基于通用多核处理器平台的并行基因表达式编程算法

吴 江[1] 唐常杰[2] 李太勇[1] 姜 玥[3] 李自力[1] 刘洋洋[1]
(西南财经大学经济信息工程学院 成都 610074)[1] (四川大学计算机学院 成都 610064)[2]
(西南民族大学计算机科学与技术学院 成都 610041)[3]

摘 要 基因表达式编程(Gene Expression Programming,GEP)是一种计算量大且通用性强的新型进化算法,其传统计算形式不能充分利用目前主流的多核处理器。为提高算法效率,提出了基于通用多核处理器平台的并行基因表达式编程算法(Parallel Gene Expression Programming Based on General Multi-core Processor,PGEP-MP)。主要工作包括:(1)分析通用多核处理器平台下并行基因表达式编程算法的机理;(2)利用MPI和OpenMP混合编程模型设计基于通用多核处理器平台的基因表达式编程算法的粗粒度与细粒度相结合的并行模型;(3)提出改进PGEP-MP算法效率的进化策略;(4)通过对函数挖掘和分类的实验证明,PGEP-MP算法提高了函数挖掘和分类的效率,在并行双核处理器数为4的情况下,PGEP-MP的平均并行加速比分别是传统GEP算法的4.22倍和4.06倍。

关键词 基因表达式编程,多核处理器,并行,进化算法
中图法分类号 TP301.6,TP18 **文献标识码** A

Parallel Gene Expression Programming Based on General Multi-core Processor

WU Jiang[1] TANG Chang-jie[2] LI Tai-yong[1] JIANG Yue[3] LI Zi-li[1] LIU Yang-yang[1]
(School of Economic Information Engineering,Southwestern University of Finance and Economics,Chengdu 610074,China)[1]
(School of Computer Science,Sichuan University,Chengdu 610064,China)[2]
(School of Computer Science and Technology,Southwest University for Nationalities,Chengdu 610041,China)[3]

Abstract Gene Expression Programming(GEP) is a new versatile evolution algorithm with huge calculation. The conventional GEP cannot take advantage of current popular multi-core processors. In order to improve the efficiency of GEP, parallel Gene Expression Programming based on general multi-core processor(PGEP-MP) was proposed. The main contributions include:(1) the mechanism of parallel GEP based on general multi-core processor is analyzed;(2) the parallel model of GEP based on general multi-core processor combined with coarse-grained and fine-grained levels is designed by the combination of MPI and OpenMP;(3) evolution strategies to improve PGEP-MP are proposed;(4) experiments on function mining and classification show that PGEP-MP improves the efficiency of function mining and classification. Compared with conventional GEP, the mean parallel speedup ratio of PGEP-MP are 4.22 and 4.02 times while the number of parallel dual core processors is 4.

Keywords Gene expression programming(GEP), Multi-core processor, Parallel, Evolution algorithm

1 引言

基因表达式编程(Gene Expression Programming,GEP)[1]是2001年12月由Candida Ferreira在遗传算法(Genetic Algorithm,GA)和遗传编程(Genetic Programming,GP)的基础上发展的新演化算法。GEP同传统的GA和GP在一些主要步骤上很相似,但在个体的编码方法及结果的表现形式等方面又有明显的区别。Candida在文献[2]中指出GEP比GA和GP快2～4个数量级。由于无须事先假定拟解决问题的任何先验模型和其很高的演化效率及精度,GEP算法被广泛应用于函数发现、关联规则、分类、聚类和多模函数优化等重要问题。

近年来,随着求解问题的规模不断扩大,对进化算法的求解质量和速度提出了更高要求。在多核处理器已成为主流处理器的背景下,并行进化算法和基于多核处理器的并行程序设计成为研究热点。目前,利用PVM或MPI进行遗传算法的并行化研究已取得一定成果,但对并行GEP算法的研究却相对匮乏。由于GEP算法在寻找全局最优解的过程中,需要进行全局搜索,随着求解问题的规模不断扩大,计算量相应增大,因此在多核处理器取代单核处理器成为主流处理器的背景下,如何充分利用通用多核处理器的并行计算能力,设计基于通用多核处理器平台的并行GEP算法,以提高算法的演化

到稿日期:2010-12-28 返修日期:2011-03-21 本文受国家自然科学基金(60773169),"十一五"国家科技支撑计划(2006BAI05A01),西南财经大学"211工程"三期青年教师成长项目(211QN09071)资助。

吴 江(1980－),男,博士,讲师,主要研究方向为数据库与知识工程,E-mail:wuj_t@swufe.edu.cn;唐常杰(1946－),男,教授,博士生导师,主要研究方向为数据库与知识工程;李太勇(1979－),男,博士,讲师,主要研究方向为数据库与知识工程;姜 玥(1978－),女,博士,副教授,主要研究方向为数据库与知识工程;李自力(1963－),男,副教授,主要研究方向为算法设计;刘洋洋(1981－),男,助教,主要研究方向为演化计算。

