文章编号: 1001-0920(2001)0S-0669-04

自适应乘子在工程优化问题中的应用

张春慨, 邵惠鹤
(上海交通大学 自动化研究所, 上海 200030)

摘　要: 在增广 Lagrange 乘子函数和协作进化算法的基础上, 采用自适应策略来解决工程优化问题。其中存在两组相互作用的进化种群 A 和 B, A 利用 B 的反馈信息来评价 A 中乘子个体的优劣, 进而进化 A 中乘子个体; B 则利用 A 中乘子个体来动态进化原问题的候选解个体。在算法迭代过程中, 罚因子是确定性逐渐增大, 以使算法逐渐收敛; A 中乘子个体则朝其最优值的方向进化, 且由 B 可得到原问题的最优解。与基于静态和动态罚函数的 GAs 相比, 该算法准确度和鲁棒性较高, 易于实现, 并适合并行化计算。

关键词: 约束优化问题; 协同进化; Lagrange 乘子; 自适应策略
中图分类号: TP 18　　文献标识码: A

Application of Self-adaptive Multiplier in Engineering Optimization Problem

ZHANG Chun-kai, *SHAO Hui-he*
(Institute of Automation, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: A new method based on coevolution is proposed where two populations interact with each other to coevolute the multiplier and solution. Compared with the standard GAs with static and dynamic penalty parameters, the test results show that this algorithm is easy to be implemented, and its efficiency is higher in accuracy and reliability, so it is effective for constrained optimization problem.

Key words: constrained optimization problem; coevolution; Lagrange multiplier; self-adaptive

1 引　　言

在工程优化问题中, 约束是不可避免的, 作为一种解决优化问题的新型算法, 进化算法(EAs)已应用在许多实际工程优化中。以往一些利用 EAs 算法处理约束优化问题的方法大致归为 4 类[1]: 1) 基于保留解的可行区域方法; 2) 基于惩罚不可行解方法; 3) 明确可行解和不可行解方法; 4) 基于解码方法。其中一些用来处理约束的策略或多或少的与求解的问题有关, 例如有关线性约束的信息可以被结合在特别的操作算子中, 或设计一个修复算子以使不可行解映射到可行解域, 或设计特殊解码和其他特殊操作算子等。就目前来看, 对于实际工程优化问题, 处理约束较为通用的方法仍是惩罚不可行解方法, 即罚函数和乘子法。由于乘子或罚因子的更新过程并不需要个别约束的梯度信息, 因此很适合一些工程应用。但由于罚函数和乘子法本身固有的问题(如何

收稿日期: 2001-04-23
基金项目: 国家 973 重点基础研究发展项目(G1998030415)
作者简介: 张春慨(1973—), 男, 山东烟台人, 博士生, 从事优化方法、人工智能研究; 邵惠鹤(1936—), 男, 浙江宁波人, 教授, 博士生导师, 从事工业生产过程计算机控制与优化、软测量技术等研究。

确定罚因子和乘子的更新策略), 使得罚函数或乘子法在 EAs 中的应用遇到较大困难。例如, 对于罚函数而言, 如果罚因子 σ 选取的太小, 则罚函数的极小点远离约束问题的最优解; 反之, 则易使算法陷入局部极小点, 且给计算增加困难; 同样问题也存在于乘子法中。

本文提出一种处理约束的合作协作进化算法, 其建立在增广 Lagrange 乘子法基础上, 利用两个相互作用的种群, 分别进化乘子和原问题候选解, 并在连续的相互作用和协同进化中, Lagrange 乘子趋向于最优值, 而使原问题的候选解收敛到最优值。

2 基于乘子法的合作协作 EAs 算法

2.1 增广 Lagrange 乘子法

考虑一般约束优化问题

$$\begin{aligned}&\min_{x\in S} \quad f(x)\\ &\text{s.t.} \quad g_i(x)=0, \quad i=1,2,\cdots,l\\ &\qquad g_i(x)\le 0, \quad i=l+1,l+2,\cdots,m \end{aligned} \tag{1}$$

