基于树结构多重最小支持度的挖掘算法研究

占美星[1],杨 颖[1],杨 磊[2]
(1. 广西大学 计算机与电子信息学院,广西 南宁 530004;
2. 广西科学院 应用物理研究所,广西 南宁 530003)

摘 要:传统的序列数据库中各数据项的最小支持度是单一的,且不能有效挖掘用户感兴趣的、稀有的数据项。为了有效提高数据挖掘的效率和准确率,文中基于 PLWAP-tree 提出了前序链接多重支持度树(Preorder Linked Multiple Supports tree,PLMS-tree)来存储序列数据库,并进一步提出了多重最小支持度条件模式增长(Multiple Support-Conditional Pattern growth,MSCP-growth)算法。算法采用对每个频繁数据项设置多重最小支持度的方法来减少空间和时间的开销。对于每个频繁数据项设置不同的支持度,来挖掘用户所需的数据序列,能有效提高数据挖掘的效率和准确率。实验结果验证了算法的有效性,对序列模式下的数据项目集挖掘的时间效率和空间效率有明显的提高。

关键词:数据挖掘;序列模式;多重最小支持度
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2014)08-0045-06
doi:10.3969/j.issn.1673-629X.2014.08.011

Study on Mining Algorithm Based on Tree Structure Multiple Minimum Supports

ZHAN Mei-xing[1], YANG Ying[1], YANG Lei[2]
(1. Institute of Computer and Electronic Information, Guangxi University, Nanning 530004, China;
2. Institute of Applied Physics, Guangxi Academy of Sciences, Nanning 530003, China)

Abstract: The minimum support of each data in traditional sequence database is single and cannot effectively mine user interest and rare items. In order to effectively improve the efficiency and accuracy of data mining, based on PLWAP-tree, the Preorder Link Multiple Supports tree (PLMS-tree) is structured to store the entire sequence databases, and then the algorithm of Multiple Supports Conditional Pattern growth (MSCP-growth) is proposed. It reduces the cost of space and time through multiple minimum supports. Each frequent item sets different supports to meet the required data pattern. Experimental results demonstrate the effectiveness of the algorithm, which can significantly improve mining time and space efficiency for the data item set in the sequence mode.

Key words: data mining; sequential patterns; multiple minimum supports

0 引 言

数据挖掘是从数据库中发现有价值的信息,已经被广泛地应用在各个领域,比如 Web 行为研究、决策支持、商业预测和客户购买行为分析等。

序列模式挖掘(Sequential Pattern Mining,SPM)最早是由 Agrawal 和 Srikant 提出的,是一种根据最小项目支持度(Minimum Item Support,MIS)从序列数据库挖掘出所需的序列模式[1],其原理与 FP-tree 以增长模式为基础[2]。针对在数据库中数据项的排序,Liu 等依照其 MIS 递增排序提出排序封闭特性(Sorted Closure Property)[3]。Pei 等针对 SPM 挖掘提出 WAP-tree 结构用来在紧凑的和压缩的前缀树挖掘出完整的系列模式[4]。Ezeife 和 Lu 针对 WAP-tree 中间过程会产生许多的模块提出了优化的 PLWAP-tree[5]。为了解决 SPM 只是概要的假设对数据库中的所有数据项只使用一个单一的最小支持度,Liu 提出了多重最小支持度(Multiple Minimum Supports,MMSs),基于 MMSs 提出了多重支持度广义序列算法(Multiple Support-Generalized Sequential Pattern,MS-GSP)[6]。

收稿日期:2013-09-30 **修回日期**:2014-01-07 **网络出版时间**:2014-04-24
基金项目:广西自然科学基金(桂财教[2013]19 号 2013jjAA70089)
作者简介:占美星(1988-),男,江西九江人,硕士研究生,研究方向为并行算法与分布式计算、数据挖掘;杨 颖,教授,硕士生导师,研究方向为并行算法、数据挖掘;杨 磊,研究员,硕士生导师,研究方向为计算机网络、信息处理技术。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140424.0830.077.html

基于 MMSs 挖掘应用研究像关联规则 ARM,根据最小支持度产生出所有的大数据项集并且利用大数据项集根据最小置信度产生出所有关联规则[7-8]。文献[9]提出了 MMSs 与 FP-growth 算法混合来挖掘频繁项,主要通过排序封闭性特征来减少搜索空间。然而目前在实际应用中只有文献[1]提出了 MS-GSP,是通过 SPM 与 MMSs 算法提出了不产生候选集的数据挖掘算法,但是排序封闭性没有被应用在 MS-GSP 算法中,这是因为在数据序列中数据项的排序是不能更改的,这样无形中增加了搜索空间。文献[10]提出基于多重最小支持度频繁模式树形挖掘算法,但介于中间存储增加导致了搜索空间的进一步扩大。为了减少搜索空间,Chen 等提出了向下封闭属性(Downward Closure Property)[11]。文献[12]提出稀少数据项(rare items)来判断不频繁序模式中其子序列可能会成为频繁序列,这样又增加了搜索空间。文献[13]通过设计多最小支持度的关联规则挖掘算法来解决。而文献[14]提出了考虑多重最小支持度的挖掘加权关联规则的算法。但该算法没有考虑到不同的项目具有的支持度是不一样的。在实际应用中,往往需要不断调整阈值才能得到期望规模的规则集,即需要一个与程序交互的过程。针对这种情况,不断有学者利用已挖掘信息,提出包括 IUA[15]、LIUA[16]、QIUA[17]、IIUA[18]等针对交互挖掘的算法及改进算法。基于上述的数据挖掘技术算法与数据仓库技术算法的发展,已有不少学者对关联规则挖掘在股票预测、财务分析中的应用进行研究并取得了一定成果[19]。

目前基于 MMSs 研究一般都是采用了候选模式产生与测试的方法,这种方法在模式数据挖掘过程中耗费了大量样本和时间,并且由于中间集产生的大大增加了搜索空间和时间开销。

文中在 PLWAP-tree 基础上,提出一种紧凑的数据结构 PLMS-tree,用来存储整个序列数据库,减少了在存储和中间模式集产生的搜索空间。并进一步提出基于多重最小支持度的多重支持条件模式增长的序列模式挖掘算法(Multiple Support-Conditional Pattern growth,MSCP-growth),用来挖掘所需的序列模式的数据,在减少时间开销和空间开销下,能较好地挖掘较为理想的序列模式。

1 基本概念

用 I 表示是数据库 S 中的项目集,i 表示项目集的项目。项目 i 的最小项目支持度设为 $\mathrm{MIS}(i)$,并且 $0<\mathrm{MIS}(i)\leqslant 1$。下面给出文中的一些定义:

定义 1:项目集 I_q 的最小项目支持度。

有一项目集 $I_q=(i_1,i_2,\cdots,i_m)$,则项目集 I_q 的最小项目支持度 $\mathrm{MIS}(I_q)$ 为:

$$\mathrm{MIS}(I_q)=\min[\mathrm{MIS}(i_1),\mathrm{MIS}(i_2),\cdots,\mathrm{MIS}(i_m)]$$

定义 2:最小支持度的阈值。

给定序列项目集 $\beta=<(\mathrm{IB}_1),(\mathrm{IB}_2),\cdots,(\mathrm{IB}_s)>(1\leqslant q\leqslant s)$,则 β 的最小支持度的阈值 $\mathrm{MIS}(\beta)$ 为:

$$\mathrm{MIS}(\beta)=\min[\mathrm{MIS}(\mathrm{IB}_1),\mathrm{MIS}(\mathrm{IB}_2),\cdots,\mathrm{MIS}(\mathrm{IB}_s)]$$

定义 3:最小序列支持度 MIS 的序列模式。

给定一个序列数据库 S 和序列 β,如果 $\mathrm{Sup}(\beta)\geqslant\mathrm{MIS}(\beta)$,则 β 在序列数据中是频繁的。若 MIN 是在数据库中数据项最小的 MIS 值,如果 $\mathrm{Sup}(\beta)\geqslant\mathrm{MIN}$,则称 β 为 MIN 序列模式。

定义 4:K - 模式的定义。

假设 L 表示一组基于 MMSs 环境下完整的序列模式和 L_{MIN} 为 MIN 序列模式。对于给定的序列数据库 S,设 L_{MIN_k} 为一组完整的 K - 最小序列模式,则 $L_{\mathrm{MIN}_k}=\{x\mid x\in L_{\mathrm{MIN}}\wedge\mathrm{length}(x)=k\}$,类似的可以把一组完整 K - 模式定义为 $L_k=\{x\mid x\in L\wedge\mathrm{length}(x)=k\}$。

定理 1:每个在 L_k 中的频繁序列模式必在 L_{MIN} 中,即 $L_k\subseteq L_{\mathrm{MIN}}$

证明:因为 $x\in L_k$ 所以 $x\in L_{\mathrm{MIN}}$

由定义 3 可知:$x\in L_k$

因为 $\mathrm{Sup}(x)\geqslant\mathrm{MIS}(x)$

由定义 4 可知:MIN 是数据项中最小的 MIS

所以 $\mathrm{Sup}(x)\geqslant\mathrm{MIN}\quad x\in L_{\mathrm{MIN}}\quad L_k\subseteq L_{\mathrm{MIN}}$

2 基于树结构多重最小支持度的序列模式挖掘算法 MSCP-growth

基于 PLWAP-tree 结构存储的数据项不能有效地判别非频繁项的子序列是否是频繁项,在此基础上提出了 PLMS-tree 结构来存储和压缩数据库。PLMS-tree 结构中增加了 iflag 和 mps 两个属性,这两个属性表示能够在判别中来标记子序列是否为频繁项以及能够减少搜索空间。

2.1 PLMS-tree 定义及构造

本节在 PLWAP-tree 基础上进行扩展改进,形成新的数据结构 PLMS-tree。

2.1.1 PLMS-tree 定义

(1)一个 PLMS-tree 结构包括一个根节点并标记为“NULL”,一组树节点和一个表头。

(2)表头中由三个属性组成,分别是 iname、mis 和 nlink。其中 iname 表示数据项的名称;mis 表示该数据项的最小项目支持度即 MIS(i);nlink 表示在 PLMS-tree 中有相同的 iname 的节点之间的链接。在表头的数据项是以 MIS 值来递增存储的。

(3)在 PLMS-tree 中的每个节点存储的信息包括

iname, icount, iflag, mps, pcode 以及节点的链接集(plink,clink,slink,nlink)。icount 表示到达该路径序列的个数,iflag 表示在一个链接下的最后一个数据项,mps 表示此节点的最小的 MIS 值,pcode 表示在 PLMS-tree 中的位置代码,节点链接 plink, clink, slink, nlink 分别表示父节点,第一个子节点,第一个兄弟节点,下一个具有相同名称的节点。

(4)所有属于 L_{MIN} 的数据项都被存储在表头中并且是以它们的 MIS 值非递减的排序存储。

相对于 PLWAP-tree,在 PLMS-tree 中,iflag 和 mps 是两个新加属性,在 PLWAP-tree 中主要是集中在 Web 日志进行挖掘,并且只是假设每个项目集只包含一个数据。在文中,设置 iflag 可以让数据项包含多个数据,mps 可以检测一个非频繁模式的数据序列在随后的模式增长过程能不能成为频繁并且能够减少不必要的搜索空间。

2.1.2　PLMS-tree 的构造

基于 PLMS-tree 的定义以及构造 PLMS-tree 结构的过程,PLMS-tree 的构造过程算法如图 1 所示。

```
算法:PLMS-tree 结构构造
输入:序列数据库 S 以及数据项集 I 和数据项 i 的最小项目支持度 MIS 值
输出:带有表头 HT 的 PLMS-tree 树结构 T
1:扫描数据库 S 并且得出每个数据项的支持度 Sup(i) ;//第一次扫描数据库
2: For each item i in I
3:if supp(i)≥MIN then 把节点 i 追加到 L_MIN中;
4:end loop
5:创建 PLMS-tree 的根节点 T,并设置标签为"NULL";
6:初始化以在 L_MIN数据项 i 的 MIS 值非递减排序的表头 HT;
7:For each sequence s in S    //第二次扫描数据库
8:删除在 S 中不在 L_MIN中的所有项,把剩下在 L_MIN中的数据项以 MIS 值来进行升序排序组成新的数据序列 S′;
9:在 S′中的每一数据项添加 mps 和 iflag;
10:Set p to root in T;
11:For each item i in S′
12:if p 存在一个子节点 e 并且 e. iname=i. iname 和 e. iflag=i. iflag
13:then e. icount++;
14:if i. mps<e. mps   then e. mps=i. mps,point p to e;
15:else   if P 没有子节点 then
16:创建一个子节点 e 并且使 e. iname=i. iname,
e. iflag=i. iflag,e. mps=i. mps 和 e. icount=1;
17:if e 是 p 的第一个子节点 then
18:e. plink=p,p. clink=e, e. pcode=p. pcode 并且追加"1"到 e. pcode 的末尾
19:else   e. plink=p,p. clink=e 并创建 p 最后子节点 d 使 d. slink=e,
e. pcode=d. pcode 并追加"0"到 e. pcode 的末尾,point p to e;
20: return T
```

图 1　PLMS-tree 的构造过程

给定如表 1 所示的序列数据库 S。根据 PLMS-tree 的算法,在数据项插入树之前它们在序列数据中的顺序是以 MIS 值升序排序的。

表 1　序列数据库 S

SID	未排序的序列	已排序的序列
10	<(ac)(ab)(bc)>	<(ca)(ab)(cb)>
20	<(ac)(ab)(c)>	<(ca)(ab)(c)>
30	<(ac)(f)(ab)(d)(e)>	<(ca)(f)(ab)(d)(e)>
40	<(a)(e)(c)(a)(b)>	<(a)(e)(c)(a)(b)>
50	<(a)(e)(c)(b)(d)(e)>	<(a)(e)(c)(b)(d)(e)>

PLMS-tree 算法的构造过程如下:

第一步:扫描序列数据库 S 并且得出各项的支持度,分别为(a:5),(b:5),(c:5),(d:2),(e:2),(f:1),若 MIN=2,只有数据项 f 不能追加到 L_{MIN} 中。

第二步:对树 T 创建一个根节点其标签为空即为"NULL",表头 HT 存储着每个数据项名称和相对应的 MIS 值并且各数据项是以其 MIS 值递增排序的,对每个数据项的扫描就建立了 PLMS-tree 树的分支。所有插入的节点都有其相对应的数据属性,包括 iname, icount,mps 以及 pcode,如果 iflag=1,在该数据项前面增加"_"。

在插入 S10 之后 PLMS-tree 结构如图 2(a),S10 在以 MIS 排序后 S10=<(ca)(ab)(cb)>。

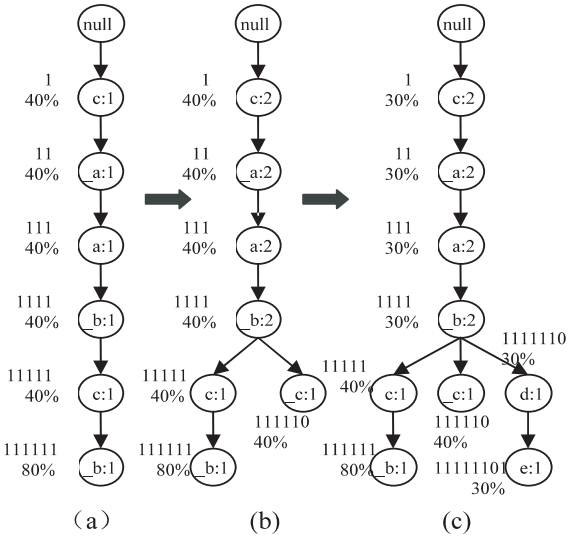

图 2　构建 PLMS-tree 的过程

第三步:找到 S10 每项的 mps 和 iflag,分别为(40%,40%,40%,40%,40%,80%)和(0,1,0,1,0,1)。S10 中第一项,因为根节点没有任何子节点,所以把 S10 中的第一项设为其子节点(c:1:40%:1),并把根节点指针指向当前的节点(c:1:40%:1);S10 中的第二项,由于(c:1:40%:1)也没有任何子节点,所以把(_a:1:40%:11)设为(c:1:40%:1)的子节点,同样把指针指向当前的节点(_a:1:40%:11);其他 S10 中的节点用同样的方法加入树中。

第四步:对于 S20=<(ca)(ab)(c)>,其数据项对

应的 mps 和 iflag 分别为(40%,40%,40%,40%,40%)和(0,1,0,1,1)。发现其与 S10 有相同的前缀(ca)(ab)(c),但是第三个数据项 c 的 iflag 是不一样的。对于前四项其 icount 是以 1 来递增的,并且其 mps 是被保留的。对于第三个数据项 c,构造一个新的数据项(_c:1:40%:111110)作为 b 的一个子节点,因为 b 有第一个子节点,所以其作为最后一个数据项且最后的 pcode 要增加一个“0”。最后 S20 插入之后树的结构如图 2(b)所示。

第五步:S30=<(ca)(f)(ab)(d)(e)>,因为(f)已经被移除了,所以其他的数据项的 mps 与 iflag 分别是(30%,30%,30%,30%,30%,30%)和(0,1,0,1,1,1),其与树有相同的前四项前缀,剩下的两个数据项(d)(e)用前面相同的方法插入到树中,S30 插入树中之后如图 2(c)所示。

至此完成了 PLMS-tree 结构的完整构造过程。为了便于树的遍历,在表头中最后建立 nlink,和有相同名称的节点建立链接路径,采用 buildLinkage(p)算法从根节点开始,递归的遍历每个节点的流程如下:

(1)找到当前节点的子节点 child,然后在表头找到有相同名称的节点 HTiname;

(2)以 HTiname. nlink 找到最后节点 Itn,产生节点的链接路径;

(3)如果 HTiname. nlink 为空号,HIiname. nlink = child,否则 Itn. nlink = child;

(4)遍历子节点(建立相同名称节点之间的链直到在前缀没有其他的子节点,否则继续遍历其兄弟节点)。

2.2 MSCP-growth 算法

2.2.1 MSCP-growth 算法简介

在 PLMS-tree 构造之后,提出了 MSCP-growth 来挖掘一整套的基于 MMSs 的频繁序列模式,挖掘算法 PLWAP-tree 的扩展,MSCP-growth 采用了分而治之的方式,即把一个分区划分成几个小分区。算法以项目 i 的前缀表示所有带有前缀的序列模式即为项目 i 的条件模式可以在 i 条件后缀的 PLMS-tree 树结构发现。这样就可以挖掘出频繁项。

图 3 给出了在 PLMS-tree 结构存储上的基于 MMSs 的 MSCP-growth 算法。算法首先通过 i. nlink 来计算条件后缀 PLMS-tree 的 HT 的每个数据项的最小项目支持度 MIS,在同一个项目集有相同的 iname 和 iflag 的节点只能计算一次。其次在 PLMS-tree 中 pcode 可以快速地定位两个属于不同项目集的节点,所以其相对应的 icount 值可以累加,而不用遍历整个树来查找具有相同名称的节点,iflag 可以检查属于不同项目集但是有相同路径的两个节点,所以有相同的 iname 但是在不同的项目集的节点可以分开计数。最后,如果节点 n 有子节点,则把 n 插入到一个新的根集 nRS 中。若节点 i. icount[0] 的数值大于或等于 MIS(β) ×| S | 与 i. mps ×| S | 的最小者,则 i 被追加到 β 之后形成新的序列 β',如果 i. count[0] 大于或等于 MIS(β') ×| S |,那么就把 β' 作为频繁序列输出,对 i. icount[1] 做同样的条件判断,那么 β' 就是所有挖掘出的基于 MMSs 的序列模式。

```
算法 MSCP-growth(RS,β,MIS(β))
输入:PLMS-tree 结构 T,表头文件 HT,序列数据库 S,根集 RS
输出:基于 MMSs 的完整的频繁序列模式
1:if RS is empty then return
2:For each item i in HT
3:Set i. mps to 1;
4:For each node n. iname=i. iname in HT
5: if n 是 RS 中节点 r 的一个子节点并且 n 没有被计数 then
6:if 在 n 与 r 之间还有一个节点 q,并且 q. iflag=1
7: i. icount[0]+=n. icount;
8:else
9:i. icount[1]+=n. icount;
10:if n 有子节点 then 把 n 追加到一个新的根集 nRS 中;
11: if n. mps<i. mps then i. mps=n. mps;
12:end if
13:end loop
14:if i. conut[0]≥min[MIS(β)×|S|,i. mps×|S|] then
15: i 是 β 最后一个集合元素形成 β′;
16:if i. count[0]≥MIS(β′)×|S| then β′作为一个频繁模式输出
17: MSCP-growth(nRS,β′,MIS(β′));
18:end if
19: if i. icount[1]≥min[MIS(β)×|S|,i. mps×|S|] then
20:节点 i 被加入到 β 形成 β′;
21:if i. icount[1]≥MIS(β′)×|S| then
22:β′作为一个频繁模式输出;
23:MSCP-growth(nRS,β′,MIS(β′));
24:end if
25:end loop
```

图 3 基于 MMSs 的 MSCP-growth 算法

2.2.2 MSCP-growth 算法分析

假设 T 表示完整的 PLMS-tree,L_{MIN_1} 表示完整的 k-MIN 序列模式和 L_k 表示完整的 k 序列模式。MSCP-growth 算法的挖掘过程如下:

1. 在 T 结构中找到所有的 L_{MIN_1} 和 L_1。

2. 对每个 $L_{\mathrm{MIN}_{k-1}}$ 中的序列模式 $\beta(k \geqslant 2)$

2.1 在 T 中遍历 β' 条件后缀,并用 $T|\beta$ 来表示并且计算 L_{MIN_1} 中每个数据项的个数。

2.2 如果在数据项被追加到 β(不论是在 β 最后的数据集加入数据项还是在 β 最后数据集中加入数据集)形成 β',若其支持度大于 MIN,则可以获得 L_{MIN_k} 频繁模式 β'。

2.3 如果所有频繁模式 β' 的支持度不小于 MIS(β'),则把所有在 L_{MIN_k} 中的频繁模式 β 加入到 L_k 中。

3. 在 $T|\beta'$ 中以前缀 β' 来递归地寻找 $L_{MIN_{k+1}}$ 频繁模式。

在上述 1 和 2.2 中,L_{MIN_1} 和 $L_{MIN_k}(k \geq 2)$ 每个频繁序列模式都会满足 MIN 的值,所以根据上述 1 和 2.3,可以得知在 L_1 和 $L_k(k \geq 2)$ 也是频繁的,所以算法 MSCP - growth 是正确的。

在 L_{MIN} 中的每个序列模式都可以被 MSCP - growth 算法发现,假设$(i_1, i_2, \cdots, i_m)$ 表示 L_{MIN} 中完整的数据项目集,并它分成 m 个子集分别为:i_1 的条件子集,i_2 的条件子集,…,i_m 的条件子集等。对于 L_{MIN} 中的每一项 i_q,从树 T 中遍历 i_q 的条件后缀 PLMS - tree 树,用 $T|i_q$ 来表示。

可以用归纳法来证明:如果 $k = 1$,则表示在每个模式中只有一个数据项,所以根据上述 1 中所以在 L_{MIN_1} 中的频繁项都可以找到;归纳步骤:假设算法 MSCP - growth 可以在 L_{MIN} 中找到不多于$(k-1)$ 个频繁项$(L_{MIN_1} \cup L_{MIN_2} \cup \cdots \cup L_{MIN_{k-1}})$。对于给定的序列模式 L_{MIN_k},即 $x = <iq_1, iq_2, \cdots, iq_k>$ 以各个数据项相对应的 iflag = $<iq_1.iflag, iq_2.iflag, \cdots, iq_k.iflag>$ 排序,因为 $x \in L_{MIN_k}$,所以 iq_k 的支持度不会小于 MIN 值。

所以在上述中的 2.1 中算法可以遍历 $<iq_1, iq_2, \cdots, iq_{(k-1)}>$ 的条件后缀 PLMS - tree 树的 $T|<iq_1, iq_2, \cdots, iq_{(k-1)}>$。所有 $iq_{(k-1)}$ 的信息都将被保存在 $T|<iq_1, iq_2, \cdots, iq_{(k-1)}>$ 中,基于算法的 2.2 步。算法可以通过 $T|<iq_1, iq_2, \cdots, iq_{(k-1)}>$ 找到 $x' = <iq_1, iq_2, \cdots, iq_{(k-1)}>$,最后 iq_k 通过追加到 x' 中形成 x。可知算法可以找出完整的 L_{MIN} 频繁模式,每个序列模式都必须在 L_{MIN} 中并且如果 L_{MIN} 中一个序列模式满足其 MIN 值,那么算法可以挖掘出在 L_{MIN} 中的所有频繁序列模式数据项。

3　实验结果及分析

3.1　数据和参数描述

实验采用模拟数据集评估 MSCP-growth 算法的性能,所有相关的数据项目集是以 IBM 数据项目集(由 Agrawal 和 Srikant 于 1995 年提出的)[1] 为标准的,这个数据集被广泛地用于测试序列模式挖掘 SPM 算法的性能。基于 MSCP-growth 算法可以对每个数据项设置不同的支持度,为了在实验中比较容易地获得 MIS 的值,采用文献[6]提出的方法,其把实际的频繁项作为 MIS 值分配的基础,其公式如下:

$$MIS(i) = \begin{cases} M(i) & M(i) \geq MINS \\ MIN & otherwise \end{cases}$$

$$M(i) = \sigma \times f(i) \quad 0 \leq \sigma \leq 1$$

其中,$f(i)$ 代表时间项 i 在序列数据库中出现的次数;MIN 代表的是在所有项中最小的 MIS 值;$\sigma(0 \leq \sigma \leq 1)$ 表示在挖掘的过程中用来控制 MIS 值的效果,如果 σ 设置为 0 的话,所有项目将具有相同的 MIS 值并且所挖掘的模式与传统的序列模式是完全相同的。

实验在 PC 上进行,配置如下:CPU 2.33 GHz 英特尔酷睿双核;内存 4 GB;操作系统 XP SP3 内存:2 GB。编程语言:C++;IDE:Dev-C++ 4.9.9.2;实验参数如下:$|D|$ 表示客户的数量(以千为单位);$|C|$ 表示每个客户的交易的平均数;$|T|$ 表示每个交易项的平均数据项;$|S|$ 表示最大潜在序列的平均长度(4);$|i|$ 表示最大潜在序列的项目集的大小(1.25);N_s 表示最大潜在序列的个数(5 000);N_t 表示最大潜在序列的数目(25 000);N 表示数据集的个数(10 000)。

3.2　实　验

图 4 对比采用基于 PLWAP-tree 结构存储数据项目集的 PLWAP-Mine 算法与基于 PLMS-tree 结构存储数据项目集的 MSCP-growth 算法的执行时间。

实验结果表明,随着客户数量 $|D|$ 的增加,PLWAP-Mine 算法执行时间增长比 MSCP-growth 算法要明显的多。当客户数量级达到 500 k 的数量级时,MSCP-growth 算法的优势更明显,PLWAP-Mine 执行时间比 MSCP-growth 要多出近两倍。随着交易的平均数和交易项的平均数据项的增加,也可以发现类似的模式。

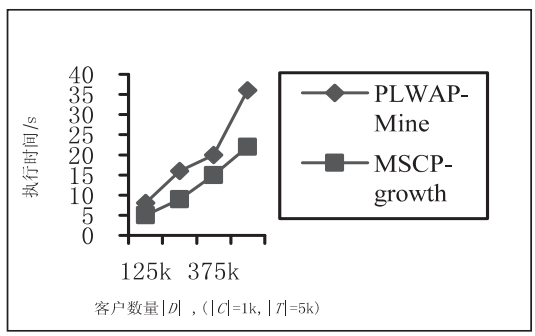

图 4　PLWAP-Mine 算法与 MSCP-growth 算法挖掘执行时间的性能比较

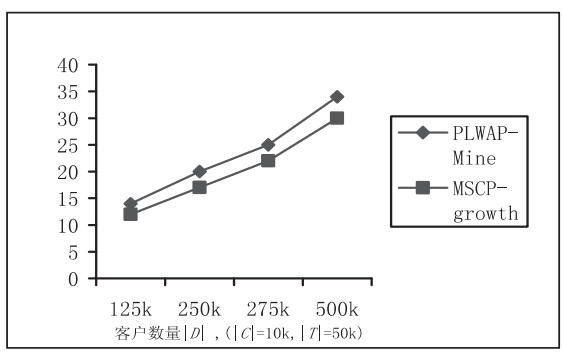

图 5　PLWAP-Mine 算法与 MSCP-growth 算法所用内存的比较

图 5 对比了两个算法在挖掘的过程中对内存的使用情况。在客户数量级比较小的时候,所用内存相差

不是很大,因为在数量级比较少的时候,符合条件的序列模式数据项目集相差不多,所占用的内存相差不大。但随着数量级越来越大,其所占用的内存就有差异了,这是因为 PLWAP-Mine 算法在执行的时候需要对内存的数据进行单个的支持度配备,并且是串行的进行数据项目最小支持度 MIS,这需要耗费大量的时间和空间。

而 MSCP-growth 算法是基于 PLMS-tree 结构的,其算法初始就开始对项目集进行频繁挖掘,在这个过程中剔除了一些不是频繁的项目集,这就大大减少了 MSCP-growth 算法要对序列项目集进行配对的阈值,节省了很多查找以及配对时间以及空间的耗费量。

综上所述,随着客户数量级的增加,传统的 PLWAP-Mine 算法执行时间比 MSCP-growth 算法要明显的多,并且其挖掘出的稀有数据项目集要比传统的算法多而符合所期望的,在空间的使用上也比传统算法要节省很多。

实验结果验证了算法的有效性,对序列模式下的数据项目集挖掘的时间效率和空间效率有明显的提高。

4 结束语

基于 PLWAP 的概念及性质提出了 PLMS-tree,并结合多重最小支持度(MMSs),给出了基于树结构多重最小支持度的序列模式挖掘算法(MSCP-growth)。使用 PLMS-tree 存储和排序整个数据库,能大幅度地减少搜索空间,从而提高 MSCP-growth 查询的效率。文中的算法可以在有效合理的时间内解决典型的序列模式挖掘问题,实验结果验证了算法的有效性。优化存储结构和 MMSs 权重因子的序列模式查询和挖掘将是一个研究趋势,笔者下一步将结合结构存储研究 MMSs 的序列模式挖掘。

参考文献:

[1] Agrawal R, Srikant R. Mining sequential patterns[C]//Proceedings of the 11th international conference on data engineering. Taipei, Taiwan: IEEE, 1995.

[2] Han Jiawei, Pei Jian, Yin Yiwen, et al. Mining frequent patterns without candidate generation: a frequent-pattern tree approach[J]. Data Mining and Knowledge Discovery, 2004, 8(1): 53-87.

[3] Liu Bing, Hsu W, Ma Yiming. Mining association rules with multiple minimum supports[C]//Proceeding of the 5th ACM SIGKDD international conference on knowledge discovery and data mining. New York, NY, USA: ACM, 1999: 337-341.

[4] Pei Jian, Han Jiawei, Mortazavi-Asl B, et al. Mining access patterns efficiently from web logs[C]//Proc of the 4th Pacific-Asia conference on knowledge discovery and data mining, current issues and new applications. London, UK: Springer-Verlag, 2000: 396-407.

[5] Ezeife C I, Lu Y. Mining web log sequential patterns with position coded preorder linked WAP-tree[J]. Data Mining and Knowledge Discovery, 2005, 10(1): 5-38.

[6] Liu Bing. Web data mining: exploring hyperlinks, contents and usage data[M]. Berlin: Springer, 2006.

[7] Lui C L, Chung F L. Discovery of generalized association rules with multiple minimum supports[J]. LNCS, 2000, 1910: 510-515.

[8] Tseng M C, Lin W Y. Efficient mining of generalized association rules with non-uniform minimum support[J]. Data and Knowledge Engineering, 2007, 62(1): 41-64.

[9] Hu Y H, Chen Y L. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism[J]. Decision Support Systems, 2006, 42(1): 1-24.

[10] Hu Yahan, Wu Fan, Liao Yichun. Sequential pattern mining with multiple minimum supports: a tree based approach[C]//Proc of 2nd international conference on software engineering and data mining. Chengdu: IEEE, 2010: 428-433.

[11] Chen S S, Huang T C K, Lin Z M. New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports[J]. Journal of Systems and Software, 2011, 84(10): 1638-1651.

[12] Chiang D A, Wang Y H, Chen S P. Analysis on repeat-buying patterns[J]. Knowledge-based Systems, 2010, 23(8): 757-768.

[13] 王振宇,白石磊,熊范纶. 多最小支持度策略的关联规则挖掘方法[J]. 小型微型计算机系统,2002,23(8):971-973.

[14] 段 军,戴居丰. 基于多支持度的挖掘加权关联规则算法[J]. 天津大学学报,2006,39(1):114-118.

[15] 冯玉才,冯剑琳. 关联规则的增量式更新算法[J]. 软件学报,1998,9(4):301-306.

[16] 宋中山,成林辉,吴立峰. 一种基于关联规则的增量数据挖掘算法[J]. 湖北大学学报:自然科学版,2006,28(3):240-243.

[17] 温 蕴. 一种基于阈值变化的增量式更新算法研究[J]. 计算机科学,2008,35(8):247-248.

[18] 钟晓桢. 基于 Apriori 和 IUA 的改进算法[J]. 江汉大学学报:自然科学版,2007,35(3):59-63.

[19] 周晓苏. 基于关联规则分析的微利企业利润质量评价研究[J]. 会计研究,2004(2):52-57.

基于树结构多重最小支持度的挖掘算法研究

作者： 占美星， 杨颖， 杨磊， ZHAN Mei-xing， YANG Ying， YANG Lei

作者单位： 占美星,杨颖,ZHAN Mei-xing,YANG Ying(广西大学 计算机与电子信息学院,广西 南宁,530004)， 杨磊,YANG Lei(广西科学院 应用物理研究所,广西 南宁,530003)

刊名： 计算机技术与发展 ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(8)