速度和精度，具有重要的理论和现实意义。

本文提出了基于通用多核处理器平台的并行基因表达式编程算法（Parallel Gene Expression Programming Based on General Multi-core Processor，PGEP-MP）。主要工作包括：(1)分析通用多核处理器平台下并行 GEP 算法的机理；(2)利用 MPI 和 OpenMP 混合编程模型设计基于通用多核处理器平台的 GEP 算法的粗粒度与细粒度相结合的并行模型；(3)提出改进 PGEP-MP 算法效率的进化策略；(4)通过对函数挖掘和分类的实验证明，PGEP-MP 提高了函数挖掘和分类的效率，在并行双核处理器数为 4 的情况下，平均并行加速比分别是传统 GEP 算法的 4.22 倍和 4.06 倍。

2 相关工作

GEP 算法的研究主要集中在理论和应用两方面。文献[3]提出了基于差分进化的 GEP 算法的全局优化算法，提高了算法的进化效率。文献[4]将 GEP 算法用于代价敏感的分类，拓宽了其在分类中的应用。文献[5]将 GEP 算法用于类对比函数挖掘。文献[6]利用 GEP 算法进行递归函数挖掘，拓宽了函数挖掘的范围。文献[7]提出了具有线性复杂度的 GEP 适应度评价算法。文献[8]将重叠基因概念引入 GEP 算法，对基于重叠表达的多基因进化算法进行了研究。为产生优势初始种群，文献[9]提出了基于基因空间均匀分布的种群初始化策略。

在并行进化算法及基于多核处理器的并行程序设计的研究中，文献[10]验证了多核处理器上多线程运算效率的提升。文献[11]引入改进的模拟退火操作，构造了一个兼顾全局搜索与局部探测的混合并行遗传算法。文献[12]提出了粗粒度并行 GEP 算法中的参数调节方法。文献[13]应用粗粒度模型设计了基于并行 GEP 算法的网格资源分配算法。

可见，在多核处理器已成为主流处理器的今天，基于多核处理器平台的并行遗传算法的研究却相对较少，而基于多核处理器平台的并行 GEP 算法目前更无研究报道。本文第 3 节对 GEP 算法进行了简介；第 4 节介绍了 MPI 和 OpenMP；第 5 节给出整个算法的模型框架；第 6 节提出改进算法效率的策略；第 7 节给出在模拟数据和真实数据上的实验结果。

3 GEP 算法简介

在 GEP 算法中，个体又称为染色体，染色体由基因通过连接运算符连接组成。基因由头和尾组成，头部由函数符（运算符或其他初等函数）或者终结符（变量或常量）组成，尾部由终结符组成。头部和尾部满足以下关系：

$$|\text{tail}|=|\text{head}|\times(n-1)+1 \quad (1)$$

式中，n 代表函数符集中参数的最大个数，例如运算符“+”带两个参数。

GEP 中的染色体编码方法可以避免在遗传操作中产生大量无效结构，大大提高了算法的效率。其编码方法为：将表达式根据其运算法则表示为表达式树（Expression Tree，ET），然后广度优先遍历该 ET，得到的符号序列即为个体编码的有效部分，称为 K 表达式（K-expression）。反之，将 K 表达式按以上过程的逆过程进行解码可得出对应的数学表达式。例如，对于表达式$(x^2+y^2)/(x+y)$，其对应的表达式树如图 1 所示。对图 1 采取从左到右的广度优先遍历，即得到该表达式对应的个体编码序列为“/++**xyxxyy”。同样，将该个体编码序列按以上过程的逆过程进行解码可得出对应的数学表达式。

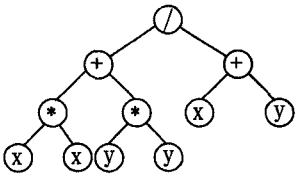


图 1 表达式树

若干个体构成整个种群。GEP 算法模拟自然界中的生物进化，按照“物竞天择，适者生存”的原则对种群实施若干次 GEP 遗传操作（选择、交叉、变异、转置和插串等），使种群一代代地进化，从中寻求出最优的个体，从而得到问题的最终解。GEP 算法的进化流程如算法 1 所示。

算法 1 GEP(Cases,GEP-Parameters)

```
/*Cases 为样本数据;GEP-Parameters 为 GEP 控制参数*/
1. Pretreat(Cases);   //样本数据预处理
2. S=Initial_Population();   //建立初始化种群
3. Best_Exp=NULL;//初始化最优解
4. Evaluation(S);//适应度评价
5. while (Iteration does not finish && Best_Exp does not reach pre-
      defined accuracy)
6.    S=Selection(S);//选择
7.    S=Mutation(S);//变异
8.    S=Invertion(S);//转置
9.    S=Transposition(S);//插串
10.   S=Recombination(S);//交叉
11.   Evaluation(S);//适应度评价
12.   Preserve(Best_Exp);//最优解保留策略
13. endwhile
14. return(Best_Exp);//返回最优解
```

4 MPI 和 OpenMP

MPI[14]（Message Passing Interface）是一种为编写消息传递程序而开发的广泛使用的标准，是消息传递并行程序设计的标准之一。但在很多情况下，采用纯的消息传递编程模式，并不能在多处理器构成的集群上取得理想的性能，原因在于没有充分利用每个处理器平台自身的处理能力。为了结合分布式内存结构和共享式内存结构两者的优势，使用分布式/共享内存层次结构作为并行算法的并行平台，应在使用 MPI 的同时结合 OpenMP。

