高性能网络安全告警信息的关联分析方法

付泽强 王晓锋 孔 军
(江南大学物联网工程学院 江苏 无锡 214122)

摘 要 在网络安全防御体系中,入侵检测系统会实时产生海量冗余、错误的网络安全告警信息,因此有必要对告警信息的关联规则和序列模式进行频繁项模式挖掘,分辨正常的行为模式,筛选出真正的攻击信息。相对于Apriori和FP-growth等算法,COFI-tree算法虽然具有较大的性能优势,但仍无法满足大规模网络安全信息快速分析的需求。为此,基于COFI-tree算法,提出了一种改进的网络安全告警信息关联分析算法。该算法通过基于倒序链表的头表节点寻址方式和基于新的SD结构的频繁项处理方法,提升了COFI-tree算法的性能。基于Kddcup99数据集的实验结果表明,与传统的Cofi算法相比,该方法在基本保证准确率的同时,能大量降低计算开销,使处理时间平均缩短21%以上,解决了在海量网络告警信息下进行关联分析时速率不高的问题。

关键词 COFI-tree,网络安全,频繁项目集,数据挖掘,关联分析
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.05.018

High-performance Association Analysis Method for Network Security Alarm Information

FU Ze-qiang WANG Xiao-feng KONG Jun
(School of Internet of Things Engineering, Jiangnan University, Wuxi, Jiangsu 214122, China)

Abstract In the network security defense system, the intrusion detection system will produce massive redundancy and wrong network security warning information in real time. Therefore, it is necessary to mine frequent item patterns from association rules and sequential patterns of alert information, distinguish normal behavior patterns, and screen out real attack information. Compared with Apriori, FP-growth and other algorithms, COFI-tree algorithm possesses bigger advantages of performance , but it still can not meet the needs of fast analysis on large-scale network security information. To this end, this paper proposed an improved network security alert information association analysis algorithm based on COFI-tree algorithm. The algorithm improve the performance of COFI-tree algorithm through node addressing mode based on reverse linked list and frequent item processing method based on new SD structure. The experimental results based on Kddcup99 dataset show that this method can basically guarantee the accuracy, reduce a lot of computing overhead, shorten processing time by more than 21% on average compared with the traditional Cofi algorithm, and solve the problem of low speed in association analysis under massive network alarm information.

Keywords COFI-tree, Network security, Frequent item sets, Data mining, Association analysis

随着信息技术的不断发展和进步,网络安全愈加重要,新时期的网络安全面临着网络攻击组织化和趋利化、攻击方法推陈出新、攻击目标范围广泛化[1]等巨大的考验。网络安全防御体系实时产生大量的冗余告警信息,包括无关告警、重复告警以及误报等,真正的攻击信息被海量低质量的告警信息所掩盖,告警信息远远超过了人工分析的能力。因此,需要对海量告警信息进行关联分析、再分析和再组织,去除无关告警信息,过滤正常攻击行为,挖掘出真正的告警信息。

目前,国内外所采用的告警关联方法主要分为:基于告警属性相似度聚类的关联方法,以TACC[2]组件为代表;基于告警因果关系的关联方法,以TIAA[3]为代表;综合两类关联方法提出的WINEPI[4]算法和A3PC[5]自适应告警模型等。针对当前筛选系统性能的不足,郑哲渊等[6]又提出了面向大规模告警数据的快速筛选系统(RSS),该系统能快速有效地筛选出有用信息,其最核心的部分是所使用的关联挖掘算法。

目前已有的关联挖掘算法可分为宽度优先搜索算法和深度优先搜索算法两种。宽度优先搜索算法以DMFIA[7],Apriori[8],CBA[9],A-Close[10]等算法为代表。深度优先搜索算法以FP-max[11],CMAR[12]等算法为代表。RSS系统在对告警信息的关联规则和序列模式进行模式挖掘时,采用了Apriori,FP-growth[13],COFI-tree[14]等数据挖掘算法。Apriori算法需要遍历整个数据集才能生成每次的候选集;并且

到稿日期:2018-05-08 返修日期:2018-07-25 本文受国家自然科学基金项目(61672264),国家重点研发计划项目(2016YFB0800803)资助。
付泽强(1991-),男,硕士生,主要研究领域为网络安全、数据挖掘,E-mail:760188504@qq.com;王晓峰(1978-),男,博士,副教授,主要研究领域为网络安全、网络仿真,E-mail:wangxf@jiangnan.edu.cn(通信作者);孔 军(1974-),男,博士,副教授,主要研究领域为人工智能、机器视觉。

