第37卷第9期 中国科学技术大学学报 Vol. 37, No. 9
2007年9月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Sep. 2007

文章编号:0253-2778(2007)09-1080-08

基因表达数据的频繁闭合模式挖掘新算法

缪裕青[1,2,3],陈国良[1,2],徐 云[1,2]

(1.中国科学技术大学计算机科学与技术系,安徽合肥 230027;2.国家高性能计算中心,安徽合肥 230027;
3.桂林电子科技大学计算机系,广西桂林 541004)

摘要:基因表达数据集与传统事务数据集相比呈现出新的特征,由于其项目数远远大于事务数,使得大量现有的基于项目枚举的频繁闭合模式挖掘算法不再适用.为此提出一种频繁闭合模式挖掘新算法TPclose,使用TP-树(tidset-prefix tree)保存项目的事务集信息.该算法将频繁闭合模式挖掘问题转换成频繁闭合事务集挖掘问题,采取自顶向下分而治之的事务搜索策略,并组合了高效的修剪技术和有效的优化技术.实验表明,TPclose算法普遍快于自底向上事务搜索算法RERⅡ,最高达2个数量级以上.

关键词:数据挖掘;关联规则;频繁闭合模式;基因表达数据;自顶向下

中图分类号:TP311 **文献标识码**:A

A new algorithm for mining frequent closed patterns in gene expression datasets

MIAO Yu-qing[1,2,3], CHEN Guo-liang[1,2], XU Yun[1,2]

(1. *Department of Computer Science and Technology, USTC, Hefei* 230027, *China*; 2. *NHPCC, Hefei* 230027, *China*;
3. *Department of Computer Science and Technology, Guilin University of Electronic Technology, Guilin* 541004, *China*)

Abstract: Unlike the traditional datasets, gene expression datasets typically contain a huge number of items and a few transactions. While there are large numbers of algorithms developed for frequent closed patterns mining, their running time increased exponentially with increasing average length of the transactions, thus such gene expression datasets render most current algorithms impractical. TPclose, a new efficient algorithm for mining frequent closed patterns from gene expression datasets was proposed. It stored the tidset of each item using a TP-tree (tidset-prefix tree). TPclose converted the problem of mining frequent closed patterns into one of mining frequent closed tidsets, adopting the top-down and divide-and-conquer search strategy to explore transaction enumeration search space and combining efficient pruning and effective optimizing. Several experiments on real-life gene expression datasets show that TPclose outperforms RERⅡ, an existing algorithm based on bottom-up search strategy, by up to two orders of magnitude.

Key words: data mining; association rules; frequent closed pattern; gene expression data; top-down

收稿日期:2006-04-30;**修回日期**:2006-10-08

基金项目:国家自然科学基金重点项目(60533020)资助.

作者简介:缪裕青,女,1966年生,博士/副教授.研究方向:数据挖掘,生物信息学. E-mail: myq@ustc.edu.cn

通讯作者:陈国良,男,中国科学院院士. E-mail: glchen@ustc.edu.cn

0 引言

随着微阵列等高通量检测技术的发展,人们积累的基因表达数据量大幅度增加,大量的基因表达数据中蕴含着丰富的基因活动信息,如何通过对基因表达数据的分析获取生物学知识是生物信息学面临的重大挑战之一[1]. 近些年,人们已研究通过对基因表达数据表达差异的显著性分析来识别与条件相关的特异性基因,通过对基因表达谱的聚类分析来预测未知基因的功能[2],通过对基因表达数据的分类分析来进行肿瘤分型的诊断等. 由于基因之间的相互作用关系比较复杂,上述这些分析并不能很好地揭示基因之间或基因与环境条件之间内在的生物关联关系. 对基因表达数据的关联规则分析能够获取这一关联信息,进而帮助诊断肿瘤等疾病和推断基因调控网络[3].

频繁闭合模式(frequent closed pattern)挖掘是关联规则挖掘和双聚类分析的基础. 在传统事务数据集上频繁闭合模式挖掘已得到广泛研究[4~7],但基因表达数据集与传统事务数据集相比呈现出新的特征. 基因表达数据集通常包含大量的列(基因)和少量的行(采样条件),列数远远多于行数. 这使得传统数据集上基于列(项目)枚举的数目随列数增加呈指数级增长的频繁闭合模式挖掘算法不再适用[8].

CARPENTER[8]和 RER Ⅱ[9]是近两年针对基因表达数据集提出的频繁闭合模式挖掘算法. CARPENTER 算法提出行枚举树(row enumeration tree)概念,即用行(事务)枚举空间替代传统的列(项目)枚举空间. 该算法通过递归扫描构造投影转置表自底向上(行组合由少到多)搜索行枚举空间. 当投影转置表较大时,该算法反复扫描转置表的代价是巨大的. RER Ⅱ算法在 CARPENTER 算法基础上,利用了求交集的简单快速特点,将 CARPENTER 算法中深度优先扫描构造投影转置表改为深度优先求行交集方式,在时间性能上有一定改进[9]. 尽管这两个算法与基于列枚举空间搜索算法相比有明显的优势[9],但所采取的自底向上行枚举空间搜索策略,由于其搜索过程中产生的模式支持度是从小到大变化的,因而无法充分利用最小支持度阈值进行搜索空间的修剪,从而造成计算时间和空间的浪费. 此外,当数据集较稠密时,自底向上搜索策略会遇到非常耗时的巨大计算问题.

