DOI :10. 3969/J. ISSN. 1672-6693. 2009. 03. 012

基于双群体的非线性约束规划进化算法*

贾礼平,邹国成
(乐山师范学院 数学系,四川 乐山 614004)

摘要:在求解非线性约束规划问题中,对其约束条件的处理是一个难点问题。本文提出了一个非线性约束规划的双群体进化算法,与以往存在的约束优化算法不同之处在于:定义个体对约束条件的函数值作为约束违犯度对群体中的个体进行度量,目标函数值作为最优解的度量。首先考虑了标准的约束规划问题,简单介绍了约束优化问题中约束条件的处理方法,给出了与这些方法不同的处理方法。针对约束违犯度,定义了两个群体,即可行群体与不可行群体。然后给出了双群体进化算法详细步骤,用 5 个 Benchmark 函数测试了此算法,并通过与其它已知算法对此 5 个函数的计算结果的比较,验证了算法的可行性和有效性。

关键词:非线性约束规划;进化算法;双群体;算子
中图分类号:O221.2;TP301.6 **文献标识码**:A **文章编号**:1672-6693(2009)03-0049-06

科学研究和工程分析中很多问题经建模都被转化为非线性约束优化问题,然而非线性约束优化问题在最优化领域一直是一类非常难处理的问题[1],但各种求解方案仍层出不穷[2-7]。由于传统优化算法,如信赖域算法[2]、拟牛顿 SQP[3] 等,它们对目标函数的解析性质要求很高,所以在实际求解工程问题中的应用并不广泛。进化算法(Evolutionary algorithm :A)[5-7] 是模拟自然界中的“优胜劣汰”原则演变成的一种优化计算方法,与传统的非线性规划算法相比,进化算法不需要梯度(导数)等信息,同时它又是一个全局性搜索方法,相对而言,它陷入局部最优的机会比较小,它只用到函数的数值信息,并不依赖目标函数的解析性能,且对群体的操作使其具有隐式并行性。所以,进化算法在解决一些大规模、复杂的非线性优化问题上具有独特的优势,进化算法的另一个显著特点是对于所解的优化问题没有太多的数学要求,可以处理任意形式的目标函数和约束,该方法被大量用来解决非线性约束优化领域内的各种问题。

近些年来,进化算法在求解约束优化问题上取得了很大进展,已有的方法大致可以分为 4 类:1)用专门设计的算法将可行个体转化为新的可行个体;2)降低非可行个体适应度值的罚函数方法;3)修改不可行个体的方法;4)其它混合方法。

虽然求解方法数不胜数,但是大多数算法都是仅对可行个体进行操作,或者是修复不可行个体成为可行个体,抛弃不可行个体。然而,不可行个体中也有许多带有大量有用信息的个体,可以利用这些个体所携带的信息来指导搜索过程,从而使得搜索过程能够快速收敛到全局最优解。

本文设计了一种新的进化算法——双群体进化算法,用以解决约束优化问题。

1 基础知识

非线性单目标约束优化问题的一般数学描述形式为

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_i(x) \leqslant 0, i = 1,2,\cdots,p \\ & h_j(x) \leqslant 0, j = p+1,\cdots,m \\ & x \in [L,U] = S \end{cases}$$

其中

$x = (x_1, x_2, \cdots, x_n) \in \mathbf{R}^n, g_i(x) \leqslant 0 (i = 1,2,\cdots,p)$

和 $h_j(x) = 0 (j = p+1,\cdots,m)$

分别表示不等式和等式约束条件,(1)式的可行域

$$D = \{x \in S \mid g_i(x) \leqslant 0, i = 1,2,\cdots,p; \; h_j(x) = 0, j = p+1,\cdots,m\}$$

* 收稿日期:2008-12-02 修回日期:2009-03-20
资助项目:四川省教育厅基金资助(No. 2006C040,No. 07ZB142);乐山师范学院校级科研启动项目资助(No. ZRC0413)
作者简介:贾礼平,男,讲师,博士研究生,研究方向为人工智能与进化算法。

