第 34 卷 第 23 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.23 Computer Engineering December 2008
·人工智能及识别技术· 文章编号：1000—3428(2008)23—0210—04 文献标识码：A 中图分类号：TP311.5

基因表达式编程的 2 种解码方法

谢大同[1]，陈巧云[2]
(1. 仰恩大学计算机与信息学院，泉州 362014；2. 仰恩大学财务部，泉州 362014)

摘 要：在基因表达式编程的基础上提出 2 种新的解码方法，分析了它们的时间和空间复杂度。第 1 种方法完全遵照原始基因表达式编程中基因型与表现型之间的映射关系，直接在基因型上计算和求解表达式。第 2 种方法在基因结构保持不变的前提下，利用栈来计算和获取表达式，使得基因型与表现型之间的映射关系已不同于前者。这 2 种方法对重组算子有着不同程度的影响。在 3 组数据上的实验结果表明，2 种新的解码方法可行而且高效，第 2 种解码方法更有利于优良子树模式的保护。

关键词：基因表达式编程；解码；基因型；表现型；复杂度

Two Decoding Methods of Gene Expression Programming

XIE Da-tong[1], CHEN Qiao-yun[2]
(1. School of Computer and Information, Yang-En University, Quanzhou 362014;
2. Department of Financial Affairs, Yang-En University, Quanzhou 362014)

【Abstract】 Two new decoding methods for Gene Expression Programming(GEP) are proposed in this paper, and their time and space complexity are analyzed. The first method conforms to the mapping relationship between genotype and phenotype of GEP, and the expression and its value are obtained directly on genotype. The second method gets the expression and its value by stack, which makes the mapping relationship between genotype and phenotype different from the foregoing. The two methods have different effects on the recombination operators. The experiments on three groups of data show the new decoding methods are feasible and efficient. The second method is more propitious to preserve the excellent schemes.

【Key words】 Gene Expression Programming(GEP); decoding; genotype; phenotype; complexity

1 概述

基因表达式编程(Gene Expression Programming, GEP)是C.Ferreira[1]提出的一种基于基因型和表现型的新型遗传算法，与遗传程序设计(Genetic Programming, GP)一样，它通过对给定的数据变量记录集进行学习，建立模型来反映这些变量之间的关系。

Ferreira在其专著[2]中对GEP进行了比较系统深入的研究。她认为，GEP与GA, GP一样遵循演化计算的基本原理，是对GA和GP的继承与发展，但又不同于它们。她解释说，GA和GP是简单的复制系统，后者比前者更复杂一些，GA中的实体是固定长度的线性结构，而GP中的实体是高度不等的、非线性的树结构，两者都是全编码利用，且只有一种实体类型或者说其个体的基因型和表现型都是同一对象。而按生物学理论，将基因型与表现型分开处理，如基因型采用GA中的线性结构，表现型采用GP中的树结构，一方面便于操作的实现，另一方面也具有灵活性。GEP就是基于这一思路发展而来，它兼具GA和GP的优点。

在 GEP 方法中，解码是演化过程中一个很重要的环节，在一定程度上它的效率影响整个程序的执行效率，故而本文对 GEP 的解码方法作了比较深入的研究，不仅探讨了 2 种新解码方法的具体实现，而且分析了它们的时间、空间复杂度以及对演化算子的影响。

2 基因表达式编程原理简介

2.1 编码规则

GEP中的基因组是由一个或多个基因组成的染色体。每个基因被划分成头、尾 2 个部分，规定头部可以出现函数符号(如运算符、初等函数甚至更复杂的函数)和终结符(包括自变量和常量，在多层基因的染色体中也可以是基因编号)，尾部只允许出现终结符。头长h由用户根据问题需要确定，而尾长t可以由式：$t = h\times(r\text{-}1)+1$ 得知。其中，r为基因所使用的函数集中函数所带的最大参数数目[2]。

