

基因表达式中含有等式约束的处理方法

何家莉, 王 培

(玉林师范学院 数学与计算机科学系, 广西 玉林 537000)

摘 要:基因表达式编程(Gene Expression Programming, GEP)算法是遗传家族的新成员,被广泛用于函数发现。在微分方程中,要寻找的函数需要满足初始值,即有时希望 GEP 找到的函数能够满足一些等式约束条件。提出了一种带拉格朗日插值函数的 GEP,对生成的种群加入插值函数使其满足等式约束,为提高 GEP 算法的进化效率和精度对目标值加入尺度变化,对其放大或者缩小。这样缩短了 GEP 算法的进化距离,从而提高了种群的进化效率。通过仿真实例,结果表明该方法可行有效。

关键词:基因表达式编程; 等式约束; 尺度

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2011)09-0092-03

Method of Gene Expression Programming for Solving Equality Constraint

HE Jia-li, WANG Pei

(Department of Mathematics and Computer Science, Yulin Normal University, Yulin 537000, China)

Abstract:Gene Expression Programming (GEP) is a new genetic algorithm for function discovery. In differential equation, sometimes hope to find the function that meet equality constraints. In order to reach this goal, algorithm with Lagrange interpolation function is proposed which makes the function fulfill the requirement of constraints. Size scale is proposed to amplify or lessen the target value in order to improve the evolutionary efficiency and precision of gene expression programming algorithm. As a result, the evolution distance of GEP algorithm shortens and the evolutionary efficiency increases. Finally, numerical experiments are performed and the results show that the method is feasible.

Key words:GEP; equality constraint; size scale

0 引言

2000年 Ferreira先生提出了一种新的遗传算法——基因表达式编程(Gene Expression Programming, GEP)^[1~4]。GEP作为一种基于基因型组和表现型组的新的遗传算法,它结合了遗传算法和遗传规划的优点。GEP的个体是由多个长度固定不变的基因组成的线性字符串,然后这些个体被表示成表达式树(Expression Trees, ET),通过遗传算子进行操作从而达到进化到较优结果的目的。GEP算法在许多领域都有很广泛的用途,特别是在函数挖掘方面^[5~7],新的研究成果不断提出。

经常在高等数学中见到微分方程。例如有这样的

微分方程 $\begin{cases} y' = f(x) \\ y(x_0) = y_0 \end{cases}$, 就是要找这样一个函数,使其

满足初始条件 $y(x_0) = y_0$ 。GEP在寻找函数方面有较强的优势,找出的函数能很大程度地逼近精确的函数,但是如果用传统的GEP找上面的函数,就很难让找的函数在 x_0 处等于 y_0 。现在的问题就是,如果情况是想找的某些函数在不影响精度的情况下,希望在某一些点上与精确值相等,那么就需要有一种新的方法来解决这类有约束^[8,9]的问题。笔者对传统的GEP进行了改进,对算法加入了拉格朗日插值多项式,使其在一定精度下一些点与精确值相等。

1 基本的基因表达式编程算法

GEP通过染色体Chromosome承载遗传信息作为基因型实体来参与遗传操作,生成的表达式树作为信息的表现型来表达遗传实体中的信息编码。经典基因表达式程序设计实现的基本算法:

1) 随机生成初始群体。

收稿日期:2011-02-25;修回日期:2011-06-07

基金项目:玉林师范学院青年项目(2009YJQN100);国家自然科学基金(11061004)

作者简介:何家莉(1981-),男,广西桂林人,硕士研究生,讲师,主要研究方向为智能计算。

2) 选择恰当的适应度函数进行评判, 根据适应度函数选择部分种群。

3) 执行遗传操作算子, 包括: 变异, 插串, 根插串, 单点重组, 两点重组, 基因重组。

4) 根据停机条件, 如果运行到事先给定的最大代数或者达到某一精度, 则停止运行, 否则转向 2)。

2 GEP 的改进算法

对于要寻找的函数, $y = f(x^0, x^1, \dots, x^s)$, 约束条件

$$\text{是} \begin{cases} y(x_0^0, x_0^1, \dots, x_0^s) = y_0 \\ y(x_1^0, x_1^1, \dots, x_1^s) = y_1 \\ \dots \\ y(x_t^0, x_t^1, \dots, x_t^s) = y_t \end{cases}$$