S 表示搜索空间。

约束优化的难点就在于如何进行约束处理,目前常用的方法是用惩罚函数,把(1)式转化为无约束优化问题,但这种方法一般都包含许多惩罚系数,在实际应用时,只有正确设置这些系数才可能获得可行解,而要获得适当的惩罚系数则需要大量的实验为基础。文献[12,13]中都采用了罚函数的策略。在文献[12]中,问题(1)通过罚函数处理成6种方法,以便对不可行解进行惩罚,分析了这几种方法的优缺点,数值例子说明结果不太满意。作为改进,文献[13]引入了一个随机平衡目标函数和对罚函数的新方法,即对罚系数用冒泡法的思想随机排序,这种排序方法用 $ES(\mu,\lambda)$ 的进化策略测试了13个Benchmark函数,数值结果说明方法是有效的,但结果受参数 P_f 的影响。文献[8]中介绍了不需要惩罚系数而用竞争选择的方法直接比较个体优劣的算法,之后又有许多基于这种思想的进化算法被提出,而这些算法虽然避免了引入惩罚系数,但为了度量个体对约束的违反程度或保持进化群体的多样化,仍然要引入新的参数。设计一种不需要惩罚系数的基于惩罚函数的进化算法一直是研究的一个方向[6,8]。

一般地,对(1)式中等式约束的处理,把其转为不等式约束,即

$$\begin{cases}\min & f(x)\\ \text{s.t.} & g_i(x)\leqslant 0, i=1,2,\cdots,p\\ & h_j(x)\leqslant 0, j=p+1,\cdots,m\\ & -h_k(x)\leqslant 0, j=m+1,\cdots,2m-p\\ & x\in[L,U]=S\end{cases}\tag{2}$$

在上面记号下,(2)式可写为

$$\begin{cases}\min & f(x)\\ \text{s.t.} & G_i(x)\leqslant 0, i=1,2,\cdots,2m-p\\ & x\in[L,U]=S\end{cases}\tag{3}$$

这里 $G(x)=(g(x),h(x),-h(x))^{\mathrm{T}}$。

本文所采取的方法是对(2)式中的约束条件不加任何处理,给出下列定义。

定义1 在问题(3)中,对 $\forall x\in S$,称 $A=|\{i\mid G_i(x)>0, i=1,2,\cdots,2m-p\}|$ 为个体 $\forall x\in S$ 对 $G_i(x)$ $(i=1,2,\cdots,2m-p)$ 的违犯度,这里 $|\cdot|$ 表示元素个数。

显然若 $A>0$,则 x 一定不是(1)式的可行解。若 $A=0$,则 x 一定是(2)式的可行解。这里通过 A 能比较方便判断 $\forall x\in S$ 是不是可行解。

通过交叉、变异后产生的个体,笔者只考虑其违犯度,从而据此把个体分为可行解与不可行解,把它们放在两个集合中:可行解集与不可行解集。

2 求解约束优化问题的新算法——双群体遗传算法

2.1 编码及初始群体产生

编码方式一般有二进制和实数两种编码方式,而实数编码方式更接近问题本质,本文对所研究问题采用实数编码进行求解。给定初始群体大小 pop_size 后,笔者在搜索区域 S 内,均匀随机产生 pop_size 个初始个体,记为 $X=(X^1,X^2,\cdots,X^{pop\_size})$,这里 $X^i=(x_{1i},x_{2i},\cdots,x_{ni})$,$n$ 为变量个数。

2.2 交叉

由于采用实数编码,在此笔者选用线性交叉方式生成新个体,对当前群体随机抽出的两个个体 x_1,$x_2\in S$,按下式进行交叉:

$$x_1'=ax_1+(1-a)x_2,\quad x_2'=(1-a)x_1+ax_2$$

上式中 a 是(0,1)之间的随机数,x_1' 与 x_2' 为交叉生成的两个新个体。

在所给的算法中,笔者分别采取在可行群体、不可行群体,可行群体与不可行群体之间进行交叉,这样可以保证群体多样性(如图1)。

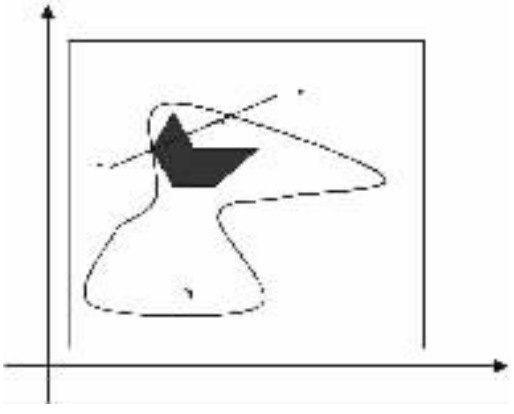

图1 二维情况下任两个个体交叉示意图

注 图中 x 表示父代个体,$*$ 表示交叉后产生的个体,$*$ 位于图中的线段上,阴影部分为可行区域 D。

从图1可以看出,对两个个体进行线性交叉,结果产生的两个个体可能全在 D 内,是可行个体,或者一个在 D 内,可行,一个不在 D 内,不可行,要么全不在 D 内,不可行。

2.3 变异

本文选用实数编码的非均匀变异方式生成新个体 x_i。随机选择一个个体 x_i,利用下式变异生成新个

体 x_i'

$$x_i' = \begin{cases} x_i + (b_i - x_i) f(G) & \text{if } r_1 \geqslant 0.5 \\ x_i - (x_i - a_i) f(G) & \text{if } r_1 < 0.5 \end{cases}$$

其中 a_i,b_i 分别为变量 x_i 的下界和上界,$f(G)$ 可由下式求出:

$$f(G) = (r_2(1 - G/G_{\max}))^b$$

其中 r_1,r_2 为(0,1) 之间均匀分布的随机数,G 为当前的进化代数,$G_{\max}$ 为设定的最大进化代数,b 为形式参数,此处取 2。

对可行群体和不可行群体的个体(可行或不可行),通过变异,有可能把其变成可行个体。

2.4 选择

本文采用精英保留策略。其思想是,把群体在进化过程中迄今出现的最好个体(称为精英个体)不进行配对交叉而直接复制到下一代中。

具体作法:

·对可行群体中的个体,把好个体(称为精英个体)不进行配对交叉而直接选择到下一代中。

·对不可行群体中的个体,根据约束违犯度 A,把 A 值较少的个体不进行配对交叉而直接选择到下一代中。

2.5 双群体进化算法

2.5.1 算法思想　Coello 在文献[9]中提到优化问题的最优解一般位于问题定义域(可行域)的边界处。因此,本算法的思想是利用位于不可行域内带有较好信息的个体,辅助搜索向可行域边界同时也向更好的解个体移动(体现在算法中的 Step 8),并用进化得到的最优性高(或者约束违犯度小)的个体替换父代中较差的个体,从而使得可行个体群中的最优个体向全局最优解移动。

本算法在同时对可行个体群和非可行群体进行遗传操作的过程中,利用目标函数值对遗传算子操作所生成的新个体群和旧个体群之间进行筛选,组成新的群体。其中遗传算子采用 2.2 ~2.4 中设计的遗传操作方法。

2.5.2 算法步骤　现在给出双群体进化算法(Bi-population evolutionary algorithm,BEA)的具体步骤。

Step 1　初始化。给出初始化群体大小 *pop_size* 的初始群体 $P(0)$,设置交叉概率 $p_c > 0$,变异概率 $p_m > 0$,设置可行群体和非可行群体集合 *FD* 和 *IFD*;后代可行集 *FDO* 与不可行集 *IFDO*,令它们为空集;令 $k = 0$;

Step 2　计算群体 $P(k)$ 中个体的约束违犯度 A,把可行解与非可行解记入 *FD* 和 *IFD* 中,对 *FD* 和 *IFD* 中的个体求目标函数值,并分别进行排序;判断是否满足停机条件,若满足,转 Step 7;否则,转 Step 3;

Step 3　对可行群体 *FD* 和非可行群体 *IFD* 中的个体,依概率 p_c 按 2.2 随机交叉,记其后代集合为 O_1;

