M-GEP:基于多层染色体基因表达式编程的遗传进化算法

彭 京[1),2)] 唐常杰[1)] 李 川[1)] 胡建军[1)]

[1)](四川大学计算机学院数据库与知识工程研究所 成都 610065)

[2)](成都市公安局科技处 成都 610017)

摘 要 该文提出了一种新的基于多层染色体基因表达式编程的遗传进化算法 M-GEP,新算法引入了多层染色体的概念,利用染色体构建的层次调用模型对个体进行表达,在解决实际函数发现、电路进化等实际问题中取得了良好效果. 该文主要贡献包括:(1)提出了基于多染色体的基因表达式编程算法(M-GEP);(2)建立了不同染色体的层次调用模型及存储结构;(3)提出并实现了基于染色体的重组算子和基因随机重组算子. 对多基因 GEP 和单基因 GEP 的对比实验结果表明,平均进化辈数仅为后者的 29%～81%.

关键词 多层染色体;M-GEP;遗传进化;基因表达式编程

中图法分类号 TP311

M-GEP: A New Evolution Algorithm Based on Multi-Layer Chromosomes Gene Expression Programming

PENG Jing[1),2)] TANG Chang-Jie[1)] LI Chuan[1)] HU Jian-Jun[1)]

[1)](*Institute of Database and Knowledge Engineering, College of Computer Science, Sichuan University, Chengdu* 610065)

[2)](*Department of Science and Technology, Chengdu Public Security Bureau, Chengdu* 610017)

Abstract This paper proposes a new evolution algorithm, M-GEP, based on the new concept of the multi-layer chromosomes in gene expression programming. The algorithm is efficient in the real applications, such as function discovery, electronic circuit evolution, etc. The main contributions include proposing algorithm M-GEP which is based on multi-layer chromosomes, establishing Level-call model and storage structure between the different chromosomes, and suggesting and implementing chromosomes reorganization operator and genes random reorganization operator. Extensive experiments on the traditional single gene and multi-genes GEP show that the average number of generations of M-GEP is reduce to 29%～81%.

Keywords multi-layer chromosome; M-GEP; heredity evolution; gene expression programming

1 引 言

遗传算法是数据挖掘和机器学习方法的重要分支. GEP(Gene Expression Programming)是借鉴生物遗传的基因表达规律提出的知识发现新技术. 文献[1～7]比较了 GEP 和传统的遗传算法(GA)与遗传编程(GP),特色如下:(1)它们在"杂交—变异—新种群—选择"的进化论思想方面相似;(2)GA 是用简单编码解决简单问题,GP 是用复杂编码解决复

收稿日期:2004-05-08;修改稿收到日期:2005-04-26. 本课题得到国家自然科学基金(60073046)、四川省重点科技计划项目基金(04SG1640)和教育部博士点专项基金(20020610007)资助.彭 京,男,1973 年生,博士研究生,主要研究方向为数据挖掘、进化计算. E-mail:pj@mail.sc.cninfo.net. 唐常杰,男,1946 年生,教授,博士生导师,主要研究方向为数据挖掘与知识发现. 李 川,博士研究生,研究方向为数据库与数据挖掘. 胡建军,博士研究生,研究方向为数据库与数据挖掘.

杂问题,GEP 是用简单编码解决复杂问题;(3)GEP 融合了 GA 和 GP 的优点. Ferreira 在文献[2]中指出 GEP 比 GA 和 GP 快 2～4 个数量级[2];(4)GEP 的先天优势使得组合爆炸(维度爆增)和早熟(或称为局域沉陷)容易克服[2];(5)基因表达式编程不需对结果公式形状做主观假定,而认为真理在训练数据中,以基本变量和算符为遗传物质,通过遗传、变异,逐代进化出更接近事物本质的公式(例如由+,-,*,/,(,),x,y,&,|,～,true,false 组合表达的结果).方法上更客观,结果上更接近事物本质.

GEP 在处理较复杂问题时,仍存在进化辈数过大或无法取得最优结果的情况.本文在 GEP 算法中引入多层染色体的概念,提出了 M-GEP(Multi-Layer Chromosome Gene Expression Programming)算法.算法有如下特点:(1)种群中每个个体由多个染色体组成;(2)不同的染色体有不同字符集和长度;(3)多个染色体分为多个层次,上层染色体基因可以调用下层染色体基因的计算结果.本文实现了新算法及相关的遗传算子,并从表达空间上分析了该算法较之传统 GEP 的优势,证明了单基因 GEP 和多基因 GEP 均为 M-GEP 一定条件下的特例.该算法在解决具体问题中取得了较好的结果,实验表明,与单基因 GEP 和多基因 GEP 算法相比,平均进化辈数仅为其 29%～81%.