OpenMP(Open Multi-Processing)是可移植多线程应用程序开发的行业标准，在细粒度与粗粒度线程技术上具有很高的效率。对于将串行应用程序转换成并行应用程序，OpenMP 指令是一种容易使用且功能强大的工具，具有使应用程序在对称多处理器或多核系统上并行执行而获得性能大幅提升的潜力[15]。OpenMP 采用 Fork/Join 并行执行模型。进程在 # pragma omp parallel 编译制导所标示的区域内产生线程级的并行，而在区域之外仍然是单线程。OpenMP 模型如图 2 所示。

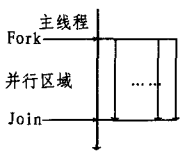


图 2 OpenMP 模型

MPI+OpenMP 混合编程模型实现了分布式/共享内存层次结构，提供了结点间和结点内的两级并行。MPI 解决处理器间的粗粒度通信，而 OpenMP 提供轻量级线程，很好地解决了每个处理器内部各内核间的并行计算。可见，MPI+OpenMP 混合模型是一种层次模型：MPI 位于顶层；OpenMP 位于底层。MPI+OpenMP 混合编程模型如图 3 所示。

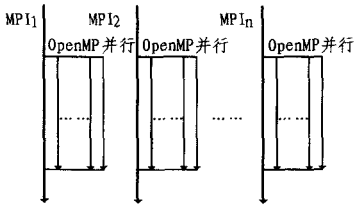


图 3　MPI+OpenMP 混合编程模型

本文利用 MPI 和 OpenMP 来设计基于通用多核处理器平台的并行 GEP 算法：在外部处理器之间，采用 MPI 作为并行的实现机制；在每个处理器内部的内核之间，采用 OpenMP 作为并行实现机制，充分利用多核处理器的性能，整体提高并行 GEP 算法的进化效率。

5　PGEP-MP

5.1　GEP 算法内部的并行性

在 GEP 算法的进化过程中，内部具有天然的并行性，包括：(1) 各种 GEP 遗传操作(选择、交叉、变异、转置和插串)的并行性；(2) 个体适应度评价的并行性。

由于这些操作属于轻量型运行级别，且内部不存在数据的相关性，因此这样的并行形式非常适合内存共享型的多核处理器计算环境。利用 OpenMP 实现内核间并行的基因表达式编程算法(Parallel Gene Expression Programming based on multi-core，PGEP-MC)的框架如图 4 所示。

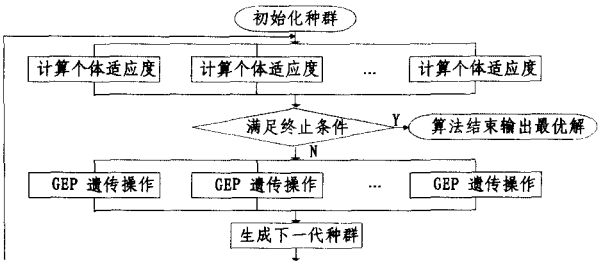


图 4　内核间并行的基因表达式编程算法框架

在该并行框架中，算法的遗传操作和适应度评价都由 c 个线程来执行。c 的大小可以根据处理器内核的数量动态决定，每个线程被一个内核执行。具体算法如算法 2 所示。

算法 2　PGEP-MC(Cases，GEP-Parameters)

/ * Cases 为样本数据；GEP-Parameters 为 GEP 控制参数 * /

1. Pretreat(Cases)；//样本数据预处理
2. S=Initial_Population()；//建立初始化种群
3. Best_Exp=NULL；//初始化最优解
4. # progma omp parallel；{ //根据处理器内核数，利用 OpenMP 产生线程级并行
5. Evaluation(S)；} //适应度评价
6. while(Iteration does not finish && Best_Exp does not reach predefined accuracy)
7. # progma omp parallel；{ //根据处理器内核数，利用 OpenMP 产生线程级并行
8. S=Selection(S)；//选择
9. S=Mutation(S)；//变异
10. S=Invertion(S)；//插串
11. S=Transposition(S)；//移植
12. S=Recombination(S)；//交叉
13. Evaluation(S)；} //适应度评价
14. Preserve(Best_Exp)；//最优个体保留策略
15. endwhile
16. return(Best_Exp)；//返回最优解

PGEP-MC 算法使传统 GEP 在多核处理器系统上并行执行，大幅提高了演化性能。并且随着未来 CPU 内核数的增加，能很方便地对算法进行扩展，无需修改程序。

5.2　GEP 算法间的并行性

GEP 算法的种群可分为若干子种群，子种群间具有天然的并行性，非常适合在计算机间并行执行。并行 GEP 算法将计算机间的高速并行性和 GEP 算法的天然并行性相结合，并行处理的引入不仅提高了求解速度，而且由于种群规模的扩大和各子种群的隔离和迁移，使种群的多样性得以丰富和保持，减少了未成熟收敛的可能性，提高了算法的效率和求解质量。

一般并行进化算法分为 4 类：全局并行算法、粗粒度并行算法、细粒度并行算法和混合并行算法，这里不再赘述。本文在子种群的并行上，利用 MPI 实现适应性最强和应用最广的粗粒度分布式并行模型：将种群按照节点机(每个节点机包含一个多核处理器)的个数分成若干个子种群，各子种群在各自的节点机的多核处理器上并行地运行由 OpenMP 实现的 PGEP-MC 算法，每经过一定的进化代，各子种群间将交换若干个个体，一方面用来引入更优良的个体，另一方面丰富了种群的多样性，防止未成熟早收敛现象。

