挖掘空间关联规则的前缀树算法设计与实现

刘君强[1),2)] 潘云鹤[1)]

[1)](浙江大学人工智能研究所，杭州 310027) [2)](杭州商学院计算机信息工程学院，杭州 310035)

摘 要 空间关联规则挖掘是在空间数据库中进行知识发现的一类重要问题.为此提出了挖掘空间关联规则的二阶段策略，通过多轮次单层布尔型关联规则挖掘，自顶向下逐步细化空间谓词的粒度，从而空间谓词的计算量大大减少.同时，设计了一种基于前缀树的单层布尔型关联规则挖掘算法(FPT-Generate)，不需要反复扫描数据库，不产生候选模式集，并在关键优化技术上取得了突破.实验表明，以 FPT-Generate 为挖掘引擎的空间关联规则发现系统的时间效率与空间可伸缩性远远优于以经典算法 Apriori 为引擎的系统.

关键词 数据库(520·4050) 海量数据库 空间数据挖掘 地理信息系统 空间关联规则

中图法分类号：TP301.6 TP18 **文献标识码**：A **文章编号**：1006-8961(2003)04-0476-05

Design and Implementation of FIPT-Based Spatial Association Rules Mining Algorithm

LIU Jun-qiang[1),2)], PAN Yun-he[1)]

[1)](*Institute of Artificial Intelligence, Zhejiang University, Hangzhou* 310027)

[2)](*School of Computer Information Engineering, Hangzhou University of Commerce, Hangzhou* 310035)

Abstract Spatial association rule discovery in spatial databases is a very important data mining task. In this paper, a two-stage strategy for the discovery of spatial association rules in geographical databases is proposed. The spatial computational overhead is greatly reduced by top down refinement of spatial predicate granularities and multiple recursions of single level boolean association rule discovery step, which is the key step of the algorithm. The single level boolean association rule mining algorithm, FPT-Generate, is detailed. FPT-Generate uses the frequent item prefix tree, FIPT, to compress and project frequent item sets, and discovers association rules by growing a frequent pattern tree, FPT, by depth first search. The algorithm FPT-Generate generates association rules without candidate generation and without redundant scans of databases. Optimizing techniques for the implementation, such as pseudo projecting and pruning, dynamic threading and hashing, and disk based partitioning, are also discussed. Experiments show that spatial association discovery systems powered by FPT-Generate are much more time efficient and space scalable than those powered by the classical algorithm, Apriori. Finally, a spatial association rule discovery system, SmartMiner, upon the support of MapInfo Professional, is developed.

Keywords Database, Very large databases, Spatial data mining, GIS, Spatial association rules

0 引 言

随着遥感技术和自动数据采集技术的广泛应用，特别是地理信息系统的开发与应用，大量空间与非空间数据被采集并存贮在大型数据库中.获取和理解在海量空间数据存贮中蕴含的知识是我们面临的重大挑战，也是传统的数据组织和检索技术所不能胜任的.空间关联规则挖掘是在空间数据库中进行知识发现的一类重要问题.

空间关联规则指明空间谓词与非空间谓词间存在的关联性.例如，通过挖掘地理信息数据库，可能发现“靠近海滩的房屋”有 90%“价格很贵”，“加油站”有 75%“靠近高速公路”，等等.

基金项目：国家 863 计划(2002AA121064)；浙江省自然科学基金(602140)；浙江省留学回国基金

收稿日期：2001-07-02；**改回日期**：2002-09-16

挖掘空间关联规则的基本策略是先计算粗粒度空间谓词,发现较高概念层次的模式与关联规则,然后自顶向下细化空间谓词,逐步发现较低概念层次的关联规则.这样就大大减少了需要精确计算细粒度空间谓词的对象数目,保证了空间谓词计算的高效率.完成空间谓词计算后,空间关联规则挖掘就转化成了单层布尔型关联规则挖掘.由于需要多轮次单层布尔型关联规则挖掘,其效率最关键.

挖掘单层布尔型关联规则的经典算法 Apriori 采用逐步试探法,其核心步骤是根据长度为 k 的频繁模式的所有可能组合形成长度为 $k+1$ 的频繁模式候选集,然后扫描数据库,验证候选集,找出长度为 $k+1$ 的所有频繁模式,最终求出关联规则.算法执行过程中需要不断产生大量候选模式、反复扫描数据库进行模式匹配,扫描次数正比于模式最大长度,时间与空间代价高,可伸缩性差.