Step 4　在可行群体 *FD* 和非可行群体 *IFD* 中的个体,依概率 p_m 按 2.3 随机选择个体进行变异操作;记其后代集合为 O_2;

Step 5　计算 O_1,O_2 中的目标值和约束违反程度;更新 *FDO* 和 *IFDO*,即把 $O_1 \cup O_2$ 中可行个体放入 *FDO* 中,把不可行个体放入 *IDFO* 中;

Step 6　将 *FDO* 与 *FD*,*IFDO* 和 *IDF* 中的个体相互比较,依 2.4 的方法把选取的个体保留到 *FD* 和 *IFD* 中。重复 Step 6,直至所有新个体与其父个体比较完毕;令 $P(k+1) = FD \cup IFD$,$k = k + 1$,转 Step 2;

Step 7　比较 *FD* 和 *IFD* 中的最优值,设其最优点分别为 x'_1、x'_0。若 x'_1 的目标函数值优于 x'_0,则 x'_1 为最优解,转 Step 9;否则转 Step 8;

Step 8　取 $x' = x'_1 + \frac{1}{4}x'_0$,若 x' 不可行,则取 $x' = x'_1 + \frac{3}{4}x'_0$。重复 Step 8,直到 x' 可行为止。若 x' 的目标函数值优于 x'_1,则 x' 为最优解;否则 x'_1 为最优解;转 Step 9;

Step 9　输出可行群体和最优解。

注　1)本算法中采取最大迭代数作为停机条件;2)在 Step 2 中,若 $A = 0$,则把对应的个体放入 *FD* 中,否则放入 *IFD* 中;3)在 Step 6 中,在两个个体 $A > 0$ 的情况下比较,取 A 值小的个体放入 *IFD* 中,对 $A = 0$ 的两个个体,取目标值小的个体放入 *FD* 中;4)在 Step 7 中,若 x'_1 的值比 x'_0 差,经过 Step 8,在有限步确实找不到比 x'_0 好的可行个体,笔者用 x'_1 近似(1)式的最优解。在算法中笔者设定经过 100 次迭代,找不到比 x'_1 好的可行个体,则停止。转入 Step 9。

3 数值模拟

3.1 试验函数

本文采用 5 个 Benchmark 进行测试,函数如下。

1)函数 1[10]:求极小问题

$$\min \quad G_1(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$\text{s.t.} \quad g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leqslant 0$$
$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0$$
$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leqslant 0$$
$$g_4(x) = -8x_1 + x_{10} \leqslant 0$$
$$g_5(x) = -8x_2 + x_{11} \leqslant 0$$
$$g_6(x) = -8x_3 + x_{12} \leqslant 0$$
$$g_7(x) = -2x_4 - x_5 + x_{10} \leqslant 0$$
$$g_8(x) = -2x_6 - x_7 + x_{11} \leqslant 0$$
$$g_9(x) = -2x_8 - x_9 + x_{12} \leqslant 0$$
$$0 \leqslant x_i \leqslant 1, i = 1,2,\cdots,9$$
$$0 \leqslant x_i \leqslant 100, i = 10,11,12, 0 \leqslant x_{13} \leqslant 1$$

2) 函数2[11] 求极小问题

$$\min \quad G_2(x) = x_1 + x_2 + x_3$$

$$\text{s.t.} \quad g_1(x) = -1 + 0.002\,5(x_4 + x_6) \leqslant 0$$
$$g_2(x) = -1 + 0.002\,5(x_5 + x_7 - x_4) \leqslant 0$$
$$g_3(x) = -1 + 0.01(x_8 - x_5) \leqslant 0$$
$$g_4(x) = -x_1x_6 - 833.332\,52x_4 + 100x_1 - 83\,333.333 \leqslant 0$$
$$g_5(x) = -x_2x_7 + 1\,250x_5 + x_2x_4 - 1\,250x_4 \leqslant 0$$
$$g_6(x) = -x_3x_8 + 1\,250\,000 + x_3x_5 - 2\,500x_5 \leqslant 0$$
$$100 \leqslant x_1 \leqslant 10\,000, 1\,000 \leqslant x_i \leqslant 10\,000, i = 2,3$$
$$10 \leqslant x_i \leqslant 1\,000, i = 4,5,\cdots,8$$