本文第 2 节给出了有关 M-GEP 的背景知识和相关定义;第 3 节描述了 M-GEP 算法;第 4 节给出相关计算方法;第 5 节描述了多染色体的遗传进化算子:染色体重组算子、多基因随机重组算子以及其它相关算子;第 6 节分析了算法的表达空间;第 7 节以函数发现和电路进化为例进行实验,并同单基因 GEP 和多基因 GEP 算法进行了比较;第 8 节总结了实验结果,并提出了下一步的研究方向.

2 算法背景及基本概念

2.1 基本定义

与遗传算法家族的其他成员如 GP、GA、GEP 类似,M-GEP 以生物进化为启示,是模仿生物进化过程的随机方法.本文采用了一系列关于遗传计算的概念,如:基因(gene)、基因型(genotype)、表现型(phenotype)、染色体(chromosome)、个体(individual)、种群(population)等,其具体定义如下.

定义 1(基因). 基因 G 是一个 5 元组,记为 $G=(Q,T,F,\delta,s)$,其中 Q 为基因型;T(Terminal)为基因终端字符集;F(Function)为基因运算符集合;δ 为基因遗传算子集合,s(score)为适应度.

在上述定义中,δ 包含变异算子、位移算子、根位移算子等;s 是基因针对特定数据集的适应度得分.例如('*++-aabcd','abcd','+-*/',0).

定义 2(染色体). 染色体 C 是一个 5 元组,记为 $C=(\Sigma,T,F,\delta,s)$.其中 Σ 为基因的集合;T(Terminal)为基因终端字符集;F(Function)为染色体中基因的连接运算符.δ 为染色体遗传算子集合,s(score)为染色体适应度.

C 中 δ 可以表示为染色体算子和基因算子的并集,即有

$$\delta=\delta_c\cup(\delta_{g1}\cup\delta_{g2}\cup\cdots\cup\delta_{gn}),\ n=\text{length}(C(\Sigma)).$$

δ_c 表示针对染色体内多个基因的操作算子,如单点重组算子、多点重组算子、基因重组算子及基因位移算子等,$\delta_{gi}(i=1,2,\cdots,\text{length}(C(\Sigma)))$ 表示染色体内每个基因的算子,通常取 $\delta_{g1}=\delta_{g2}=\cdots=\delta_{gn}$ $(n=\text{length}(C(\Sigma)))$,即染色体内所有基因取相同的遗传算子.

例如:$C_1=(\{G_1,G_2,G_3\},\text{'ab'},\text{'+'},\delta,0.35)$.表示染色体 C_1 由 3 个基因组成,染色体的终端字符集为(a,b),基因的连接运算符为'+',适应度得分为 0.35.

定义 3(个体及种群). 个体 I 是一个 3 元组,记为 $I=(\Sigma,\delta,s)$,其中 Σ 为染色体集合,δ 为个体遗传算子集合,s(score)为个体适应度.种群 $P=(\Sigma)$,Σ 为个体集合.

同定义 2 类似,染色体 $I(\delta)$ 可表示为

$$\delta=\delta_i\cup(\delta_{c1}\cup\delta_{c2}\cup\cdots\cup\delta_{cn}),\ n=\text{length}(I(\Sigma)).$$

δ_i 表示针对多个染色体的操作算子,如染色体重组算子、基因随机重组算子,$\delta_{ci}(i=1,2,\cdots,\text{length}(I(\Sigma)))$ 表示每个染色体的遗传算子.

根据定义 1～3,单基因 GEP 算法具有如下特征:种群中任意个体 $I=(\Sigma,\delta,s)$ 满足:$\text{length}(I(\Sigma))=1$(即只有一个染色体),且 $C\in I(\Sigma)$,$\text{length}(C(\Sigma))=1$(即只有一个基因).而多基因 GEP 算法的特征是:种群中任意个体 $I=(\Sigma,\delta,s)$ 满足:$\text{length}(I(\Sigma))=1$(即只有一个染色体).

命题 1(GEP 算法的包含关系). 设 As 是单基因 GEP 算法,Ag 是多基因 GEP 算法,Am 是多染色体算法.令≤表示算法的包含关系,则有 As≤Ag≤Am,即多基因 GEP 算法包含单基因 GEP 算法,而 M-GEP 算法包含多基因 GEP 算法.

