第 33 卷 第 6 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.6 Computer Engineering March 2007
·软件技术与数据库· 文章编号：1000—3428(2007)06—0053—03 文献标识码：A 中图分类号：TP312

IDS 日志数据挖掘的改进算法

崔 玮，刘建伟，张其善
（北京航空航天大学电子信息工程学院，北京 100083）

摘 要：提出了一个基于最小完美哈希函数的关联规则的挖掘算法。基于 Apriori 的算法，在综合了传统哈希剪枝技术的同时，利用最小完美哈希函数的优点，保证了静态数据库关联规则挖掘，可以对关联规则的哈希结构数据进行动态的调整。该算法提高了挖掘效率，通过抑制哈希地址冲突提高了算法的稳定性和可用性。

关键词：数据挖掘；入侵检测；关联规则；最小完美哈希函数

Improved Data Mining Algorithm for IDS Log

CUI Wei, LIU Jianwei, ZHANG Qishan
(School of Electronics and Information Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100083)

【Abstract】 This paper proposes an optimized direct Hash and pruning algorithm based on minimal perfect Hash scheme. This algorithm based on Apriori, integrates the advantages of conventional Hash pruning technique and those of minimal perfect Hash scheme. Besides dealing with mining association rules of static database, this algorithm also perform greatly in updating Hash structure. This algorithm improves the efficiency of mining, stability and availability by restraining the Hash address conflict. The experiments show that ODHP can be more efficient than conventional mining algorithm.

【Key words】 Data mining; Intrusion detection; Association rule; Minimal perfect Hash function

随着电脑技术的逐渐普及和网络规模的迅猛增大，数据库的容量也以几何级数增长。在日常生活和研究过程中，因为需要面对数据库的分析和管理操作，所以需要研究从大量的网络数据中自适应地发现可用数据的挖掘理论和技术。基于数据挖掘的入侵检测系统[1]，即利用数据挖掘技术分析网络流量和主机日志数据，从中提取黑客入侵模式和用户正常行为模式，用于检测攻击。

在非并行挖掘算法中，Apriori的改进算法(Direct Hashing and Pruning, DHP)[2]在效率的改进上是非常有效的。这些算法都使用一种称作逐层搜索的迭代方法，通过多次扫描事务数据库来找出频繁项目集。但这些改进算法在前 2 次扫描事务数据库时开销很大，导致了算法的效率出现瓶颈，在实际使用过程中效率也不尽如人意[3]。

本文针对 DHP 算法提出了一种改进算法 ODHP。该算法应用了最小完美哈希函数，不仅能胜任数据库静态数据的挖掘工作，还能适应如 IDS 日志数据库这样更新较快的动态结构，从而能够更快、更准确地挖掘出关联规则信息。

1 传统的 DHP 算法描述

1.1 数据挖掘基本思想

为了建立精确有效的检测模型，往往需要收集大量的训练数据，从中识别出一组有意义的特征集，需要提取事件数据之间的关联规则。事件数据关联规则挖掘过程就是发现频繁模式(频繁关联规则和频繁情节规则)的过程。

实际上，挖掘关联规则的任务就是从已知数据中，得出所有满足最小支持度和最小置信度的关联规则。通常这一问题可以被分解为 2 个子问题，即：

(1)求出 D 中满足最小支持度的所有频繁项目集；

(2)利用频繁项目集生成满足最小置信度的所有关联规则。

事实上，挖掘关联规则的整个执行过程中第(1)个子问题是核心问题。当找到所有的频繁项目集后，相应的关联规则将很容易生成。第(1)个子问题是挖掘关联规则的重点。

1.2 DHP 算法基本思想

DHP(Direct Hashing and Pruning)算法[4]在继承传统Apriori算法一些优势的同时，利用哈希技术有效地改进了候选频繁项目集的生成过程，产生了比Apriori算法体积更小的候选频繁项目集，同时也缩减了事务数据库的大小，减小了I/O操作时间，其效率比Apriori算法有明显提高。本文对图 1 中的数据资料进行了计算。

数据库 D

事件 ID	属性
1	ACD
2	BCE
3	ABCE
4	BE

图 1 原始数据

应用传统的 DHP 算法对数据库 D 进行扫描后，首先建立了一个哈希表(Hash Table)，分别将其频繁项目集的可能组合罗列出来；然后，将罗列的结果代入哈希函数。

h({x y}) = (order of x)*10+(order of y) mod 7

求余后得到一个字节中的具体位置。将所有的子集全部经过哈希函数计算之后，分别对该字节的每一位计数。然后判断其支持度是否大于 minsup=2。

再从结果中将不属于L_1的部分去掉，便得到了K=1 时的候选集，以备K=2 阶段的计算。其具体的计算步骤如图 2 所

作者简介：崔 玮(1980－)，男，博士生，主研方向：网络安全与信息传输；刘建伟，副教授；张其善，教授、博导

收稿日期：2006-03-24 **E-mail**：liujianwei@buaa.edu.cn

示，其中D代表事件资料数据库，C_1是候选集，L_1为频繁项目集，Sup为支持度(Support)。在本例中设最小支持度minsup=2。

应用传统的 DHP 算法，将得到如图 2 所示的哈希表。

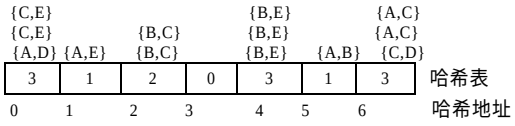

图 2 传统 DHP 算法得到的哈希表

从图 2 中可以看出，哈希地址为 0、2、4、6 的事件的支持度超过了minsup。事实上，经过DHP算法得到C_2的过程比起Apriori算法要大大地简化了。

2 算法改进

2.1 传统算法的缺陷

用传统的 DHP 算法进行挖掘面临着哈希冲突和哈希函数不通用的限制。例如在图 2 所示的 DHP 算法流程中就出现了 2 个事件拥有同样的地址的情况，在大数据量的情况下，这种情况是必须要杜绝的，因为它的出现使得运算效率大为降低。

由于面向的是网络环境，因此可将网络数据流经过处理得到提取出数据包量、时长、传输字节量等属性作为特征属性存储到数据库中。所采用的属性越多，其判断的准确率越高，当然时空复杂度也就越大，也越需要高效率的算法。

传统的算法在应用哈希技术的同时并没有很好地解决哈希地址冲突及哈希函数的选取问题，甚至在最坏的情况下需要回归到 Apriori 算法，所以在实际的入侵检测系统日志分析工作中，其运算效率还是不能令人满意。

在应用 DHP 算法的入侵检测系统日志挖掘过程中，待处理的数据特征不断被更新，新入侵方法的数据特征也不断出现，而与此同时哈希函数也必须随之不断更换。很显然，传统的 DHP 算法并不能适应入侵检测系统日志的规则挖掘要求。

2.2 应用最小完美哈希函数的 ODHP 算法

最小完美哈希函数是根据中国古典的余数定理发展出来的演算法[5]，它既能保证占用最少哈希地址数、完全无冲突，同时又能保证在数据增加或减少的时候不需要更换哈希函数。

在实际的最小完美哈希函数的构建过程中，首先需要为数据项目集I中的每一项I_i，各分配一个质数m_i。要求对每个I_i，$m_{i+1}>m_i$。对这组质数显然有$m_i>i$。那么，构建后的最小完美哈希函数为

$$H(I_x，I_y)=d(I_x)+B(I_y)$$

其中，d(a)表示第 1 次 a 出现的位址-1。

例如在图 1 中，需要先对子项目集进行分组，并确定第 1 次出现的位置。其中，d(A)= 1-1 = 0，而 d(B)= 4-1 = 3。

$$B(I_y)=C(I_y) \bmod m_y$$

$$C(Iy)=\sum_{i=1}^{n} M_I b_i i$$

其中，$M_i=m_1*m_2*\ldots*m_{i-1}*m_{i+1}*\ldots*m_n$ ；bi是能让mi能被Mibi-1整除的最小正整数。

将最小完美哈希函数应用到入侵检测系统日志数据关联规则挖掘的 DHP 算法中，可以大幅度增加规则挖掘的效率和稳定性。

如果应用最小完美哈希函数来解决图 2 的问题，具体的步骤如图 3 所示。

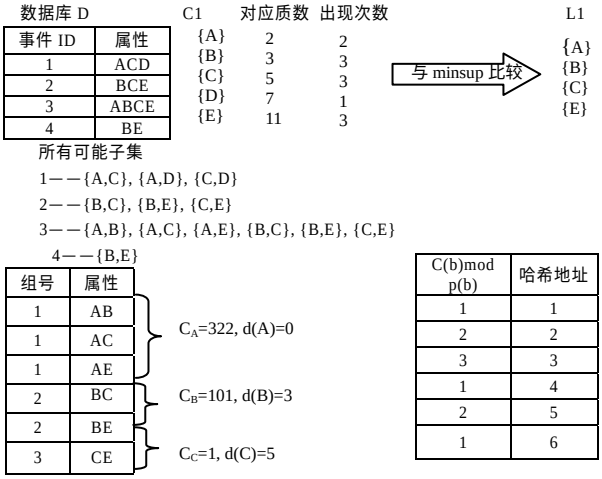

图 3 使用最小完美哈希函数进行关联规则挖掘

2.3 算法描述

ODHP 算法的内容如下：

```
/*ODHP 算法*/
Main start
s= minimum support ;
set all the buckets of H2 to zero;   //哈希表初始化
for each transactionin Database do begin
//第 1 次扫描数据库，对出现的项目计数
    for each item do begin
        i.count++;
        hash table(i)++;
    end
end
L1={c|c.count≥s, c是哈希树上的一个节点}
For (k=2;Fk-1≠ϕ;k++) do begin;   //F是频繁项目集
    For all transaction , Dk do begin
Assign prime integer to each items in Dk;
//对数据集中的每一个元素赋一个质数值
Insert and count 1-items occurrences in a hash tree;
Select key-character-items from t ;
//从数据集中各单数据内选出代表属性
Group ki1 ;    //依照代表属性进行分组
End
Gen_candidate(Lk-1,Hk,Ck); //生成候选集
    Set all the buckets of Hk+1 to zero;   //哈希表初始化
    Dk+1=ϕ;
For all (k+1)-subsets x of t do
Assign d(ki1);      //对每个数据指定哈希头地址
Calculate Ci ;       //计算 Ci
H(ki1 ki2)++;       //应用最小完美哈希函数计算
End
Save Hash Table;   //将哈希望表结果存储
For each item in Ck do begin
If H(ki1 ki2)<s;   //判断是否为频繁项目集
Remove from Ck table;
        End
Save Ck table;   //存储Ck，以便在重新扫描数据库时使用
Lk={c|c.count≥s , c 是哈希树上的一个节点};
End
//进行第 2 次 pruning，也应用最小完美哈希函数，方法同上
Gen_candidate(Lk-1,Hk,Ck); //生成候选集
While(|Ck|>0)
```

```
{       D_{k+1}=φ;
        For all transaction t in D_k do begin
              Count_support;
              If(t.count>k) then
                     Prune_second(k+1);
save to D_{k+1};
End
Lk={c in C_k|c.count≥s};
if(|D_k|=0) then finish;
C_{k+1}=apriori_gen(L_k);//判断所有的k-1 项目子集是否频繁项目集
k++;
}
Main end
```

对剪枝后代数据进行 2 次剪枝主要是针对特定事务项的项目。在算法中的Prune_second函数对应于make_hasht函 数[6]，将数据集中所有事务项中潜在的k+1 项目集重新散列到哈希结构这一过程。

3 算法实现及实验结果

本文测试环境：主频为 1.6GHz，内存为 256MB，操作系统为 Windows 2000，编译器为 Visual C++ 6.0，数据库系统为 SQL Server。入侵检测系统采用基于 Snort 的 C/S 结构的 IDS 的情况下针对传统的 DHP 算法和 ODHP 算法的挖掘速度和效率进行了比较。

数据库访问方式采用 ADO 方式。而数据结构方面，在候选项目集集合采用了前缀树结构(prefix-tree)，而频繁项目集则采用了数组结构，频繁项目集之间也使用数组结构组织。

在测试中，对最大频繁项目集的大小分别选取了 10、15、20、25 等多个不同的测试点。最小支持度设定在 0.4%。

从图 4 中可以看出，在实验中 ODHP 算法比传统的 DHP 算法更为有效。实验证明，ODHP 算法对各类型的频繁项目集挖掘都十分有效。

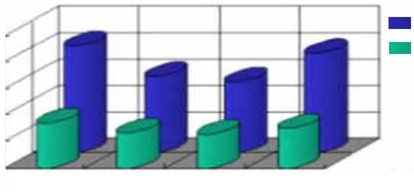

图 4 算法测试比较

4 结束语

针对传统 DHP 算法在入侵检测系统中实际应用的缺陷，本文提出了一种改进算法 ODHP。实验结果显示，ODHP 算法效率比传统的 DHP 算法高，后续研究也将集中把这一算法的思想更深入地应用于实际系统的优化中。

参考文献

1 胡和平, 肖述超. 一种分布式入侵检测系统模型[J]. 计算机工程与科学, 2005, 27(7).

2 Mannila H, Toivonen H. Discovering Generalized Episodes Using Minimal Occurrences[C]//Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining. 1996.

3 杨华兵, 叶新郢, 张宁蓉. 入侵检测中频繁模式的有效挖掘算法[J] 情报指挥控制系统与仿真技术, 2005, 27(1).

4 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]. Proceedings of the 20th VLDB Conference. 1994.

5 张真诚. 中国古典数学在咨讯科学上的应用[J]. 咨讯与教育杂志, 1991, (8).

6 Chen Mingyan, Yu P S. Using a Hash-based Method with Transaction Trimming for Mining Association Rules[J]. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(5): 813-825.

~~~~~~

(上接第 52 页)

```
//实现切割面要素坐标拼接；
        if(切割边边界点是实边界点)
则 2 个边界点指向的面状要素坐标合并时保留 1 个边界点；
        else   //切割点
则 2 个边界点指向的面状要素坐标合并时删除 2 个切割点；
        }   };
```

上述拼接过程是理想情况下的实现，在切割边关联索引表支持下拼接过程简单。

为减少拼接错误，可在切割边结构中增加地理要素编码字段，确保实现同类地理要素的拼接。图块块边界面要素拼接前后示意见图 5，图块拼接后实现切割面状地理要素在接边处是无缝的，屏幕视觉效果良好。

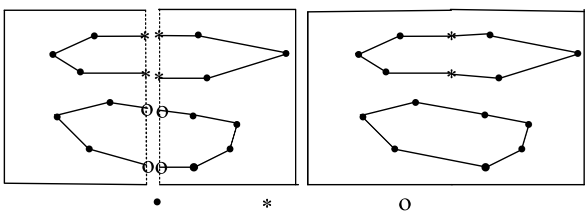

图 5 图块拼接前后示意图

3 结论

嵌入式 GIS 系统上地理数据是分幅分块记录存储的，某一时刻以图块为单位仅读取屏幕可视范围的地理数据到内存实现无缝拼接，执行显示和缩放处理；漫游时重新读取漫游区域的地理数据，此时较长时间没有使用的图块数据可从内存清除。这样嵌入式系统上基本实现数据按需读取和处理过程，系统执行速度快。根据实现分析和实践证明，本文提供的嵌入式 GIS 面要素坐标格式数据量小、算法实现简单，能够在一定位置误差范围内自动化实现面要素无缝拼接处理，能够用于任意数量的图块拼接。数据量小和算法简单高效的特点使之可以用于 C/S 体系结构的网络 GIS 系统中。

参考文献

1 隋春光, 彭认灿. 数字海图无缝拼接的实现及相关问题研究[J]. 测绘科学, 2004, 29(4): 28-31.

2 王 卉. 无缝 GIS 相关理论与技术的研究[D]. 郑州: 解放军信息工程大学, 2004.

3 朱欣焰, 李德仁. 无缝空间数据库的概念、实现与问题研究[J]. 武汉大学学报, 2002, 27(4): 382-386.

4 王 卉, 王家辉. 无缝 GIS 发展的两个关键技术[J]. 测绘通报, 2002, (4).
~~~~~~