第 14 卷 第 28 期 2014 年 10 月
1671—1815(2014)28-0231-05

科 学 技 术 与 工 程
Science Technology and Engineering

Vol. 14 No. 28 Oct. 2014
ⓒ 2014 Sci. Tech. Engrg.

基于前缀共享树的频繁情节挖掘算法

丁 勇[1] 朱辉生[2] 高广银[1]
(南京理工大学泰州科技学院[1];泰州学院计算机科学与技术学院[2],泰州 225300)

摘 要 经典的频繁情节挖掘算法 NONEPI 及其改进算法 NONEPI + 存在时空复杂度高、“重复计算”等问题,基于最小且非重叠发生的支持度定义,提出一个基于前缀共享树的频繁情节挖掘算法 PST_NONEPI,该算法采用深度优先搜索策略,将发现的频繁情节压缩到前缀共享树中,通过动态维护前缀共享树来发现所有的频繁情节。该算法只需扫描事件序列一次,大大提高了频繁情节挖掘的效率。实验证明,PST_NONEPI 算法能有效地挖掘频繁情节。

关键词 事件序列 频繁情节 最小且非重叠发生 前缀共享树
中图法分类号 TP311; **文献标志码** A

情节是由若干事件类型组成的序列,挖掘频繁情节可以发现事件类型之间的紧随关系,从而揭示用户或系统潜在的行为模式。为了解决频繁情节挖掘问题,Manilla 等人[1]首先引入了情节的概念,并提出了两个经典算法 WINEPI[2]、MINEPI[3]。其中,WINEPI 基于滑动窗口来定义支持度,而 MINEPI 是基于情节的最小发生来定义支持度。这些算法在计算支持度时都可能包含了情节的多次重叠发生,从而导致“重复计数”问题。为此 Laxman 等人[4]引入了“非重叠发生”的概念来计算一个情节的支持度,并且提出了一个快速的频繁情节挖掘算法 NONEPI[5],该算法通过采用有限状态机来跟踪情节的状态变化从而达到计数的目的,不足之处,该算法与 Apriori 算法类似,需要多遍扫描事件序列,并产生大量候选情节。改进算法 NONEPI +[5,6]对产生的候选情节预先进行“剪枝”,某种程度上提高了挖掘效率。然而,当事件序列很长时,多遍扫描序列仍然需要较高的空间和时间代价。

为了克服 NONEPI 算法的不足,本文提出一个事件序列上的频繁情节挖掘算法 PST_NONEPI。该算法基于最小且非重叠发生的支持度定义,克服了情节发生的“重复计数”问题;算法通过动态维护前缀共享树来产生频繁情节,节省了存储空间,提高了频繁情节挖掘的效率。

1 相关定义

1.1 事件、事件序列

给定事件类型集合 $\varepsilon = \{E_1, E_2, \cdots, E_n\}$,事件是一个二元组$(E,T)$, E 表示某事件名称,T 表示该事件的发生时间。事件序列 S 是由 n 个事件按其发生的时间先后顺序排列而成的序列,表示为 $S = <(E_1,T_1,),(E_2,T_2),\cdots,(E_n,T_n)>$

1.2 情节、子情节

情节 α 是由若干事件类型组成的序列,表示为 $\alpha = <E_1E_2\cdots E_k>$, $E_i \in \varepsilon$,且对于所有的 i 和 $j(1 \leqslant i < j \leqslant n)$ 满足 E_j 在 E_i 之后发生。情节 α 包含事件的个数称为 α 的长度,记为 $|\alpha|$。情节 $\alpha = <E_1E_2\cdots E_m>$ 和 $\beta = <E_1'E_2'\cdots E_k'>$,若至少存在一个整数序列 $1 \leqslant \gamma_1 < \gamma_2 < \cdots \leqslant \gamma_k$,满足 $E_i' = E_{\gamma i}(1 \leqslant i \leqslant k)$,称 β 是 α 的子情节,记为 $\alpha \supseteq \beta$。