证明. (1)由多基因 GEP 及 M-GEP 算法特

点知 Am 对个体 I 及 $I(\Sigma)$ 均没有限制，而 Ag 限制了染色体数，故有 Ag≤Am. (2)根据单基因 GEP 算法特点，由于 Ag 对基因数量没有限制，而 As 中将基因数限制为 1 故有 As≤Ag. 综合(1)(2)知 As≤Ag≤Am.　　证毕.

2.2　编码方法

GEP 中，编码是在从基因型到表现型的映射. 在传统遗传算法中，基因型和表现型相同，都是 Tree，并且遗传操作算子(变异、交叉等)是直接作用于 Tree，所以，在遗传操作过程中，产生大量的无效结构. 而 Ferreira 提出的 ET 表达式则采用不同的编码方法[1]，例如对表达式(a∗b+c)/(c∗a)，GP 中的基因型和表现型都是图 1 所示的结构，这个 Tree 在 GP 中既是基因型又是表现型，但是在进化过程中，容易出现非法表达式. 在 GEP 中表达式(表现型)与图 1 相同，但基因型则采用不同的策略，对图 1 采取从左至右，从上至下的编码方式，得到/+∗∗ ccaab.

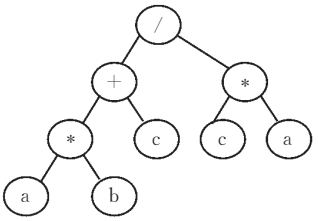

图 1　GP 的结构

为保证表达式的合法性，在 GEP 中，基因分为头部和尾部，头部可以包括 F(连接运算符集合)和 T(终端字符集合)，尾部只包括 T，头部和尾部只要满足式(1)[7]，就一定可以得到一个正确的基因型，不会产生无效的 ET 表达式，

$$t = h(n-1)+1 \tag{1}$$

其中，h 代表头部长度，t 代表尾部长度，n 代表 F 中参数的最大个数，比如操作运算函数“∗”带 2 个参数.

3　M-GEP 算法

M-GEP 算法过程与传统遗传算法基本过程类似，即：变异—杂交—新种群—选择. M-GEP 算法设计中主要涉及了多染色体的处理：(1)针对染色体的遗传算子. 由于在进化中引入了多染色体的概念，因此本文专门定义了关于染色体算子操作，即染色体重组算子，记为 P_c 和基因随机重组算子，记为 P_r. 关于染色体算子的细节，在第 5 节中描述. (2)染色体之间的调用模型.

针对一个具体问题，个体最终的适应度只有一个，因此必须设计不同染色体之间适应度的结合方式. 下面就此问题具体讨论.

3.1　调用结构

M-GEP 算法将个体 I 中的染色体 $I(\Sigma)$ 设计为层次结构，按从下到上持续，记为 $C_1, C_2, \cdots, C_k$，其中 $k=\text{length}(I(\Sigma))$，即染色体个数. 设 $C_1(\Sigma)=(G_{(1,1)}, G_{(1,2)}, \cdots, G_{(1,n)})$，$n=\text{length}(C_1(\Sigma))$，即第 1 个染色体的基因个数，则可定义第 2 个染色体 C_2 的终端字符集：

$$C_2(T)=(G_{(1,1)}, G_{(1,2)}, \cdots, G_{(1,n)}) \tag{2}$$

其中 $n=\text{length}(C_1(\Sigma))$. 一般的有第 i 个染色体 C_i 的终端字符集：

$$C_i(T)=(G_{(i-1,1)}, G_{(i-1,2)}, \cdots, G_{(i-1,n)}),\ i>1 \tag{3}$$

即将所有的非第 1 个染色体的终端字符集设置为下一级染色体的基因，如图 2 所示.

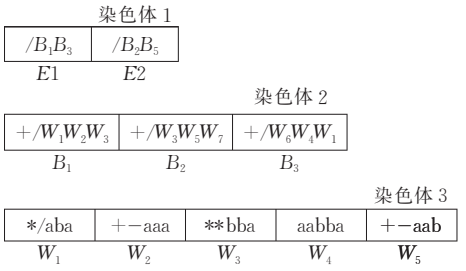

图 2　染色体调用模型

在图 2 中共定义 3 个染色体，从上到下分为 3 层. 最下层染色体有 5 个基因，分别记为 $W_1, W_2, \cdots, W_5$. 第 2 个染色体有 3 个基因，记为 B_1, B_2, B_3，染色体的终端字符集为 $(W_1, W_2, \cdots, W_5)$，表示对第 3 个染色体基因的调用. 第 1 个染色体有 2 个基因，其终端字符集为 (B_1, B_2, B_3)，表示对第 2 个染色体基因的调用.