其中, $S\subset R^n$ 为搜索空间, 是一个 n 维立方体, 即由 $l_i\le x_i\le u_i(i=1,2,\cdots,n)$ 组成; $f:S\to R$ 为目标函数; 集合 $F=\{x\in S; g_i(x)=0,\ i=1,2,\cdots,l,\ g_i(x)\le 0,\ i=l+1,l+2,\cdots,m\}$ 为可行区域。

为了处理约束, Hestenes 和 Powell 于 1968 年各自提出了乘子法[2]。增广 Lagrange 乘子函数定义为

$$\begin{aligned}\Phi(x,\theta,r)=&\\ f(x)+\frac{1}{2}\sum_{i=1}^{l}r_i[(g_i+\theta)^2-\theta^2]+&\\ \frac{1}{2}\sum_{i=l+1}^{m}r_i[\max(g_i+\theta,0)^2-\theta^2]&\end{aligned} \tag{2}$$

其中, r_i 是罚因子, $u_i=r_i\theta_i$ 是 Lagrange 乘子, 为了简化起见, $r_i=r$。

增广 Lagrange 乘子函数与罚函数的区别在于增加了乘子项, 这种区别使得增广 Lagrange 乘子函数与 Lagrange 乘子函数及罚函数相比, 具有不同的特点: 对于 $\Phi(x,\theta,r)$ 而言, 如果事先知道最优乘子 u_i, 只要取足够大的罚因子 r, 就可通过极小化 $\Phi(x,\theta,r)$ 求得原问题的局部最优解[2], 在某种程度上克服了罚函数法中因罚因子趋向无穷大而带来的病态性[3]。

2.2 基于乘子法的合作协作 EAs 算法

在求解增广 Lagrange 乘子函数时, 由于变量之间关系复杂, 且目标函数或约束不可微时采用确定性方法来更改 Lagrange 乘子是不现实的。比较可行的方法是采用自适应策略, 即先给定充分大的罚因子 r 和 Lagrange 乘子的初步估计 u_i^0, 然后在算法迭代中根据反馈信息自适应地修正第 k 代中的乘子 u_i^k, 以使第$(k+1)$ 代的乘子 u_i^{k+1} 趋向最优值。

本文提出基于乘子法的合作协作 EAs 算法, 其利用相互作用的进化种群[4], 分别进化 Lagrange 乘子和原问题的候选解。其中种群 A 利用种群 B_j 的适配值信息来评价种群 A 中乘子个体的优劣, 进而进化乘子个体; 种群 B_j 则利用种群 A 的乘子个体而动态地改变其适应值函数以进化原问题的候选解。在算法迭代过程中, 罚因子逐渐增大, 而 Lagrange 乘子则逐渐趋向于最优值, 并且原问题的最优解可在种群 B_j 中得到。

基于乘子法的合作协作 EAs 算法的步骤如下:

1) 初始化

种群 A: 规模为 $S1$, 浮点数编码, 编码内容是乘子 u_i。均匀地随机产生 $S1$ 个个体, 并确定交叉概率和突变概率。

种群 B: 对应于种群 A 中的第 j 个个体, 均匀随机产生一个规模为 $S2$ 的种群 B_j, 浮点数编码, 编码内容是 $x_i(i=1,2,\cdots,n)$, 确定交叉概率和突变概率。这样共有 $S1$ 个种群 $B_j(j=1,2,\cdots,S1)$。

2) 确定罚因子 r 的更新规则

$$r=(Ck)^{\alpha} \tag{3}$$

其中, $C=0.5,\alpha=2,k$ 是算法迭代次数。

3) 算法迭代

for $k=1,2,\cdots,$ max-cycles

对应于种群 A 中第 j 个乘子个体的编码内容, 确定种群 B_j 的适应值函数

for $j=1,2,\cdots,S1$

for $i=1,2,\cdots,$ max-gen-b

进化种群 B_j, 其适配值函数如

$$\begin{aligned}\text{fitness}_B=f(x)+\frac{1}{2}\sum_{j=1}^{l}r_j[(g_i+\theta_i)^2-\theta_i^2]+&\\ \frac{1}{2}\sum_{i=l+1}^{m}r_i[\max(g_i+\theta,0)^2-\theta^2]&\end{aligned} \tag{4}$$

end

end

for $i=1,2,\cdots,$ max-gen-a

进化种群 A, 其适配值函数如

$$\text{fitness}_A=\sum_{i=1}^{S2}\left(\frac{\text{feasible-fitness}_i+\text{unfeasible-fitness}_i}{S2}\right) \tag{5}$$

式(5) 中, 种群 A 的第 j 个个体的适配值 fitness_j 等于对应种群 B_j 中所有个体的适配值平均值; $\text{feasible-fitness}_i$ 和 $\text{unfeasible-fitness}_i$ 分别是种群 B_j 中位于可行域和非可行域的个体适配值函数, 分别如

$$\text{feasible-fitnesss}_i = f(x) \tag{6}$$

$$\text{unfeasible-fitnesss}_i = f_{\max} + \frac{1}{2}\sum_{j=1}^{m} g_j^2 + \frac{1}{2}\sum_{j=1}^{l} h_j^2(x) \tag{7}$$

其中, $f_{\max}$ 是种群 B_j 中可行解中最大的目标函数, 即种群 B_j 中位于可行域中最差个体的目标函数, 如果种群 B_j 中不存在可行解, 那么 $f_{\max}$ 等于种群 B_j ($j = 1, 2, \cdots, S1$) 中可行解中最大的目标函数, 否则如果都不存在可行解, 那么 $f_{\max} = 0$。

end

根据式(3) 更新罚因子

end

4) 算法结束

在式(5) 中, 定义种群 A 的个体适应值函数的原则是: ① 任何可行解都要好于任何不可行解; ② 在两个可行解之间, 选择具有好的目标函数的解; ③ 在两个不可行解之间, 选择具有小的约束违犯值的解。

进一步地说, 为了进化所有解, 通常先检查解是否违犯约束, 如果解是不可行的, 算法就不需要计算其目标函数。因为在实际中这样的解不能实现, 而且不可行解的适应值大小除了与其违犯约束的大小有关, 还与最差可行解的适应值有关。

在算法迭代过程中, 罚因子是逐渐增大的, Lagrange 乘子自适应地趋向于最优值, 而原问题的最优解可在种群 B_j 中得到。对于种群 B_j, 其进化代数是恒定的, 且每次进化均是在前一次进化基础上进行的。其优点是种群具有继承性, 这在一定程度上可以防止算法不稳定。但这样可能导致种群多样性降低, 因此在进化初期, 取突变概率较大, 且随着进化进行, 突变概率降低。

3 测试函数

为了测试该算法的性能, 考虑在文献中提到的几个约束优化问题。为合理评价本文算法, 在进化单个种群时使用了标准 EAs 算法, 且其参数没有精细调整。文献中提出的 GAs 算法, 为了处理约束, 使用了罚函数方法, 包括自适应罚函数和静态罚函数(取不同的罚因子), 且其参数进行了精细调整。在本文算法中, Lagrange 乘子的范围为 $1 \le u_i \le 999$。

问题 **1**

$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$$

约束条件

$$x_1 - 2x_2 + 1 = 0$$

$$-x_1^2/4 - x_2^2 + 1 \ge 0$$

$$0 \le x_1 \le 10, \quad 0 \le x_2 \le 10$$

该问题的参考全局最优点为 $x^* = (0.823, 0.911)$, 参考最优目标函数值为 $f^* = 1.393$。本文算法和文献中的算法各自运行 10 次所得到的最好结果如表 1 所示, 其中本文算法运行 10 次所得到的平均最优目标函数为 1.408 7, 所找到的最差目标函数为 1.431 2。

问题 **2**　Himmelblau 的非线性优化问题[6]

$$f(x) = 5.357\,854\,7x_3^2 + 0.835\,689\,1x_1x_5 + 37.293\,29x_1 - 40\,792.141$$

