文章编号:1000 - 582X(2006)06 - 0078 - 04

基于自适应惩罚函数法的混合遗传算法*

刘琼荪,周声华
(重庆大学 数理学院,重庆 400030)

摘　要:引入了自适应的惩罚因子,将约束问题转化为无约束问题. 通过遗传算法求得无约束问题的可行解,再将此解作为约束变尺度法的初始可行点,由约束变尺度法得到精度较高的解. 数值实验表明该混合算法比单纯使用遗传算法效率高,而且在多数情况下能得到全局最优解.

关键词:非线性规划;惩罚函数;遗传算法;约束变尺度法
中图分类号:O221.2; O231　　**文献标识码**:A

考虑一般的非线性约束优化问题:

$$\left.\begin{array}{ll}\min & f(x)\\ \text{s.t.} & hi(x)=0 \quad i=1,\cdots,m_2\\ & g_j(x)\le 0 \quad j=1,\cdots,m_2\\ & I\le x\le u\end{array}\right\} \tag{1}$$

式中 $x=(x_1,\cdots,x_n)^{\mathrm{T}}$, $I=(l_1,\cdots,l_n)^{\mathrm{T}}$, $u=(u_1,\cdots,u_n)^{\mathrm{T}}$.

此类问题是优化中涉及最广的问题. 传统的方法主要有可行方向法、惩罚函数法[1-4]、约束变尺度法[1-3],其中约束变尺度法最为流行,但这些方法都极易陷入局部极小点. 1975 年出现了具有全局搜索性能的遗传算法[3-4],但其局限于仅含变量上下限约束的优化问题. 近年来,用遗传算法解决含复杂约束的非线性规划问题的研究一直是一个学术热点,并取得了一系列可喜的成果.

目前,最流行的方法就是借鉴传统优化算法中的惩罚函数法,将非线性约束优化问题转化为仅含变量上下限约束的优化问题,再由遗传算法进行求解. 然而这种方法满足约束条件的精度往往达不到要求,并且运行效率也比较低,为此,又出现了遗传算法和传统优化算法相结合的混合遗传算法[3-9]. 然而对于惩罚函数法 x 惩罚因子常难以合理选取. 笔者引入了随自变量变化的惩罚因子,由此构造了一种自适应的惩罚函数,并将遗传算法和约束变尺度算法相结合,实例证明这样可有效地解决约束非线性规划问题.

1　惩罚函数的构造

一般地,由问题(1)构造的辅助函数为:

$$f_p(x)=f(x)+\sum_{i=1}^{m_1}K_i\cdot|h_i(x)|+\sum_{j=1}^{m_2}M_j\cdot\max\{0,g_j(x)\}, \tag{2}$$

其中 $p(x)=f(x)+\sum_{i=1}^{m_1}K_i\cdot|h_i(x)|+\sum_{j=1}^{m_2}M_j\cdot\max\{0,g_j(x)\}$,为惩罚函数,$Ki(i=1,\cdots,m_1)$、$M_j(j=1,\cdots,m_2)$ 为惩罚因子,通常取较大的值. 因此,问题(1)可转化为如下问题:

$$\left.\begin{array}{l}\min f_p(x)\\ \text{s.t. } I\le x\le u\end{array}\right\}. \tag{3}$$

在实际计算中,惩罚因子的选取很重要. 如果惩罚因子过大(一般取一系列趋向于无穷大的严格递增正数列$\{K_i^n,\ n\to\infty\}$, $\{M_i^n,\ n\to\infty\}$),函数 $f_p(x)$ 会因对 $f(x)$ 分配的权重(固定取 1)过小而忽略对目标函数 $f(x)$ 的影响,往往得到的是非线性优化问题(1)的局部最优解,同时也给计算增加困难. 如果惩罚因子过小,惩罚项得不到足够的惩罚,满足约束条件的精度就会降低. 因此,惩罚因子常难以合理地确定.