本文针对基因表达数据集的特点和自底向上搜索策略的不足,提出了基于基因表达数据的频繁闭合模式挖掘新算法 TPclose(based on tidset-prefix tree mining frequent closed pattern). 该算法应用概念格理论,将频繁闭合模式挖掘问题转换成频繁闭合事务集(frequent closed tidset)挖掘问题,充分利用事务数远远小于项目数,计算和存储代价都较小的特点,采取自顶向下分而治之的行枚举空间搜索策略,并结合使用本文提出的 TP-树(tidset-prefix tree)结构、高效的修剪技术和有效的优化技术,获得了较好的结果.

1 问题描述

设 $I=\{i_1,i_2,\cdots,i_m\}$是项目(item)集合,I 的一个非空子集称为一个项目集(itemset),数据集 $D=\{T_1,T_2,\cdots,T_n\}$是事务(transaction)集合. 其中,事务 $T_k=(\mathrm{tid}_k,X_k)$,tid_k为事务标识符,X_k为集合 I 中由项目组成的子集,即 $X_k\subseteq I$. 假如某个项目集 $Y\subseteq X_k$,则称事务 T_k包含项目集 Y. 数据集 D 中包含项目集 Y 的事务数称为项目集 Y 在 D 中的支持数,用 $\sigma(Y)$表示. Y 的支持数在 D 中所占的比例称为项目集 Y 在 D 中的支持度(support),用 $\mathrm{Sup}(Y)$表示. 所有事务标识符组成集合 $T=\{\mathrm{tid}_1,\mathrm{tid}_2,\cdots,\mathrm{tid}_n\}$,$T$ 的一个非空子集称为一个事务集(tidset). 为了简便起见,在不引起混淆的情况下,项目集$\{a,e,h\}$将简写成"aeh",事务集$\{2,4\}$将简写成"24".

定义 1.1 给定数据集 D 和一个最小支持度阈值 minSup(minimum support threshold),对项目集 X,若 $\mathrm{Sup}(X)\geqslant\mathrm{minSup}$,则称 X 是频繁项目集(frequent itemset)或频繁模式(frequent pattern).

由于一旦求出项目集的支持数,就很容易求出其支持度,所以相对最小支持度阈值 minSup 可定义最小支持数阈值 $\min\sigma$.

定义 1.2 设项目集 X,若不存在任何真超集 Y,满足 $Y\supset X$,并使得 $\mathrm{Sup}(Y)=\mathrm{Sup}(X)$,则称 X 为闭合模式(closed pattern).

若项目集 X 既是频繁的又是闭合的,则称 X 是频繁闭合模式.

对于项目集 $Y\subseteq I$,其对应的事务集定义为包含 Y 的所有事务的标识符集合,用 $\mathcal{T}(Y)$表示. 对于事务集 $S\subseteq T$,其对应的项目集定义为 S 中所有事务共同包含的最大项目集,用 $I(S)$表示.

由这两个定义可推出以下性质和引理.

性质 1.1 设项目集 $X\subseteq I$,则 $\mathcal{T}(X)=\bigcap_{x\in X}\mathcal{T}(x)$.

性质 1.2 设事务集 $S\subseteq T$,则 $I(S)=\bigcap_{s\in S}I(s)$.

性质 1.3 设项目集 $X\subseteq I$,则 $X\subseteq I(\mathcal{T}(X))$.

性质 1.4 设事务集 $S\subseteq T$,则 $S\subseteq\mathcal{T}(I(S))$.

由性质 1.1 可知,对于项目集 $X,Y\subseteq I$,若 $X\subseteq Y$,则 $\mathcal{T}(X)\supseteq\mathcal{T}(Y)$;由性质 1.2 可知,对于事务集 $U,V\subseteq T$,若 $U\subseteq V$,则 $I(U)\supseteq I(V)$.

引理 1.1 设事务集 $U\subseteq T$,则 $I(U)$是一个闭合模式.

引理 1.2 设项目集 $X\subseteq I$,则 X 是一个闭合模式,当且仅当 $X=I(\mathcal{T}(X))$.

例 1.1 如图 1 所示,图 1(a)表示数据集 D 由 5 个事务、15 个项目组成. 图 1(b)表示由 D 得到的转置表 TT,即 D 中每个项目对应的事务集列表. 由图 1(a)可得到,$\mathcal{T}(aeh)=234$,表示项目集 aeh 对应的事务集为 234,即包含项目集 aeh 的事务有 2,3,4 三个事务. $I(24)=aehpr$,表示事务集 24 对应的项目集为 $aehpr$,即在 2,4 事务中共同包含的最大项目集为 $aehpr$. 此外,由图 1 还可得到,$\mathcal{T}(aeh)=\mathcal{T}(a)\bigcap\mathcal{T}(e)\bigcap\mathcal{T}(h)=1234\bigcap234\bigcap234=234$,$I(24)=I(2)\bigcap I(4)=adehlpr\bigcap aehpr=aehpr$.