当最大项集的维数较高时，算法的效率严重下降。FP-growth 算法在建立 FP-tree 时需要多次计算共享前缀路径，递归建树与遍历的时间很长，影响了算法的执行效率[15]。COFI-tree 算法不需要递归建立条件模式树，同时在一定时间内只有一棵 COFI 树在内存中，占用的空间结构小，因此，其综合性能是 3 种算法中最好的。

然而，在处理海量的事务集合时，COFI-tree 算法依然存在不足：在第一次遍历事务数据库建立 FP-tree 时，需要建立 FP-tree 的频繁项头表[16]，传统的 COFI-tree 算法需要迭代找出每个头表节点链表的最后一个节点，数据量越大，花费的时间越长；在处理频繁项时，建立并处理每一棵 COFI-tree 会占用大量的空间结构，而且一系列树的计算也使得效率不高。本文针对上述不足，通过重新构建 FP 树频繁项头表的数据结构，以及使用新的 SD 结构来处理频繁项的方式等方法，提出了改进的 COFI-tree 算法，提高了算法的性能。

1　COFI-tree 算法的原理分析

1.1　相关定义

定义 1　项集为数据项的集合。

定义 2　$D=\{T_1,T_2,T_3,T_4,T_5\}$ 是一事务数据库，每一个 $T_i(i=1,2,3,4,5)$ 称为一个事务。

定义 3　设事务 $T=\{I_1,I_2,\cdots,I_m\}$ 是 m 个项目的集合，其中每个 I_i 称为数据项，包含 k 个数据项的项集称为 k-项集。

定义 4　给定事务数据库 D，数据项 Y 在 D 中的支持度是指 D 中含有数据项 Y 的事务个数，D 中包含 Y 的事务个数在 D 中所有事务总数中所占的百分比称为项集 Y 的支持率，若数据项 Y 的支持度大于或等于用户指定的最小支持度 min_sup，则数据项 Y 称为频繁项。

1.2　COFI-tree 算法

COFI-tree 的每个节点包含节点名称、参与值（*participation_counter*）和数量（*count*）。

COFI-tree 树方法主要分为 3 个阶段。

第一阶段：设定一个最小支持度，扫描一遍事物数据库，确定每个数据项的支持度，找出所有的频繁项，舍弃支持度小于最小支持度的项，按频繁项的支持度递减排列事务数据库中的每一组事务；再一次扫描数据库，建立 FP 树。

第二阶段：依据 FP 树依次建立每个频繁项 X_i 的 COFI-tree。步骤如下：在 FP-tree 中找到包含 X_i 的所有路径，X_i 的支持度为所在路径的支持度；将包含 X_i 的每条路径接入 COFI 树的根节点 X_i，如果 COFI 树中的路径与 X_i 中的某个路径有相同的部分，则相同部分中所有节点的 *count* 值要加上 X_i 在该路径的支持度的值；另外，还要建立 COFI-tree 的局部频繁项头表。

第三阶段：对 COFI 树进行处理。将 COFI 树中支持度小于最小支持度的项舍弃。在 COFI 树的每个节点上增加一个参与值（*participation_counter*），参与值的初始值为零，按 COFI 树中局部频繁项支持度递减的顺序依次取出 COFI 树头表中的频繁项 Y_i。在 COFI-tree 中，找出包含 Y_i 的所有路径，以及每条路径上所有包含 Y_i 的子集，若某个子集尚不存在，则把这个子集添加到列表中，频数为 $Y_i.count-(Y_i.participation\_counter)$；否则，这个子集的频数增加 $Y_i.count-(Y_i.participation\_counter)$，每条路径上每个节点的参与值等于原来的参与值加上在该路径中 $Y_i.count-(Y_i.participation\_counter)$ 的值，如果 $Y_i.count$ 已经等于 $Y_i.participation\_counter$，则不对此条路径做以上操作。最后将列表中的支持度大于最小支持度的子集取出，删去重复项，去除包含项，所得结果即为这个频繁项的 COFI 树的最终频繁项目集。