3) 函数3[12] 求极大问题

$$\max \quad G_3(x) = \left| \frac{\sum_{i=1}^{20} \cos^4(x_i) - 2\prod_{i=1}^{20} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{20} ix_i^2}} \right|$$

$$\text{s.t.} \quad g_1(x) = 0.75 - \prod_{i=1}^{20} x_i \leqslant 0$$
$$g_2(x) = \sum_{i=1}^{20} x_i - 150 \leqslant 0$$
$$0 \leqslant x_i \leqslant 10, i = 1,2,\cdots,10$$

4) 函数4[13] 求极大问题

$$\max \quad G_4(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

$$\text{s.t.} \quad g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0$$
$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0$$
$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0$$
$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0$$
$$-10.0 \leqslant x_i \leqslant 10.0, i = 1,2,\cdots,7$$

5) 函数5[13] 求极小问题

$$\min \quad G_5(x) = e^{x_1x_2x_3x_4x_5}$$

$$\text{s.t.} \quad g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10$$
$$g_2(x) = x_2x_3 - 5x_4x_5 = 0$$
$$g_3(x) = x_1^3 + x_2^3 + 1 = 0$$
$$-2.3 \leqslant x_i \leqslant 2.3, i = 1,2$$
$$-3.2 \leqslant x_i \leqslant 3.2, i = 3,4,5$$

对于测试函数5,在用各种算法比较时,将其化为如下不等式约束。

$$\min \quad G_5(x) = e^{x_1x_2x_3x_4x_5}$$

$$\text{s.t.} \quad g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 \leqslant 0$$
$$g_2(x) = -x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_5^2 + 10 \leqslant 0$$
$$g_3(x) = x_2x_3 - 5x_4x_5 \leqslant 0$$
$$g_4(x) = -x_2x_3 + 5x_4x_5 \leqslant 0$$
$$g_5(x) = x_1^3 + x_2^3 + 1 \leqslant 0$$
$$g_6(x) = -x_1^3 - x_2^3 - 1 \leqslant 0$$
$$-2.3 \leqslant x_i \leqslant 2.3, i = 1,2$$
$$-3.2 \leqslant x_i \leqslant 3.2, i = 3,4,5$$

3.2 算法结果分析

3.2.1 参数设置 为了与文献[12-14]中的算法结果比较,笔者采用如下参数:交叉概率 $p_c = 0.008$;变异概率 $p_m = 0.008$;群体大小 $pop\_size = 70$;最大迭代代数:$Max\_gen = 20$;对BEA,独立运行20次。

表1给出了这5个函数的详细性质和标准结果。

表1 $G_1 \sim G_5$ 的性质和标准结果

	n	MLC	$MNLC$	Optimal solution	Optimal value
1	13	9	0	$(1,1,1,1,1,1,1,1,1,3,3,3,1)^T$	-15
2	8	3	3	$(579.316\,7, 1\,359.943, 5\,110.071, 182.017\,4, 295.598\,5, 217.979\,9, 286.416\,2, 395.597\,9)^T$	7 049.330 9
3	20	1	1	NA	0.804
4	7	0	4	$(2.330\,499, 1.951\,372, 0.477\,541\,4, 4.365\,726, 0.624\,487\,0, 1.038\,131, 1.594\,227))^T$	680.630 1
5	5	0	6	$(-1.717\,143, 1.595\,709, 1.827\,247, -0.763\,641\,3, -0.763\,645)^T$	0.053 95

注:表中左边表示函数 $G_1 \sim G_5$ 的序号,NA 表示目前还没求出其优结果,N 表示函数维数,MLC 表示问题中线性约束数目,$MNLC$ 表示非线性约束数目。目前函数3无确切的最优解,表中所给有解是从文献[13]中取得。