定理 依据 GEP 编码规则产生的个体编码串总能得到一个有效的表达式。

证明：因为尾部不会出现函数符号，所以在基因对应的表达式树(Expression Tree, ET)中，基因尾部的符号总是作为叶子节点出现在 ET 的最后一层。而根据树的特点，只有在满树的情况下，叶子节点才是最多的，并且都出现在最后一层，满树情况下的叶子节点数目 t 与非叶子节点数目 f 之间存在这样一个关系 $t=(r\text{-}1)\times f+1$，因为每个非叶子节点引出 r 个分支，那么总共有 $r\times f$ 个分支。除了根之外，每个分支引出一个节点，从而树共有 $r\times f+1$ 个节点，因此叶子节点数为 $r\times f+1-f=(r\text{-}1)\times f+1$。显然如果基因头部长度为 h，且头部符号全都为带最大参数数目 r 的函数符号，则尾部节点数只要达到$(r-1)\times h+1$即可组建一棵合法的 ET。

2.2 演化算子

对于演化算法来说，好的算子在解的搜索过程中起了一个非常重要的作用。C.Ferreira 在其论著中主要提到了基本

作者简介：谢大同(1977－)，男，讲师、硕士，主研方向：演化计算及其并行化；陈巧云，助理会计师、学士

收稿日期：2008-01-08 **E-mail：**xiedt2004@163.com

GEP 中的 7 个算子，它们分别是变异、单点重组、两点重组、基因重组、插串换位、根插串换位和基因插串换位。限于本文篇幅，这里不再赘述，有关这些算子的详细介绍请参考相关文献[1-2]。

2.3 算法步骤

GEP 算法与其他演化算法一样，主要包括种群初始化、适应值评估、遗传操作等步骤。

3 基因表达式编程的解码方法

3.1 基因表达式编程的解码过程

从 GEP 基因型和表现型之间的映射关系可以看出解码过程实际上就是一个层次遍历建树，后序遍历求值的过程。根据基因型建立 ET 的做法是：逐位读取字符，将基因的第一位字符作为树的根节点，如果根节点是带 2 个参数的函数符号，则连续读取 2 个字符分别作为其左、右孩子，若根节点是带一个参数的函数符号，则读取一个字符作为其左孩子，若根节点为终结符则不给其分配孩子节点；接下来再依次考察其左右孩子，依此类推，直到下一层的节点均为终结符。

在多数 GP 和 GEP 论著中，都特别强调 ET 这样一个概念。对于 GP 来说，遗传操作是针对 ET 进行的，建立显式的 ET 有助于操作的实施；对于 GEP 来说，遗传操作是在基因型上进行的，建树过程有可能使 GEP 算法在时间和空间耗费上有所增加，因此只要能找到一个合适的适应值求解方法，则建立 ET 并非必要。

3.2 新的解码方法

(1)解码方法一

前面谈到对基因进行解码的常用方法是先根据基因编码建立ET，然后采用后序遍历树的方法求出表达式的值。假设单基因染色体头长为n，则建树的时间复杂度$O(n)$，后序遍历树求解表达式值的时间复杂度$O(n)$[3]，因此解码总的时间复杂度为$O(n)$，考虑到函数集的最大角数*maxArity*(即函数所带的最大参数个数)通常不超过 3，空间复杂度则为$(n\times maxArity+1)\times maxArity = O(n)$。

从对 ET 的后续遍历过程可以看出，计算是从其最后一棵以一个函数符号为根的子树(总共 2~3 个节点)开始的，依次往根节点靠拢。因此若不通过先建树来计算，首先必须找到基因型的有效部分以及最后一个函数符号的位置。对单个基因来说，可使用 3 个标志变量进行搜索和记录。下面是找基因有效部分及定位最后一个函数符的算法：

```
1  p = -1;
2  q = 0;
3  i = 0;
4  do{
5      if(ch[i]是函数)
6          q+=arity(ch[i]);
7          p = i;
8      endif
9  }while(++i ≤ q);
```