tid_k	X_k
1	*a*, *b*, *c*, *l*, *o*, *s*
2	*a*, *d*, *e*, *h*, *l*, *p*, *r*
3	*a*, *c*, *e*, *h*, *o*, *q*, *t*
4	*a*, *e*, *h*, *p*, *r*
5	*b*, *d*, *f*, *g*, *l*, *q*, *s*, *t*

(a) 数据集D

i_j	$\mathcal{T}(i_j)$
a	1, 2, 3, 4
b	1, 5
c	1, 3
d	2, 5
e	2, 3, 4
f	5
g	5
h	2, 3, 4
l	1, 2, 5
o	1, 3
p	2, 4
q	3, 5
r	2, 4
s	1, 5
t	3, 5

(b) D的转置表TT

图 1 数据集及转置表

Fig. 1 Dataset and transposed table

很容易推出,$\sigma(X)=|\mathcal{T}(X)|$,即项目集的支持数就等于其对应事务集元素的个数. 例如,$\sigma(aeh)=|\mathcal{T}(aeh)|=|234|=3$,$\sigma(aehpr)=|\mathcal{T}(aehpr)|=|24|=2$. 因此,求项目集的支持数可转换为求项目集对应事务集元素的个数.

定义 1.3 设事务集 $U\subseteq T$,若 $|U|\geqslant\min\sigma$,则称 U 为频繁事务集;若 $U=\mathcal{T}(I(U))$,则称 U 为闭合事务集;若 U 既频繁又闭合,则称 U 为频繁闭合事务集.

由性质 1.3,1.4 和定义 1.3 可推出如下引理:

引理 1.3 设项目集 $X\subseteq I$,则 $\mathcal{T}(X)$是一个闭合事务集.

问题描述:给定数据集 $D=\{T_1,T_2,\cdots,T_n\}$是事务的集合. 其中,项目集合 $I=\{i_1,i_2,\cdots,i_m\}$,事务 $T_k=(\mathrm{tid}_k,X_k)$,tid_k为事务标识符,$X_k\subseteq I$,$n\ll m$. 对于用户给定的最小支持度阈值 minSup,列举出 D 中所有频繁闭合模式及其支持度.

2 TP-树与算法 TPclose

2.1 概念格

给定数据集 D,设 D 中所有闭合事务集构成集合 C^T,所有闭合模式构成集合 C^I,则称数据对(U,X)为一个概念(concept),其中,$U\in C^T$,$X\in C^I$,$U=\mathcal{T}(X)$,$X=I(U)$. 例如,数据集 D 如图 1 所示,则$(24,aehpr)$即是一个概念. 其中,24 是闭合事务集,$aehpr$ 是闭合模式,且 $24=\mathcal{T}(aehpr)$,$aehpr=I(24)$. 设 D 中所有的概念构成集合 C_f,在 C_f 上定义偏序关系 $\leqslant_f$:$(U_1,X_1)\leqslant_f(U_2,X_2)\Leftrightarrow U_1\subseteq U_2(X_2\subseteq X_1)$,则$\langle C_f,\leqslant_f\rangle$为一个概念格(concept lattice)或 Galois 格(Galois lattice)[10]. 图 2 为一个概念格的 Hasse 图,其中,数据集为图 1 中的 D.

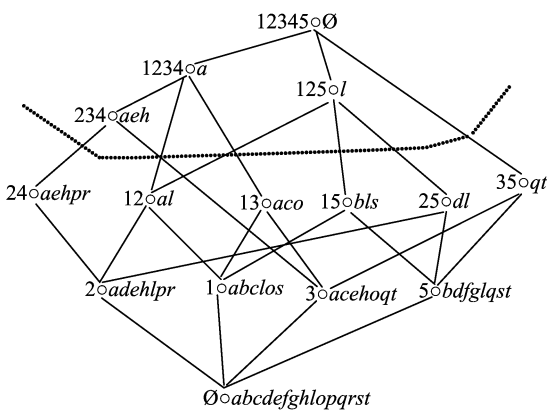

图 2 概念格的 Hasse 图

Fig. 2 The Hasse diagram of the concept lattice

在 C_f 上定义运算 $\vee$,$\wedge$:

$$\vee_{i=1}^{k}(U_i,X_i)=(\mathcal{T}(I(\bigcup_{i=1}^{k}U_i)),\bigcap_{i=1}^{k}X_i),$$

$$\wedge_{i=1}^{k}(U_i,X_i)=(\bigcap_{i=1}^{k}U_i,I(\mathcal{T}(\bigcup_{i=1}^{k}X_i))).$$

如图 2 所示,$(24,aehpr)\vee(12,al)=(\mathcal{T}(I(124)),aehpr\bigcap al)=(1234,a)$,$(24,aehpr)\wedge(12,al)=(24\bigcap12,I(\mathcal{T}(aehlpr)))=(2,adehlpr)$.