1.3 最小且非重叠发生

给定事件序列 S 和情节 α,若至少存在一个序列 $S' = <(E_1,T_1),(E_2,T_2),\cdots,(E_k,T_k)>$,满足 $S' \subseteq S$ 且 $T_i < T_i + 1(1 \leqslant i < k)$,则称情节 α 在 S 上发生。

设 $[T_s,T_e]$ 是情节 α 在 S 上的一次发生,若不存在另一次发生 $[T_s',T_e']$,使得 $T_s < T_s'$ 且 $T_e' \leqslant T_e$,则 $[T_s,T_e]$ 是 α 在 S 上的一次最小发生。若 $[T_s,T_e]$ 和 $[T_s',T_e']$ 是 α 在 S 上的两次最小发生且 $T_e < T_s'$,则称 $[T_s,T_e]$ 和 $[T_s',T_e']$ 是 α 在 S 上的两次最小且非重叠发生。α 所有最小且非重叠发生记为 $\alpha.\mathrm{mano}$。

1.4 支持度、频繁情节

情节 α 的所有最小且非重叠发生组成的时间戳集合的基数称为支持度,记为 $\alpha.\mathrm{sup}$。给定支持度阈值 min_sup,若 $\mathrm{min_sup} \leqslant \alpha.\mathrm{sup}$,则称 α 是频繁情节。

2014 年 4 月 24 日收到 国家自然科学基金项目(61003001,61103009)资助
第一作者简介:丁 勇(1980—),男,汉,江苏泰州人,硕士,讲师。研究方向:数据库理论与数据挖掘。E-mail:4383526@qq.com。

2 算法 PST_NONEPI

2.1 算法描述

PST_NONEPI 算法首先定义一种特殊的数据结构 PST（prefixsharedtree），用于压缩挖掘的频繁情节，树中每个结点对应一个频繁情节 α，包括属性：α. name 情节的事件名称，α. sup 情节的支持度，α. mano 情节的最小且非重叠发生的时间戳集合，α. child 情节的孩子结点。

PST_NONEPI 算法的基本思想是：首先单遍扫描给定的事件序列 S，从中发现所有的频繁 l 情节 F_1。然后，以生成的 F_k 情节 α 为前缀和 F_1 中每个情节 β 进行串接，形成一个长度为 $k+1$ 的新情节 $\gamma=\alpha+\beta$，这个过程称为“情节增长”，由函数 EpisodeGrow(Episode α)实现。对于 γ 可能的子情节 sub_γ，根据函数 PSTFind(sub_γ) 快速查找其在前缀共享树中是否存在。若存在，则根据函数 ComputeMano(α, β)计算情节 γ 发生的时间戳，时间戳集合的长度$|\gamma.\text{Mano}|$即为 γ 的支持度(γ. sup)，若 γ. sup $<$ min_sup，则 γ 为非频繁情节，无需进行下一轮情节增长；若 γ. sup $\geq$ min_sup，则将 γ 压入 PST 中，并以 γ 为前缀递归进行下一轮情节增长。若不存在，根据情节的 Apriori 性质可以判断 γ 一定不是频繁的，因而无需进行时间戳计算，直接返回上一层。当没有任何情节可以增长时算法终止，代码如下。

```
算法:PST_NONEPI
输入:事件序列 S = <(E1,T1,),(E2,T2),…,(En,Tn)>
      最小支持度 min_sup
输出:前缀共享树 PST
(1) 创建一个根节点 Root;
(2) 扫描事件序列,得到频繁 1 情节 F1
(3) For each α ∈ F1
(4)     EpisodeGrow(α)
(5) Output(Root)

Procedure EpisodeGrow(Episode α)
输入:待增长的情节 α
(1) For each β ∈ F1
(2)      γ = α + β
(3)      If PSTFind(sub_γ ∈ γ)
(4)         γ.Mano = ComputeMano(α, β)
(5)         If |γ.Mano| ≥ min_sup
(6)            Add β to α as a child node
(7)            EpisodeGrow(Episode β)
```

