sum_{i=1}^{30}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2), -30\leqslant x_i\leqslant 30 \quad (10)$$

$$f_3=\frac{1}{4\,000}\sum_{i=1}^{30}x_i^2-\prod_{i=1}^{30}\left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right)+1, -50\leqslant x_i\leqslant 50 \quad (11)$$

参数初始化如下:三种算法种群规模 Pop 均为 100,CADE 变异率 F 取 0.5,初始交叉率 CR_0 取 0.3,最大进化代数 $g_{\max}$ 为 1 000,缩放因子 m 取 8.0,试验中,若 15 次迭代后最优解依然保持不变则进行 150 次混沌搜索。DE 算法变异算子 F 取 0.5,CR 取 0.6 固定不变。GA 采用实数编码方式,其中交叉率 W_c=0.8,变异率 W_m=0.02。

表 1 列出了上述三种算法求解各测试函数运行 20 次最优解的平均值和方差。由表 1 可知,对各测试函数,本文提出的 CADE 的优化性能明显好于另外两种算法。其中,GA 在求解这三个高维多峰函数均无法收敛到理论最优解,优化效果较差。DE 的优化结果也不理想,特别是优化 f_2 函数时,DE 求得的最优解的平均值与其理论最小值相差较大。而 CADE 在相同参数设置下均能以较高的精度收敛到最优解,表现出较强的稳定性和鲁棒性。

图 1~图 3 是上述三种算法求解各测试函数运行 20 次最优解平均值的进化曲线。从图中可知,对于这三个高维多峰函数,GA 均提前收敛,出现早熟现象;DE 的求解精度虽然好于 GA,但是在计算过程中仍早熟收敛,得不到令人满意的优化结果;而 CADE 的求解精度比以上两种方法高很多,具有很强的寻优能力。以上结果说明 CR 的自适应变化以及混沌搜索的引入使得 CADE 能有效地避免早熟收敛现象的发生,算法求解精度可以随着进化代数的增加进一步提高,因此,CADE 在性能上较 DE 有明显的改善,能有效避免早熟,具有很强的全局收敛性。

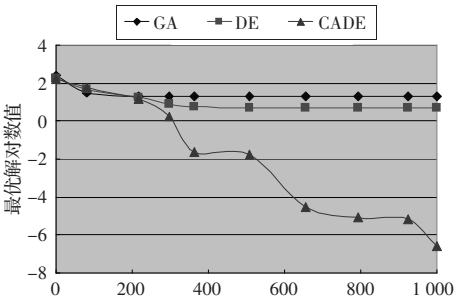

图 1　f_1 最优解平均值进化曲线

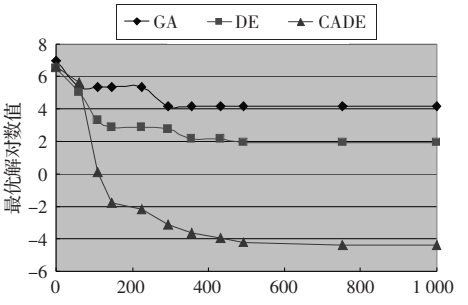

图 2　f_2 最优解平均值进化曲线

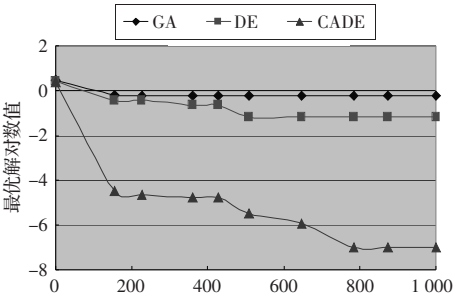

图 3　f_3 最优解平均值进化曲线

5　结论

DE 算法是一种采用浮点矢量编码,原理简单的并行随机搜索算法,其主要缺点是容易出现早熟现象,求解精度不高。本文在 DE 中引入交叉算子的自适应调整策略以及局部遍历搜索能力较好的混沌变异操作,提出一种新的基于混沌搜索的自适应差分进化算法 CADE。该算法不但保留了 DE 实现简单,并行搜索的特点,并且又通过根据搜索进程调整交叉率以及对提早收敛的最优的个体进行指定步数的混沌搜索使算法同时兼顾种群多样性以及求解精度。通过对测试函数的测试结果表明,与 GA 以及 DE 相比,本文提出的 CADE 算法全局收敛性强,稳定性好,求解精度高,是求解目标函数是复杂高维连续函数的实际工程应用问题的一种新途径。

研究结合 CADE 以解决实际工程问题,确定各参数的取值使 CADE 求得的解能满足工程应用需求以及研究适合 CADE 处理约束条件的机制是笔者下一步的研究内容。

表 1　三种算法运 20 次各测试函数的最优解及方差

函数	理论最优	GA		DE		CADE	
		平均最优解	标准差	平均最优解	标准差	平均最优解	标准差
f_1	0	20.643	4.367	5.487	3.068	2.645×10^{-7}	2.858×10^{-7}
f_2	0	1.893×10^{4}	1.230×10^{4}	97.462	35.518	4.791×10^{-5}	2.176×10^{-5}
f_3	0	0.464	0.050	0.049	0.023	7.049×10^{-8}	3.545×10^{-8}

(下转 39 页)