文章编号: 1001-0920(2004)07-0804-04

一种基于粒子群算法求解约束优化问题的混合算法

李炳宇, 萧蕴诗, 吴启迪
(同济大学 电子与信息工程学院, 上海 200092)

摘 要: 通过将粒子群算法(PSO)与差别进化算法(DE)相结合, 提出一种混合算法PSODE, 用于求解约束优化问题. PSODE是在PSO算法中适当引入不可行解, 将粒子群拉向约束边界, 加强对约束边界的搜索, 同时与DE算法结合以加强搜索能力. 基于典型高维复杂函数的仿真表明, 该算法简单高效, 鲁棒性强.

关键词: 约束优化问题; 粒子群优化算法; 群体智能; 差别进化
中图分类号: TP301.6 **文献标识码**: A

Hybrid algorithm based on particle swarm optimization for solving constrained optimization problems

LI Bing-yu, XIAO Yun-shi, WU Qi-di
(School of Electronic and Information Engineering, Tongji University, Shanghai 200092, China. Correspondent: LI Bing-yu, E-mail: yv.tou@163.com)

Abstract: A hybrid algorithm, PSODE, is proposed by combining particle swarm optimization (PSO) with different evolution (DE), for solving constrained optimization problems. PSODE is a powerful searching algorithm by keeping properly infeasible solution to attracting the swarm to the constraint boundary and by combining with DE to enhance the searching ability. Simulation results on benchmark complex functions with high dimension show that the hybrid algorithm is effective, efficient and fairly robust to initial conditions.

Key words: constrained optimization problem; particle swarm optimization; swarm intelligence; different evolution

1 引 言

在社会和生产实践中, 许多问题可归结为函数的优化问题, 而且是带有约束条件的优化问题. 以往的许多算法是基于梯度信息求解约束优化问题, 只适用于目标函数和约束条件可微的情况. 目前, 通常用于处理约束问题的是惩罚不可行解方法, 即罚函数法和乘子法[1], 它们的更新过程都不需要个别约束的梯度信息, 因此很适于一些工程应用. 但惩罚因子和乘子更新策略的确定限制了其应用, 因为如果惩罚因子选择不当, 将使解或陷入局部最小, 或远离约束条件下的最优解; 而乘子法则容易在鞍点附近发生振荡, 甚至发散不收敛. 不使用乘子法或罚函数法的其他算法的计算精度又不高.

针对上述问题, 本文将粒子群算法(PSO)[2]与差别进化(DE)[3]相结合, 提出一种混合算法PSODE. 粒子群算法(PSO)是Eberhart和Kennedy共同发明的一种新的群体智能计算技术, 源于对鸟群和鱼群群体运动行为的研究. 差别进化(DE)是一种全局并行直接搜索算法, 它通过随机选取空间中两个矢量, 加以不同的权重得到另一个矢

收稿日期: 2003-07-09; **修回日期**: 2003-09-17.
基金项目: 国家自然科学基金资助项目(70271035, 60104004); 国家973子项目资助(2002CB312202); 上海市启明计划(03QG14053).
作者简介: 李炳宇(1977—), 男, 山东成武人, 博士生, 从事群体智能、复杂系统理论与方法等研究; 萧蕴诗(1946—), 男, 湖北武汉人, 教授, 博士生导师, 从事智能自动化理论与应用、复杂系统理论与方法等研究.

量, 再与父代矢量比较, 根据适应值的大小得到子代矢量. 这两种算法的共同特点是, 不要求目标函数和约束可微, 原理简单直观, 容易实现, 搜索速度快且高效实用.

本文将两者相结合, 用于处理高维复杂约束优化问题, 不依赖于具体问题, 可有效避免因单一搜索机制引起的停滞现象, 加强搜索能力, 提高搜索效率. 同时针对约束优化问题中, 一大类问题的最优解是在约束边界附近的情况, 在粒子群算法中适当引入不可行解, 将更多的粒子"拉向"约束边界, 加强对约束边界的搜索, 能以更高的概率获得全局最优解. 最后基于典型高维测试函数的仿真结果验证了该算法的有效性和高效性.