本文在提出挖掘空间关联规则的二阶段策略基础上,设计了一种基于前缀树 FIPT 的关联规则挖掘算法 FPT-Generate,该算法不需要反复扫描数据库,不产生候选模式集,并在算法实现的关键优化技术上取得了突破.实验表明,采用算法 FPT-Generate,在地理信息数据库中挖掘空间关联规则的时间效率与空间可伸缩性远远优于经典算法 Apriori.

1　空间关联规则及其挖掘策略

定义 1　空间数据库 SDB 由空间对象集合 $\boldsymbol{O}=\{o_i\}$ 和描述这些对象的非空间属性的关系型数据库 $\boldsymbol{R}=\{r_i\}$ 组成.空间谓词以空间对象集合 $\boldsymbol{O}$ 的某个子集 $\boldsymbol{O}_i$ 为参照系,描述空间对象的空间属性和相互关系.空间关联规则表示空间对象/(空间)谓词关系,其形式为

$$P_1 \wedge P_2 \wedge \cdots \wedge P_m \rightarrow Q_1 \wedge \cdots \wedge Q_n(c\%)$$

其中,$c\%$ 是可信度,P_i,Q_j 是谓词,且至少有一个为空间谓词,如 close_to、adjacent_to、intersect、inside 等.以下是空间关联规则的例子.

$$\text{is\_a}(x,\text{house}) \wedge \text{close\_to}(x,\text{beach}) \rightarrow \text{is\_expensive}(x)(90\%)$$

$$\text{is\_a}(x,\text{gas\_station}) \rightarrow \text{close\_to}(x,\text{highway})(75\%)$$

挖掘空间关联规则分为两个阶段:第 1 阶段,计算各个空间对象的空间谓词,进一步可以将空间谓词组织成一个事务数据库,同一个对象的所有谓词组成一个事务;第 2 阶段,在(谓词)事务数据库中进行单层布尔型关联规则挖掘.

为减少空间计算的工作量,空间谓词可以组织成一个粒度由粗到细的层次结构,自顶向下逐步细化.挖掘过程则是两阶段不断交替的循环过程,首先,挖掘高层粗粒度空间谓词的频繁模式和关联规则;然后,对频繁模式中的空间对象进一步计算低层细粒度的空间谓词,再挖掘相应的频繁模式和关联规则,直至不能发现新的频繁模式为止.

由于高层粗粒度空间谓词可以采用 R 树和最小界盒等空间近似计算方法计算,并且这种计算方法限制和减少了需要计算低层细粒度空间谓词的对象数目,空间谓词计算效率大大提高.在(谓词)事务数据库中挖掘单层布尔型关联规则是提高整体挖掘效率的关键.

单层布尔型关联规则定义如下.

定义 2　项目集 $\boldsymbol{I}=\{i_1,i_2,\cdots,i_m\}$,文字 i_k 是一个项目.数据库 $\boldsymbol{T}=\{t_1,t_2,\cdots,t_n\}$,$t_k \subseteq \boldsymbol{I}$ 是一个事务.模式 $\boldsymbol{p} \subseteq \boldsymbol{I}$ 被事务 t 所包含,如果 $\boldsymbol{p} \subseteq t$.$\boldsymbol{T}$ 对 $\boldsymbol{p}$ 的支持率 $supp(\boldsymbol{p},\boldsymbol{T})$ 是 $\boldsymbol{T}$ 中包含 $\boldsymbol{p}$ 的事务数.$\boldsymbol{p}$ 是频繁模式,如果 $supp(\boldsymbol{p},\boldsymbol{T}) \geq \xi$(阈值).$\boldsymbol{x} \rightarrow \boldsymbol{y}$ 是关联规则,如果 $\boldsymbol{x} \cup \boldsymbol{y}$ 是频繁模式,可信度 $conf(\boldsymbol{x} \rightarrow \boldsymbol{y}) = supp(\boldsymbol{x} \cup \boldsymbol{y},\boldsymbol{T})/supp(\boldsymbol{x},\boldsymbol{T}) \geq \theta$(阈值).

2　单层布尔型关联规则挖掘方法

2.1　频繁项目前缀树

针对算法 Apriori 的缺陷,提出了以压缩方式表达数据库 $\boldsymbol{T}$ 所含频繁模式信息的前缀树.