约束条件

$$g_1(x) = 85.334\,407 + 0.005\,685\,8x_2x_5 + 0.000\,26x_1x_4 - 0.002\,205\,3x_3x_5$$

$$g_2(x) = 80.512\,49 + 0.007\,131\,7x_2x_3 + 0.002\,995\,5x_1x_2 + 0.002\,181\,3x_3^2$$

$$g_3(x) = 9.300\,961 + 0.004\,702\,6x_3x_5 + 0.001\,254\,7x_1x_3 + 0.001\,908\,5x_3x_4$$

$$0 \le g_1(x) \le 92, \quad 90 \le g_2(x) \le 110$$

$$20 \le g_3(x) \le 25, \quad 78 \le x_1 \le 102$$

表 **1**　本文算法与其它方法的比较

设计变量	得到的最好解			
	本文算法	Gen[1]	Homaifar[5]	GRG[6]
x_1	0.822 8	0.808 0	0.810 2	0.822 9
x_2	0.911 2	0.885 44	0.902 6	0.911 5
g_1	4.00×10^{-4}	3.7×10^{-2}	5.00×10^{-3}	1.00×10^{-4}
g_2	- 0.043	0.052	0.025	-5.15×10^{-5}
$f(x)$	1.393 7	1.433 9	1.425 1	1.393 4

表 2 本文算法与其它方法的比较

设计变量	得到的最好解			
	本文算法	Gen[1]	Homaifar[5]	GRG[6]
x_1	78.008	81.490	78.000	78.620
x_2	33.011	34.090	33.000	33.440
x_3	27.109	31.240	29.995	31.070
x_4	45.000	42.200	45.000	44.180
x_5	44.822	34.370	36.776	35.220
g_1	91.980 9	90.523	90.714 6	90.520 7
g_2	96.209 6	99.319	98.840 5	98.892 9
g_3	20.000 1	20.060	19.999 9	20.131 6
$f(x)$	− 31 025.141 2	− 30 183.586	− 30 665.609	− 30 373.949

$33 \leq x_2 \leq 45, \quad 27 \leq x_i \leq 45, \quad i = 3, 4, 5$

本文算法运行 10 次所得到的平均最优目标函数为 − 31023.547, 所找到的最差目标函数为: − 31 021.862, 表 2 给出了几个算法在 10 次运行中的最好结果。

4 结 语

本文在乘子法的基础上, 把非线性约束优化问题转化为无约束优化问题, 然后利用合作协同进化算法分别进化 Lagrange 乘子和原问题候选解, 以得到近似最优Lagrange 乘子和近似最优解。仿真结果表明, 与其他算法相比, 该方法具有较好的约束处理能力及可以获得较好的近似全局最优解, 且算法鲁棒性较强。该算法的优点在于: 算法迭代过程中, 乘子和原问题的候选解之间是相互作用的, 从而使乘子自适应变化, 且种群的适应值函数也是动态变化的。

参考文献:

[1] Mitsuo Gen, Runwei Cheng. Genetic algorithms and engineering design[M]. New York: John Wiley & Sons, 1997.

[2] J S Arora, A I Chahande, J K Paeng. Multiplier methods for engineering optimization[J]. Int J for Numerical in Engineering, 1991, 32: 1485–1525.

[3] Richardson J T, M R Palmer, G Liepins *et al*. Some guidelines for genetic algorithms with penalty functions [A]. Proc of the 3rd Int Conf on Genetic Algorithms [C]. Morgan Kaufmann, 1989. 191–197.

[4] M A Potter, K A De Jong. A cooperative co-evolutionary approach to function optization[A]. In Third Parallel Problem Solving from Nature[C]. Jerusalem, 1994. 463-465.

[5] A Homaifar, S H Y Lai, X Qi. Constrained optimization via genetic algorithms[J]. Simulation, 1994, 62 (4): 242–254.

[6] David M Himmelblau. Applied nonlinear programming. New York: McGraw-Hill, 1972.