观察图 2 中的概念格可发现,要找出数据集中

所有的闭合模式，既可在概念格中通过自底向上概念求“∨”运算得到，也可通过自顶向下概念求“∧”运算先找到全部闭合事务集后，再通过映射 I 求出所有的闭合模式. 如图 2 所示，当 $\min\sigma=3$ 时，含频繁闭合模式的所有概念均在虚线上方，虚线下方是不含频繁闭合模式的概念. 显然，当数据集中项目数较多且最小支持度阈值较高时，在概念格虚线下方的非频繁闭合模式往往很多且长度较长，这样自底向上策略就会由于先找出不频繁闭合模式而花费过多的计算和存储代价. 相比之下，自顶向下策略是寻找频繁闭合事务集，当数据集中的事务数远远小于项目数时，频繁闭合事务集长度较短，因而所花费的计算和存储代价也较小，TPclose 算法正是基于以上思想设计的.

2.2　TP-树与等价类

为了高效搜索闭合事务集，我们提出了一种新的 tidset-prefix tree(TP-树)结构，以压缩存储数据集中每个项目的事务集信息. TP-树是一棵 tid 前缀树，当 tid 按一定次序排序时，多个项目的事务集共享公共的 tid 前缀. 为了方便树的遍历，构造一个 tid 头指针表(header table)，表中的 tid 在树中的所有出现都通过一个边链连接起来，边链的头指针则保存在头指针表中.

TP 树由一个根节点“root”、root 的子树集合以及一个 tid 头指针表组成. 子树中每个节点由三个域组成(tid，level，side_link)，其中，tid 表示该节点所代表的事务标识符，level 表示该节点在 TP 树中的层数，side_link 指向边链中下一个节点. 定义根节点所在的层数为 0，其孩子节点的层数为 1，孙子结点的层数为 2，依次类推. 树中事务标识符相同的节点按照层数升序用边链连起来，头指针表中的 tid 通过头指针(head)指向其边链的第一个节点.

TP-树构造方法：给定数据集 TT，首先计算其各事务所包含的频繁项目个数，根据这个数降序排序事务标识符，然后将 TT 表中每个项目的事务集都按照这一次序依次插入到 TP-树中. 每个项目事务集的插入都是从 TP-树的根节点开始. 假设要插入事务标识符 u，(Ⅰ)判断根节点的孩子节点中是否包含 u；(Ⅱ)若没有包含 u，则将 u 作为新孩子节点插入到根节点下，u 节点的层数为根节点层数加 1，并根据这一层数将 u 节点插入到事务标识符 u 的边链中，然后再将 u 作为新子树的根节点继续下一个要插入事务标识符的判断；(Ⅲ)若已包含 u，则直接将 u 节点作为新子树根节点继续下一个要插入事务标识符的判断；(Ⅳ)依次下去，直到所有项目的事务集都插入到 TP-树中为止. 图 3 是由图 1 生成的 TP-树，树中每个节点的第一个数字表示该节点所代表的事务标识符，括号中的数字表示该节点的层数，虚线箭头表示边链.

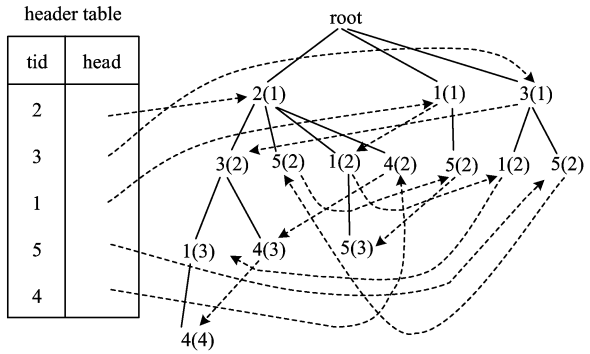

图 3　TP-树

Fig. 3　TP-tree

给定事务标识符集合 T，当定义事务标识符的某种次序(<)后，集合 T 为一个序列. 设 $U\subseteq T$，定义函数 $p(U,k)$ 为取 U 中含 k 个标识符的前缀，定义函数 $s(U,k)$ 为取 U 中含 k 个标识符的后缀，$0\leqslant k\leqslant|U|$.

在 TP-树中，从根节点到达每个节点的路径都可形成一个事务标识符序列，该序列实际上是序列 T 的一个子序列. 设树中所有这样形成的事务标识符序列构成集合 $\mathscr{R}$，在 $\mathscr{R}$ 上定义等价关系 θ：$\forall U,V\in\mathscr{R},U\theta V\Leftrightarrow s(U,1)=s(V,1)$. 也就是说，如果两个事务集等价，则它们的最后一个事务标识符相同. $\mathscr{R}$ 关于 θ 的等价类可惟一确定 $\mathscr{R}$ 的划分，以事务标识符 r 为生成元的等价类简记为$[r]$. 例如，由图 3 的 TP-树可得到 $\mathscr{R}$ 的划分：$\pi_\theta(\mathscr{R})=\{[2],[3],[1],[5],[4]\}$. 其中，$[2]=\{2\}$，$[3]=\{3,23\}$，$[1]=\{1,21,31,231\}$，$[5]=\{35,25,15,215\}$，$[4]=\{24,234,2314\}$.

设事务标识符为 r，等价类$[r]$中元素 U 分别与类中其他元素通过交运算(∩)生成的元素称为 U 的孩子，U 的孩子再通过两两交运算生成的元素称为 U 的孙子，以此类推. 等价类$[r]$中所有元素的后代统称为$[r]$的生成类.