其中 s 表示有 s 个自变量, t 表示样本点个数。笔者希望在那些点上等式成立。

如果函数仅是一维的 $y = f(x)$, 则可以采用拉格朗日插值多项式^[10,11]:

$$L_n = y_0 l_0(x) + y_1 l_1(x) + \dots + y_n l_n(x),$$

其中 $l_i(x) =$

$$\frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1})(x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1})(x_i - x_n)}.$$

而对于高维的函数, 则可以延展拉格朗日函数:

$$L_n(x^0, x^1, \dots, x^s) = y_0 l_0(x^0, x^1, \dots, x^s) + y_1 l_1(x^0, x^1, \dots, x^s) + \dots + y_n l_n(x^0, x^1, \dots, x^s)$$

其中 $l_i(x^0, x^1, \dots, x^s) = l_i(x^0) \cdot l_i(x^1) \cdot \dots \cdot l_i(x^s)$ 。

设用 GEP 随机生成的函数为 $g(x^0, x^1, \dots, x^s)$, 则发现的函数就为 $g'(x^0, x^1, \dots, x^s) = g(x^0, x^1, \dots, x^s) \cdot (x_0 - y_0) \cdot (x_1 - y_1) \cdot \dots \cdot (x_t - y_t) + L_n$

这样就能保证在要求的点上等式成立了。

由于 GEP 算法比较容易进入局部最优解, 为了能达到较好的结果, 现对基本的算法进行改进。适应度

函数选用 $f_j = \sum_{j=1}^n (M - |C_{i,j} - T_j|)$ 。 M 为一常量, 用于控制适应度的 f_j 的取值范围, $C_{i,j}$ 表示第 i 个基因对应的函数表达式中利用第 j 个样本变量数据求得的函数值; T_j 表示第 j 个样本的目标值。加入了约束条件的 GEP, 在寻找一维函数时还能够达到满意解。一旦到了多维的函数时, 因为对 GEP 生成的函数进行的变形比较大, 所以精度就受到了影响。为了能更好地达到精度, 现对目标函数值 f 乘以一个常数 C , 即 $|g'(x^0, x^1, \dots, x^s) - f(x^0, x^1, \dots, x^s)| \times C = e$ (e 表示生成的函数与目标函数的误差)。然后从绝对值里提出一个常数 C 。

$$C \times \left| \frac{g'(x^0, x^1, \dots, x^s)}{C} - f(x^0, x^1, \dots, x^s) \right| = e$$

$$\left| \frac{g'(x^0, x^1, \dots, x^s)}{C} - f(x^0, x^1, \dots, x^s) \right| = \frac{e}{C}$$

这时候误差就比原来的缩小了。但是在实际操作过程中, 不是 C 取得越大越好。因为当 C 取很大时, GEP 的寻优能力也发生了改变。这样做的目的是对原函数值经行尺度放大或者缩小, 从而能达到精度。试验证明找到一个合适的常数 C 比不进行任何改变的目标函数值更能逼近精确解。

3 改进的具体算法

下面给出算法:

1) 确定各个算法的各个参数: GEP 算法的变异概率, 变换概率, 倒置概率, 重组概率。

2) 随机生成 N 组种群, 每一组种群就是一个函数。确定一个目标函数的倍数 C , 根据具体的目标函数值确定范围。文中例 2 的 C 从 1 取, 步长为 0.1, 当陷入局部最小值时, 重新更换种群并调整 C 。

3) 进行适应度评价。文中采用的是绝对误差的方法。评价完之后采用最优选择方法, 即每次都选择最优的种群, 其它的种群按转轮盘选择。

4) 进行 gep 算法的变异、倒置、重插、重组等操作。在交叉算子概率的选择时, 先给出一个概率范围, 迭代的阶段分别根据不同的时段选择具体的值^[12]。每经过数代就进行全局大变异一次, 大变异的概率为 1。