定义 3　带标记和权重的根树 $LWRT=\langle \boldsymbol{V},\boldsymbol{E},\boldsymbol{L}\rangle$,其中 $\boldsymbol{V}$ 是结点集,记为 $V(LWRT)$.$\forall v \in \boldsymbol{V}$ 用有序对 $\langle i,w\rangle$ 标注,文字 $i \in \boldsymbol{L}$ 记为 $v.label$,整数权重 w 记 $v.weight$.根结点用 $\langle,\rangle$ 标注.如果 $v.label=i$,则称 v 为 i 结点.$\boldsymbol{E}$ 是有向边集,从结点 v_i 到 v_j 的边记为 $\langle v_i,v_j\rangle \in \boldsymbol{E}$,称 v_i 是 v_j 的父亲,v_j 是 v_i 的子女.如果存在结点序列 $\langle v_{i_1},\cdots,v_{i_p}\rangle$,$\langle v_{i_k},v_{i_{k+1}}\rangle \in \boldsymbol{E}$,$v_{i_1}=v_i$,$v_{i_p}=v_j$,则称存在从结点 v_i 到 v_j 的路径,v_i 是 v_j 的祖先,v_j 是 v_i 的后代.结点 v 的祖先集合记为 $anct(v,LWRT)$,后代集合记为 $desc(v,LWRT)$.$\boldsymbol{L}$ 是 $LWRT$ 标注所用文字集.

定义 4　项目前缀树 $IPT=\langle \boldsymbol{V},\boldsymbol{E},\boldsymbol{L},\Omega,\boldsymbol{D}\rangle$ 是一棵 $LWRT$,其中 $\boldsymbol{L} \subseteq \boldsymbol{I}$ 是 IPT 所表达的项目子集.$\Omega(\langle i_1,w_1\rangle,\cdots,\langle i_q,w_q\rangle)$,记为 $\Omega(IPT)$,$i_k \in \boldsymbol{L}$,$w_k=v.weight(v \in \boldsymbol{V},v.label=i_k)$.$\langle i_1,\cdots,i_q\rangle$ 称 Ω

序，记为 $\Omega.iseq$. 如果 s 是 t 的子序列，则记为 $s\subseteq t$. 如果$\langle i,j\rangle\subseteq\Omega.iseq$，记为 $i\prec j$. 如果 $i\prec j$ 或 $i=j$，记为 $i\preceq j$. IPT 任意路径的项目序列$\subseteq\Omega.iseq$. $\boldsymbol{D}$ 为数据库 $\boldsymbol{T}$ 在 $\boldsymbol{L}$ 上的投影，即$\forall\ t\in\boldsymbol{T},\exists\ s=t\cap\boldsymbol{L}\in\boldsymbol{D}$，$s$ 按$\Omega.iseq$与一条从根起始的路径唯一对应，路径上各结点的权重等于其所表达的事务数. 频繁项目前缀树 $FIPT=\langle\boldsymbol{V},\boldsymbol{E},\boldsymbol{L},\Omega,\boldsymbol{D}\rangle$是一棵 IPT，并且$\forall\ i\in\boldsymbol{L},supp(i,\boldsymbol{T})\geq\xi,\forall\ \langle\langle i_k,w_k\rangle,\langle i_j,w_j\rangle\rangle\subseteq\Omega,w_k\geq w_j$.

$FIPT$ 不表达非频繁项目，并以支持率降序排列频繁项目，各个事务更有可能合并表达共同的频繁项目前缀，因此 $FIPT$ 的规模远远小于数据库 $\boldsymbol{T}$.

构造 $FIPT$ 的算法如下：

step1 CountAndSortItemsBySupports

扫描 $\boldsymbol{T}$，统计各项目支持率，将频繁项目、支持率二元组按降序排列得 Ω.

step2 BuildFrequentItemsPrefixTree

创建 $FIPT$ 的根，并标注为$\langle,\rangle$；

for each 事务 t in 数据库 $\boldsymbol{T}$ do begin

将 t 的频繁项目按 $\Omega.iseq$ 排列得子序列 s；

令指针 v 指向 $FIPT$ 的根结点；

for each 项目 i in 序列 s do begin

if 存在 v 的子女 u，并且 $u.label=i$

then $u.weight$ 加 1；

else begin

新建 v 的子女 u；

$u.label\leftarrow i$；

$u.weight\leftarrow 1$；

end

$v\leftarrow u$；

end

end

如表 1 所示，如果 ξ 取 4（即 40%），则 $\Omega=\langle\langle i_3,6\rangle,\langle i_6,6\rangle,\langle i_1,5\rangle,\langle i_2,4\rangle,\langle i_{13},4\rangle,\langle i_{16},4\rangle\rangle$，$FIPT$ 如图 1 所示.