算法的第 4 步，调用过程 ComputeMano(α, β)用于计算情节 γ 的时间戳，γ 为频繁情节 α, β 进行串接后产生的新情节，容易证明 γ. sup $=|\gamma.\text{Mano}|$

```
Procedure ComputeMano(α, β)
输入:情节 α,F1 情节 β
输出:γ.Mano
(1) Timestamp i,j,k
(2) For each i ∈ α.Mano
(3)    j = k
(4)    For each j ∈ β.Mano
(5)       If i.Ts > (j-1).Te and i.Te < j.TS
(6)                          and (i+1).Ts > j.Ts
(7)          add [i.Ts,j.TS] to γ.Mano
(8)          k = j + 1
```

算法关键的一步是对于情节增长操作产生的新情节 γ，如何快速地在 PST 中查找其子情节在 PST 中是否存在，从而避免不必要的时间戳计算过程。

```
Procedure PSTFind(Episode γ)
输入:情节 γ,前缀共享树 PST
输出:True/False
(1) Current = Root
(2) for each sub_γ ∈ γ
(3)    if Current.name.contains(sub_γ.name)
(4)    Current = CurrentNode.NextNode
(5)    else
(6)       return false
(7) return true
```

2.2 算法优化

算法中的 PSTFind(Episode γ)过程的第(2)步：for each sub_$\gamma \in \gamma$ 时间开销是比较大的。如：频繁情节 $\alpha = <ABC>$ 与 $\beta = <D>$ 串接产生 $\gamma = <ABCD>$，则 sub_$\gamma = \{<AB>, <AC>, <AD>, <BC>, <BD>, <CD>, <ABC>, <ABD>, <ACD>, <BCD>\}$，若查找每个子情节在 PST 中是否存在必然带来不必要的时间开销。因为 α 是频繁的，α 的所有子情节必是频繁的，因此 $<AB>$ $<AC>$ $<BC>$ $<ABC>$ 无需在 PST 中查找。因为 $<AD>$. sup $=$ $<ABD>$. sup，$<AD>$. sup $=$ $<ACD>$. sup，$<BD>$. sup $=$ $<BCD>$. sup，若 ABD 是频繁的，则 AD 必然是频繁的，同理若 BCD 是频繁的，则 BD 必是频繁的。此可以约简 sub_$\gamma = \{ABD, ACD, BCD\}$，从而提高了算法执行的效率。可以调用以下过程对候选子情节剪枝。

```
Procedure CutSub(Episode γ)
输入:情节 γ
输出:sub_γ
(1) Set sub_γ = Φ
(2) for i = 1 to γ.len-1
(3)    α = sub(γ.name,1,i-1) + sub(γ.name,i+1, γ.len)
(4)    sub_γ.add(α)
```

2.3 应用实例

给定事件序列

$S = <(A,1),(B,2),(C,4),(A,5),(B,7),(A,9),(B,11)(C,12)>$，设最小支持度 min_sup = 2，算法执行的步骤如下。

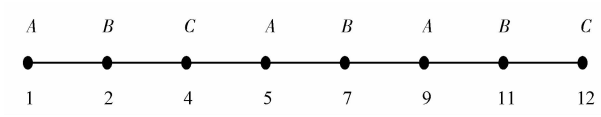


图 1　事件序列

Fig. 1　Event sequence

Step1:构造空的前缀共享树 PST,单遍扫描事件序列,得到 $F_1 = \{ <A> <B> <C> \}$,并将其作为孩子结点压缩到 PST 根结点下。

Stept2:对频繁情节 $\alpha = <A>$ 与 F_1 中的每个情节进行情节增长操作,首先与 $\beta = <A>$ 串接形成情节 $\gamma = <AA>$,在 PST 中快速查找 γ 的子情节 $<A>$,结果存在,所以计算 γ 的时间戳为{[1,5]},因为 $|\gamma.\text{Mano}| < \text{min_sup}$,所以 γ 为非频繁的,无需以 γ 为前缀继续操作。

Stept3:返回上一层,将情节 $\alpha = <A>$ 与 $\beta = <B>$ 进行同样操作,形成情节 $\gamma = <AB>$,计算 $|\gamma.\text{Mano}| \geqslant \text{min_sup}$ 所以 $\gamma = <AB>$ 是频繁的。

Stept4:令 $\alpha = \gamma$,继续以 $\alpha = <AB>$ 为前缀,与 $\beta = <A>$ 进行增长操作,形成情节 $\gamma = <ABA>$,因 γ 的子情节 $\text{sub_}\gamma = <AA>$ 在 PST 中不存在,所以无需计算 γ 的时间戳,直接返回上一层。

Stept5:以此类推,当无任何情节可增长时,算法终止。挖掘过程形成的频繁情节和产生的前缀共享树见表 1 和图 2。

表 1　频繁情节

Table 1　Frequent episodes

序号	频繁情节	最小且非重叠发生时间戳	支持度
1	$<A>$	{[1,1][5,5][9,9]}	3
2	$<AB>$	{[1,2][5,7][9,11]}	3
3	$<ABC>$	{[1,4][9,12]}	2
4	$<AC>$	{[1,4][9,12]}	2
5	$<B>$	{[2,2][7,7][11,11]}	3
6	$<BA>$	{[2,5][7,9]}	2
7	$<BC>$	{[2,4][11,12]}	2
8	$<C>$	{[4,4][12,12]}	2

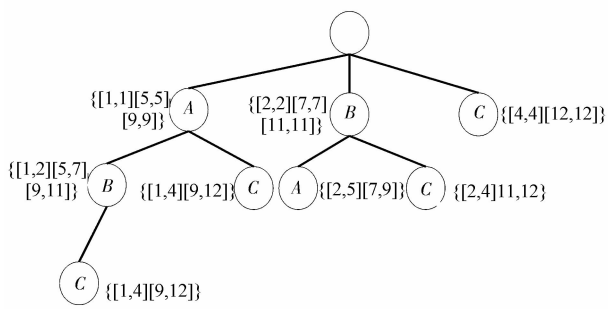


图 2　前缀共享树

Fig. 2　Prefix shared tree

2.4　算法复杂度分析

设事件序列 S,事件类型集合 ε,前缀共享树中压缩的频繁情节个数为 $|\text{PST}|$。

时间复杂度:算法 PST_NONEPI 对前缀共享树中的每个频繁情节和 F_1 中的情节进行增长操作,树中频繁情节个数为 $|PST|$,F_1 的上界为 $|\varepsilon|$,因此增长的次数为 $|PST| * |\varepsilon|$。而完成一次增长需要扫描一次前缀共享树和计算 $(\alpha + \beta)$. Mano,在前缀共享树中快速查找的时间上界为 $O(|\varepsilon|)$;计算 $(\alpha + \beta)$. Mano 需要单遍扫描 α. Mano 和 β. Mano,其时间复杂度为 $O(|S|)$。因此,算法总的时间复杂度为 $O(|PST|.|\varepsilon|.|\varepsilon|.|S|)$。

空间复杂度:前缀共享树共存储 $|PST|$ 个频繁情节,每个情节至少有 min_sup 个最小且非重叠发生,至多有 max_sup 个最小且非重叠发生,因此所需要的空间复杂度为 $O(|PST|.\text{max_sup})$。

3　实验评估

实验采用的硬件环境为 Win7 操作系统、3.4 GHz Intel i3-4130 CPU、4 GB 内存。程序基于Eclipse 平台和 Java 语言实现。

实验数据选择南京理工大学泰州科技学院图书馆中国知网 CNKI 的文献引用记录,下载 Web 服务器上从 2014 年 5 月 1 日至 2014 年 5 月 31 日的日志数据,该数据包括了近万名读者对 12 648 种不同参考文献的引用记录,共 68 461 条。通过对不同的文献进行标号作为事件类型,按照其引用的时间顺序排列为一个事件序列。

图 3 显示随着支持度阈值的增加,PST_NONEPI 算法挖掘的非重叠发生的频繁情节(NONEPI)及最小且非重叠发生的频繁情节(MANEPI)的个数均在减少。但随着持度的增大, NONEPI 逼近于 MANEPI,因此情节个数相近。

图 4 显示 PST_NONEPI 算法比 NONEPI 算法执行效率高。当支持度阈值较小时,因为 PST_NONEPI 算法需要对 PST 进行频繁的深度搜索和大量的时间戳计算,所以相比 NONEPI 算法性能没有太大提升,当支持度较大时,PST_NONEPI 算法的执行效率有明显提升。

图 5 显示随着支持度阈值的增大,两种算法的内存开销均在减小,PST_NONEPI 算法要优于 NONEPI 算法,因为 PST 的搜索及剪枝策略避免了无效的时间戳计算带来的内存开销。

4　结束语

PST_NONEPI 算法主要用于事件序列上最小且

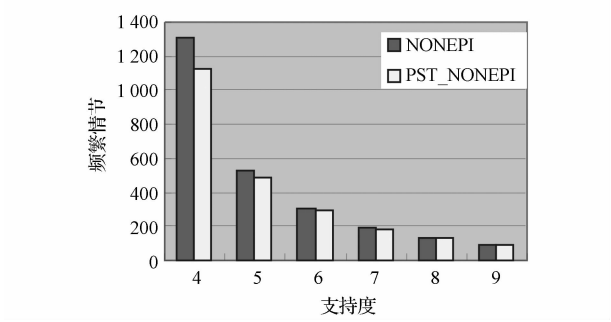


图 3 频繁情节与支持度阈值

Fig. 3 Frequent episode *vs.* min_sup

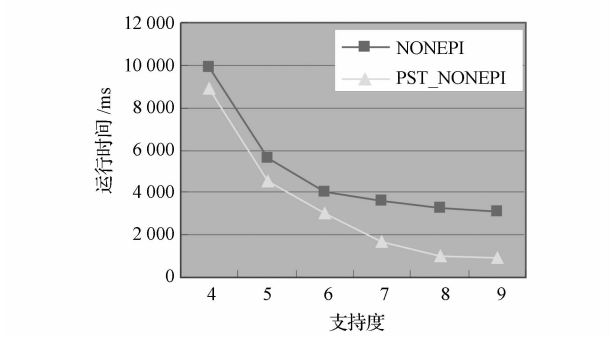


图 4 运行时间与支持度阈值

Fig. 4 Run time *vs.* min_sup

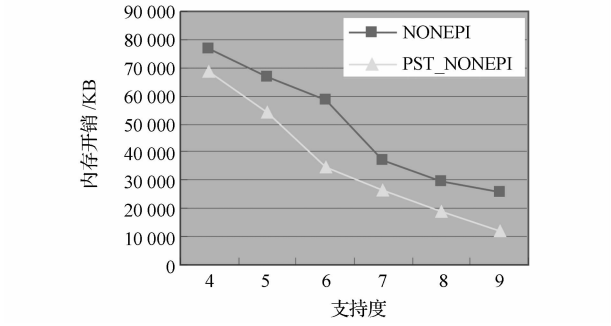


图 5 内存开销与支持度阈值

Fig. 5 Memory *vs.* min_sup

非重叠发生的频繁情节挖掘,算法通过深度优先搜索和时间戳计算进行情节增长操作,从而挖掘所有的频繁情节,避免了重复扫描事件序列。同时,引入前缀共享树这种数据结构来压缩已发现的频繁情节,并通过快速查找前缀共享树减少了冗余的时间戳计算过程,从而提高了算法的执行效率。理论和实验证明 PST_NONEPI 算法能有效地挖掘事件序列上的频繁情节。

参 考 文 献

1 Manilla H, Toivonen H, Verkamo A. Discovering frequent episodes in sequences. Proceedings of the First International Conference on Knowledge Discovery and Data Mining,1995: 210—215

2 Mannila H, Toivonen H, Verkamo A I. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1997; 1(3): 259—289

3 Mannila H, Toivonen H. Discovering generalized episodes using minimal occurrences. KDD, 1996;96: 146—151

4 Laxman S, Sastry P S, Unnikrishnan K P. Discovering frequent episodes and learning hidden markov models: a formal connection. Knowledge and Data Engineering, IEEE Transactions, 2005; 17 (11): 1505—1517

5 Zhou W, Liu H, Cheng H. Mining closed episodes from event sequences efficiently. Advances in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2010: 310—318

6 Zhu H, Wang P, He X, *et al*. Efficient episode mining with minimal and non-overlapping occurrences. Data Mining (ICDM), 2010 IEEE 10th International Conference, IEEE, 2010: 1211—1216

7 朱辉生, 汪 卫, 施伯乐. 基于最小且非重叠发生的频繁闭情节挖掘. 计算机研究与发展, 2013;50(4): 852—860
Zhu Huisheng, Wang Wei, Shi Baile. Frequent closed episode mining on minimal and non-overlapping occurrences. Research and Development of Computer, 2013; 50(4): 852—860

8 朱辉生, 汪 卫, 施伯乐. 基于频繁闭情节及其生成子的无冗余情节规则抽取. 计算机学报, 2012, 1: 53—63
Zhu Huisheng, Wang Wei, Shi Baile. Extracting non-redundant episode rules based on frequent closed episodes and their generators. Chinese Journal of Computers, 2012; 1: 53—63

(下转第 246 页)

Wright Jr R S, Lipchak B. OpenGL superBible. Beijing: Post & Telecom Press, 2005: 3—187

10 Donald M, Pauline Baker. Computer graphics with OpenGL(Third Edition). Beijing: Publishing House of Electronics Industry, 2005: 175—185

The Lithologic Pattern Filling of Drilling Histogram

WANG Jia-jia, TU Wen-sen, XIAO Liang

(School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, P. R. China)

[**Abstract**] Drilling histogram is commonly used in GIS tools, and lithologic pattern is needded when drawing the histogram. Lithologic pattern is one to one correspondence with the lithologic name. A new method of vector pattern fill is mainly proposed aims at the characteristics of drilling histogram. The pattern is decomposed into several type of lines, then drawing the line according to the parameters of the line. At the same time, given to the complexity of layer, it also discusses a composite vector pattern filling algorithm which considers the scaling factor relationship between patterns. This algorithm can fill several patterns into one area. The experimental result about the histogram of zijin northwest mine, is fully proved the high efficiency of the algorithms.

[**Key words**] drilling histogram lithologic pattern vector composite pattern

(上接第 234 页)

Algorithm for Mining Frequent Episodes Based on the Prefix Shared Tree

DING Yong[1], ZHU Hui-sheng[2], GAO Guang-yin[1]

(Taizhou College of Science and Technology[1];
School of Computer and Science, Taizhou College[2], NJUST, Taizhou 225300, P. R. China)

[**Abstract**] Algorithm NONEPI and its improved algorithm NONEPI + to find non-overlapped frequent episodes exist some defects such as high complexity and "over computing", etc. In this paper, support based on minimal and non-overlapped occurrence is definited, presents an algorithm called PST_NONEPI for mining frequent episodes based on the prefix shared tree, the algorithm uses a depth-first search strategy, compress frequent episodes have been found to the prefix shared tree, to discover all frequent episodes by maintaining the prefix shared tree. the algorithm only needs to scan the event sequences once, which improves the efficiency of mining frequent episodes. Experiments show that, PST_NONEPI algorithm can effectively mine frequent episodes.

[**Key words**] event sequence frequent episode minimal and non-overlapped occurrence prefix shared tree