3.2.2 结果分析　由于 G_3 的标准结果目前未知,笔者只对 G_1,G_2,G_4,G_5 独立运行20次所得到的结果与文献[14]进行比较。从表2可以看出,由于采用了双群体策略,通过可行群体之间,不可行群体,可行群体与不可行群体之间交叉,可行群体与不可群体变就异,产生出的后代中可行个体的比例有所不同,对于 G_1,BEA 算法产生的可行个体的比例占总个体的 8.11%,而采用罚函数的文献[14],可行解只占到 1.11%;对于 G_2,BEA 算法产生的可行个体的比例占总个体的 3.11%,而采用罚函数的文献[14],可行解只占到 0.10%;对于 G_2,BEA 算法产生的可行个体的比例占总个体的 61.62%,采用罚函数的文献[14],可行解只占到 51.21%;对等式约束的 G_5,BEA 算法产生的可行个体远远大于文献[14]。故本算法优于基于罚函数为基础的传统算法。

从表3可以看出,在同样的实验条件下(独立运行20次),对于函数 G_1,*BEA* 的结果与 *RY* 一样好,比 *KM* 更好;均值较高于 *RY*,但低于 *KM*,对于方差与均值有类似结果;对于 G_2,*BEA* 的结果与 *RY* 不相上下,比 *KM* 更好,*BEA* 的方差小于 *RY*;对于 G_3,*BEA* 的结果比 *RY* 和 *KM* 差,考虑到函数的性态,说明本算法还有改进的地方;对于 G_4,*BEA* 的结果比 *RY* 和 *KM* 好,但方差比 *RY*、*KM* 大;对于 G_5,*BEA* 的结果和 *RY*、*KM* 差不多,*BEA* 和 *RY* 的方差大致一样,都比 *KM* 小,这说明算法在处理等式约束时也是有效的。

表2　算法 BEA 独立运行 20 次最优结果与文献[14]比较

FCN	*R* ~ *F*		Optimal Value		Mean Value	
	BEA	文献[14]	*BEA*	文献[14]	*BEA*	文献[14]
1	0.081 1	0.011 1	−15	−15.002	−14.89	−15.002
2	0.031 1	0.001 0	7 048.3	2 282.723	7 325.91	2 449.798
4	0.616 2	0.512 1	680.13	680.771	624.06	681.262
5	0.001 7	0	0.054 2	0.084	0.107 2	0.955

注:独立 20 次运行中,*R* ~ *F* 表示产生可行解的平均比例,Optimal Value 表示最优值,Mean 表示均值,在文献[14]中取的是 Method#1。

表3　*BEA* 与 RY[13]、KM[12] 的比较

FCN	Best			Mean			STD		
	Opt	*BEA*	*RY*	*KM*	*BEA*	*RY*	*KM*	*BEA*	*RY*	*KM*
1	−15	−15	−15	−14.786	−14.89	−15	−14.71	0.046	0	0.054
2	7 049.3	7 048.3	7 054.3	7 147.9	7 325.91	7 559.2	8 163.6	140.7	141.7	616.1
3	0.804*	0.914	−0.804	−0.799	0.827	−0.782	−0.780	0.060	0.006	0.002
4	680.63	680.13	680.63	680.91	624.06	680.66	681.16	20.397	0.009	0.127
5	0.054 0	0.054 2	0.054 0	0.054	0.107 2	0.057	0.064	0.065	0.064	0.076

注:数据采用最多保留3位有效数字,故与文献[14]中的结果有出入,但不影响结果分析。表中 Best 表示最优解,*Opt* 表示目前已知的标准最优解,Mean 表示独立运行 20 次的均值,STD 表示方差。

4 结论

本文给出了一种基于双群体进化的实数编码遗传算法,用以解决单目标约束优化问题。该方法没有使用传统的罚函数方法,而是计算个体的违反约束条件的程度。根据个体的可行性对群体进行分类,并在进化过程中采用精英保留策略,分别对可行个体群和非可行个体群进行操作。在进化过程中,利用非可行个体群中的个体所携带信息,对整个搜索过程进行修正,使搜索过程逐步向最优解移动。从数值试验的结果可以看出,随着进化(迭代)代数的增加,群体的最优性逐渐向预期的方向发展,搜索结果的标准差也呈减少的趋势,从而验证了该算法的有效性。