表 1 事务数据库 $\boldsymbol{T}$

t_1	$i_1\ i_2\ i_3\ i_6\ i_{12}\ i_{13}\ i_{15}$
t_2	$i_1\ i_3\ i_8\ i_{17}$
t_3	$i_2\ i_6\ i_8\ i_{10}\ i_{13}\ i_{15}$
t_4	$i_2\ i_3\ i_{11}\ i_{19}\ i_{16}$
t_5	$i_6\ i_1\ i_3\ i_4\ i_7\ i_9\ i_{13}\ i_{16}$
t_6	$i_9\ i_{19}\ i_{20}\ i_{17}$
t_7	$i_5\ i_6\ i_8\ i_1\ i_{10}\ i_{18}$
t_8	$i_{14}\ i_{18}\ i_{20}$
t_9	$i_1\ i_6\ i_3\ i_5\ i_{12}\ i_{16}\ i_{13}\ i_{14}$
t_{10}	$i_4\ i_3\ i_7\ i_2\ i_{11}\ i_{16}$

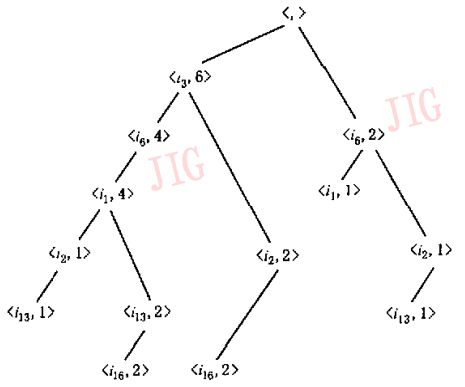

图 1 $FIPT$

2.2 基于 $FIPT$ 的关联规则挖掘方法

设 $\Omega=\langle\langle i_1,w_1\rangle,\cdots,\langle i_q,w_q\rangle\rangle$，按 $\Omega.iseq$ 排列 $\boldsymbol{T}$ 所含频繁模式，则可将频繁模式划分成若干子集，$\forall\ \langle i_k,w_k\rangle\in\Omega,w_k\geq\xi$ 对应子集 $\boldsymbol{FP}_{i_k}$ 的每个频繁模式均以结尾. 显然$\langle\{i_k\},w_k\rangle\in\boldsymbol{FP}_{i_k}$，$\boldsymbol{T}$ 的投影子集 $\boldsymbol{T}_{i_k}^{\mathrm{EX}}=\{t\cap\{i_1,\cdots,i_{k-1}\}\mid i_k\in t\in\boldsymbol{T}\}$所含的频繁模式拼接$\langle\{i_k\},w_k\rangle$亦属于 $\boldsymbol{FP}_{i_k}$，并且

$$\boldsymbol{FP}_{i_k}=\{\langle\{i_k\},w_k\rangle\}\cup\{\langle p\cup\{i_k\},w\rangle\mid w=supp(p,\boldsymbol{T}_{i_k}^{\mathrm{EX}})\geq\xi\}$$

定义 5 项目 j 剪裁 $IPT=\langle\boldsymbol{V},\boldsymbol{E},\boldsymbol{L},\Omega,\boldsymbol{D}\rangle$得到 $subtree(IPT,j)=\langle\boldsymbol{V}^S,\boldsymbol{E}^S,\boldsymbol{L}^S,\Omega^S,\boldsymbol{D}^S\rangle$是一棵 IPT，且 $\boldsymbol{V}^S\subset\boldsymbol{V},\forall\ v\in\boldsymbol{V}^S,v.weight=\sum_{\substack{u.label=j\\ u\in desc(v,IPT)}}u.weight$. $\boldsymbol{E}^S\subset\boldsymbol{E},\boldsymbol{L}^S\subset\boldsymbol{L},\forall\ i\in\boldsymbol{L}^S,i\prec j,\Omega^S\subset\Omega,\boldsymbol{D}^S$ 是 $\boldsymbol{D}$ 在 $\boldsymbol{L}^S$ 上的投影子集.