其中，*ch* 为基因串；*p* 指示有效函数符的位置；*q* 指示终结符的位置；*i* 为当前搜索的字符位置。

找到有效基因串后，可定义一个大小为基因有效长度的浮点类型数组存放每一条数据记录的中间结果以及最后的计算值。在使用每一条数据记录计算前先将对应位置的终结符值填入，计算从最后一个运算符开始，每次从与运算符位置相对应的浮点数单元之后按运算符所带的参数数目取数，计算后将结果填入与运算符位置相对应的浮点数单元。如此反复直到获得第一个浮点数单元中的数值即为表达式最终结果。若要获得中缀表示的最终表达式，可参考基因适应值评价的过程。

从这个解码过程来看，其时间复杂度在数量级上与建树后再后序遍历树相同，同样为 $O(n)$，但若不是基于顺序结构建立树，则节点空间的动态分配所带来的时间耗费比较大。此外，由于每个树节点需多用 *maxArity*(最大参数数)个指针，使这种解码方法在空间复杂度上比建树解码的方式要占优。

(2)解码方法二

解码方法一与建树解码方式只在形式上有差异，因此，最终获得的表达式完全一致。下面要提到一种基于 GEP 的基因结构构造不仅在解码形式上，且在最终表达式结果上有很大差异的解码方式。

按 GEP 编码方式，基因头长一确定，尾长就可确定下来，以后只要按其编码规则变化，所得编码串总能表示一个有效的基因。GEP 算法总是按照层次遍历建树、后序遍历求值的方式进行解码。既然其编码串在编码规则限制范围内变动总有效，那是否可以按别的方式来解码？在研究第 1 种解码过程中发现，当从基因尾部往前计算时，若用堆栈来存储中间结果而不是将中间结果放入与运算符位置对应的浮点数单元中，可发现最终得到的表达式通常有所不同。下面是第 2 种解码方式下表达式求值的算法：

```
1   double Calculate(char * s)
2   {
3       k = valid_gene_length – 1;
4       do{
5           if(s[k]是函数符号)
6               从栈中取出数目为函数参数数量的数值;
7               计算，并将结果压栈;
8           else
9               将终结符对应的数值压栈;
10          Endif
11          }while (--k≥0);
12          return  栈顶数值;
13  }
```

从上面的表达式求值算法，易知第 2 种解码方法的时间复杂度也是 $O(n)$，空间耗费主要在栈上，空间复杂度为 $O(n)$，其中 n 同样为染色体头长。

GEP 中遗传算子实施的对象是一个线性结构的染色体，而在 GP 中算子操作对象是 ET，通常情况下 GP 中的算子不会造成 ET 在一次遗传操作之后变得面目全非，它能有针对性地保留或者在个体间交换一些好的基因片段即 sub-ETs。GEP 的解读方式使得其大多数算子极具破坏性，譬如基因有效部分的双参函数变为单参函数或者反过来会使得后面基因段中的字符在 ET 中的位置发生强烈的变化。C.Ferreira 对这种强变化的特点看作是 GEP 相对 GP 的优势，她认为 GP 的遗传算子只是使得遗传信息在变化点周围迁移。经本实验发现，变异算子是 GEP 中最具破坏力的算子，在算法中甚至只使用变异算子就可获得不错的效果，其他算子在算法中能将好的基因信息保留下来且不至于造成整个染色体被强烈破坏的仅有基因重组算子和基因插串换位算子。对比 GEP 中基因的 ET 和 K-表达式，可以发现，ET 的子表达式中的字符在 K-表达式中并非连续出现，如对于一个最简单的子表达式(a+b)在 K-表达式中可能是+和 ab 之间还有一些字符。下面举

一个具体的例子说明。对于基因*-+abaa，其 ET 如图 1 所示，其对应的 k-表达式为*-+abaa，-的操作数是 a, b，而-与 ab 之间存在符号+。因此，如果-, a, b 构成的是一个优秀的子树，要通过重组将其提取出来植入到别的染色体中，必须分别截取-和 ab，而 GEP 重组算子要完成这样一个过程难度较大。

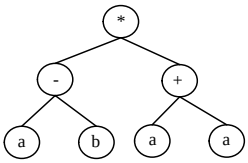

图 1　基因*-+abaa 对应的 ET

若使用第 2 种解码方式，只需让单点、两点算子针对这种方式作一些适当的处理就可在个体间共享有用的子树信息。下面考察一下第 2 种解码方式下的 K-表达式和对应的 ET 之间的微妙关系。如基因一+*a/Qa+aaababab 和基因二 Q/*b-+abbabbaaa，其对应的 K-表达式和 ET 如图 2 所示。