从所选的5个测试函数的试验中可以看出试验结果从数值精度上基本上不低于已知结果,甚至在某些函数上比已知结果还要好,而且,更重要的是,本文的算法可以在进化(迭代)代数很少的情况下,能够较迅速地收敛到预期的结果。从以上分析中可以得出,本算法性能比较稳定且有效可行。关于对此算法的理论收敛性分析和算法改进将是以后研究的一个问题。

参考文献:

[1] Garey M R, Johnson D S. Computer and intractability-A guide to the theory of NP-completeness[M]. New York: W. H. Freeman & Co, 1990.

[2] Peng Y H, Yao S B. A feasible trust-region algorithm for inequality constrained optimization[J]. Applied Mathematics and Computation, 2006, 173(1): 513-522.

[3] Wei Z X, Liu L Y, Yao S W. The superlinear convergence of

a new Quasi-Newton-SQP method for constrained optimization[J]. Applied Mathematics and Computation,2008,196(2):791-801.

[4] Jian J B,Tang C M,Hu Q J,et al. A new superlinearly convergent strongly subfeasible sequential quadratic programming algorithm for inequality-Constrained optimization[J]. Numerical Functional Analysis and Optimization,2008,29(3):376-409.

[5] Zhang M,Luo W,Wang X F. Differential evolution with dynamic stochastic selection for constrained optimization[J]. Information Sciences,2008,178(15):3043-3074.

[6] Huang F Z,Wang L,He Q. An effective co-evolutionary differential evolution for constrained optimization[J]. Applied Mathematics and Computation,2007,186(1):340-356.

[7] Jian J B,Tang C M. A self-organizing migrating genetic algorithm for constrained optimization[J]. Applied Mathematics and Computation,2008,198(1):237-250.

[8] Deb K,Agrawal S. A niched penalty approach for constraint handling in genetic algorithms[C]//Proceedings of thd ICANNGA299,Portoroz,Slovenia,1999:234-239.

[9] Coello C,Carlos A. Constraint-handing using an evolutionary multiobjective optimization technique[J]. Civil Engineering and Environmental Systems,2000(17):319-346.

[10] Floudas C A,Pardalos P M. A collection of test problems for constrained global optimization algorithms[M]. New York:Springer-Verlag New York,1990.

[11] Hock W,Schittkowski K. Test examples for nonlinear programming codes[C]. Springer-Verlag:Lecture Notes in Economics and Mathematical Systems,1981,187:127-129.

[12] Koziel S,Michalewicz Z. Evolutionary algorithms, homomorphous mapping, and constrained parameter optimization[J]. Evolutioanry computation,1999,7(1):19-44.

[13] Runarsson T P,Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation,2000,4(3):284-294.

[14] Michalewicz Z. Genetic algorithms numerical optimization and constraints[C]//In Eshelman LJ, ed. Proceedings of the 6th International Conference on Genetic Algorithms, San Mateo:Morgan Kaufmann Publishers,1995:151-158.

Bi-population Based on Evolutionary Algorithm for Solving Nonlinear Constrained Programming

JIA Li-ping, ZOU Guo-cheng

(Dept. of Mathematics, Leshan Normal University, Leshan Sichuan 614000, China)

Abstract: It is difficult to handle constrained conditions in solving nonlinear constrained programming. In this paper a bi-population based on evolutionary algorithm for solving nonlinear constrained programming is proposed. The algorithm is different from other algorithms in that one defines the violation based on constrained conditions to measure the individuals, and defines the optimal value based on objective value to measure quality of individuals in population. Firstly, we consider the standard constrained optimization problem and state different methods to handle constraints, then present a different method. For degree of violation, we define two populations: feasible and infeasible population. Finally, we present the detailed steps of bi-population evolutionary algorithm. The feasibility and effectiveness are verified by comparing other existed algorithms with the same five benchmark functions.

Key words: nonlinear constrained programming; evolutionary algorithm; bi-population; operator

(责任编辑 游中胜)