根据定义 5，用项目 j 剪裁 IPT 的过程是：由 IPT 上所有从根结点至 j 结点的路径组成子树；叶结点 IPT 的权重取 IPT 上相应结点的权重；除根外，内点的权重取其子女权重之和；然后从该子树中删除叶结点，得到 $subtree(IPT,j)$.

图 2 为用$\langle i_1,5\rangle$剪裁图 1 $FIPT$ 的过程示例，图 2(c)是所得 $FIPT$ 子树. 对应频繁模式子集由$\langle\{i_1\},5\rangle$和从图 2(c)得出的频繁模式拼接而成，即

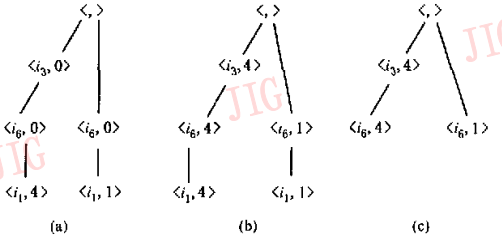

图 2

$\{\langle\{i_1\},5\rangle,\langle\{i_3,i_1\},4\rangle,\langle\{i_5,i_1\},5\rangle,\langle\{i_3,i_6,i_1\},4\rangle\}$. 可以证明,$subtree(IPT,j)$ 表达的投影子集 $\boldsymbol{D}^s=\boldsymbol{T}_j^{FX}$,因此有

定理　令 $\Phi(IPT,\xi)$ 是 IPT 所表达的数据库 $\boldsymbol{T}$ 的投影 $\boldsymbol{D}$ 中支持率$\geq\xi$ 的模式集合,则

$$\Phi(IPT,\xi)=\bigcup_{\langle i,w\rangle\in\Omega(IPT)\wedge w\geq\xi}(\{\langle\{i\},w\rangle\}\cup\Phi(subtree(IPT,i),\xi)\times\{\langle\{i\},w\rangle\})$$

其中,$\langle I_1,w_1\rangle\circ\langle I_2,w_2\rangle=\langle I_1\cup I_2,\min(w_1,w_2)\rangle$,

$$\boldsymbol{P}\times\boldsymbol{Q}=\{\boldsymbol{p}\circ\boldsymbol{q}\mid\boldsymbol{p}\in\boldsymbol{P}\wedge\boldsymbol{q}\in\boldsymbol{Q}\}$$

定义 6　频繁模式树 $FPT=\langle\boldsymbol{V},\boldsymbol{E},\boldsymbol{L}\rangle$ 是一棵 $LWRT$,FPT 从根到任意结点 v 的路径代表一个频繁模式,路径终点权重 $v.weight$ 是该模式的支持率.

根据定理,挖掘频繁模式可描述为建立频繁模式树 FPT 的过程:创建 FPT 的根,并标注$\langle,\rangle$;$\forall\langle i,w\rangle\in\Omega(IPT)\wedge w\geq\xi$,创建根的子女 v,并标注$\langle i,w\rangle$;$\forall\langle i^s,w^s\rangle\in\Omega(subtree(IPT,i))\wedge w^s\geq\xi$,建立 v 的子女 v^s,并标注$\langle i^s,w^s\rangle$,以此类推. 图 3 是挖掘图 1 所示 $FIPT$ 得到的 FPT,除根外共有 12 个结点,代表 12 个频繁模式.