5) 经过数次迭代后, 当抗体的适应度无法达到预期目标时, 也就是说陷入了局部最优解时, 保留最优解, 删掉其他种群, 同时随机生成一部分种群填补到删除的种群。

6) 这样逐步循环下去, 直到找到满意解为止。

4 仿真实例

例 1 一元函数发现问题^[12] $G(x) = x^3 + x^2 + x + 1$, 约束条件是 $y(1) = 1; y(2) = 4$, 取从 0 到 10, 步长为 1 的 11 个点。GEP 生成的函数用 $f(x)$ 表示, 生成的函数形式为 $g(x) = f(x) \cdot (x - 1) \cdot (x - 2) + 4 \times \frac{(x - 4)}{(1 - 2)} + 15 \times \frac{(x - 1)}{(2 - 1)}$ 。使用文中的算法, 实验参数如表 1 所示。

最后用 Matlab 算出的结果是:

$$g(x) = (((\cos(\sin(-4.096))) + x) + (\cos(5.0277))) + (-5.9834 + 8.9881)) * (x - 1) * (x - 2) + (4 * (x - 2) / (1 - 2) + 15 * (x - 1) / (2 - 1))$$

其绝对误差为: $\sum_{i=1}^{11} e_i = \sum_{i=1}^{11} |G(x_i) - g(x_i)| = 0.0064$, 达到满意的精度。在例 1 中, 因为函数不复杂, 所以不需要调整尺度 C 就可以达到满意解了。但

随着函数的维数增多,约束增多,复杂度增加,就需调整参数 C 了。下面给出例 2 说明问题。

表 1 F_1 试验参数

参数名	参数值
种群大小	30
函数符集 F	{+, -, *, √}
终结符集 T	{ x }
头部长度 H	6
进化代数 N	400
基因个数	3
连接符	+
最小交叉概率	0.02
最大交叉概率	0.05
最小变异概率	0.03
最大变异概率	1
C 的取值	1

例 2 五元函数发现问题^[12] $F_2(a, b, c, d, e) = \frac{\sin a \cos b}{\sqrt{\exp^c}} + \tan(d - e)$, 约束条件是: $F_2(0.3, 0.3, 0.3, 0.3, 0.3) = 0.2430, F_2(0.8, 0.8, 0.8, 0.8, 0.8) = 0.3350$ 。取从 0 到 1, 步长为 0.1 的 11 个点。GEP 生成的函数用 $f(x)$ 表示, 生成的函数形式为 $g(x) = f(x) \cdot [(a - 0.3) + (b - 0.3) + (c - 0.3) + (d - 0.3) + (e - 0.3)] \cdot [(a - 0.8) + (b - 0.8) + (c - 0.8) + (d - 0.8) + (e - 0.8)] + 0.2430 \cdot [(a - 0.8)(b - 0.8)(c - 0.8)(d - 0.8)(e - 0.8)](0.3 - 0.8)^{-5} + 0.3350 \cdot [(a - 0.2)(b - 0.2)(c - 0.2)(d - 0.2)(e - 0.2)](0.8 - 0.3)^{-5}$ 使用文中的算法求解, 试验参数如表 2(其余与例 1 参数一致)所示。

表 2 F_2 试验参数

参数名	参数值
函数符号集 F	{+, -, *, √, sin, cos, tan, sqrt}
终结符集 T	{ a, b, c, d, e }
进化代数 N	3500
C 的取值	2.5

最后用 Matlab 计算的结果是:

$$2.5 \times g(x) = (((\cos((a + 0.042324) + 8.6456) - (\sqrt{b}))) + (\sin(\cos(4.0575 + 0.071443)))) + (\sin(\cos((\sqrt{d}) + 0.075596))) * ((a - 0.3) + (b - 0.3) + (c - 0.3) + (d - 0.3) + (e - 0.3)) * ((a - 0.8) + (b - 0.8) + (c - 0.8) + (d - 0.8) + (e - 0.8)) + 0.2430 * ((a - 0.8) * (b - 0.8) * (c - 0.8) * (d - 0.8) * (e - 0.8)) / (0.3 - 0.8)^5 + 0.3350 * ((a - 0.2) * (b - 0.2) * (c - 0.2) * (d - 0.2) * (e - 0.2)) / (0.8 - 0.3)^5$$

其中 $g(x)$ 是所要找的函数, $C = 2.5$, sqrt 表示开平方, ^ 表示几次方, * 表示乘号, / 表示除号。其绝对

误差 $\sum_{i=1}^{11} e_i = 0.4750$ 。而如果不改变目标函数值尺度, 或者取其它数值, 得到的结果没有这么好。试验每取一个 C 值, 就进行 20 次试验, 取其中最好的结果。每次都迭代 4000 次。具体见表 3。