引理 2.1　设 $U,V\in\mathscr{R}$，若 $W=U\cap V$，其中，$U\in[u]$；$V\in[v]$；$u,v\in T$，$u\neq v$，则 W 一定属于 $\mathscr{R}$ 的某个等价类或其生成类.

证明 (Ⅰ)假设 $u<v$ 且 $U\subset V$,则 $W=U\cap V=U$,故 W 属于等价类$[u]$.

(Ⅱ)假设 $u<v$ 且 $U\not\subset V$. 设 $w=s[W:1]$,因为 $W=U\cap V$,所以 $w\in U$ 且 $w\in V$. 由 U 是从根节点到节点 u 形成的事务集可知,w 为此路径上的一个节点,由 $\mathcal{R}$ 集合的组成可知,从根节点到达节点 w 时可得到一事务集 $M\in\mathcal{R}$,且 $M\subset U$,$M\in[w]$. 同理,V 是从根节点到节点 v 形成的事务集,w 也为此路径上的一个节点,所以从根节点到达节点 w 时也得到一事务集 $N\in\mathcal{R}$,且 $N\subset V$,$N\in[w]$. 因此,$W=U\cap V=M\cap N$,故 W 属于等价类$[w]$或其生成类.

由 TP-树的构造可知,对于任意项目 $x\in I$,$\mathcal{T}(x)$均包含在 $\mathcal{R}$ 中. 因此,由性质 1.1 和引理 2.1 可得到:

推论 2.1 设项目集 $X\subseteq I$,若 X 是闭合模式,则 $\mathcal{T}(X)$一定属于 $\mathcal{R}$ 的某个等价类或其生成类.

由于对于任意项目 $x\in I$,$\mathcal{R}$ 不但包含了 $\mathcal{T}(x)$还包含了 $p(\mathcal{T}(x),k)$,其中,$k=\{1,2,\cdots,|\mathcal{T}(x)|-1\}$,所以,由推论 2.1 和引理 1.3 可知,$\mathcal{R}$ 的等价类或其生成类中的元素或者是闭合事务集或者是闭合事务集的前缀.

2.3 TPclose 算法

TPclose 算法基本思想是:首先由 TT 表构造 TP-树,然后按照头指针表自上而下的顺序沿着 tid 边链搜索 TP-树得到集合 $\mathcal{R}$ 的各个等价类;再采取分而治之的策略对各个等价类中的元素进行深度优先求交集,得到全部频繁闭合事务集;最后再通过映射 I 求出全部的频繁闭合模式. 算法描述见算法 2.1.

算法 2.1 Algorithm TPclose(TT,minσ)
输入:数据集 D 的转置表 TT,minσ;
输出:所有频繁闭合模式及其支持度 FCP;
步骤:

步 1 构造 TP-树;
步 2 按照自上而下的次序,从头指针表中第 minσ 个事务标识符开始搜索边链; //使用修剪技术 1
步 3 找到此事务标识符在 TP-树中层数≥minσ 的各个节点,从根节点到达这些节点形成的事务标识符序列构成以此事务标识符为生成元的等价类 N; //使用修剪技术 2
步 4 调用 subTPclose(N); //寻找频繁闭合模式
步 5 搜索头指针表中的下一个事务标识符边链,当头指针表中最后一个事务标识符的边链都已搜索过,结束;否则返回步 3;

```
procedure SubTPclose(N)//深度优先搜索频繁事务集,N 初始为 TP-树产生的等价类
begin
  for each node n_i in N
    if |n_i|>minσ then      //使用修剪技术 3,4
      for each node n_j in N,其中 n_i <_α n_j 且 |n_j|>minσ,
      <_α为 N 中元素生成的先后次序
        n'=n_i∩n_j;
        if |n'|≥minσ then      //使用修剪技术 4
          if n_i=n_j then 从 N 中删除 n_j;
          if n_i⊃n_j then 将 n_j加到 N'中且从 N 中删除 n_j;
              //N'存放 n_i的孩子
          if n_i≠n_j and n' is not discovered before then 将
          n'加到 N'中;      //冗余修剪
        endif
      endfor
      if N'≠Ø then subTPclose(N');endif
      GFCP(n_i);      //调用子程序求 I(n_i)
    else if n_i is not discovered before then GFCP(n_i);//冗余修剪
  endfor
end

procedure GFCP(U)      //求频繁事务集 U 的频繁闭合模式 I(U),并将 U,I(U)及其支持数保存到 FCP 中
begin
  for FCP 中每个元素
    if FCP 中存在 U 的前缀
      then 利用优化技术求 I(U);
        if I(U)已存在 then 用新的支持数替代旧的支持数且用 U 替代其前缀,return;endif
      else 直接求 I(U);
    endif
  endfor
  保存 U,I(U)及其支持数|U|到 FCP 中;
end
```

在 TPclose 算法中,通过支持度修剪、冗余修剪等技术对搜索空间进行高效修剪,支持度修剪技术具体体现在四个方面:

(Ⅰ)在头指针表中,按照自上而下的次序搜索事务标识符的边链. 当事务标识符的秩小于最小支持数阈值时,在 TP-树中由根节点到达该事务标识

符节点构成的事务标识符序列不可能是频繁事务集,因此不需要搜索此事务标识符边链.