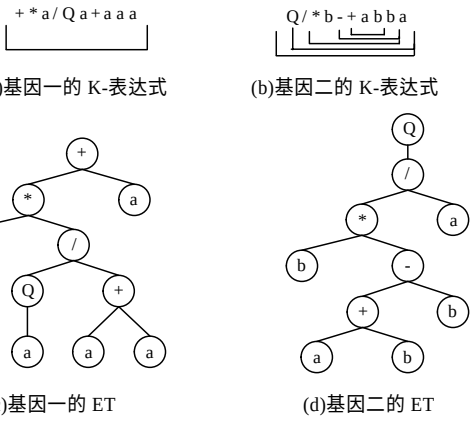

图 2　第 2 种解码方式下的 K-表达式和 ETs

从图 2 中可以看出，Qa, +aa, /Qa+aa 等可构成其中的 sub-ETs，而这些 sub-ETs 对应的前序序列都是它的子串，并且一般说来越靠近基因头部，sub-ET 越复杂。因此，若根据这个特点对一点重组和两点重组做一些改变，将更有利于优秀子树的保留与遗传。譬如，对于两点重组，先在一个染色体的某个基因头部或身部随机选一个位置，往后扫描以找到第一个函数符号的位置作为第一个交叉点，根据这种解码方法的特点，再往后扫描必定可以找到一个终结符使得从第一个交叉点到该终结符的字符串能解码成一个有效的子表达式，最后执行交叉操作。

上面的基因一和基因二按照 C.Ferreira 所提的基因型与表现型之间的映射关系可获得 ETs，如图 3 所示。

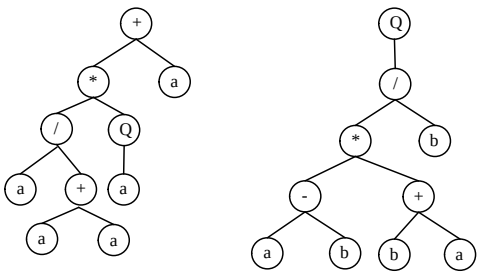

图 3　基因一和基因二在第 1 种解码方式下的 ETs

假设这 2 个基因在实施两点交叉时交叉点选择第 5 位和第 9 位，则交叉后得到后代个体：+*a/-+abbababab 和 Q/*bQa+aaabbaaa，对应的 ETs 如图 4 所示。

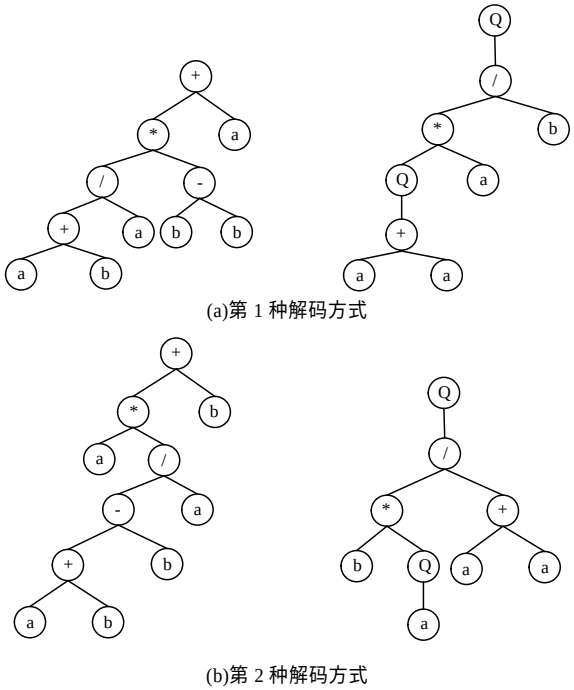

图 4　基因一和基因二经两点交叉后在 2 种解码方式下的 ETs

从基因执行交叉操作之后的图 4 可看出，第 1 种解码方式下的重组仅能交换单个符号的信息，不仅无法交流子树信息，而且经过交叉操作后很可能破坏进化得到的优秀子树分支，而第 2 种解码方式下的重组能交流子树信息。这个实例说明解码方式能够影响到算子的演化能力，从而影响演化的效率。