5.3　PGEP-MP

利用 MPI+OpenMP 所设计的基于通用多核处理器平台的并行基因表达式编程算法(PGEP-MP)的总框架如图 5 所示。

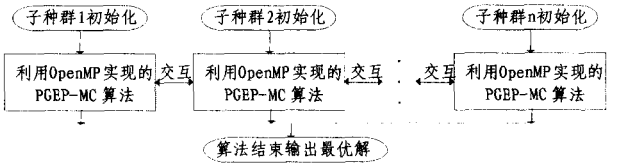


图 5　基于通用多核处理器平台的并行基因表达式编程算法的框架

下面描述 PGEP-MP 算法，即算法 3。

算法 3　PGEP-MP(Cases，GEP-Parameters，Parallel-Parameters)

/ * Cases 为样本数据；GEP-Parameters 为 GEP 控制参数；Parallel-Parameters 为并行控制参数 * /

1. MPI_CreateThread()；//初始化生成 N 个进程，每个处理器一个进程
2. Read (Cases，GEP-Parameters，Parallel-Parameters)；
//处理器 0 读入数据文件和各参数
3. Send (Cases，GEP-Parameters，Parallel-Parameters)；
//将这些数据发送到其他各进程；
4. while (Iteration does not finish && Best_Exp does not reach predefined accuracy)
5. PGEP-MC()；//各个处理器并行进行演化过程：在各自的进化过程中调用算法 2
6. Migration()；//按照设定的迁移率和迁移周期执行迁移选择；
7. endwhile

8. return(Best_Exp);//返回最优解

PGEP-MP 算法扩展较好，只需在程序开始运行时初始化进程个数以分配给每个处理器并行执行，无需修改程序。

5.4 迁移操作

PGEP-MP 算法涉及如下迁移操作问题：

(1) 迁移拓扑

迁移拓扑确定子种群之间个体的迁移路径。基本的 PGEP-MP 算法采用如图 6 所示的单向环模型的迁移拓扑。

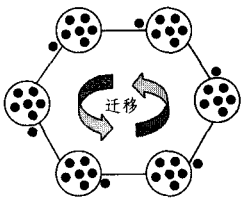


图 6 单向环迁移拓扑模型

(2) 迁移规模

迁移率：每次迁移时迁移的个体的数目。迁移率过大会破坏子种群的稳定性；迁移率过小，将使得子种群不能充分利用其他子种群的信息。

迁移周期：决定各种群间的迁移间隔。迁移间隔过大，会出现早熟收敛现象；间隔过小，会导致在可接受的时间内得不到高质量的解[16]。

(3) 迁移策略

迁移选择：选出迁移个体的策略。

迁移替换：把最差或者有限数目的最差个体替换掉。

6 提升 PGEP-MP 效率的策略

6.1 提高子种群自身的进化效率

在整个并行进化算法中，提高子种群各自的进化效率对提高整个并行算法的进化效率至关重要。将提高 GEP 算法的优化策略引入并行算法的各子种群的进化中，理论上可以提高整个并行 GEP 算法的进化效率。

6.2 子种群差异进化

借鉴文献[17]，将子种群差异进化策略应用于 PGEP-MP。将子种群分为探索型和精耕型两种。探索子种群的 GEP 遗传参数值较大，增大了探索到最优个体的可能性；精耕子种群的 GEP 遗传参数值较小，更易保持个体的稳定性，可在局部范围内寻找优秀个体，并尽可能保存下来。子种群差异进化的拓扑结构如图 7 所示。

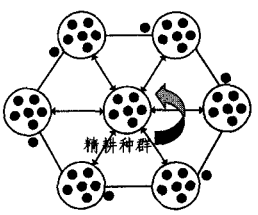


图 7 子种群差异进化的迁移拓扑模型

精耕子种群只有一个，负责局部搜索全局最优解；探索子种群分为若干个，在大空间内为精耕子种群提供候选个体。子种群差异进化的迁移拓扑模型能避免早熟，提高算法的进化效率。

6.3 自适应的迁移策略

在种群的迁移中，迁移方向和迁移率是影响并行算法效率的关键因素之一。合适的迁移方向能有效提高整个种群并行的进化效率。而对迁移率来说，如果过大会破坏子种群的稳定性；迁移率过小，将使得子种群不能充分利用其他子种群的信息。为此，借鉴文献[18]的自适应思想，设计了一种自适应的迁移策略，包括自适应的迁移方向和迁移率。设相邻子种群 i 和 j 中个体的最高适应度是 fit_i 和 fit_j。$\Delta fit=fit_i-fit_j$。迁移方向由 Δfit 的符号确定，如果 $\Delta fit>0$，由种群 i 迁移到 j；否则，则由种群 j 迁移到 i。迁移率由下式决定：

$$\lambda=\left\lceil\frac{\Delta fit}{\max(fit_i,fit_j)}\times N\right\rceil \tag{2}$$