(Ⅱ) 在 TP-树中,当节点的层数小于最小支持数阈值时,从根节点到达该节点构成的事务标识符序列不可能是频繁事务集,因此不需要搜索此分枝.

(Ⅲ) 当等价类(或其生成类)中某个元素的长度等于最小支持数阈值时,与其交集不可能产生新的频繁事务集,因此,该元素的后代不需要再搜索.

(Ⅳ) 当等价类的生成类中某个元素的长度小于最小支持数阈值时,该元素不可能是频繁事务集,因此,该元素及其后代均不需要再搜索.

引理 2.2　冗余修剪. 当等价类的生成类中某个新生成的元素已被搜索过时,该元素及其后代均不需要再搜索.

证明　在 TPclose 算法中,因为等价类的生成类中的元素是从等价类中的元素开始,按照深度优先方式两两求交集生成的,所以,等价类的生成类中的元素可看作是等价类中某些元素取交集. 设等价类$[r]=\{r_1,r_2,\cdots,r_i,\cdots,r_j,\cdots,r_k,\cdots,r_m,\cdots,r_s,\cdots,r_n\}$有 n 个元素,$[r]$的生成类中有 $A<_\alpha C<_\alpha D$,其中,$<_\alpha$是用于标识生成类中元素的生成次序的符号,D 是 C 的兄弟,$C=r_k\cap R_c\cap r_m$,$D=r_k\cap R_c\cap r_s$,$A=r_i\cap R_a\cap r_j$,R_c 和 R_a 是$[r]$中多个元素的交集,且 $C=A$. 当求 C 的孩子 $V=C\cap D$ 时,由算法可知,$V=r_k\cap R_c\cap r_m\cap r_s$,且在 A 的兄弟中也一定存在一个元素 B,使 $B=r_i\cap R_a\cap r_s$,使得 A 的孩子 $U=A\cap B=r_i\cap R_a\cap r_j\cap r_s=r_k\cap R_c\cap r_m\cap r_s=V$. 这就说明,当等价类的生成类中某个新生成的元素在此之前已被搜索过时,它的后代也已被搜索过,所以该元素及其后代均不需要再搜索.

为了快速由 I 映射求出全部频繁闭合模式,TPclose 算法使用了一项优化技术,该技术采取的是基于搜索头指针表自上而下次序,该次序保证了在产生各等价类时,一条事务标识符序列的前缀一定是在该事务标识符序列之前产生. 设执行 TPclose 算法最后求出的频繁事务集集合为 S,$<_\beta$是用于标识 S 中元素求映射 I 的次序的符号,则有以下引理:

引理 2.3　设$\forall U_i,U_j\in S,U_i<_\beta U_j$. 若 $I(U_i)=I(U_j)$,则 U_i 是 U_j 的前缀.

证明　由 TPclose 算法的修剪技术可知,S 中包含所有非冗余的频繁闭合事务集或其前缀. 根据引理 1.1 和定义 1.3 可知,对于事务集 $U,V\subseteq T$,若 U,V 为闭合事务集,则 $U=V$,当且仅当 $I(U)=I(V)$,因此,$I(U_i)=I(U_j)$说明 U_i,U_j 不可能同时为频繁闭合事务集.

(Ⅰ) 设 U_i 是某个频繁闭合事务集的前缀,$U_j=\mathcal{T}(X)$,其中,$X\subseteq I$. 因为 $I(U_i)=I(U_j)$,则 $I(U_i)=I(\mathcal{T}(X))$,由此可推知 $\mathcal{T}(I(U_i))=\mathcal{T}(I(\mathcal{T}(X)))$. 根据引理 1.3 知,$\mathcal{T}(I(U_i))=\mathcal{T}(X)$,根据性质 1.4 可知,$U_i\subseteq\mathcal{T}(I(U_i))$,所以 $U_i\subseteq U_j$. 因此 U_i 也是 U_j 的前缀.

(Ⅱ) 设 U_i 是 $\mathcal{T}(X)$的前缀,U_j 是 $\mathcal{T}(Y)$的前缀,其中,$X,Y\subseteq I$. 因为 $I(U_i)\supseteq I(\mathcal{T}(X))$,根据引理 1.3 有 $\mathcal{T}(I(U_i))\subseteq\mathcal{T}(I(\mathcal{T}(X)))=\mathcal{T}(X)$;又因为已知 $I(U_i)=I(U_j)$,所以 $\mathcal{T}(I(U_j))\subseteq\mathcal{T}(X)$. 又由性质 1.4 知,$U_j\subseteq\mathcal{T}(I(U_j))$,于是 $U_j\subseteq\mathcal{T}(X)$,因此可推知 U_j 也是 $\mathcal{T}(X)$的前缀;因为已知 $U_i<_\beta U_j$,所以 U_i 是 U_j 的前缀.