在针对特定训练集计算个体适应度中，采用逐级方式求得染色体每个基因表达式得分，代入作为下一条染色体的终端字符集. 最终个体的适应度得分根据最后一条染色体每个基因表达式得分求出，适应度公式如下：

$$f_i=\sum_{j=1}^{C_t}(M-|\ C_{(i,j)}-T_j\ |) \tag{4}$$

或

$$f_i=\sum_{j=1}^{C_t}\left(M-\left|\frac{C_{(i,j)}-T_j}{T_j}\cdot 100\right|\right) \tag{5}$$

其中 M 为适应度规范因子，下标 i 表示种群中的第 i 个个体，j 表示适应度用例中的第 j 个用例，C_t 表

示适应度用例的数目，$C_{(i,j)}$ 表示第 i 个个体代入第 j 个测试用例后得到的结果，T_j 表示测试用例中的目标函数值. 例如给定表 1 所示的训练集，自变量为 A,B，目标函数值 $T_j(j=1,2,\cdots,C_t)$ 对应 Result 列，C_t 等于 4，表示共有 4 个测试用例.

表 1 训练集

A	B	Result
1	3	7
2	4	12
3	7	82
4	2	30

3.2 算法描述

M-GEP 算法完成具体遗传进化过程，其算法框架描述如下所示.

算法 1. M-GEP 算法.

输入：训练集及 M-GEP 配置

输出：最佳适应度个体

```
Init(StepConfig);//加载配置和种群初始化
//调用评价函数
DefaultJudge(self,Population);
//循环遗传进化,直到达到最大辈数或成功
while (FCurrentGenaration<FMaxGenaration) do begin
  //首先将适应度排名第一的继承下来
  SetLength(tmpPopulation,1);
  tmpPopulation[0] := Population[0];
  //依次进行各项遗传算子操作,如变异、重组等等
  for x := 0 to FOperationCount-1 do
    FOperationList[x](self,x,Population,
        tmpPopulation);
  //调用评价函数
    DefaultJudge(self,tmpPopulation);
  //将结果排序并复制到下一代
  Sort(tmpPopulation,Population);
  //进化辈数加 1
  Inc(FCurrentGenaration);
  //如果最好进化结果满足要求,则退出
  if tmpPopulation[0].Score <Fprecision then
    break;
end;
//返回最佳适应度个体
  Result := tmpPopulation[0];
```

引理 1. 算法 1 的复杂度是 $O(m\times n\times k)$，其中 n 为种群大小，m 为总进化辈数，k 为训练集的长度.

证明. 在算法 1 计算个体针对 k 个训练样本的复杂度为 $O(k)$；算法需计算种群中每个个体的适应度得分，故种群适应度计算的复杂度为 $O(n\times k)$，因算法 1 最多进化 m 代，故算法 1 的复杂度为 $O(m\times n\times k)$. 证毕.

4 适应度计算结构

第 3.2 节中 $DefaultJudge$ 过程实现评价函数计算，即个体适应度计算，其要点是：(1)对每个个体完成语法分析，得到计算步骤；(2)分配相应的计算缓冲池，用于保存中间计算结果；(3)依次调入训练数据，通过计算步骤及计算缓冲池得到最终适应度得分. 个体适应度算法(Individual Fitness Algorithm，IFA)的形式化描述如下.

算法 2. 个体适应度算法(IFA).

输入：个体 $I(\Sigma)$(包括多个染色体及内含基因)

输出：$I(s)$(即个体适应度得分)

1. 分析 I 每个染色体 C 及内含基因，得到内部计算步骤数组，每个计算步骤数组包括计算操作符、运算操作的每个参数偏移量和结果保存偏移量.

2. 根据总计算次数分配计算缓冲池.

3. 读取一条样本数据，将样本数据写入计算缓冲池指定位置.

4. 按照步 1 得出的计算步骤数组，依次进行计算，并将结果写入计算缓冲池，在计算池首部得到最终个体针对特定样本的适应度得分.

5. 重复第 3，4 步，直到完成所有样本数据计算. 适应度公式(4)或(5)，计算该个体对样本数据最终的适应度.

由以上算法可以看到，无论个体的染色体数目和每个染色体基因的数量有多少，训练均可以通过一个统一的循环处理过程完成，且该算法循环次数为 k，k 为训练集的大小. 例 1 说明了 IFA 算法的计算过程.