表 3 C 值的变化与最后算法结果的精度关系

C 的变化	1	2	2.2	2.5	3	3.2	3.5	10
绝对误差值	0.9847	0.7847	0.5811	0.4750	0.6450	1.0235	1.4235	1.7423

从表 3 可以看出, 目标值的尺度大小在求解最优值时是有一定效果的。当没有使用尺度值时, 试验多次的最好结果是 0.9847, 而找到了合适的 C 值时, 就能比之前的精度更进一步。但不同的问题取的尺度有些差别。如果问题的数值较大则, 用尺度 C 来缩小, 可以取 (0, 1) 之间的数。如果数值小则尺度可以适当放大些。

5 结束语

传统的 GEP 算法在挖掘函数时, 没有约束条件。但当需要微分方程之类的问题, 需要找的函数是有约束条件时, 传统的 GEP 就不好计算了。通过在 GEP 算法中加入拉格朗日插值函数, 能使得找到的函数达到等式约束条件。但由于有了约束条件, 寻找的函数在精度上又很难达到满意解。于是对目标函数值经行尺度放大或者缩小, 试验证明, 找到合适的尺度能够使得解更加精确。

参考文献:

- [1] Ferreira C. Gene Expression Programming in Problem Solving [M]//Soft Computing and Industry Recent Applications. [s. l.]:Springer-Verlag, 2002:635-654.
- [2] Ferreira C. Gene Expression Programming Mathematical Modeling by an Artificial Intelligence [M]. New York: Springer Verlag, 2002.
- [3] Zuo Jie, Tang Changjie, Zhang Tianqing. Mining predicate association rule by gene expression programming [C]//Proceedings of International Workshops for Web Information Age. Berlin: Springer-Verlag, 2002: 92-103.
- [4] 唐勇波, 丰娟. 基因表达式编程算法的应用[J]. 计算机应用与软件, 2009, 26(12): 259-261.
- [5] 柳秋云, 王翰虎. 基于基因表达式编程的核 k 近邻分类算法[J]. 计算机技术与发展, 2009, 19(8): 19-22.
- [6] 涂艳琼. 基因表达式编程在函数挖掘中的应用研究[J]. 江西理工大学学报, 2008, 29(3): 66-68.
- [7] 陆昕为, 蔡之华. 一种改进的 GEP 方法及其在演化建模预测中的应用[J]. 计算机应用, 2005, 25(12): 2783-2786.
- [8] 王勇, 蔡自兴. 约束优化进化算法[J]. 软件学报, 2009, 20(1): 11-26.

3.2 时间复杂性和空间复杂性分析

3.2.1 时间复杂性分析

由于 XML 片段形态和查询都较大地影响算法的时间复杂性,因此本研究仅分析 XTOP-K 算法在最坏的情况下时间复杂性。

设树的深度是 h , 数据流中接收到的叶节点个数是 N_1 , 非叶节点个数是 N_2 。

显然有处理 $\text{Begin}(x)$ 事件的时间复杂性是 $O(l)$ 。当处理 $\text{Text}(x)$ 事件时,判断当前节点的值是否包含关键字结合中某个关键字所需要的时间复杂性假设是 f , 做删除算法的时间复杂性显然是 $O(l)$, 该关键字集合大小是 N , 故 Text 事件的时间复杂性是 $O(N * \max\{f, O(l)\})$ 。处理 $\text{End}(x)$ 事件的时间复杂性: 将 XLCA 节点保存到数组的时间复杂性是 $O(l)$, 删除以 XLCA 为根节点的相关连通子树 RCT 在最坏的情况下的时间复杂度是 $O(N * h)$ 。

此算法的时间复杂度是 $\max(N_1 * (N * \max\{f, O(l)\}), N_2 * O(N * h))$ 。

3.2.2 空间复杂性分析

将 XML 数据流保存在滑动窗口模型里面,所需的存储空间是 $O(N_1 + N_2)$ 。所需栈的存储空间是 $O(N_1)$ 。保存查询结果的数组 Q 所需要的存储空间是 $O(\lambda * N)$ (λ 表示查询返回的个数)。此算法的空间复杂度为 $\max(O(N_1 + N_2), O(\lambda * N))$ 。

以上从理论上对 XTOP-K 算法的正确性、完备性、时间复杂性及空间复杂性进行了分析,证明了算法的正确性和完备性,且有较好的时间复杂度。

4 结束语

在 XML 流数据库中, TOP-K 查询是一类重要查询。本研究在 Dewey 编码的基础上,提出用滑动窗口模型保存 XML 数据流信息;且针对 XML 数据流关键字查询中 SLCA 丢失一部分有意义查询结果,改进了 XLCA 松弛语义查询结果,并提出了基于 XLCA 的 XML 数据流上的 TOP-K 关键字查询算法。从理论上证明了此算法的正确性和查询的完整性,分析了该算法有较好的时间复杂性。本研究是在尽量不丢失 XML 数据流信息的前提下进行尽可能全的查询,故对如何利用最少的存储空间去存储 XML 数据流数据没有太多的考虑。进一步的工作包括 XML 数据流查询

中滑动窗口的大小等问题的讨论。

参考文献:

- [1] 周爱武,李孙长,程博,等. XML 数据库的研究与应用[J]. 计算机技术与发展,2009,19(9):218-221.
- [2] Min J K, Park M J, Chung C W, et al. XTREAM: An efficient multi-query evaluation on streaming XML data[J]. Information Sciences, 2007, 177: 3159-3538.
- [3] Bose S, Fegaras L, Leine D, et al. A Query Algebra for Fragmented XML Stream Data[C]//In Proceedings of the 9th International Conference on Data Base Programming Languages (DBPL). Postdam, Germany: [s. n.], 2003: 195-215.
- [4] Weim M Z, Rundensteiner E A, Mani M, et al. Processing recursive XQuery over XML streams: The Raindrop approach[J]. Data & Knowledge Engineering, 2008, 65: 243-265.
- [5] 李波,杨卫东. XML 数据流上的关键字查询算法[J]. 计算机工程, 2009, 35(4): 35-37.
- [6] 王春华. 2008 年中国计算机科学技术发展报告[R]. 北京: 机械工业出版社, 2009: 61-64.
- [7] 孟小峰,周龙骧,王珊. 数据库技术发展趋势[J]. 软件学报, 2004, 15(12): 1822-1836.
- [8] 王小峰,孟小峰,周军峰,等. XML 数据流上的关键字查询[J]. 计算机研究与发展, 2006, 43(S): 484-489.
- [9] 黎玲利,李建中,骆吉洲,等. XML 数据流上的 TOP-K 关键字查询处理[J]. 计算机科学, 2008, 35(10): 159-164.
- [10] Xu Y, Papakonstantinou Y. Efficient Keyword Search for Smallest LCAs in XML Databases[C]//In Proceedings of SIGMOD. [s. l.]: [s. n.], 2005: 527-538.
- [11] Sun C H, Chan C Y, Goenika A K. Multi-way SLCA-based Keyword Search in XML Data[C]//In Proceedings of World Wide Web. [s. l.]: [s. n.], 2007: 1043-1052.
- [12] 谢涛,王晓玲,周傲英,等. XML 关键字检索的最低公共祖先快速查找方法[J]. 计算机研究与发展, 2006, 43(S): 477-483.
- [13] Luo Y, Lin X M, Wei W, et al. SPARK: Top-k Keyword Query in Relational Databases[C]//In Proceedings of SIGMOD. [s. l.]: [s. n.], 2007: 115-126.
- [14] Jin C Q, Yi K, Chen L, et al. Sliding-Window Top-k Queries on Uncertain Streams[J]. VLDB Journal, 2010, 19(3): 411-435.
- [15] 朱皓,杨卫东,施伯乐,等. XML 关键字搜索中一个高效的寻找 XLCA 的算法[J]. 计算机研究与发展, 2008, 45(S): 383-389.

(上接第 94 页)

- [9] 盛跃宾,陈定昌. 有等式约束优化问题的粒子群优化算法[J]. 计算机工程与设计, 2006, 27(13): 2412-2418.
- [10] 贺俐,陈桂兴. 计算方法[M]. 武汉: 武汉大学出版社, 1998.

- [11] 李庆阳,王能超. 数值分析[M]. 北京: 清华大学出版社, 施普林格出版社, 2004.
- [12] 李太勇,唐常杰. 基因表达式编程种群多样性自适应调控算法[J]. 电子科技大学学报, 2010, 39(2): 279-283.