利用以上引理可得到下列优化技术. 例如求 $I(2314)$,因为 $I(2314)=I(231)\cap I(4)$,而 231 是 2314 的前缀,$I(231)$在求 $I(2314)$之前已求出,所以只需将 $I(231)$与 $I(4)$求交集就可求出 $I(2314)$,而无需再求 $I(2)\cap I(3)\cap I(1)\cap I(4)$,这显然可大大减少交运算的次数,同时通过判断是否满足 $I(2314)=I(231)$,也可避免产生冗余.

例 2.1　给定数据集 D,如图 1 所示,$\min\sigma=2$,构造如图 3 所示的 TP-树. 按照 TPclose 算法有:

(Ⅰ) 利用支持度修剪技术搜索 tid=3 的边链,得到等价类[3]中的频繁事务集 23,求 $I(23)=aeh$ 保存到 FCP[1]中.

(Ⅱ) 搜索 tid=1 的边链,得到[1]中的频繁事务集集合{21,31,231},因为元素 21 和 31 的长度均等于 $\min\sigma$,所以根据支持度修剪技术,不需要与其他元素求交集,直接求 $I(21)=al$ 保存到 FCP[2]中,$I(31)=aco$ 保存到 FCP[3]中. 因为元素 23 是 231 的前缀,且 $I(23)$已保存在 FCP[1]中,所以可利用优化技术先求 $I(231)=I(23)\cap I(1)=a$,然后保存到FCP[4]中.

(Ⅲ) 同样,搜索 tid=5 的边链,得到[5]={35,25,15,215},直接将求得的 $I(35)=qt$ 保存到 FCP[5]中,$I(25)=dl$ 保存到 FCP[6]中,$I(15)=bls$ 保存到 FCP[7]中. 元素 21 是 215 的前缀,且 $I(21)$已保存在FCP[2]中,于是将求得的 $I(215)=$

$I(21)\cap I(5)=l$ 保存到 FCP[8]中.

(Ⅳ) 搜索 tid=4 的边链,得到[4]={24,234,2314},直接求 $I(24)=aehpr$ 并保存到FCP[9]中. 元素 234 与 2314 求交集仍为 234,由于 23 是 234 的前缀,且 $I(23)$已保存在 FCP[1]中,所以求 $I(234)=I(23)\cap I(4)=aeh=I(23)$,于是用 234 替代 23,支持数 3 替代 2 仍保存在 FCP[1]中. 同理,由于元素 231 是 2314 的前缀,且 $I(231)$已保存在 FCP[4]中,因此求 $I(2314)=I(231)\cap I(4)=a=I(231)$后,用 2314 替代 231,支持数 4 替代 3 仍保存在 FCP[4]中. 至此,头指针表已搜索完毕,挖掘结束.

定理 2.1 TPclose 算法仅列举了所有频繁闭合模式.

证明 由推论 2.1 和引理 1.2 可知,所有闭合模式可由 TPclose 算法列举出. 由引理 2.2 和引理 2.3 可知,TPclose 算法列举出的仅是非冗余的闭合模式. 在求闭合模式过程中,算法通过修剪技术将那些不可能产生频繁事务集的搜索空间剪去,仅列举了频繁事务集. 因此,根据引理 1.1 和定义 1.3 可知,TPclose 算法仅列举了所有频繁闭合模式.

3 性能评估

评估 TPclose 算法性能的所有实验都是在 PC Pentium Ⅲ上执行的,操作系统为 Windows XP,使用 C++编程. 实验使用了真实数据集 BCR_ABL[11]和 ALL_AML[12],在 BCR_ABL 数据集中有 15 个组织采样,每个采样中描述了12 625个基因的活动水平. 在 ALL_AML 数据集中有 38 个组织采样,每个采样中描述了5 000个基因的活动水平. 数据集 BCR_ABL 的离散处理直接采用 $P/A/M$ 值,P 表示基因有表达,A,M 看作基因没有表达. 数据集 ALL_AML 的离散处理是将第一列样本作为参照样本,取比率的对数,然后对每一行取平均值,各值再与平均值比较,大于 1 表示基因呈高表达(highly expressed),小于 −1 表示基因呈高抑制(highly repressed). 假设基因表达值是平均值的 2 倍或 1/2 时为显著表达,这样每个基因就用两个标识符表示,一个标识符代表基因的高表达,另一个代表基因的高抑制.

图 4 显示了在两个数据集上两种算法运行时间的实验结果. 从图中可看出,TPclose 普遍比 RERⅡ快. 在图 4(a)中,当最小支持数不小于 9 时,TPclose 比 RERⅡ快 2~6 倍;在图 4(b)中,当最小支持数不小于 17 时,TPclose 比 RERⅡ快 2 个数量级以上;当最小支持数小于 17 时,由于 RERⅡ运行时间太长(两个小时以上),所以没有运行完. 表 1、表 2 列举了两个数据集在不同最小支持度下挖掘出的频繁闭合模式数量,从数量上可看出两个数据集都较稠密. 在实验中,我们也观察了内存使用情况,通常 TPclose 比 RERⅡ所占内存少.