当某个频繁项的 COFI 树操作完毕后，释放该频繁项的 COFI 树，建立另一个频繁项的 COFI 树继续操作，直至结束。

1.3　存在的问题

COFI 算法虽然是关联规则挖掘算法中较为高效的一种算法，但还是存在着一些不足：1）为了进行 FP-tree 频繁项头表的构建，FP-tree 头表链所使用的正序链表需要逐个找出每个头表中的最后一个指针，空间和时间消耗过大；2）用 COFI 树处理 FP 树得出的频繁项时，建树和计算占用了大量的空间和时间，且 COFI 树对频繁项的处理步骤冗长，速度较慢。

2　改进的 COFI-tree 算法

对 COFI-tree 算法的改进分为两部分：1）在 FP-tree 的频繁项头表进行数据结构的改进，使用倒序链表代替迭代寻找最后一项正序链表，不需要找到最后一项，将新的数据项直接指向头表中对应的项就可以了；2）使用新的 SD 结构处理频繁项代替 COFI-tree 的树处理方式。

2.1　倒序链表的生成

本节将举例说明针对 FP-tree 树的频繁项头表进行数据结构改进及建立倒序链表的具体步骤。设有如表 1 所列的事务数据表。

表 1　事务数据表示例

Table 1　Example of transaction data table

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
A	1	1	1	1	1	1	1	0	0	1	1
B	1	1	0	1	0	1	0	1	1	1	0
C	1	1	1	0	1	1	0	1	0	1	0
D	0	1	0	1	1	0	0	1	1	0	0
E	0	1	1	0	1	0	0	1	1	0	1
F	0	0	1	0	0	1	1	1	1	1	1
G	0	0	1	1	1	0	1	0	1	0	1
H	0	0	1	1	1	1	1	0	1	0	1
W	0	0	0	1	0	0	1	0	0	0	0
L	0	0	0	0	0	0	0	0	0	1	0

计算每个数据项的支持度并将其保存在频繁项的集合中，设 3 为最小支持度。筛选出集合中支持度大于或等于 3 的数据项（频繁项），按照支持度降序排列，并将其放入列表 L 中，$L=\{(A:9),(B:7),(C:7),(F:7),(H:7),(E:6),(G:6),(D:5)\}$。将每个事务中小于最小支持度的数据项删除，即删除 W 项和 L 项，将剩余的数据项按照 L 中的顺序排列。再次扫描事物数据库，建立 FP 树。

根据列表 L 建立一个频繁项头表，包括频繁项和支持度两部分。头表中的频繁项的指针（*node_next*）指向树中具有

相同数据项名称的节点。FP 树如图 1 所示。

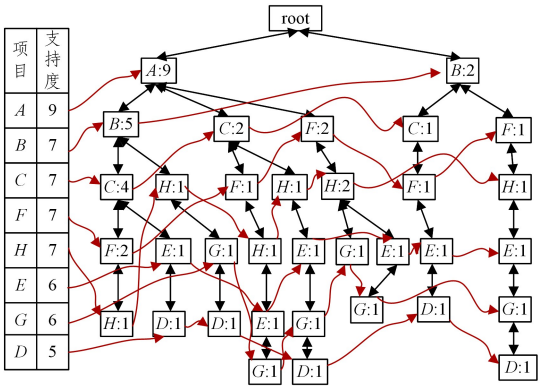

图 1 FP 树

Fig. 1 FP-tree

建立频繁项头表时，每当在 FP-tree 中找到与头表中有相同数据项名称的节点时，都需要对频繁项头表中对应的频繁项进行迭代处理，找出最后一个指针 *node_next* 的位置，才能将 FP-tree 中相同的数据项加入。这种做法的不足在于：每次有一个新的项出现时就必须从头表链中对应的频繁项的第一个 *node_next* 开始查找，直到找出最后一个 *node_next*，时间复杂度为 O(n)。在图 1 中，以 C 节点为例，每插入一个新的 C 节点，首先需找到频繁项头表中的 C 节点，通过其指针 *node_next* 找到第一个 C 节点的位置，再通过第一个 C 节点的指针 *node_next* 找到下一个 C 节点所在的位置；重复以上步骤，直至找到最后一个 C 节点，才将新的节点插入。当数据量很大时，上述方式会降低速率。

为了解决这个问题，需要对头表的数据结构进行改进，将头表链由正序变为倒序，其中头表记录的是当前最后一个节点的位置，每次插入一个新的节点时：1)在头表链中找到这个节点；2)使得插入的节点的指针域指向头表链中的这个节点；3)头表链中的节点等于插入的这个节点。具体步骤如图 2 所示。