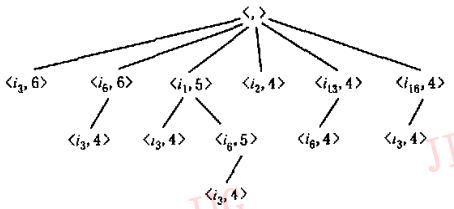

图 3

因此,挖掘频繁模式可以由以下非递归算法完成:

```
step3 GenerateFrequentPatternsTree
创建 FPT 的根,并标注为空⟨,⟩;
∀⟨i,w⟩∈Ω(IPT)∧w≥ξ,建立根子女,标注⟨i,w⟩;
将根的第 1 个子女压入栈;
while 栈非空 do begin
  从栈中弹出一个 FPT 结点 v,令 i=v.item;
  设 v 父亲项目前缀树 IPT_p,
  求 IPT_c←subtree(IPT_p,i);
  ∀⟨i_c,w_c⟩∈Ω(IPT_c)∧w_c≥ξ,建 v 子女⟨i_c,w_c⟩;
  v 的下个兄弟压入栈,再将 v 的第 1 子女压入栈;
end
最后,可以从 FPT 中挖掘出关联规则.
step4 GenerateAssociationRules
for each 结点 v in FPT do begin
获得从根到 v 的项目序列 pattern(设长度为 n);
for each LHS∪RHS=pattern∧LHS∩RHS=∅ do
begin
  搜索 FPT 从根起始项目序列为 LHS 的路径终点 x;
  if v.weight/x.weight≥θ then 输出 LHS→RHS;
end
end
```

3　算法实现及其关键技术

3.1　基本算法的优化

剪裁 IPT 子树操作:$IPT_c\leftarrow subtree(IPT_p,i)$ 是决定挖掘效率的关键步骤. 采用虚拟剪裁技术、动态穿线技术获得了最佳的空间与时间效率.

虚拟剪裁技术　在同一数据结构中存贮 IPT 及其剪裁出的 $subtree(IPT,i)$,具体操作是

① $\forall v\in V(IPT)(v.label<i\wedge v.weight>0\Rightarrow v.weight$ 清 0). IPT 项目按 $\Omega.iseq$ 排在 i 前、权重>0 结点权重清 0;

② $\forall u\in V(IPT)\left\{\begin{array}{l}u.label=i\wedge u.weight>0\Rightarrow\forall v\in V(IPT)\\(v\in anct(u,IPT)\Rightarrow v.weight\text{ 加 }u.weight)\end{array}\right\}$.

沿各权重>0 的 i 结点向根逆行,累计其祖先的权重.

由于剪裁操作不改变树的拓朴关系,剪裁依据是各权重>0 的 i 结点,因此,只要按 $\Omega.iseq$ 自顶向下执行,就可在同一数据结构中完成所有的剪裁操作.

动态穿线技术　在虚拟剪裁过程中,始终使权重>0 结点按项目穿入对应的横向线索中,从而大大缩小搜索范围.

另外,使用 hashing 技术来计算 Ω:扫描数据库 $\boldsymbol{T}$ 过程中,采用线性表加开放式链表的数据结构来记录各个项目的支持数,用杂凑函数进行项目寻址,以大大提高统计效率;

使用 hashing 技术与穿线技术快速得出每个事务的频繁项目前缀序列,避免对 Ω 作线性搜索. 每个 IPT 结点的子女按 $\Omega.iseq$ 排序,以提高在 IPT 上进行子序列搜索和插入的效率.

动态管理 FPT　边生成、边输出、边删除 FPT 结点,使 FPT 内存代价最小化,可忽略不计.

3.2　基于辅存的算法实现

尽管大部分情况下,可以将 IPT 放在内存中,但是这并不能保证 IPT 一定不会超出内存容量的限制,为此提出了基于辅存的算法 Disk-based FPT-Generate,其基本思路是分而治之,即将 $\boldsymbol{T}$ 划分为若干投影子集,分别挖掘其中的频繁模式.

(1)扫描 $\boldsymbol{T}$,得频繁项目权重二元组序列(降序)$\Omega^0=\langle\langle i_1,w_1\rangle,\cdots,\langle i_q,w_q\rangle\rangle$.

(2)分割原则　选取 $\Omega^0.iseq$ 的子序列$\langle i_{j_1},\cdots,$

i_{j_r}>,设 $i_{j_q}<i_1$ 是特殊文字,$i_{j_r}=i_q$,将 T 划分为 r 个子集,第 k 个子集

$$T^*_{i_{j_k}}=\{t\cap\{i_1,\cdots,i_{j_k}\}\mid t\in T\wedge\exists\, i\in t\wedge i_{j_{k-1}}<i\le i_{j_k}\},k=1,\cdots,r$$

(3)扫描 T,根据分割原则,在辅存中建立 $T^*_{i_{j_k}}$ 对应的子树 $IPT^*_{i_{j_k}}$,$k=1,\cdots,r$.

(4)挖掘频繁模式

$$\bigcup_{1<k\le p}\ \bigcup_{(i,w)\in\Omega(IPT^*_{i_{j_k}})}^{i_{j_{k-1}}<i\le i_{j_k}\wedge w\ge\xi}(\{\{i\},w)\}\cup\Phi(subtree(IPT^*_{i_{j_k}},i),\xi)\times\{(\{i\},w)\})$$

(5)从频繁模式中挖掘出关联规则

4 性能评价和结论

本文以 800MHz Pentium Ⅳ CPU、256MB 内存、20GB 硬盘和 Windows 2000 Server 操作系统为实验平台,用大量数据集对基本算法进行性能测评.图 4 数据集是 T25I20D100kN20kL5k,其中,D××k 是数据集的大小,支持率为 0.1%至 1.0%.图 5 数据集是 T25I20D(100k ~ 1m)N20kL5k,支持率为 0.25%.

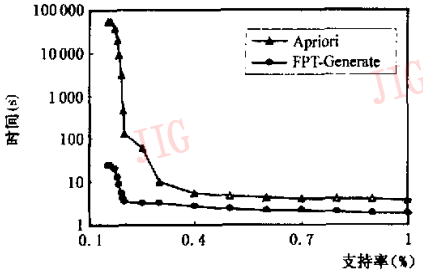

图 4 算法的时间性能比较

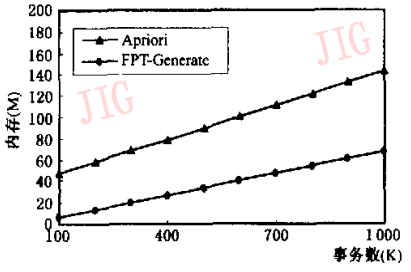

图 5 算法所占内存性能比较

测试表明:FPT-Generate 的时间效率优于 Apriori[9].特别是当阈值不断降低时,关联规则爆炸性增加,使 Apriori 的缺陷突显出来,FPT-Generate 的效率可高出 Apriori 几个数量级(图 4).FPT-Generate的空间可伸缩性优于 Apriori.FPT-Generate占用的内存空间远小于 Apriori(图 5).

以 FPT-Generate 算法为引擎,在 MapInfo Professional v4.1.2 基础上进行二次开发,实现了用于空间关联规则挖掘的实验系统 SmartMiner,并用 Business Points Data[10]进行了实验.实验表明,基本挖掘策略是可行的,实验系统的时间效率与空间可伸缩性是令人满意的.

参 考 文 献

1 Agrawal R, Srikant R. Fast algorithms for mining association rules [A]. In: VLDB'1994 [C], Santiago, Chile, Sept. 1994: 487~499.

2 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[A]. In: SIGMOD'1993 [C], Washington, D.C., May 1993.

3 Han J, Fu Y. Discovery of multiple-level association rules from large databases[A]. In: VLDB'1995 [C], Znrich Switzerland, 1995:420~431.

4 Srikant R, Agrawal R. Mining generalized association rules[A]. In: VLDB'1995[C], Zurich, Switzerland, Sept. 1995: 407~419.

5 Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [A]. In: SIGMOD'2000 [C], Dallas, TX, May 2000.

6 Agrawal R, Aggarwal C, Prasad V V V. A tree projection algorithm for generation of frequent itemsets [J]. Journal of Parallel and Distributed Computing, 2000,61(3): 350~371.

7 Srikant R, Agrawal R. Mining quantitative association rules in large relational tables[A]. In: SIGMOD'1996 [C], Montreal, Canada, 1996: 1~12.

8 Bayardo R J. Efficiently mining long patterns from databases [A]. In: SIGMOD'1998[C], Seattle, Washington, June 1998: 85~93.

9 Christian Borgelt. Apriori: a program to find association rules [CP/OL]. http://fuzzy.cs.uni-magdeburg.de/~borgelt

10 MapInfo Corporation. MapInfo Data: Business Points Sample [DB/OL]. http://www.mapinfo.com/free/index.cfm

刘君强 1962 年生,杭州商学院副教授,硕士生导师,1984 年获北京大学计算机软件专业学士学位,1987 年获浙江大学管理工程专业硕士学位,现为浙江大学计算机系博士生.主要从事人工智能、数据挖掘研究.

潘云鹤 1946 年生,教授,博士生导师,浙江大学校长,中国工程院院士,1970 年毕业于同济大学建筑系,1981 年获浙江大学计算机应用专业硕士学位.主要从事人工智能、智能 CAD/CG 研究.