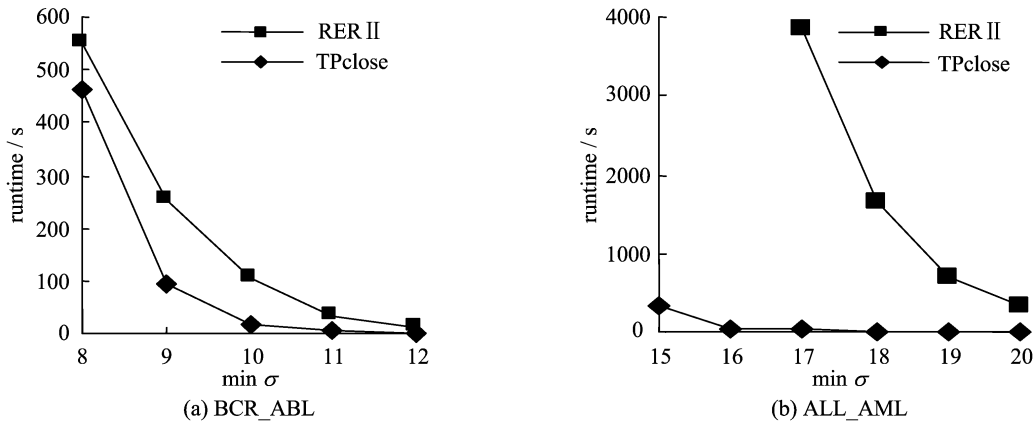

图 4 算法运行时间比较

Fig. 4 Runtimes of different algorithms

表 1 BCR_ABL 的频繁闭合模式数

Tab. 1 The number of frequent closed pattern in BCR_ABL

最小支持数(度)	8(53%)	9(60%)	10(67%)	11(73%)	12(80%)
频繁闭合模式数	16 216	9 787	4 809	1 856	540

表 2　ALL_AML 的频繁闭合模式数

Tab. 2　The number of frequent closed pattern in ALL_AML

最小支持数(度)	15(41%)	16(43%)	17(46%)	18(49%)	19(51%)	20(54%)
频繁闭合模式数	18 771	8 954	4 308	2 153	1 091	548

4　结论

本文针对基因表达数据集呈现出的新特点,提出了在基因表达数据集上挖掘频繁闭合模式的新算法 TPclose. 该算法应用概念格理论,借助本文提出的一种新的 TP-树结构,将频繁闭合模式挖掘问题转换成频繁闭合事务集挖掘问题,通过采取自顶向下分而治之的行枚举空间搜索策略,既利用了事务数远远小于项目数,求事务集交集计算与存储开销都较小的特点,又充分利用了最小支持度阈值对搜索空间进行修剪. 实验表明,TPclose 算法采取的策略、修剪技术和优化技术都是有效的,算法获得了较高的性能.

由于基因表达数据集较少的事务数和大量的项目数,造成这种数据集往往是稠密的且模式较长,因此当最小支持度阈值较低时,数量巨大的频繁闭合模式,会带来高昂的计算和存储开销. 利用并行计算机强大的计算和存储能力来解决频繁闭合模式挖掘问题将是我们下一步研究的内容.

参考文献(References)

[1] 孙啸,陆祖宏,谢建明. 生物信息学基础[M]. 北京:清华大学出版社,2005:282-314.

[2] Madeira S C, Oliveira A L. Biclustering algorithm for biological data analysis: a survey [J]. ACM Transactions on Computational Biology and Bioinformatics, 2004, 1(1):24-45.

[3] Creighton C, Hanash S. Mining gene expression databases for association rules [J]. Bioinformatics, 2003, 19(1):79-86.

[4] Han J W, Pei J, Yin Y W. Mining frequent patterns without candidate generation[C]//Proc. of 19th ACM SIGMOD Int'l Conf. on Management of Data. Dallas: ACM Press, 2000:1-12.

[5] Pasquier N, Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules[C]// Proc. of the 7th Int'l Conf. on Database Theory. Jerusalem: Springer-Verlag, 1999:398-416.

[6] Zaki M J, Hsiao C J. CHARM: An efficient algorithm for closed itemset mining[C]//Proc. of the 2nd SIAM Int'l Conf. on Data Mining. Arlington, 2002:12-28.

[7] Liu J Q, Sun X Y, Zhuang Y T, et al. Mining frequent closed patterns by adaptive pruning [J]. Journal of Software, 2004, 15(1):94-102.
刘君强,孙晓莹,庄越挺,等. 挖掘闭合模式的高性能算法[J]. 软件学报,2004,15(1): 94-102.

[8] Pan F, Cong G, Tung A, et al. CARPENTER: finding closed patterns in long biological datasets[C]// SIGKDD'03. Washington: ACM Press, 2003: 637-642.

[9] Cong G, Tan K L, Tung A, et al. Mining frequent closed patterns in microarray data[C]//Proc. of the 4th IEEE Int'l Conf. on Data Mining. 2004, 4: 363-366.

[10] Valtchev P, Missaoui R, Godin R. Formal concept analysis for knowledge discovery and data mining: the new challenges [C]//Proc. of ICFCA'04. 2004: 352-371.

[11] Supplemental data for classification, subtype discovery, and prediction of outcome in pediatric lymphoblastic leukemia by gene expression profiling [EB/OL]. http://www.stjuderesearch.org/data/ALL1/all_datafiles.html.

[12] Cancer program data sets [EB/OL]. http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi.