第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)03—0028—03　　文献标识码：A　　中图分类号：TP311

缩减投影数据库规模的增量式序列模式算法

刘佳新[a]，严书亭[b]，任家东[a]

(燕山大学 a. 信息科学与工程学院；b. 科学技术研究院，河北 秦皇岛 066004)

摘　要：在增量式序列模式挖掘算法中，数据库更新只有插入和扩展 2 种操作，未考虑序列删除的情况。为此，提出一种基于频繁序列树的增量式序列模式更新算法(IUFST)。在数据库和支持度发生变化时，IUFST 算法分不同情况对频繁序列树进行更新操作，缩减投影数据库的规模，提高算法效率。实验结果表明，该算法在时间性能上优于 PrefixSpan 算法和 IncSpan 算法。

关键词：数据挖掘；增量式挖掘；序列模式；投影数据库；频繁序列树；深度优先

Incremental Sequential Pattern Algorithm of Reducing Projected Database Size

LIU Jia-xin[a], YAN Shu-ting[b], REN Jia-dong[a]

(a. College of Information Science and Engineering; b. Science and Technology Administration Office, Yanshan University, Qinhuangdao 066004, China)

【Abstract】This paper proposes an incremental sequential patterns updating algorithm based on frequent sequence tree, called IUFST, in order to solve the problem that when the database is updated, the existed incremental mining algorithms of sequential patterns only mention two kinds of database updates, insert and append rather than the delete operation. When the database is updated and the support is changed, IUFST is divided into four kinds of situations to update the frequent sequence tree. It reduces the size of the projected database and improves the efficiency. Experimental results show that IUFST outperforms PrefixSpan and IncSpan in time cost.

【Key words】data mining; incremental mining; sequential pattern; projected database; frequent sequence tree; depth-first

DOI: 10.3969/j.issn.1000-3428.2012.03.010

1　概述

序列模式挖掘是数据挖掘[1]的一个重要研究领域，最初是由文献[2]提出来的。在许多应用中，序列数据库是增量更新的，为了提高挖掘效率，一般通过增量式更新算法对数据库中新增数据序列进行挖掘。文献[3]提出一种基于投影的增量式挖掘算法 IncSpan。但是 IncSpan 算法有一些缺陷，因此不能找出完备的序列模式。文献[4]通过证明 IncSpan 算法中基本属性的不正确来阐明这些缺陷，并对 IncSpan 进行改进，提出一种新的算法 IncSpan+。文献[5]提出一种有效的基于约束的增量序列模式挖掘算法 ISC。该算法使用格结构存储先前挖掘结果，通过扫描数据库一次把数据集转化成二进制字符串图。文献[6]提出 RePL4UP 算法和 PL4UP 算法，这 2 种算法能快速更新老的频繁模式，不需要重新扫描整个更新后的数据库。

在数据库更新时，现有的增量式序列模式挖掘算没有考虑到序列的删除情况，并且这些算法不能充分利用上一次挖掘的结果，当支持度发生变化时，需要对数据库进行重新挖掘。因此，本文提出一种基于频繁序列树的增量式序列模式更新算法(IUFST)，用于解决序列从数据库中删除时，如何找到新的序列模式。

2　频繁序列树结构

频繁序列树是一棵前缀树，频繁序列树中存储数据库中满足频繁序列树支持度阈值的所有序列模式及其支持度信息。频繁序列树的构造过程与在数据库中使用 PrefixSpan 算法挖掘序列模式的过程是相似的。把每一次在投影数据库中挖掘出的所有频繁项作为孩子结点，插入到以投影数据库前缀的最后一项为父亲结点的频繁序列树中。下面给出频繁序列树的定义。

定义　频繁序列树的根节点包含一个属性，用于存储频繁序列树支持度阈值。除了根结点，频繁序列树中每个结点都包含 2 个属性，分别存储数据库中满足频繁序列树支持度阈值的序列模式及其支持度。从根结点的孩子结点到任何一个叶结点的路径都代表数据库中的一个序列模式，其支持度等于叶结点的支持度。频繁序列树中任何结点的支持度都不小于其子结点的支持度。

本文用一个实例来说明频繁序列树结构。为简单起见，本文假设每一个元素中仅包含一个项，当元素包含多个项时，可以使用相似方法进行推论得到结果。序列数据库 D 如表 1 所示。假设频繁序列树支持度阈值为 5。序列数据库 D 的频繁序列树如图 1 所示。

表 1　序列数据库 D

序列号	序列
1	F A M C
2	A M B C F
3	A F K M C
4	A M C F
5	A M C
6	A M C J F

基金项目：河北省教育厅科学研究计划基金资助项目(2008498)；河北省自然科学基金资助项目(F2010001298)；秦皇岛市科学技术研究与发展计划基金资助项目(201001 A018)；

作者简介：刘佳新(1978－)，女，博士研究生，主研方向：数据挖掘；严书亭，硕士；任家东，教授、博士生导师

收稿日期：2011-06-03　**E-mail：**ljx@ysu.edu.cn

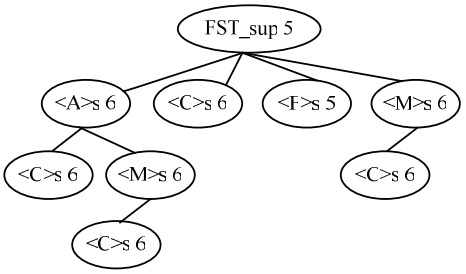


图 1　表 1 中序列数据库 D 的频繁序列树

从图 1 可以看出，序列数据库 D 中满足频繁序列树支持度阈值的序列模式及其支持度为{<A>:6, <C>:6, <F>:5, <M>:6, <AC>:6, <AM>:6, <MC>:6, <AMC>:6}。

由于频繁序列树中存储了数据库中满足频繁序列树支持度阈值的所有序列模式及其支持度，因此 IUFST 算法能够充分利用上一次的挖掘结果，在最小支持度不小于频繁序列树支持度阈值的情况下，当支持度发生变化时，IUFST 算法无需对数据库进行重复挖掘，只需通过深度优先遍历频繁序列树就能找到所有序列模式。文献[7]给出了频繁序列树的构造算法 Con_FST。先用 Con_FST 构造原有数据库的频繁序列树，当数据库更新时，再使用 IUFST 算法处理删除序列。

3　基于频繁序列树的增量式序列模式更新算法

在数据库更新时，现有的增量式序列模式挖掘算法只提到序列的插入和扩展 2 种情况，没有考虑到序列的删除情况。为了解决序列的删除问题，并且使算法能够充分利用上一次挖掘的结果，本文提出一种基于频繁序列树的增量式序列模式更新算法，称为 IUFST。IUFST 是一种基于投影的增量式序列模式挖掘算法，其主要思想是通过对频繁序列进行更新操作，找到满足支持度的所有序列模式。

当数据库和支持度发生变化时，IUFST 算法分以下 4 种情况对频繁序列树进行更新：

(1)数据库发生变化，最小支持度不小于频繁序列树支持度阈值。

(2)数据库发生变化，最小支持度小于频繁序列树支持度阈值。

(3)支持度发生变化，最小支持度不小于频繁序列树支持度阈值。

(4)支持度发生变化，最小支持度小于频繁序列树支持度阈值。

算法 1　IUFST(D, db, min_sup, FST)

输入 原有数据库 D，删除数据库 db，最小支持度 min_sup，频繁序列树 FST

输出 更新后的频繁序列树 FST，更新后的数据库 D，更新后数据库的频繁序列集 FS’

Stpe1 If FST 为空

Stpe2　　Con_FST(, D, min_sup, FST);

Stpe3　ElseIf db 不为空

Stpe4　　If min_sup >= FST_sup

Stpe5　　FST_updated(D, db, min_sup, FST);

Stpe6　　ElseIf min_sup < FST_sup

Stpe7　　对 D 构造投影数据库，找到支持度不小于 min_sup 并且小于 FST_sup 的所有序列模式及其支持度，存储到 FST 中;

Stpe8　　FST_sup = min_sup;

Stpe9　　FST_updated(D, db, min_sup, FST);

Stpe10　ElseIf db 为空

Stpe11　　If min_sup ⩾ FST_sup

Stpe12　　遍历 FST，找到 FS’;

Stpe13　ElseIf min_sup < FST_sup

Stpe14　　对 D 构造投影数据库，找到支持度不小于 min_sup 并且支持度小于 FST_sup 的所有序列模式及其支持度，存储到 FST 中;

Stpe15　　FST_sup = min_sup;

Stpe16　　遍历 FST，找到 FS’;

Stpe17　Return;

算法 2　FST_updated(D, db, min_sup, FST)

输入 D, db, min_sup, FST

输出 更新后的频繁序列树 FST，更新后的数据库 D，更新后数据库的频繁序列集 FS’

Stpe1 对 db 构造投影数据库，找到满足频繁序列树支持度阈值的所有序列模式及其支持度，FS_db;

Stpe2 把 FS_db 中的序列模式及支持度从 FST 中删除;

Stpe3　D=D – db;

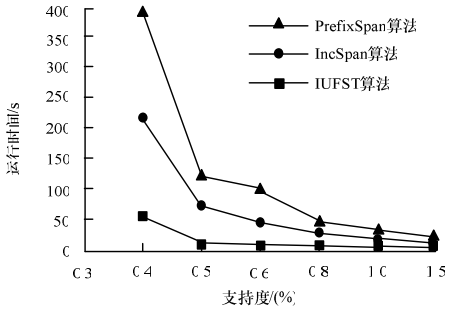
Stpe4　遍历 FST，找到 FS’;

Stpe5　Return;

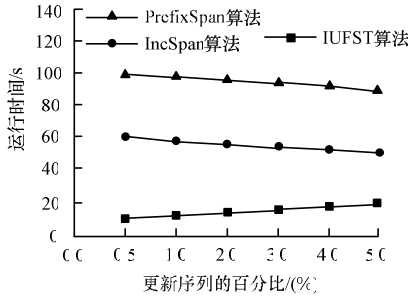
频繁序列树结构在 IUFST 算法中的引用，使 IUFST 算法能够充分利用先前的挖掘结果。在最小支持度不小于频繁序列树支持度阈值的情况下，当数据库发生变化时，IUFST 算法不需要对原有数据库构造投影数据库，只需要对删除数据库构造投影数据库，大大减小了投影数据库的规模。当支持度发生变化时，IUFST 算法不需要对数据库进行重复挖掘，只需通过遍历频繁序列树就能找到所有序列模式。

4　实验结果与性能分析

在实验中，对 IUFST、IncSpan 和 PrefixSpan 算法的性能进行比较。实验运行环境为 1.5 GHz 奔腾 CPU、1 GB 内存、操作系统为 Windows XP。所有算法使用 Microsoft Visual Studio 2005 和 Microsoft SQL Server 2005 实现。采用的数据集是人工合成数据，用字母来标记项目。项目集包含 26 个元素，数据集的大小为 2×10^5。3 种算法性能分析如图 2 所示。



(a)支持度变化



(b)更新序列的百分比变化

图 2　3 种算法性能分析

图 2(a)给出当删除序列数目是原有数据库序列总数 1%的情况下，当支持度发生变化时 3 种算法的运行时间。假设数据库的频繁序列树已经存在，并且频繁序列树支持度阈值

为10。从图中看出，当 min_sup=0.4%时，由于 PrefixSpan 和 IncSpan 算法需要构造大量的投影数据库，因此算法运行的时间较长。IUFST 算法不需要对原有数据库构造投影数据库，只需要对删除数据库构造投影数据库，大大减小了投影数据库的规模。因此，IUFST 算法运行速度比 PrefixSpan 算法和 IncSpan 算法快。当支持度变大时，由于 PrefixSpan 算法和 IncSpan 算法不具有增量挖掘功能，需要重新构造投影数据库，但由于投影数据库的规模大大减小，因此算法的运行速度有了明显的提高。IUFST 算法采用频繁序列树结构作为序列存储结构，频繁序列树中存储数据库中满足频繁序列树支持度阈值的所有序列模式，由于支持度大于频繁序列树支持度阈值，因此 IUFST 算法无需对数据库进行重新挖掘，只需要遍历频繁序列树就能找到所有的序列模式，算法的运行时间只需要几秒钟。因此，IUFST 算法的运行速度最快。

图 2(b)给出在 min_sup=0.5%时，数据库中删除序列的百分比发生变化时 3 种算法所受到的影响。由于 PrefixSpan 算法和 IncSpan 算法不具有增量挖掘功能，因此当数据库发生变化时，需要对整个数据库重新构造投影数据库，但是由于删除序列的增加，需要构造投影数据库的规模变小，因此算法的运行时间变小。当数据库发生变化时，由于 IUFST 算法只对删除数据库中的序列构造投影数据库，大大减小了投影数据库的规模，提高了算法的效率。因此，IUFST 算法的运行速度最快。

5 结束语

本文提出了一种基于频繁序列树的增量式序列模式更新算法(IUFST)。IUFST 使用频繁序列树结构作为序列存储结构，频繁序列树中存储满足频繁序列树支持度阈值的所有序列模式及其支持度信息。当数据库和支持度发生变化时，IUFST 算法分 4 种情况对频繁序列树进行更新，最后通过深度优先遍历频繁序列树找到所有序列模式。频繁序列树的结构特点使 IUFST 算法能够充分利用先前挖掘的结果，在最小支持度不小于频繁序列树支持度阈值的情况下，当数据库发生变化时，IUFST 算法不需对原有数据库构造投影数据库，只需对删除数据库中的序列构造投影数据库，大大减小了投影数据库的规模。当支持度发生变化时，IUFST 算法不需要对数据库进行重复挖掘，只需通过深度优先遍历序列树就能找到满足支持度的所有序列模式。通过实验表明了 IUFST 算法在时间性能上优于 PrefixSpan 算法和 IncSpan 算法。

参考文献

[1] Zou Xiaohong, Zhao Li, Guo Jingfeng, et al. An Advanced Algorithm of Frequent Subgraph Mining Based on ADI[J]. ICIC Express Letters, 2009, 3(3): 639-644.

[2] Agrawal R, Srikant R. Mining Sequential Patterns[C]//Proc. of the 11th International Conference on Data Engineering. [S. 1.]: IEEE Press, 1995.

[3] Cheng Hong, Yan Xifeng, Han Jiawei. IncSpan: Incremental Mining of Sequential Patterns in Large Database[C]//Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, USA: [s. n.], 2004.

[4] Nguyen S N, Sun Xingzhi, Orlowska M E. Improvements of IncSpan: Incremental Mining of Sequential Patterns in Large Database[C]//Proc. of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Hanoi, Vietnam: [s. n.], 2005.

[5] Ren Jiadong, Sun Yafei, Guo Sheng. Incremental Sequential Pattern Mining Based on Constraints[J]. Journal of Computational Information Systems, 2008, 4(2): 571-576.

[6] Ezeife C I, Liu Yi. Fast Incremental Mining of Web Sequential Patterns with PLWAP Tree[J]. Data Mining and Knowledge Discovery, 2009, 19(3): 376-416.

[7] Liu Jiaxin, Yan Shuting, Ren Jiadong. The Design of Frequent Sequence Tree in Incremental Sequential Patterns Mining[C]//Proc. of International Conference on Software Engineering and Service Sciences. Beijing, China: [s. n.], 2011.

编辑 索书志

(上接第 21 页)

参考文献

[1] Ammann P, Wijesekera D, Kaushik S. Scalable, Graph-based Network Vulnerability Analysis[C]//Proc. of the 9th ACM Conference on Computer and Communications Security. New York, USA: ACM Press, 2002.

[2] Ou Xinming, Govindavajhala S, Appel A. MulVAL: A Logic-based Network Security Analyzer[C]//Proc. of the 14th USENIX Security Symposium. Baltimore, USA: [s. n.], 2005.

[3] Ingols K, Lippmann R, Piwowarski K. Practical Attack Graph Generation for Network Defense[C]//Proc. of the 22nd Annual Computer Security Applications Conference. Miami Beach, USA: [s. n.], 2006.

[4] 宋舜宏, 陆余良, 夏 阳, 等. 基于贪心策略的网络攻击图生成方法[J]. 计算机工程, 2011, 37(2): 126-128.

[5] Ou Xinming, Boyer W F. A Scalable Approach to Attack Graph Generation[C]//Proc. of the 13th ACM Conference on Computer and Communications Security. New York, USA: ACM Press, 2006.

[6] Wang Lingyu, Tania I. An Attack Graph-based Probabilistic Security Metric[C]//Proc. of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security. London, UK: [s. n.], 2008.

[7] Homer J, Ou Xinming, Schmidt D. A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks[EB/OL]. (2010-11-21). http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.157.4048.

[8] 叶 云, 徐锡山, 贾 焰, 等. 基于攻击图的网络安全概率计算方法[J]. 计算机学报, 2010, 33(10): 1987-1996.

编辑 刘 冰