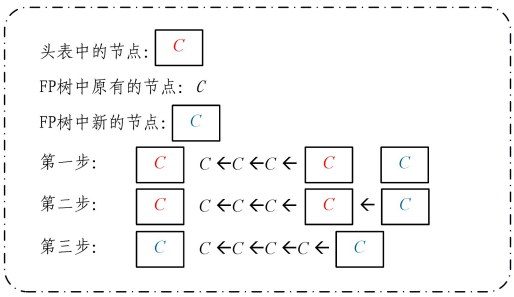

图 2 头表结点变换示例

Fig. 2 Example of head table node transformation

这样，当头表加入一个新项时，就不需要迭代地找到最后一个 *node_next*，直接将新增加的项指向头表链中的项即可。比如新增加一个 C 节点时，先到头表中寻找 C 节点，如果不存在 C 节点，则直接将新增加的 C 节点放入头表中；若 C 节点已经存在，则将新增加的 C 节点指向头表中的 C 节点，而后使头表中的 C 节点等于新增加的 C 节点的地址。头表节点改进的伪代码如下：

```
If header.node==null
    header.node=新增节点;
    此头表节点各个参数=新节点参数;
else
    新增节点→node_next=header.node;
    header.node=Node(新增节点);
end
```

2.2 SD 结构处理频繁项

2.2.1 COFI-tree 的构建

构建每个频繁项的 COFI 树。举例说明：根据如图 1 所示的数据构建频繁项 G 的 COFI 树，从 FP-tree 中找到包含 G 的每条路径，G 所在的每条路径的支持度等于该路径中 G 的支持度。在 FP 树中，G 的路径如下：

$(G-H-B-A):1$

$(G-E-H-F-C-A):1$

$(G-E-H-C-A):1$

$(G-H-F-A):1$

$(G-E-H-F-A):1$

$(G-E-H-F-B):1$

根据 G 的路径得出 G 的 4 个局部频繁项：$H[6]$，$A[5]$，$F[4]$，$E[4]$。将 G 的每条路径去掉非局部频繁项 C、B，然后每条路径按照局部频繁项的支持度递增排列如下：

$(G-A-H):1$

$(G-E-F-A-H):2$

$(G-E-A-H):1$

$(G-F-A-H):1$

$(G-E-F-H):1$

创建树 G-COFI-tree 的根节点 G，然后对 G 的每条路径执行如下操作：将每条路径接入 COFI 树的根节点 G 中，如果 COFI 树中的路径与 G 中的某条路径有相同的部分，则相同部分中的所有节点的 *count* 值要加上 G 在该路径的支持度的值；否则，创建一条新路径，使其每个节点的支持度为该路径的支持度。例如，第一条路径$(G-A-H):1$，它的支持度是 1，则根节点 G 的支持度为 0+1=1，$A-H$ 与 G 中现有的路径没有相同的部分，所以创建支持度为 1 的新节点 A 和 H。第二条路径$(G-E-F-A-H):2$，其支持度为 1，G 的支持度增加 1，为 2。同时也创建新节点 E，F，A，H，支持度为 1。对所有路径依次如上操作，直至路径为空。操作完成后，每个节点增加一个参与值(*participation_counter*)，参与值的初始值为零。建立一个局部频繁项头表，用头表项的 node-link 连接具有相同 item-name 域的节点。

图 3 即为频繁项 G 的 G-COFI-tree。

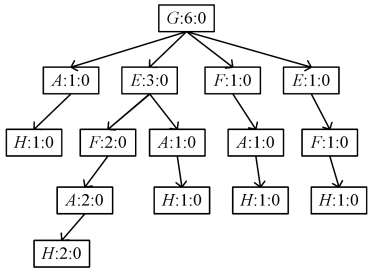

图 3 G-COFI-tree

Fig. 3 G-COFI-tree

2.2.2　基于 SD 结构处理频繁路径基

首先，根据图 3(G-COFI-tree)产生频繁项 G 的频繁路径基，然后在 G-COFI-tree 的频繁项头表中按照局部频繁项的支持度递减的顺序分别对每一个局部频繁项作如下操作：在 G-COFI-tree 中从下到上找出以这个频繁项为头节点的所有路径。当头节点的支持度(*count*)大于 *participation_counter* 时，设值 T 为该路径的头节点的支持度减去 *participation_counter* 的值，每条路径中的节点支持度值增加 T；同时，这一条路径即为一个频繁路径基，支持度为 T，当头节点的支持度和 *participation_counter* 都相同时，则不进行操作。图 4 即为 G-COFI-tree 产生所有频繁路径基后的图。