例 1. 考察个体 $I=(\{C_1,C_2\},\delta,s)$ 的计算过程，其中 $C_1=(\{G_1\},$'B_1','+',$\delta_1,s)$，$C_2=(\{G_2\},$'ab','+',$\delta_2,s)$，且 $G_1=($'$+B_1B_1$','B_1','+-*/',-)，$G_2=($'* ab','ab','+-*/',-). 即个体 I 有两个染色体，每个染色体均只有一个基因，每个基因的表达式分别为：'$+B_1B_1$'，'* ab'，其中 B_1 表示第 2 个染色体第 1 个基因的计算结果.

根据算法 IFA 第 1 步，得到计算步骤数组，如下所示：

$$f_1=(fMul,ResultNo:3,ParamNo:2,Params:(1,2)),$$
$$f_2=(fAdd,ResultNo:4,ParamNo:2,Params:(3,3)).$$

由以上可见：(1) f_1 由保存在偏移位置 1 和 2 的数值执行乘法操作，并将执行结果保存在偏移位置 3. (2) f_2 将偏移位置 3 的数据读出，并与自己相加，然后将结果保存到偏移位置 4.

根据算法步 2 可知计算缓冲池长度为 4，最终计算结果保存在偏移位置 4，缓冲池如图 3 所示.

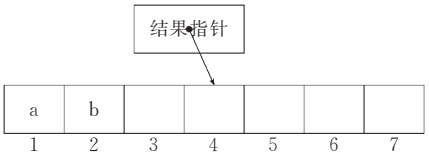

图 3　计算缓冲池结构

根据步 3，4，依次读取样本数据，将计算缓冲池中偏移位置 1 设为样本中 a 的值，偏移位置 2 设为 b 的值；然后根据计算数组 f_1 将偏移位置 3 存入偏移位置 1 和 2 相乘的结果；再根据计算数组 f_2，在偏移位置 4 存入偏移位置 3 与自己相加的结果；从计算缓冲池偏移位置 4 读出个体针对特定样本的表达式结果.

最后在所有的样本计算完成后，根据公式(4)或(5)，得出最终的适应度.

5　遗传算子操作

遗传操作是遗传算法中极为重要的一环. 在第 3 节中已经对 M-GEP 算法作了介绍，其中对各项遗传操作采用了循环方式完成，本节中将介绍 M-GEP 算法中的具体遗传算子.

遗传算子操作过程可归纳为选择—计算—生产三个步骤. 首先遗传算子通过轮盘赌等算法确定作用的个体，然后根据选择执行具体的遗传计算，最后产生下一代并返回. 在 GEP 算法中遗传算子主要有变异、位移、根位移、单点重组、两点重组、基因位移、基因重组[1~3]等.

针对 M-GEP 中存在多个染色体的情况，本文提出了采用染色体重组算子 Pc 和基因随机重组算子 Pr，用于解决染色体之间的遗传操作. 具体算法描述如下.

算法 3.　染色体重组算子(Pc)操作.

输入：种群 $P(\Sigma)$

输出：后代种群 $P_{\text{next}}(\Sigma)$

1. 计算需要由 Pc 算子生成的后代数；
2. 根据轮盘赌的方式随机选择两个需要重组的个体；
3. 以基因为单位，随机产生染色体重组位置；
4. 交换两个个体的重组位置后的所有基因；
5. 产生两个后代个体；
6. 重复步 2～步 5，直到达到步 1 指定辈数.

算法 4.　基因随机重组算子(Pr)操作.

输入：种群 $P(\Sigma)$

输出：后代种群 $P_{\text{next}}(\Sigma)$

1. 计算需要由 Pr 算子生成的后代数；
2. 根据轮盘赌的方式随机选择两个需要重组的个体；
3. 对两个重组个体的所有基因随机选择是否交换；
4. 用选择结果生成两个后代个体；
5. 重复步 2～步 4，直到达到步 1 指定辈数.

本文第 7 节的实验将证明这两个算子对最终进化结果具有明显的影响.

6　表达空间分析

定义 4(表达空间).　设个体 $I=(\Sigma,\delta,s)$ 最终得到的表达式为 E，则称 $\text{length}(E)$ 为个体 I 的表达空间，记为 D_I.

通过下面的分析和实验，可以看到表达空间的大小与问题的求解有直接关系.

引理 2.　单基因算法 As 的最大表达空间 $\max(D(As))=n$，其中 n 为基因长度，即基因的最大表达空间等于基因的长度.

证明. 已知个体由一个基因组成，由式(1)知，在最好情况下，有 $\text{length}(E(I))=n$，即 $\max(D(As))=n$. 证毕.

引理 3.　多基因算法 Ag 的最大表达空间 $\max(D(Ag))=n+k-1$，其中 n 为染色体长度，k 为基因个数.