式中，N 为子种群大小。

7 实验与性能分析

7.1 实验环境

实验环境是由 8 台 Intel 双核 E3300 处理器 PC 通过 100Mbps 以太网互联而成的分布式并行环境，操作系统为 Windows XP。

7.2 实验数据和参数设置

(1) 函数挖掘

函数挖掘是数据挖掘中的重要内容，选取下列函数集进行函数挖掘实验。以下 3 个函数分别代表了不同类型的函数，常用于测试 GEP 挖掘性能，具有代表性和可比性。

$$F1=\pi r^2 \tag{3}$$

$$F2=5x^4+4x^3+3x^2+2x+1 \tag{4}$$

$$F3=\frac{(1.0+x_1^2+x_2^2)^2}{(\sqrt{x_1^2+x_2^2}+1.5)} \tag{5}$$

在实验中，对每个函数的每个自变量分别随机产生 1000、10000 和 100000 个[0.00，100.00]之间的随机数作为训练数据集的参数值，然后分别求出以上 3 个函数对应的目标值。对每个数据集重复 100 次挖掘实验，最后取统计结果的平均值作为最后的实验结果。算法中的基本原始参数设置见表 1。设置函数 $F1$、$F2$ 和 $F3$ 挖掘成功的最低相对误差为 0.0028、0 和 0.005。

(2) 分类

分类也是数据挖掘中的重要研究内容。训练分类器的实验数据来自 UCI 机器学习数据库中的 5 个数据集[19]。数据集的基本情况和最后训练得到的分类器拟达到的分类精度见表 2。算法中的基本原始参数设置见表 1。

表 1 PGEP-MP 算法的基本参数

参数	数值	参数	数值	参数	数值
子种群大小	50	基因个数	3	基因连接符	+
基因头长	11	染色体长	69	变异率	0.044
单点交叉率	0.3	两点交叉率	0.3	基因交叉率	0.3
转置率	0.1	RIS 插串率	0.1	IS 插串率	0.1
迁移周期	50	迁移率	1	最大进化代数	3000

表 2 UCI 数据集合和拟达到的分类精度

数据集	记录数	分类数	属性个数		训练分类器拟达到的精度
			连续	离散	
Credit	690	2	6	9	84.1
Diabetes	768	2	8	—	75.9
House-votes-84	435	2	—	16	94.7
Sick	3772	2	6	21	94.3
Sonar	208	2	60	—	82.9

7.3 评价指标

采用并行加速比 S_p 作为评价指标，并行加速比 S_p 的定

义如下：

$$S_p=\frac{\overline{T_l}}{\overline{T_p}} \tag{6}$$

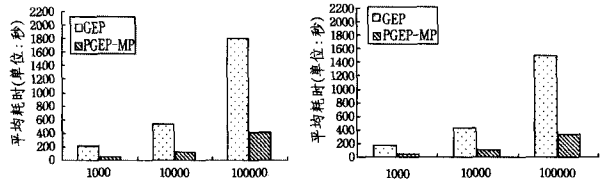
式中，$\overline{T_l}$为进化成功时串行平均运算时间；$\overline{T_p}$为进化成功时并行平均运算时间。

7.4 实验结果

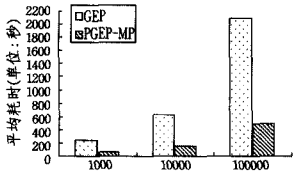
7.4.1 函数挖掘

(1) 并行加速比分析

当并行处理器数为 4 时，传统 GEP 算法和 PGEP-MP 算法挖掘成功的平均进化时间如图 8 所示。图 8 表明，在并行处理器数为 4 的情况下，对于不同大小的数据集，PGEP-MP 算法的平均耗时明显少于 GEP 算法。在数据集大小为 1000、10000 和 100000，成功挖掘到目标函数的情况下，PGEP-MP 算法的并行加速比是传统 GEP 算法的 4.14、4.19 和 4.32 倍，平均为 4.22 倍。这是因为传统的 GEP 算法在计算每个个体的适应度和进行各种遗传操作时，只能串行计算，而 PGEP-MP 充分利用了 OpenMP 技术，使得个体的适应度和各种遗传操作在处理器的各内核间并行处理，发挥了多核处理器的优势，提高了算法效率。另一方面，由于利用 MPI，各处理器间的子种群并行演化，并且执行了个体迁移，进行了信息的交换，因此减少了整体算法陷入局部最优的可能性，有效地提高了算法的整体效率。



(a) 函数 F1 挖掘成功的平均进化时间比较　(b) 函数 F2 挖掘成功的平均进化时间比较



(c) 函数 F3 挖掘成功的平均进化时间比较

图 8　函数挖掘成功的平均进化时间比较

(2) 处理器数与 PGEP-MP 算法效率分析

表 3 显示了 3 个函数在不同大小的训练数据集上，随着并行节点(每个节点包含一个多核处理器)个数的增加，PGEP-MP 算法的平均并行加速比的变化情况。随着并行节点个数的增加，PGEP-MP 算法的并行加速比也随之提高，不过随着并行节点个数的增加，PGEP-MP 算法的并行加速比并没有显著呈线性增加。原因在于过多的并行节点在进行演化时，某些节点的演化会重复，这样就使得 PGEP-MP 算法的并行加速比不能呈线性增加。因此，过多地选择并行的节点数并不能线性地提高 PGEP-MP 算法的效率。

表 3　PGEP-MP 算法的并行加速比

	处理器数量			
	2	4	6	8
F1	2.25	4.21	6.12	7.51
F2	2.32	4.20	6.18	7.42
F3	2.18	4.27	5.94	7.43