4　实验

该实验的数据分别来自文献[4-6]。第 1 组数据是关于某矿 3 个工作面 18 个回采月份的采煤工作面瓦斯涌出量的统计资料，要求建立一个模型反映煤层埋藏深度、煤层厚度、煤层瓦斯含量、工作面煤层与邻近煤层的层间距、工作面平均日进度、工作面平均日产量和采煤工作面瓦斯涌出量之间的关系；第 2 组数据是关于天津市 1980 年~1996 年资金、就业人员与国内生产总值的统计数据，要求建立模型反映资金和就业人员数量对国内生产总值的影响；第 3 组数据是 1900 年~1998 年关于地震的时间序列数据，要求建立模型反映该段时间地震发生的趋势，并可以预测未来地震情况。

分别以这 3 组数据对第 1 种解码方法和第 2 种解码方法作对比实验。

表 1 给出了数据集的说明。3 组数据的函数集分别为{+,-,*,/,E,S,C}, {+,-,*,/,L,S,C,T,Q}, {+,-,*,/,E,L,S,Q}，终结符集分别为{a,b,c,d,e,f,?}, {a,b,?}, {a,b,c,d,e,f,g,h,i,j,?}(嵌入维度为 10)，常量范围分别为{-10,10}, {-1,1}, {-1,1}，常量数组大小均为 50，基因头长分别为 8, 3, 8，身长分别为 20, 22, 20，单点重组概率为 0.3，两点重组概率分别为 0.3, 0.5, 0.3，变异概率为 0.044，选择算子采用竞争规模为 3 的锦标赛选择，适应值函数为

$$fitness = 1\ 000 - \sum_{i=1}^{m} \left| \frac{y_i' - y_i}{y_i} \times 1\ 000/m \right|$$

其中，y_i和y_i'分别为实测值和拟合值；m为总记录数，父代和子代群体大小分别为 50, 30。停机条件为适应值达到　1

000 或者演化代数达到 5 000，实验运行 200 次。

表 1　数据集信息

名称	采煤工作面瓦斯涌出量	国内生产总值统计信息	地震时间序列
自变量数	6	2	需要根据嵌入维度确定
数据集大小	18	17	90

从表 2 中的实验结果可以看出，在其他实验参数不变的情况下，3 组数据在第 2 种解码方式下实验 200 次的平均最好适应值都比第 1 种解码方式好，取得的最好结果也优于后者。

表 2　2 种解码方式下的演化结果

解码方式	演化结果	第 1 组数据	第 2 组数据	第 3 组数据
第 1 种解码	平均适应值	945.875	941.574	965.871
	最佳适应值	973.580	975.398	970.437
第 2 种解码	平均适应值	957.636	944.547	966.646
	最佳适应值	977.505	983.391	971.105

从表 3 的标准差数据和 t-test 结果可知，第 2 种解码方式下的演化结果更稳定。这说明，解码方式能影响重组算子的演化能力，从而影响到演化结果，而第 2 种解码方式下之所以能获得适应值更好的表达式在于它能有效地保护优良子树模式。

表 3　2 种解码方法下平均演化结果精度的比较

演化结果		第 1 组数据	第 2 组数据	第 3 组数据
标准差	第 1 种解码	17.846	30.685	2.820
	第 2 种解码	10.751	20.791	2.108
t-test 值		-7.983	-1.134	-3.112

5　结束语

本文探索了基因表达式编程的解码方法，并提出了 5 种新的解码方法。在一定程度上提高了算法的执行效率，使基因表达式编程在求解更大规模的问题时将比其他传统方法优势更明显。同时，本文的研究也表明基因表达式编程有待进一步完善，不仅在其编码解码、演化算子等具体实现和方法上有进一步发掘的潜力，而且其理论基础还相当薄弱。在下一步的研究中，笔者将重点分析基因表达式编程的收敛性，并拓展基因表达式编程的应用范围。

参考文献

[1] Ferreira C. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems[J]. Complex Systems, 2001, 13(2): 87-129.