证明.　设个体 I 每个基因长度为 h，则有 $n=k\times h$，由引理 2 知，每个基因的最大表达空间等于基因的长度，故有 k 个基因的最大表达空间为 $k\times h=n$，另外多个基因需通过操作函数将结果连接后输出，而 k 个基因需 $k-1$ 个连接符，故最终的表达式长度为 $n+k-1$，即 $\max(D(Ag))=n+k-1$. 证毕.

引理 4.　设个体由两层染色体组成，其中每层染色体为 $k/2$ 个基因，每层染色体的基因的长度均为 h，则有多染色体算法 Am 的最大表达空间 $\max(D(Am))\sim O(n^2/k)$，其中 $n=k\times h$.

证明.　设个体染色体总长度为 n，由定义知 $n=h\times k$，根据引理 2 下层染色体单个基因表达式最大长度为 h，由 M-GEP 算法知上层染色体的终端字符集为基因 $G_{(1,1)},G_{(1,2)},\cdots,G_{(1,k-1)}$. 所以上层每个基因的最大表达空间为 $T(h)\times h+H(h)$. 由引理 3 知上层染色体最大表达空间：

$$D=(T(h)\times h+H(h))k/2+k/2-1 \quad (6)$$

其中，T 表示基因尾部长度，H 表示基因头部长度，由式(6)可以推出：

$$D=T(h)n/2+H(h)k/2+k/2-1 \quad (7)$$

由式(1)知 $T(h)\sim O(h)=O(n/k)$，故有 $D\sim O(n^2/k)$，即 $\max(D(Am))\sim O(n^2/k)$.

例 2. 根据引理 2～4，在 $n=24$ 和 $n=100$ 情况下，三种算法的表达空间比较如表 2，表 3 所示.

为方便比较，表 2，表 3 均特别设置基因头部和尾部长度相等，在实际情况中通常不会出现.

表 2 $n=24$ 时表达空间比较

	染色体个数	染色体长度合计	总基因个数	基因长度	表达空间
M-GEP	2	24	6	4	47
多基因 GEP	1	24	6	4	29
单基因 GEP	1	24	1	24	24

表 3 $n=100$ 时表达空间比较

	染色体个数	染色体长度合计	总基因个数	基因长度	表达空间
M-GEP	2	100	10	10	279
多基因 GEP	1	100	10	10	109
单基因 GEP	1	100	1	100	100

由引理 2～4 和例 2 可以看出，在仅有两条染色体的情况下，M-GEP 算法的表达空间就已远远大于单基因和多基因 GEP 算法，这是 M-GEP 算法的一个重要特征.

7 实验结果分析

实验对三种不同类型的 GEP 程序进行了比较，并就 M-GEP 参数对进化的影响做了相应实验. 各类 GEP 算法使用的主要参数如表 4 所示.

表 4 算法参数

	最大进化辈数	种群大小	训练集数	运算符集合	染色体数	基因连接符	基因数量	头部大小	总长度	变异率	单点重组率	两点重组率	基因重组率	基因位移率	位移率	根位移率	Pc率	Pr率	精度
M-GEP	5000	60	20	+−*/	3	+	(3,3,1)	(3,3,4)	51	0.044	0.3	0.2	0.05	0.05	0.1	0.1	0.1	0.1	0.001
多基因 GEP	5000	60	20	+−*/	1	+	3	8	51	0.044	0.4	0.2	0.1	0.1	0.1	0.1	—	—	0.001
单基因 GEP	5000	60	20	+−*/	1		1	25	51	0.044	0.4	0.2	0.1	0.1	0.1	0.1	—	—	0.001

实验 1. 一元函数的发现问题，根据以下公式：

$$Y=X^6-2\times X^4+X^2 \quad (6)$$

随机产生 20 个实数，数据范围为 $[-3,3]$，$M=10000$，其它参数如表 4 所示. 对每个算法运行 100 次，结果如表 5 所示.

表 5 实验 1 对比结果

	成功率(%)	平均成功辈数	最长成功辈数	最短成功辈数
M-GEP	99	342	2681	7
多基因 GEP	89	811	3908	21
单基因 GEP	98	614	4737	30

由实验结果可以看出，在一元函数问题的发现中，M-GEP 算法明显优于多基因和单基因 GEP，平均成功辈数仅为后者的 42%和 56%. 函数发现的成功率也明显高于后者.

实验 2. 二元函数的发现问题，根据以下公式：

$$Z=X^3-X^2\times Y+Y \quad (7)$$

随机产生 20 个实数，数据范围为 $[-10,10]$，$M=10000$，其它参数如表 4 所示. 对每个算法运行 100 次，结果如表 6 所示.