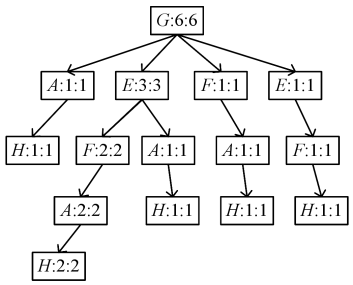

图 4　产生所有频繁路径基后的 G-COFI-tree

Fig. 4　G-COFI-tree after producing all the frequent path bases

此时，G-COFI-tree 中每个节点的支持度都等于参与值，最终频繁路径基为 $(GAH):1$，$(GEFAH):2$，$(GEAH):1$，$(GFAH):1$，$(GEFH):1$。

创建 SD 结构处理频繁路径基。SD 结构是一个环形单向链表，链表的每段对应着长度不同的频繁路径基(频繁路径基的长度沿着链表的顺时针方向递增)。设最终的频繁路径基的最大长度为 N，则 SD 结构有 N 段。例如 G-COFI-tree 的最终频繁路径基的最大长度为 5，则对应的 SD 结构就有 5 段。如图 5 所示，按照长度对频繁路径基进行分类，相同长度的频繁路径基连接在一起。

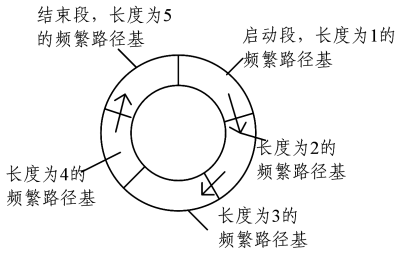

图 5　SD 结构

Fig. 5　Structure of SD

基于 SD 结构处理频繁路径基的步骤如下：

1)将得到的最终频繁路径基分别按长度放入 SD 结构，SD 结构的每一段将长度相同的频繁路径基连在一起。例如，频繁路径基 GAH 加上支持度为 $GAH:1$。因为频繁路径基 GAH 的长度为 3，所以 $GAH:1$ 放入 SD 结构长度为 3 的段中。当所有频繁路径基都放入 SD 结构后，使所有频繁路径基两两相交，所有交集也按照长度放入 SD 结构。如果某个交集与 SD 结构中已有的频繁路径基相同，则放弃将此交集放入 SD 结构中。将放入 SD 结构的每一个交集的支持度设为 0。此操作完成后的 SD 结构如图 6 所示。

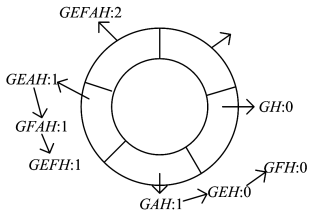

图 6　放入频繁路径基和两两交集的 SD 结构

Fig. 6　Structure of SD after adding frequent path base and intersection

2)从 SD 结构长度为 1 的启始段开始，每一段中的每个路径基依次按顺时针方向(路径基长度递增)与其他段中的路径基进行比较，直至到达结束段。每一段中的每个路径基不与本段中的路径基作比较。如果此路径基是其他段中的某个路径基 B 的子集，则此路径基的支持度等于原来的支持度加上路径基 B 的支持度。如长度为 3 的段中的路径基 $GAH:1$ 是长度为 4 的段中 $GEAH:1$ 和 $GFAH:1$ 的子集，也是长度为 5 的段中 $GEFAH:2$ 的子集，则 GAH 的支持度由 1 变为 $1+1+1+2=5$。同样，SD 结构长度为 3 的段中的路径基 $GEH:0$ 变为 $GEH:4$。此操作完成后的 SD 结构如图 7 所示。

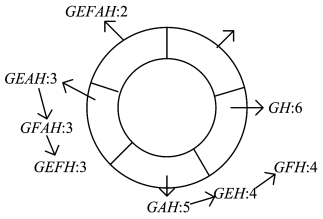

图 7　支持度变化后的 SD 结构

Fig. 7　Structure of SD after changing support degree

3)此时，SD 结构中支持度大于或等于最小支持度(*min_sup*)的路径基即为 G-COFI-tree 的最终频繁项目集，为 $GH:6$，$GEH:4$，$GFH:4$，$GAH:5$，$GEAH:3$，$GFAH:3$，$GEFH:3$。