[2] Ferreira C. Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence[M]. New York, USA: Springer-Verlag, 2002.

[3] 严蔚敏, 吴伟民. 数据结构(C语言版)[M]. 北京: 清华大学出版社, 2002.

[4] 赵朝义, 袁修干, 孙金镖. 遗传规划在采煤工作面瓦斯涌出量预测中的应用[J]. 应用基础及工程科学学报, 1999, 7(4): 387-392.

[5] Cai Zhihua, Jiang Siwei, Zhu Li, et al. A Novel Algorithm of Gene Expression Programming Based on Simulated Annealing[C]//Proc. of International Symposium on Intelligence Computation & Applications. Wuhan, China: University of Geosciences Press, 2005: 605-611.

[6] Hyndman R J. Time Series Data Library[EB/OL]. (1998-12-06). http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/.

~~~~~~

（上接第 209 页）

表 2　水声信号的预测结果

比较项目	PSO-RBFNN 算法	GA-RBFNN 算法	*K*-means 算法
训练集的 MSE	0.161 8	0.175 3	0.193 1
测试集的 MSE	0.165 9	0.173 6	0.102 8
隐层节点个数	16	18	24
迭代次数	20	30	…

图 2 为水声信号时间序列实际输出与所建模型预测值的比较曲线。从中可以看出，预测值与实际值吻合较好，其 MSE 误差为 0.165 9。

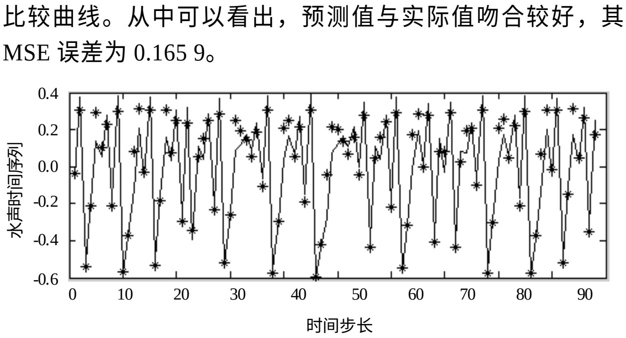

图 2　水声信号序列实际值和预测值比较

5　结束语

本文在自增加聚类算法确定 RBF 神经网络基函数中心个数的基础上，运用改进的粒子群优化算法对其距离阈值进行优化，结合最小二乘法确定网络输出权值，并将得到的 RBF 网络用于混沌时间序列和水声信号预测。从对实际舰船信号的预测结果可以看出，利用 PSO 和 RBF 神经网络相结合的方法进行舰船信号建模和预测是可行的，同时还验证了混沌时间序列的短期可预测性。

参考文献

[1] 章新华, 张晓明, 林良骥, 等. 舰船辐射噪声的混沌现象研究[J]. 声学学报, 1998, 23(2): 134-140.

[2] Packard N H, Crutchfield J P, Farmer J D, et al. Geometry from A Time Series[J]. Physics Review Letters, 1980, 63(6): 712-716.

[3] Takens F. Detecting Strange Attractor in Turbulence[M]. Berlin, Germany: Springer, 1981.

[4] Chen Guanrong. Fuzzy Modeling, Prediction, and Control of Uncertain Chaotic Systems Based on Time Series[J]. IEEE Trans. on Circuits and Systems, 2000, 47(10): 1527-1531.

[5] Bai Yunfei. Genetic Algorithm-based Self-growing Training for RBF Neural Network[C]//Proc. of Int’l Joint Conf. on Neural Networks. [S. l.]: IEEE Press, 2002.

[6] Clerc M, Kennedy J. The Particle Swarm Explosion, Stability, and Convergence in a Multi-dimensional Complex Space[J]. IEEE Trans. on Evolutionary Computation, 2002, 6(1): 58-73.
~~~~~~