在二元函数问题的发现中，M-GEP 平均成功辈数仅为多基因和单基因 GEP 的 61%和 29%.

表 6 实验 2 对比结果

	成功率(%)	平均成功辈数	最长成功辈数	最短成功辈数
M-GEP	100	32	115	2
多基因 GEP	100	52	270	4
单基因 GEP	100	110	3301	4

实验 3. 测试基因个数对 M-GEP 算法的影响. 针对公式 6 和实验 1 中的训练数据，分别将 M-GEP 第 2 个染色体的基因数据设置为 1～8，分别做 30 次测试，结果如图 4 所示，其中横坐标为基因个数，纵坐标为进化辈数.

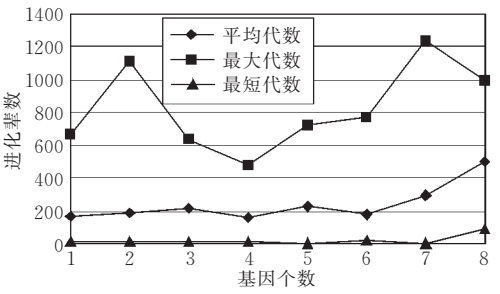

图 4 不同基因个数与进化辈数比较

通过图 4 可以发现，基因数目在 4 个左右时进化结果最好，当基因数过大或过小时效果都较差.

实验 4. 实验染色体算子对 M-GEP 算法的影

响. 去掉 Pc，Pr 算子后与实验 3 的平均进化辈数结果对比如图 5 所示.

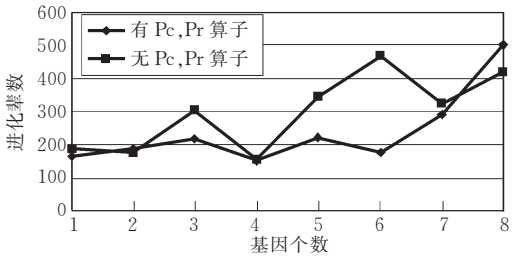

图 5　Pc，Pr 算子分析

由图 5 可以看出，去掉 Pc，Pr 算子后，平均进化辈数有明显增加，证明了这两个算子在遗传进化中具有明显的作用.

实验 5.　以 2 位乘法器为例，比较三种算法在电路进化[8,9]中的差别. 实验中将运算符集合修改为“ANOX”，即“AND，NOT，OR，XOR”，多基因连接符设为“OR”. 2 位乘法器有 2×2 个电路输入，4 个电路输出，分别设为 $P0$，$P1$，$P2$，$P3$. 实验结果中的进化辈数等于依次进化 $P0$，$P1$，$P2$，$P3$ 后累计的辈数.

具体实验结果如表 7 所示.

表 7　2 位乘法器电路进化比较

	成功率(%)	平均辈数	最长辈数	最短辈数
M-GEP	100	346	889	34
多基因 GEP	51	3922	6162	123
单基因 GEP	100	425	1103	40

可以看到 M-GEP 算法和单基因 GEP 算法在 2 位乘法器电路进化中表现都不错，成功率均为 100%，而多基因 GEP 算法成功率却较低，通过分析具体进化过程，发现问题出在进化 $P2$ 电路的输出上. 主要原因估计是对多基因的连接运算符设置，因为在计算染色体最终表达式时，强制将 4 个基因的计算结果使用“OR”连接不太符合电路设计的实际情况.

比较 M-GEP 算法和单基因 GEP 算法的平均进化辈数，可以看出 M-GEP 还是有明显优势，仅为单基因 GEP 算法的 81%.

8　结论及下一步的工作

本文提出了一种新的遗传进化算法：M-GEP，该算法在原有基因表达式编程基础上引入了多层染色体的概念，上级染色体对下级染色体基因具有重复调用的功能. 另外依据多染色体的理论，本文提出并实现了基于染色体的重组算子和基因随机重组算子. M-GEP 算法在相同染色体长度下极大地扩展了问题的表达空间，同时问题的关键部分能以基因的形式得以继承. 实验证明 M-GEP 算法比单基因 GEP 和多基因 GEP 算法性能有明显提高.

下一步作者将考虑如下研究工作：

(1)通过实验发现 M-GEP 各项参数的设置对进化辈数和成功率影响巨大，因此拟研究各项算子和参数对进化的影响规律及如何自动发现最优的参数组合.

(2)通过分析 M-GEP 算法实际进化过程，作者发现问题的关键组成部分通常以基因的方式继承到了下一代，很明显这样对提高进化效率、减少遗传辈数具有极大的帮助，下一步拟研究如何自动发现良序基因，并将其注入染色体.