(3) 改进策略性能提升分析

采用 6.2 节和 6.3 节提出的策略对 PGEP-MP 算法的迁移拓扑和迁移策略进行改进。在并行处理器数为 4 的情况下，PGEP-MP 与改进后的 PGEP-MP 在 3 个函数上的平均并行加速比的对比如表 4 所列。表 4 表明了 6.2 节和 6.3 节提出的策略有效地提高了 PGEP-MP 算法的演化效率，证明了该改进策略的有效性。

表 4　改进后的 PGEP-MP 算法的并行加速比

	PGEP-MP	改进后的 PGEP-MP
F1	4.21	4.36
F2	4.20	4.29
F3	4.27	4.61

(4) 并行开销分析

首先，将传统 GEP 算法与在一个双核处理器上运行的 PGEP-MP 算法进行比较，此 PGEP-MP 算法没有使用 MPI 机制进行处理器间的并行，仅采用 OpenMP 开启一个线程进行演化。挖掘函数 $F1$ 的实验结果如表 5 所列。

表 5　OpenMP 开销(单位：秒)

训练数据集大小	GEP	PGEP-MP′	开销	开销比例
1000	210	225	15	6.67%
10000	532	550	18	3.27%
100000	1801	1831	30	1.64%

注：PGEP-MP′指在一个双核处理器上运行的 PGEP-MP 算法，此算法仅采用 OpenMP 开启一个线程进行演化

其次，将传统 GEP 算法与在一个双核处理器上运行的 PGEP-MP 算法进行比较，此 PGEP-MP 算法没有使用 OpenMP 机制进行内核间的并行，仅仅采用 MPI 机制进行处理器间的并行演化。由于只有一个处理器，事实上也并没有进行多个处理器间的并行，此实验旨在对 MPI 机制的开销进行分析。在函数 $F1$ 上的实验结果如表 6 所列。

表 6　MPI 开销(单位：秒)

训练数据集大小	GEP	PGEP-MP″	开销	开销比例
1000	210	223	13	5.83%
10000	532	562	30	5.34%
100000	1801	1898	97	5.11%

注：PGEP-MP″指在一个双核处理器上运行的 PGEP-MP 算法，此算法仅采用 MPI 进行处理器间并行演化

表 5 表明随着训练数据的增大，OpenMP 并行开销所占程序的总开销的比率逐渐减小，总体来说，OpenMP 并行开销所占比例较小。表 6 表明 MPI 并行开销所占程序的总开销的比率较为固定，总体来说，MPI 并行开销所占比例较小。

7.4.2 分类

(1) 并行加速比分析

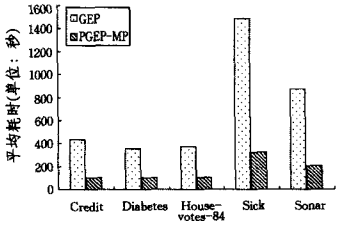


图 9　分类器演化的平均进化时间比较

当并行处理器数为 4 时，传统 GEP 算法和 PGEP-MP 算法训练分类器达到预先设定的精度的平均进化时间如图 9 所

示。图 9 表明，在并行处理器数为 4 的情况下，对于不同分类器训练数据集，PGEP-MP 算法的平均耗时明显优于 GEP 算法。对 5 个分类训练集，在成功挖掘满足预先设定精度的分类器的情况下，PGEP-MP 算法的并行加速比是传统 GEP 算法的 4.24、3.48、3.60、4.61 和 4.36 倍，平均为 4.06 倍。

(2) 处理器数与 PGEP-MP 算法效率分析

表 7 显示了在 5 个分类训练集上时，PGEP-MP 算法的平均并行加速比随并行节点个数的增加而变化的情况。随着并行节点个数的增加，PGEP-MP 算法的并行加速比也随之提高，不过随着并行节点个数的增加，PGEP-MP 算法的并行加速比同样没有显著呈线性增加。

表 7 PGEP-MP 算法的并行加速比

	处理器数量			
	2	4	6	8
Credit	2.33	4.24	6.14	7.69
Diabetes	2.05	3.48	5.04	6.89
House-votes-84	1.98	3.60	5.96	6.78
Sick	2.69	4.61	6.03	7.93
Sonar	2.27	4.36	6.07	7.81

(3) 改进策略性能提升分析

采用 6.2 节和 6.3 节提出的策略对 PGEP-MP 算法的迁移拓扑和迁移策略进行改进。在并行处理器数为 4 的情况下，PGEP-MP 与改进后的 PGEP-MP 在 5 个分类器训练数据集上的并行加速比的对比如表 8 所示。表 8 表明了采用了 6.2 节和 6.3 节提出的策略后，PGEP-MP 算法有效地提高了并行加速比，证明了该改进策略的有效性。

表 8 改进后的 PGEP-MP 算法的并行加速比

	PGEP-MP	改进后的 PGEP-MP
Credit	4.24	4.41
Diabetes	3.48	3.86
House-votes-84	3.60	3.81
Sick	4.61	4.72
Sonar	4.36	4.43

(4) 并行开销分析

首先，将传统 GEP 算法与在一个双核处理器上运行的 PGEP-MP 算法进行比较，此 PGEP-MP 算法没有使用 MPI 进行处理器间的并行，仅采用 OpenMP 开启一个线程进行演化，结果如表 9 所列。