为了较好地解决惩罚因子的确定问题,笔者将惩罚因子选取为关于自变量 x 的函数,并且为了加快收敛速度,借鉴"多级惩罚[7]"的思想——对违反约束大的段给予较大的惩罚而违反约束小的段给予较小的惩罚,按下式构造惩罚函数:

$$p(x)=\sum_{i=1}^{m_1}K_i\cdot|h_i(x)|+\sum_{j=1}^{m_2}M_j\cdot\max\{0,g_j(x)\}, \tag{4}$$

式中,

$$K_i(x)=\frac{|h_i(x)|}{\sum_{i=1}^{m_1}|h_i(x)|+\sum_{j=1}^{m_2}\max\{0,g_j,(x)\}},\ i=1,\cdots,m1, \tag{5}$$

* 收稿日期:2006 - 02 - 01
作者简介:刘琼荪(1956 -),女,重庆人,重庆大学教授,主要从事数理统计、智能算法的研究.

$$M_j(x)=\frac{\max\{0,g_j(x)\}}{\sum_{j=1}^{m_2}\max\{0,g_j(x)\}},\ j=1,\cdots,m_2 \tag{6}$$

辅助函数构造如下:

$$f_p(x)=f(x)+C(x)\cdot p(x), \tag{7}$$

式中

$$C(x)=1+\frac{|f(x)|}{1+p(x)}. \tag{8}$$

2 惩罚函数的自适应性

对于因子 $K_i(x)(i=1,\cdots,m_1)$, $M_j(x)(j=1,\cdots,m_2)$, $C(x)$和惩罚函数$p(x)$,具有如下性质:

1) $\sum_{i=1}^{m_1}K_i(x)+\sum_{j=1}^{m_2}M_j(x)=1, 0\leqslant K_i(x)\leqslant 1, i,\cdots,m_1$, $0\leqslant M_j(x)\leqslant 1, j=1,\cdots,m_2$.

2) 当 x 是非可行点,如果 $\exists i=l$, $|h_i(x)|>|h_l(x)|\neq 0$ 时,则 $K_i(x)>K_l(x)$;如果 $\exists j\neq k, g_j(x)>g_k(x)>0$,则 $M_j(x)>M_k(x)$.

3) 当 x 是非可行点,对 $\forall\ i\neq j$,如果 $|h_i(x)|>g_j(x)$,则 $K_i(x)>M_j(x)$,反之亦然.

4) $p(x)\geqslant 0, C(x)\geqslant 1$,且如果 x 是非可行点,则 $p(x)>0$,反之亦然.

上述性质显然成立.

关于此惩罚函数法的自适应性有如下两个定理:

定理1 对非可行点 x_0;如果 $0<\frac{|f(x_0)|}{p(x_0)}\leqslant 1$ 时,则 $1<C(x_0)<2$,且当 $\frac{|f(x_0)|}{p(x_0)}\to 0$, $(|f(x_0)|\ll p(x_0))$ 则 $C(x_0)\to 1$.

证明: 显然,由 $C(x_0)=1+\frac{|f(x_0)|}{f+p(x_0)}$ 可直接推得.

定理2 对非可行点 x_0,设 $|f(x_0)|=M\cdot p(x_0)$, $M>0$ 为常数. 若 $M\to+\infty$,且 $p(x_0)\to+\infty$,则 $\frac{C(x_0)\cdot p(x_0)}{f(x_0)}\to 1$;若 $M\to+\infty$;且 $p(x_0)\to 0$,则 $\frac{C(x_0)\cdot p(x_0)}{|f(x_0)|}\to 0$.

证明: 由 $\frac{C(x_0)\cdot p(x_0)}{f(x_0)}=\frac{p(x_0)}{f(x_0)}+\frac{p(x_0)}{1+p(x_0)}=\frac{1}{M}+\frac{p(x_0)}{1+p(x_0)}$ 可直接推得.

上述性质2)、3)和定理说明了惩罚因子 $K_i(x)(i=1,\cdots,m), M_j(x)(j=1,\cdots,n), C(x)$ 具有"多级惩罚"的功能,并且在算法的迭代过程中起到了"自适应"调节的作用. 定理1说明,当目标函数绝对值相对惩罚函数值较小时,惩罚函数的系数会很小并在(1,2)内,以增加目标函数值对辅助函数的影响,当目标函数值为0时,辅助函数就退化为惩罚函数. 定理2说明,当目标函数值相对于惩罚函数值较大且惩罚函数值本身就很大时,辅助函数会权衡目标函数和惩罚函数的影响(各自影响均约50%);当目标函数值相对惩罚函数值较大且罚函数值近似为0时,辅助函数会着重考虑目标函数的影响(当然这会降低可行解近似满足约束的精度,这也是惩罚函数法与约束变尺度法相结合的原因).

3 基于惩罚函数法的遗传算法与约束变尺度法的结合

基于惩罚函数法的遗传算法通常需要提高计算精度就要增大惩罚函数项的权重,而这样又会陷入局部极小甚至得不到最优解. 因此,为了既能得到全局最优解同时又能提高精度,笔者将其与约束变尺度法进行结合. 先通过遗传算法,尽量搜索到全局最优解区域,再利用局部搜索能力强的约束变尺度法搜索到高精度的可行解,具体步骤如下:

Step1 对优化问题(1),按(4)到(8)式构造辅助问题:

$$\max\ \tilde{f}_p(x)=-f_p(x),$$
$$\text{s.t. } \boldsymbol{l}\leqslant x\leqslant \boldsymbol{u}.$$

Step2 利用遗传算法对上问题求解,得可行解 x_0,若精度满足要求,终止;否则转Step3.

Step3 以 x_0 为起始点利用约束变尺度法对问题(1)求解,获得最优解 x^*.

4 计算实例

1) $\min f_1(x)=e^{x_1x_2x_3x_4x_5}+10(x_1-2x_2^2+5x_3+x_4x_5)$, s.t. $x\in A$,

2) $\min f_2(x)=0.1f_1(x)$, s.t. $x\in A$,

3) $\min f_3(x)=-20e^{-0.2\sqrt{\frac{1}{3}\sum_{i=1}^{3}xi^2}}-e^{\frac{1}{3}\sqrt{\sum_{i=1}^{3}\cos(2\pi x_i)}}$ s.t. $x\in B$,

4) $\min f_4(x)=100f_3(x)$, s.t. $x\in B$,

$$\boldsymbol{A}=\left\{\begin{pmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{pmatrix}\middle|\begin{array}{l}x_1^2+x_2^2+x_3^2+x_4^2+x_5^2=10\\x_2x_3-5x_4x_5=0\\x_1^3+x_2^3=-1\\-10\leqslant x_i\leqslant 10,\ i=1,\cdots,5\end{array}\right\}$$

$$\boldsymbol{B}=\left\{\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\middle|\begin{array}{l}x_1+x_2+x_3-20=0\\1.5+x_1x2+x_2x_3-2x_1-3x_2-x_3\leqslant 0\\x_1x_2x_3-100\leqslant 0\\1\leqslant x_1,x_2,x_3\leqslant 20\end{array}\right\}$$

采用上述4个算例对本文算法进行检验,其中算例(2)和(4)分别是将算例(1)与(3)目标函数减小10

倍和放大100倍,主要是为了检验提出的惩罚函数法的自适应能力. 算例(1)、(2)的最优解为$(-1.559\,9\ 1.408\,8\ -2.097\,6\ \pm0.768\,8\mp0.768\,8)^{T}$,目标函数值分别为-166.017、-16.601 7;算例(3)、(4)的最优解为$(8.997\,7\ 1.005\,6\ 9.996\,7)^{T}$,目标函数值分别为-6.931 2、-6 931.2.

在单纯采用遗传算法时:算例(1)和算例(2)种群中的个体数目取为50,迭代次数为500,交叉率0.95,变异率0.08;算例(3)和(4)种群中的个体数目为30,迭代次数为300次,交叉率0.95,变异率0.08. 在采用混合遗传算法时:种群中的个体数目皆为20,迭代次数为200,交叉率0.95,变异率0.1. 另外,对常系数惩罚函数法,系数统一取为20,遗传算法均采用二进制编码,对各问题分别计算6次,其结果如表1-2:

表1 算例(1)和(2)计算结果

计算次数		1	2	3	4	5	6
函数值	Meth1	-161.747 1	-164.957 5	-160.530 1	-144.778 9	-157.295 9	-158.656 1
	Meth2	-152.978 0	-158.761 5	-155.288 9	-150.956 9	-148.187 8	-164.023 7
	Meth3	-166.017 0	-106.771 4	-156.836 8	-166.017 0	-166.017 0	-166.017 0
f_1 最大约束误差	Meth1	$5.762\,9\times10^{-5}$	0.004 5	$8.244\,3\times10^{-6}$	$1.879\,7\times10^{-4}$	$2.942\,4\times10^{-6}$	$3.044\,4\times10^{-6}$
	Meth2	$4.287\,2\times10^{-6}$	$1.609\,6\times10^{-6}$	$1.491\,8\times10^{-4}$	$1.403\,6\times10^{-4}$	$2.315\,2\times10^{-4}$	$8.623\,3\times10^{-6}$
	Meth3	$1.184\,5\times10^{-9}$	$4.656\,1\times10^{-9}$	$2.547\,1\times10^{-9}$	$3.603\,9\times10^{-9}$	$2.230\,5\times10^{-9}$	$2.698\,8\times10^{-9}$
计算时间	Meth1	12.528 0	11.887 0	11.717 0	11.857 0	12.127 0	11.427 0
	Meth2	12.037 0	12.748 0	12.308 0	12.298 0	12.178 0	12.107 0
	Meth3	6.237 0	6.098 0	6.149 0	7.210 0	6.179 0	6.119 0
函数值	Meth1	-15.703 9	-5.940 9	-13.729 8	-14.703 8	-11.417 4	-9.243 9
	Meth2	-15.485 9	-14.954 7	-15.899 9	-15.618 7	-14.432 0	-14.830 8
	Meth3	-16.601 7	-15.683 7	-15.683 7	-16.601 7	-16.601 7	-15.683 7
f_2 最大约束误差	Meth1	$6.566\,8\times10^{-6}$	$5.132\,6\times10^{-7}$	1.960×10^{-6}	$4.037\,1\times10^{-6}$	$2.506\,0\times10^{-5}$	$5.891\,2\times10^{-6}$
	Meth2	$1.069\,7\times10^{-5}$	$2.507\,9\times10^{-6}$	$4.989\,3\times10^{-5}$	$2.868\,0\times10^{-5}$	$4.051\,6\times10^{-5}$	$1.784\,6\times10^{-5}$
	Meth3	$6.290\,3\times10^{-9}$	$7.752\,5\times10^{-10}$	$7.010\,1\times10^{-9}$	$1.807\,9\times10^{-9}$	$1.798\,6\times10^{-8}$	$4.331\,7\times10^{-8}$
最大约束误差	Meth1	11.577 0	11.786 0	11.346 0	11.446 0	11.427 0	11.737 0
	Meth2	12.170 0	12.448 0	13.990 0	12.427 0	13.269 0	12.077 0
	Meth3	6.179 0	5.928 0	5.909 0	6.349 0	6.139 0	6.760 0

表2 算例(3)和(4)计算结果

计算次数		1	2	3	4	5	6
函数值	Meth1	-5.775 1	-5.204 8	-3.395 5	-6.017 7	-3.944 1	-6.525 4
	Meth2	-6.857 9	-6.528 3	-6.295 7	-5.471 6	-6.712 7	-5.205 2
	Meth3	-5.480 7	-6.037 5	-6.532 3	-5.480 7	-5.480 7	-6.037 5
f_1 最大约束误差	Meth1	$5.640\,5\times10^{-10}$	$1.301\,4\times10^{-8}$	$1.859\,4\times10^{-8}$	$5.692\,2\times10^{-10}$	$8.663\,7\times10^{-9}$	$2.843\,4\times10^{-9}$
	Meth2	$5.681\,7\times10^{-8}$	$8.296\,6\times10^{-9}$	$1.406\,8\times10^{-9}$	$3.088\,3\times10^{-7}$	$1.249\,8\times10^{-7}$	$1.055\,8\times10^{-9}$
	Meth3	$1.360\,7\times10^{-10}$	$1.882\,9\times10^{-12}$	$7.105\,4\times10^{-15}$	$5.435\,7\times10^{-13}$	$2.065\,8\times10^{-10}$	$2.344\,8\times10^{-13}$
计算时间	Meth1	8.572 0	7.501 0	7.531 0	7.481 0	7.641 0	7.791 0
	Meth2	8.402 0	8.062 0	7.982 0	8.002 0	8.091 0	8.131 0
	Meth3	7.401 0	6.129 0	6.049 0	6.229 0	6.229 0	6.309 0
函数值	Meth1	-1 909.3	-1 909.3	-1 909.3	-1 909.3	-1 909.3	-1 909.3
	Meth2	-629.780 0	-515.190 4	-608.663 6	-575.741 5	-608.730 5	-672.201 9
	Meth3	-548.072 3	-608.730 6	-672.372 4	-603.746 1	-672.372 4	-693.124 4
f_2 最大约束误差	Meth1	17	17	17	17	17	17
	Meth2	$1.679\,9\times10^{-9}$	$1.704\,9\times10^{-9}$	$1.058\,7\times10^{-9}$	$3.265\,3\times10^{-6}$	$4.424\,3\times10^{-8}$	$5.140\,8\times10^{-8}$
	Meth3	$6.394\,9\times10^{-14}$	$3.769\,8\times10^{-11}$	$4.419\,6\times10^{-12}$	$2.245\,3\times10^{-12}$	$8.896\,0\times10^{-12}$	0
计算时间	Meth1	7.6900	7.1900	7.8450	7.2870	7.3960	7.4320
	Meth2	8.6030	8.1220	8.1210	8.1210	8.2510	8.1320
	Meth3	6.5200	6.3790	6.3390	6.2590	6.1880	6.2490