2　处理约束条件的一般方法

目前, 惩罚函数法是处理约束条件的常用方法, 即在目标函数中加上一个惩罚项, 以反映该点是否位于可行域内, 使得算法在惩罚项的作用下找到原问题的最优解. 虽然可以证明当惩罚因子趋于无穷大时, 带有惩罚项的目标函数的解将收敛到原问题的解[4]. 但在实际应用中, 却很难选择适当的惩罚因子. 若惩罚因子选取过大, 将造成计算机计算数值的溢出和较早收敛于局部最小解; 若选取过小, 可能会造成所得解不是原问题的最优解. 对于要解决的约束优化问题往往需要通过多次试验来确定惩罚因子.

乘子法在实际计算时是采取逐步迭代的方式求解原问题的最优解. 但在鞍点附近会出现振荡甚至发散的情况. 虽然文献[5]提出了MaxQ 方法来避免在鞍点的振荡, 但该方法需要对参数Q 进行试验, 而且收敛速度较慢. 文献[6]利用遗传算法使用竞争选择策略, 引入了不需要罚因子而直接比较个体优劣的方法, 但该算法搜索能力不强, 且没有针对边界的处理方法, 所以这种方法的计算效果并不能令人十分满意. 本文针对上述算法存在的局限性提出了PSODE 算法, 可有效克服上述缺点.

3　PSODE 算法简介

3.1　PSO 的基本原理

PSO 算法虽然是基于群体的, 但与传统的遗传算法不同, 它通过个体之间的协作来寻找最优解. 最初, Eberhart 和Kennedy 是为了建立模型以模拟鸟群在二维空间优美而不可预测的运动; 后来从该模型中得到启示, 并将其用于解决优化问题; 又将其推广到N 维空间, 每个优化问题的解就是搜索空间中的一只鸟, 鸟被抽象为没有质量和体积的微粒(点), 第 i 个粒子在N 维空间里的位置表示为矢量$X_i = (x_1, x_2, \cdots, x_N)$, 飞行速度表示为矢量$V_i = (v_1, v_2, \cdots, v_N)$. 每个粒子都有一个由被优化函数决定的适应值, 并且知道自己到目前为止发现的最好位置(pbest) 和现在的位置X_i, 可将它看作是粒子自己的飞行经验. 除此之外, 每个粒子还知道目前为止整个群体中所有粒子发现的最好位置(gbest, 即在pbest 中的最好值), 可将其看作是粒子同伴的经验. PSO 是一种基于迭代的优化工具. 对于第k 次迭代, 粒子按下式进行变化[2]:

$$v_{id}^{k+1} = w \times v_{id}^{k} + c_1 \times \mathrm{rand}() \times (p_{id} - x_{id}^{k}) + c_2 \times \mathrm{rand}() \times (p_{gd} - x_{id}^{k}), \tag{1}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}. \tag{2}$$

式中: $i = 1, 2, \cdots, M$, M 为该群体中粒子的总数; v_{id}^{k} 为第k 次迭代粒子i 飞行速度矢量的第d 维分量; x_{id}^{k} 为第k 次迭代粒子i 位置矢量的第d 维分量; p_{id} 为粒子i 个体最好位置pbest 的第d 维分量; p_{gd} 为群体最好位置 gbest 的第 d 维分量; c_1 和 c_2 为权重因子; rand() 为随机函数, 产生[0, 1] 的随机数; w 为惯性权重函数.

式(1) 主要通过 3 部分来计算粒子 i 的新速度: 粒子 i 前一时刻的速度; 粒子 i 当前位置与自己最好位置之间的距离; 粒子 i 当前位置与群体最好位置之间的距离; 粒子 i 通过式(2) 计算新位置的坐标. 通过式(1) 和(2) 粒子i决定下一步的运动位置. 如图 1 所示, 以两维空间为例描述了粒子根据式(1) 和(2) 从位置X^k 到X^{k+1} 移动的原理.

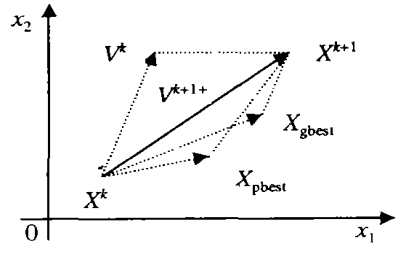

图 1　粒子移动原理图

3.2　DE 算法的基本原理

差别进化(DE) 是在N 维空间中选取NP 个N 维矢量作为一个群体, 在整个运算过程中群体规模不变. 差别进化也有类似遗传算法的变异、交叉和选择等操作. 其中变异操作的定义如下[3]:

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}). \tag{3}$$

其中: $r_1, r_2, r_3 \in \{1, 2, \cdots, NP\}$; F 是位于[0, 2] 的

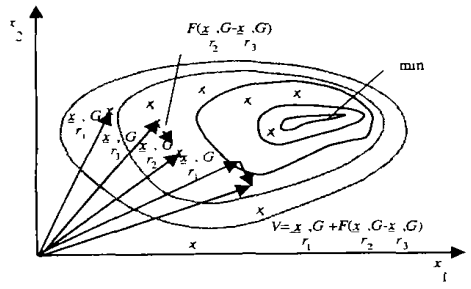

图 2 变异原理图

一个常数. 式(3) 表示从群体中随机取出两个矢量 $x_{r_2,G}$ 和 $x_{r_3,G}$, 将两个矢量相减, 两者的差 $(x_{r_2,G} - x_{r_3,G})$ 经 F 放大后加到第 3 个矢量 $x_{r_1,G}$ 上, 得到下一代新的矢量 $v_{i,G+1}$. 其两维空间的变异原理如图 2[3] 所示

差别进化的交叉操作定义如下[3]:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+2}, \cdots, u_{Di,G+1}),$$

其中各分量由下式决定:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, \text{randb}(j) \leq CR \text{ 或} \\ \quad j = \text{rnbr}(i); \\ x_{ji,G}, \text{randb}(j) > CR \text{ 且} \\ \quad j \neq \text{rnbr}(i); \end{cases} \quad (4)$$

$$j = 1, 2, \cdots, N.$$

式中: randb(j) 为位于 $[0, 1]$ 的随机数, CR 为位于 $[0, 1]$ 的交叉常数, rnbr(i) 为位于 $[1, N]$ 的随机整数, 用以确保 $u_{i,G+1}$ 中至少有一个分量来自于 $v_{ji,G+1}$. 在差别进化中, 选择操作采取的是贪婪策略, 即只有当产生的子代个体 $u_{i,G+1}$ 优于父代个体 $x_{i,G}$ 时才被保留, 令 $x_{i,G+1} = u_{i,G+1}$, 否则 $x_{i,G}$ 仍被保留

3.3 PSODE 算法

在处理高维复杂问题时, PSO 不可避免地会以一定概率停滞在一些局部最小点. 如何避免过早停滞和加强搜索能力是解决好复杂问题的关键. 考虑到DE 算法和PSO 算法的编码方式相似, 易结合、易实现, 且两者的搜索能力都很强, 本质上都是并行搜索算法, 所以将两者结合为一个高效的混合算法. 兼顾到算法效率, 在本文算法中, 每次只从粒子群中随机选取10% 的个体用于DE 搜索. 这样, 在群体向最优点 gbest 聚集的过程中, 有些粒子将偏离其运动轨迹. 对其轨迹的邻域进行搜索, 不但有效避免了粒子过早停滞, 而且加强了搜索范围

现实中一大类约束优化问题的最优解在约束边界附近. 对于这类问题, 最优解附近不可行解的适应值很可能优于位于可行域内部的可行解的适应值. 鉴于 PSO 算法的群体搜索策略, 引入一部分接近边界的不可行解, 通过比较它们的适应值, 保留一定比例有较好适应值的不可行解, 因为这样可吸引群体向约束边界移动, 有助于群体发现更好的解.

定义函数

$$v(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\}, 1 \leq i \leq m,$$

表示粒子与边界的接近程度. 在此, 将 l 个等式约束代入目标函数, 同时消减 l 个变量, 降低了问题的复杂度, 因此只需考虑不等式约束.

在群体搜索过程中采用死亡罚函数的方法, 直接对不可行解赋予较差的适应值. 但每隔G 代, 根据函数 $v(x)$ 从群体不可行解中选取 q 个粒子, 在随后的G 代计算中仅保持 q 个不可行解的粒子的空间位置不变, 如果出现其他不可行解可直接对其赋予较差的适应值. 然后在随后的 G 代计算中仍采取死亡罚函数法去除这 q 个粒子, 并在其后 G 代保留一部分不可行解. 上述过程如图 3 所示

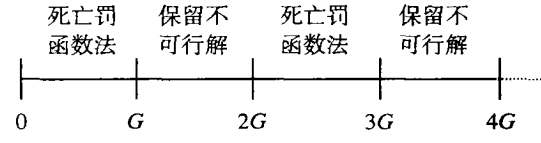

图 3 PSODE 算法的迭代过程

PSODE 算法流程如下:

Step 1: 初始化参数;

Step 2: 不满足停止条件, 则执行 Step 3 ~ Step 6; 否则, 转 Step 7;

Step 3: iter + + ;

Step 4: 判断是否应该保留不可行解, 如果是, 转到 Step 6; 否则, 转 Step 5;

Step 5: 采用死亡罚函数法搜索;

Step 6: 计算 $v(x)$, 保留不可行解;

Step 7: 输出最优结果

算法中 Step 5 的目的是"逼迫"粒子群离开不可行域, 只在可行域内搜索; Step 6 是通过不可行域的点"吸引"可行域内的部分粒子穿过约束边界向不可行域飞行. 通过 Step 5 和 Step 6 的不断交替执行, 加强了对不同约束边界附近区域的搜索

4 数值实验

在此用 Keane's Bump 优化问题[7] 作为测试函数, 该问题定义如下:

$$\max f_n(x) = \left| \sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i) \right| /$$

$$\max \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|,$$

$$\text{s.t.} \begin{cases} \prod_{i=1}^{n} x_i \geq 0.75, \\ \sum_{i=1}^{n} x_i \leq 7.5n, \\ 0 \leq x_i \leq 10,\ 1 \leq i \leq n. \end{cases}$$

由于Bump 问题具有的"三超"特性(超非线性、超多峰、超高维), 已成为国际上通用的衡量优化算法的测试函数[8]. 图 4 显示了两维Bump 问题的曲面, 其中不符合约束要求的点赋值为 0. 由此可以看出, Bump 函数是一个多峰的非线性函数, 随着维数的增加该函数极难优化, 而且约束条件 $\prod_{i=1}^{n} x_i \geq 0.75$ 中的 $\prod_{i=1}^{n} x_i$ 容易溢出. 本文以维数 $n = 20$ 的 Bump 函数作为测试函数. PSODE 算法中参数设计如下: $M = 600, w_{max} = 0.9, w_{min} = 0.4$, $iter_{max} = 3\,000$, $c_1 = c_2 = 2$, $F = 0.8$, $q = 6$, $CR = 0.3$, $G = 10$, 独立运行 10 次, 选取最优解, 计算平均最优解和均方差.

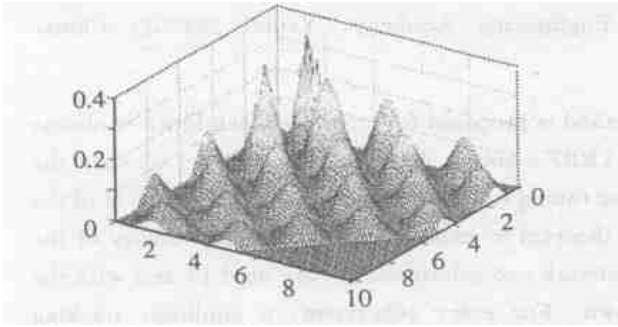

图 4　两维 Bump 问题曲面

本文算法与其他算法的比较见表 1. 本文提出的 PSODE 算法在点 $X = (3.162\,547, 3.128\,140, 3.094\,457, 3.061\,642, 3.028\,128, 2.993\,436, 2.958\,732, 2.922\,748, 0.495\,475, 0.488\,337, 0.481\,788, 0.476\,854, 0.470\,682, 0.466\,036, 0.460\,866, 0.456\,295, 0.453\,248, 0.448\,711, 0.444\,562, 0.440\,326)$ 取得最好结果 0.803 619. 从表 1 可以看出, PSODE 算法在最优解和收敛性上都优于其他算法. 其中 GC 算子法取得的最优结果较好, 但该算法是特别为这个问题设计的几何交叉算子法, 所以它不适合于推广到其他问题.

同时值得注意是, 本文算法不是为某一特定问题设计的, 它具有很强的通用性.

表 1　本文算法与其他算法的比较

比较项	算法				
	本文算法	GC 算子法[9]	FPDC-GA[10]	PGA[11]	DC-G[12]
最优解	0.803 619	0.803 553	0.802 747	0.76	0.713 9
平均最优解	0.803 604	*N/A*	0.796 5	*N/A*	0.669 4
优化值均方差	0.000 2	*N/A*	0.005 6	*N/A*	0.025 1

5　结　　语

对于带有约束的高维复杂优化问题, 关键是如何克服过早收敛于局部最小和处理约束边界条件. 本文针对以上两点, 将PSO 算法与DE 算法相结合, 有效地避免了算法因单一搜索机制引起的停滞现象. 通过引入不可行解导引群体加强对约束边界的搜索, 避免了算法在边界出现振荡甚至发散的情况. 最后数值仿真证明了该算法的有效性. PSODE 算法编码简单, 容易实现, 不要求目标函数和约束条件可微, 不依赖于具体问题, 通用性强, 为求解约束优化问题提供了一条可行的方法.

参考文献(References):

[1] 张春慨, 邵惠鹤. 自适应乘子在工程优化问题中的应用[J]. 控制与决策, 2001, 16(6): 669-672.
(Zhang C K, Shao H H. Application of self-adaptive multiplier in engineering optimization problem [J]. *Control and Decision*, 2001, 16(6): 669-672.)

[2] Kennedy J, Eberhart R. Particle swarm optimization [A]. *Proc IEEE Int Conf on Neural Networks* [C]. Perth, 1995. 1942-1948.

[3] Rainer Storn, Kenneth Price. Different evolution——A simple and efficient heuristic for global optimization over continuous space [J]. *J of Global Optimization*, 1997, (11): 341-359.

[4] 秦寿康. 最优化理论与方法[M]. 北京: 电子工业出版社, 1986.

[5] Wang Tao. *Global Optimization for Constrained Nonlinear Programming* [M]. Doctoral Dissertation: University of Illinois at Urbana-Champaign, 2001.

[6] Deb K, Agrawal S. A niched-penalty approach for constraint handling in genetic algorithms [A]. *Proc of the ICANNGA-99* [C]. Portoroz, 1999. 234-239.

[7] Keane A J. Experiences with optimizers in structural design [A]. *Proc of the Conf on Adaptive Computing in Engineering Design and Control* 94 [C]. Plymouth, 1994. 14-27.

(下转第 812 页)

般 η 的取值范围为 0.01～1.

4 结 语

本文方法放宽了文献[8,9]中单输入单输出系统的要求,同时利用已知信息,解决了一类非线性多变量系统控制系数矩阵未知情况的控制器设计问题,避免了可能出现的控制器奇异问题.并首次应用 Lyapunov 稳定性理论推导出全调节RBF 神经网络各参数的自适应调节律,保证了整个闭环系统的稳定性.同时,通过引入非线性微分跟踪器和神经网络抑制了Backstepping 设计存在的"计算膨胀"问题.

参考文献(References):

[1] Sastry S S, Isidori A. Adaptive control of linearizable system [J]. *IEEE Trans on Automatic Control*, 1989, 34(11): 1123-1131.

[2] Seto D, Annaswamy A M, Baillieul J. Adaptive control of nonlinear systems with a triangular structure [J]. *IEEE Trans on Automatic Control*, 1994, 39(7): 1411-1428.

[3] Polycarpou M M. Stable adaptive neural control scheme for nonlinear systems [J]. *IEEE Trans on Automatic Control*, 1996, 41(3): 447-451.

[4] Krstic M, KanEllakopoulos I, Kokotovic P. *Nonlinear and Adptive Control Design* [M]. New York: Wiley-Interscience Publication, 1995.

[5] Vadim IUtkin, De-Shiou Chen, Hao-Chi Chang. Block control principle for mechanical systems [J]. *J of Dynamic Systems, Measurement, and Control*, 2000, 122(1): 1-10.

[6] Loukianov A, Castillo-Toledo B, Dodds S J. Nonlinear sliding surface design in the presence of uncertainty [A]. *Proc of the 14th IFAC* [C]. Beijing, 1999. 55-60.

[7] Li Y, Sundararajan N, Saratchandran P. Neuro-controller design for nonlinear fighter aircraft maneuver using fully tuned RBF networks [J]. *Automatica*, 2001, 37(8): 1293-1301.

[8] Zhang T, Ge S S, Hang C C. Adaptive neural network control for strict-feedback nonlinear systems using backstepping design [J]. *Automatica*, 2000, 36(12): 1835-1846.

[9] Ge S S, Wang C. Adaptive NN control of uncertain nonlinear pure-feedback systems [J]. *Automatica*, 2002, 38(4): 671-682.

[10] 晋玉强. 导弹非线性自适应控制系统设计[D]. 烟台: 海军航空工程学院, 2003.

[11] 韩京清, 王伟. 非线性跟踪-微分器[J]. 系统科学与数学, 1994, 14(2): 177-183.
(Han J Q, Wang W. Nonlinear tracking differentiator [J]. *J of System Science and Mathematic Sciences*, 1994, 14(2): 177-183.)

[12] Raul Ordonez, Jeffrey T Spooner. Stable multi-input multi-output adaptive fuzzy control [A]. *Proc of the 35th CDC* [C]. Japan, 1996. 610-615.

[13] Khalil H K. Adaptive output feedback control of nonlinear systems represented by input-output models [J]. *IEEE Trans on Automatic Control*, 1996, 41(2): 177-188.

(上接第807 页)

[8] Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems [J]. *Evolutionary Computation*, 1996, 4(1): 1-32.

[9] Michalewicz Z, Eaguvel S, et al. The spirit of evolutionary algorithms [J]. *J of Computing and Information Technology*, 1999, 7(1): 1-18.

[10] 林丹, 李敏强, 寇纪松, 等. 基于遗传算法求解约束优化问题的一种算法[J]. 软件学报, 2001, 12(4): 628-632.
(Lin D, Li M Q, Kou J S. A GA-based method for solving constrained optimization problems [J]. *J of Software*, 2001, 12(4): 628-632.)

[11] Schoenauer M, Michalewicz Z. Boundary operators for constrained optimization problems [A]. *Proc of the 7th Int Conf on Genetic Algorithms* [C]. CA: Morgan Kaufman Publishers, 1997. 322-329.