其次，将传统 GEP 算法与在一个双核处理器上运行的 PGEP-MP 算法进行比较，此 PGEP-MP 算法没有使用 OpenMP 进行内核间的并行，仅仅采用 MPI 进行处理器间的并行演化。由于只有一个处理器，事实上也并没有进行多个处理器间的并行，结果如表 10 所列。

表 9 OpenMP 开销(单位：秒)

	GEP	PGEP-MP′	开销	开销比例
Credit	432	457	25	5.47%
Diabetes	351	371	20	5.39%
House-votes-84	367	388	21	5.41%
Sick	1490	1539	49	3.18%
Sonar	876	919	43	4.68%

注：PGEP-MP′指在一个双核处理器上运行的 PGEP-MP 算法，此算法仅采用 OpenMP 开启一个线程进行演化

表 9 表明随着训练数据的增大，OpenMP 并行开销所占程序的总开销的比率逐渐减小，总体来说，OpenMP 并行开销所占比例较小。表 10 表明 MPI 并行开销所占程序的总开销的比率较为固定，总体来说，MPI 并行开销所占比例较小。

表 10 MPI 开销(单位：秒)

	GEP	PGEP-MP″	开销	开销比例
Credit	432	449	17	3.79%
Diabetes	351	367	16	4.36%
House-votes-84	367	383	16	4.18%
Sick	1490	1568	78	4.97%
Sonar	876	912	36	3.95%

注：PGEP-MP″指在一个双核处理器上运行的 PGEP-MP 算法，此算法仅采用 MPI 进行处理器间的并行演化

结束语 为提高 GEP 算法的求解质量和速度，利用 MPI 和 OpenMP 技术设计了基于通用多核处理器平台的并行 GEP 算法：PGEP-MP。该算法在每个处理器间，采用 MPI 作为并行实现机制；在每个处理器内部，采用 OpenMP 作为并行实现机制，充分利用多核处理器的性能。这种基于混合编程模型的并行 GEP 算法充分利用两种编程模式的优点：MPI 解决各处理器间的粗粒度通信；OpenMP 提供轻量级线程，解决每个多核处理器内核间的并行，最大限度地提高了并行效率。通过对函数挖掘和分类的实验证明，本文提出的 PGEP-MP 提高了函数挖掘和分类的效率，在并行双核处理器数为 4 的情况下，平均并行加速比分别是传统 GEP 算法的 4.22 倍和 4.06 倍。在未来的工作中，如何更加有效地降低并行过程中的额外开销以及改进迁移操作中的迁移拓扑、迁移规模和迁移策略，以提高并行 GEP 算法的效率，将是研究的重点。

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]. Portugal: Angra do Heroismo, 2002

[3] 李太勇，唐常杰，吴江，等. 基于差分进化基因表达式编程的全局函数优化[J]. 计算机科学，2009，36(11)：140-143

[4] 吴江，李太勇，刘洋洋. 基于基因表达式编程的代价敏感分类算法[J]. 吉林大学学报：信息科学版，2009，27(4)：418-423

[5] Duan Lei, Tang Chang-jie, Tang Liang, et al. Mining class contrast functions by gene expression programming[C]// Proc. of the 5th International Conference on Advanced Data Mining and Application. Beijing, Springer-Verlag, 2009, 5678: 116-127

[6] 吴江，唐常杰，姜玥，等. 基于基因表达式编程的递归函数挖掘[J]. 四川大学学报：工程科学版，2007，39(5)：127-131

[7] 陈瑜，唐常杰，李川，等. LDecode：具有线性复杂度的 GEP 适应度评价算法[J]. 四川大学学报：工程科学版，2008，40(1)：107-112

[8] 彭京，唐常杰，元昌安，等. 基于重叠表达的多基因进化算法[J]. 计算机学报，2007，30(5)：775-785

[9] 胡建军，唐常杰，段磊，等. 基因表达式编程初始种群的多样化策略[J]. 计算机学报，2007，30(2)：305-310

[10] 谷照升. 基于多核 CPU 的并行计算设计[J]. 长春工程学院学报：自然科学版，2009，10(3)：92-94

[11] 唐天兵，谢祥宏，韦凌云，等. 多核 CPU 环境下小生境混合遗传算法的研究[J]. 计算机应用研究，2009，26(11)：4073-4075

[12] Park H-H, Grings A, Santos M V D. Parallel hybrid evolutionary computation: automatic tuning of parameters for parallel gene expression programming[J]. Applied Mathematics and Computation, 2008, 201: 108-120

[13] 邓松，王汝传，张羽，等. 基于并行基因表达式编程的网格资源分配算法[J]. 电子学报，2009，37(2)：272-277

[14] 都志辉. 高性能计算并行编程技术——MPI 并行程序设计[M]. 北京：清华大学出版社，2001

[15] Cuvillo J D, Zhu W, Gao G R. Landing OpenMP on cyclops64: an efficient mapping of OpenMP to a Many-core System on a Chip [C]// Proceedings of CF'06. Ischia, Italy, 2006: 41-60

[16] 王小平，曹立明. 遗传算法：理论、应用与软件实现[M]. 西安：西安交通大学出版社，2002

[17] 王文义，秦广军，王若雨. 自适应的多种群并行遗传算法研究[J]. 计算机工程与应用，2006，15：78-81

[18] 赖鑫生，张明义. 基于渗透原理迁移策略的并行遗传算法[J]. 计算机学报，2005，28(7)：1146-1152

[19] Blake C L, Merz C J. UCI repository of machine learning databases[OL]. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Department of Information and Computer Sciences, 1998

(上接第 270 页)

[2] Lin Bo, Fang Bin, Li Dong-hui. Character Recognition of Licencense Palte Image Based on Multiple Classifiers[C]// Proceedings of the 2009 International Conference on Wavelet Analysis and Pattern Recognition. Baoding, July 2009: 12-15

[3] Hsu C-W, Lin C-J. A Comparison of Methods for Multiclass Support Vector Machines[J]. IEEE Transactions on Neural Networks, 2002, 13(2)

[4] Wang Shen-zheng, Lee H-J. A Cascade Framework for a Real-Time Statistical Plate Recognition System[J]. IEEE Transactions on Information Forensics and Security, 2007, 2

[5] Guo Jing-ming, Liu Yun-fu. License Plate Localization and Character Segmentation with Feedback Self-learning and Hybrid Binarization Techniques[J]. IEEE Transactions on Vehicular Technology, 2008, 57(3)

[6] 罗鑫，吴炜，杨晓敏，等. 一种基于 PCA 的多模板字符识别[J]. 电子测量技术，2007，30(1)

[7] 邹明明，卢迪. 基于改进模板匹配的车牌字符识别算法实现[J]. 研究与开发，2010，19(1)

[8] 甘龙，高隽，梁栋，等. 基于分形维数的车牌字符识别[J]. 中国公路学报，2002，15(4)

[9] Duda R O, Hart P E, Stork D G. Pattern Classification[M]. China Machine Press, 2009: 94-95

[10] Theodoridis S, Koutroumbas K. Pattern Recognition(Third Edition)[M]. Publishing House of Electronics Industry, 2008: 177-180

[11] Otsu N. A Threshold Selection Method from Gray-level Histograms[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1979, 9(1): 62-66

[12] 邹岚，张桂林. 主成分分析方法(PCA)在车辆牌照识别中的应用[J]. 计算机与数字工程，2007，35(3)

[13] 杨晓敏，周强，吴炜，等. 基于支持向量机的车牌字符识别[J]. 四川大学学报：自然科学版，2009，46(4)

(上接第 281 页)

表 2　推荐参数对应的识别率及消耗时间

	0(%)	1(%)	2(%)	3(%)	4(%)	5(%)	6(%)
$n_1=14$	100	90.00	100	96.67	100	100	93.33
$n_2=13$	93.33	76.67	66.67	70.00	93.33	96.67	100
(8,83)	100	100	100	100	95.00	100	100
(9,93)	100	95.00	100	100	100	100	100
(9,96)	100	100	100	100	95.00	100	100

	7(%)	8(%)	9(%)	识别率(%)	训练时间(s)	测试时间(s)
$n_1=14$	100	90.00	100	97.00	2.275	5.518
$n_2=13$	96.67	83.33	93.33	87.00	2.226	4.428
(8,83)	100	95.00	100	99.00	4.619	0.908
(9,93)	100	95.00	100	99.00	4.084	1.000
(9,96)	100	100	100	99.50	4.055	0.990

结束语　本文介绍了基于统计分析理论的手写数字识别，首先提出了基于投影间隔比率和间隔变化的特征提取方法，然后从信息论的角度分析特征提取，提出了基于旋转投影的识别方法。最后理论分析和实验证明了旋转投影在特征数量相同的情况下具有更高的识别率，并且解决了倾斜数字的识别问题。在今后的工作中，将对几何结构和统计分析相结合的数字识别方法作进一步研究。

参 考 文 献

[1] Gonzalez, Woods. Digital Image Processing[M]. New Jersey: Prentice Hall Publisher, 2008

[2] 求是科技. Visual C++数字图像处理典型算法及实现[M]. 北京：人民邮电出版社，2006

[3] 盛骤，谢式千. 概率论与数理统计(第 4 版)[M]. 北京：高等教育出版社，2008

[4] 付庆铃，韩力群. 基于人工神经网络的手写数字识别[J]. 北京工商大学学报：自然科学版，2004，122(13)：44-45

[5] 耿西伟，张猛，沈建京. 基于结构特征分类 BP 网络的手写数字识别[J]. 计算机技术与发展，2007，17(1)：130-132

[6] 罗成，孙越恒. 基于加强贝叶斯分类的手写数字识别[J]. 微处理机，2009(03)：77-79

[7] 宫淑兰. 手写数字识别的研究与应用[D]. 青岛：山东大学，2006

[8] Duda R O, Hart P E, Stork D G. Pattern Classification(Second Edition)[M]. New York: John Wiley & Sons, 2001

[9] Webb A R. Statistical Pattern Recognition(Paperback)(2nd Edition)[M]. New York: John Wiley & Sons, 2002

[10] Duda R O, Hart P E. Pattern Classification and Scene Analysis [M]. New York: John Wiley & Sons, 1973

[11] (美)Mitchell T. 机器学习[M]. 曾华军，张银奎，译. 北京：机械工业出版社，2003

[12] Bloch G, Lauer F, Ching Y, et al. A trainable feature extractor for handwritten digit recognition[J]. Pattren Recognition, 2007, 40