表中:Meth1——基于常系数惩罚函数法的遗传算法;Meth2——基于自适应惩罚函数法的遗传算法;Meth3——基

于自适应惩罚函数法的遗传算法与约束变尺度法的结合.

5 结 论

可以看出:常系数惩罚函数法对于算例(1)和(3)比较理想,但当惩罚函数的系数保持不变,用于算例(2)和(4)时,就出现了问题. 对于算例(2),由于其目标函数值相对算例(1)中的目标函数值减为原来的$\frac{1}{10}$,从而辅助函数$f_p(x)$中的目标函数值的影响就会变弱,使计算得到的最优目标函数值降低(第二次计算得到的值仅为 -5.940 9). 相反,对于算例(4),其目标函数值相对算例(3)中的目标函数值放大一百倍后,辅助函数中的目标函数值的影响变大,约束条件的影响变弱,从而最大约束误差项$f_p(x)$就会变大(值为17). 而采用笔者的自适应惩罚函数法,不会受目标函数值缩小或放大的影响,只是在计算时间上相对常系数惩罚函数法有微小的增加. 若采用自适应惩罚函数法和约束变尺度法相结合,不仅在计算精度上得到了提高,而且在计算时间上也会减少,并且计算结果都稳定在全局最优解附近.

参考文献:

[1] 盛昭瀚,曹忻. 最优化方法基本教程[M]. 江苏:东南师范大学出版社,1992.

[2] 刘宝光. 非线性规划[M]. 北京:北京理工大学出版社,1988.

[3] 飞思科技产品研发中心. MATLAB6.5 辅助优化计算与设计[M]. 北京:电子工业出版社,2003.

[4] JOINES J A, HOUCK C R. On the Use of Non - stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with Gas[Z]. Proceedings of the Evolutionary Computation Conference, Orlando, 1994.

[5] 王等刚,刘迎曦. 求解一类非线性规划问题的混合遗传算法[J]. 上海交通大学学报,2003,37(12):1952 - 1956.

[6] 吴浩扬,朱长纯. 基于种群过早收敛程度定量分析的改进自适应遗传算法[J]. 西安交通大学学报, 1999, 33(11):27.

[7] HOMAIFAR A, LAI S H Y, QI X. Constrained Optimization via Genetic Algorithms [J]. Simulation, 1994, 62(4): 242 - 254.

[8] 王小平,曹立明. 遗传算法理论、应用与软件实现[M]. 西安:西安交通大学出版社,2002. 76 - 79.

[9] MICHALEWICA Z, SCHOENAUER M. Evolution algorithms for constrained parameters optimization problems[J]. Evolutionary Computation, 1996, 4(1):61 - 64.

Hybrid Genetic Algorithm Based on Novel Adaptive Penalty Function

LIU Qiong-sun, ZHOU Sheng-hua

(College of Mathematics and Physics, Chongqing University, Chongqing 400030, China)

Abstract: The authors introduce a sort of novel adaptive penalty gene, transform the constrained problem into unconstrained problems. An solution is given for this unconstrained problem with genetic algorithm, and then it is used as initial values for the constrained variable metric method to get precise solution. The numerical experiments illustrate that this hybrid genetic algorithm is more efficient than the genetic algorithm, and at most situations globally optimal solution can be gotten.

Key words: nonlinear programming problems; penelty function; genetic algorithm; the constrained variable metric method

(编辑　张小强)