(3)考虑将该算法应用于一些具体问题的解决中，如流动人口变动规律研究[10,11]、中药的复方配伍规律研究等等.

参　考　文　献

1　Ferreira C.. Gene expression programming: A new adaptive algorithm for solving problems. Complex Systems, 2001, 13 (2): 87～129

2　Ferreira C.. Gene Expression Programming. First Edition. Portugal: Angra do Heroismo, 2002

3　Ferreira C.. Gene expression programming in problem solving. In: Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications, 2001, 635～654

4　Ferreira C.. Discovery of the Boolean functions to the best density-classification rules using gene expression programming. In: Lutton E. *et al*. eds.. Proceedings of the 4th European Conference on Genetic Programming. Lecture Notes in Computer Science 2278. Berlin: Springer-Verlag, 2002, 51～60

5　Ferreira C.. Analyzing the founder effect in simulated evolutionary processes using gene expression programming. In: Abraham A., Ruiz-del-Solar J., Kpen M. eds.. Soft Computing Systems: Design, Management and Applications. Netherlands: IOS Press, 2002,153～162

6　Ferreira C.. Function finding and the creation of numerical constants in gene expression programming. In: Benitez J. M. *et al*. eds.. Advances in Soft Computing: Engineering Design and Maufacturing. Springer-Verlag, 2003, 257～266

7　Zuo Jie, Tang Chang-Jie, Zhang Tian-Qing. Mining predicate association rule by gene expression programming. In: Meng Xiao-Feng, Su Jian-Wen, Wang Yu-Jun eds.. Proceedings of the International Conference for Web Information Age 2002. Lecture Notes in Computer Science 2419. Berling Heidelberg:

Springer-Verlag, 2002, 92～103

8 De Garis H. Evolvable hardware: The genetic programming of Darwin machines. In: Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria, 1993, 441～449

9 Zhao Shu-Guan, Yang Wan-Hai. A novel approach for evolutionary design of circuits based on typical topology and adaptive genetic operations. Journal of Circuits and System, 2003, 8(2): 113～115(in Chinese)
(赵曙光,杨万海. 基于典型结构的电路自适应进化设计新方法. 电路与系统学报,2003,8(2):113～115)

10 Peng Jing, Tang Chang-Jie, Li Chuan, Chen An-Long, Hu Jian-Jun. New replication architecture for distributed databases based on UD-Tree. Mini-Micro Systems, 2004, 25(12): 2065～2069(in Chinese)
(彭 京,唐常杰,李 川,陈安龙,胡建军. 一种基于 UD-Tree 的分布式数据库新型复制架构. 小型微型计算机系统,2004,25(12): 2065～2089)

11 Peng Jing, Tang Chang-Jie, Hu Jian-Jun, Chen An-Long, Li Chuan. DIRM: A model for data query based on dynamic information route approach. Journal of Sichuan University, Engineering Science Edition, 2005, 37(1): 108～115(in Chinese)
(彭 京,唐常杰,胡建军,陈安龙,李川. DIRM:基于动态信息路由的数据检索模型. 四川大学学报(工科版), 2005, 37(1): 108～115)

PENG Jing, born in 1973, Ph. D.. His research interests include data mining and evolutionary computing.

TANG Chang-Jie, born in 1946, professor, Ph. D. supervisor. His main research interests include data mining and knowledge discovery.

LI Chuan, born in 1978, Ph. D.. His research interests include database theory and data mining.

HU Jian-Jun, born in 1970, Ph. D.. His research interests include database theory and data mining.

Background

This research belong to the National Science Foundation of China project, "The Research on Key Techniques in Knowledge Discovery Based on Gene Expression Programming" (60073046), and is also supported by Sichuan Major Science and Technology Project, "The Outliner Floating Population Analyses Based on Knowledge Discovery "(04SG1640). The projects are mainly focus on research of key techniques based on Gene Expression Programming (GEP). GEP is a new technique in knowledge discovering which borrow idea from the creature evolution rules of gene expression. Comparing the traditional evolution algorithm: GA (Genetic Algorithm) and GP(Genetic Programming) with GEP, authors present following key points: (1) GEP combines the advantage of GA and GP, GEP is 100-60000 times fast than GA or GP. (2) GEP has an inborn advantage to weaken the combination explosion. (3) Gene expression programming does not need the hypothesis, but trusts 'truth is inside the training data'.

This paper proposes a new evolution algorithm: M-GEP, which is based on the new concept of the multi-layer chromosome in GEP. The algorithm is efficient in the real applications, such as function discovery, electronic circuit evolution, etc.