至此，频繁项 G 的 SD 结构处理完成，删除 G 的相应链表和数据，然后对其他频繁项进行类似处理。以此类推，得出事务数据库的所有最终频繁项目集。

算法具体思想如下所示。

算法 1　SD 结构处理频繁路径基算法

输入：FP 树，一项频繁集，最小支持度(min_sup)

输出：最终频繁项目集

1. for 一项频繁集中的每个项 x_i。
2. 创建 x_i 的 x_i-cofi-tree。
3. for x_i 的局部频繁项的每个项 y_i，利用节点的 count 和 participation 的不同得出所有频繁路径基，并将所有频繁路径基和对应支持度放入构建的 SD 结构中。
4. 将所有频繁路径基两两相交，将所有交集放入 SD 结构，支持度为 0。如果某个交集与 SD 结构里已存在的数据重复，则此交集不放入 SD 结构。
5. 从 SD 结构长度为 1 的启始段开始，每一段中的每个路径基依次按顺时针方向(路径基长度递增)与其他段中的路径基进行比较，并修改自己的支持度。
6. 找出 SD 结构中支持度大于或等于 3 的路径基，此即为 x_i 的最终频繁项目集。
7. Release (x_i) and go to 1。

3 实验结果与分析

3.1 实验环境

为便于比较，本节在统一平台上进行了所有实验。实验的硬件环境为 PC 电脑，CPU 为 Intel(R) Core(TM) i5-7200U，内存为 8GB，在 64 位 Window10 操作系统中，算法的源代码使用 C++语言编写，在 VS2010 上运行，数据库采用 Kddcup99 数据集[17]。Kddcup99 数据集是林肯实验室模拟美国空军局域网的一个网络环境，收集了 9 周 TCPdump 网络连接和系统审计数据，并对各种用户类型、各种不同的网络流量和攻击手段进行了实验仿真，可以视为一个真实的网络环境[18-19]。实验图中的时间反映了几次实验中算法的平均时间，包括所有的预处理时间、读入时间和结果输出时间。

3.2 实验结果

Kddcup99 数据集中每条事务共由 41 个特征进行描述，选取每个事务的 16 个特征进行测试。实验从两个方面对改进前后的算法进行对比：1)固定最小支持度为 50%，每次加载不同的事务数量对 Apriori 算法、Cofi 算法和改进的 Cofi 算法进行测试，设定事务数量为 2000，5000，8000，10000，20000，实验结果如图 8 所示；2)选取 10000 条事务，设定支持度范围在 3%～20%之间。图 9 对比 Apriori 算法、Cofi 算法和改进的 Cofi 算法在最小支持度为 20%，10%，7%，5%，3%等情况下的运行时间。

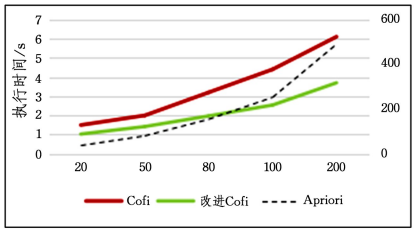

图 8 不同事务数量的效率对比

Fig. 8 Efficiency comparison of different transaction quantities

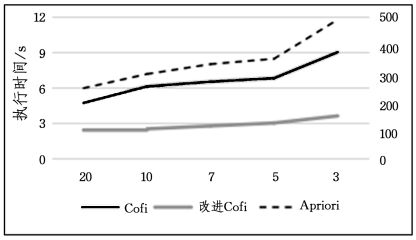

图 9 不同支持度的效率对比

Fig. 9 Efficiency comparison of different support degrees

图 8 中，Apriori 算法的曲线对应的是图右边的次坐标轴，Cofi 算法和改进的 Cofi 算法曲线对应的是图左边的主坐标轴。从图 8 的测试结果可以看出，在相同的最小支持度下，改进的 Cofi 算法效率有了很大的提高，领先于 Apriori 算法。Apriori 算法的时间开销随着事务数量的增大，最后已接近 500s，而改进的 Cofi 算法的时间开销基本都在 4s 以内。同时，与 Cofi 算法相比，改进的 Cofi 算法平均节省 20%以上的时间，特别是在事务数量逐渐增大的情况下，改进后的 Cofi 算法的时间开销明显小于传统 Cofi 算法的时间开销，效果更加明显。其主要原因在于事务数量越多，倒序链表在构造头表节点时节省的时间就越多，新的 SD 结构节省的重复操作的时间也越多。图 9 中，图右边的次坐标轴对应的是 Apriori 算法的曲线。Cofi 算法和改进的 Cofi 算法曲线对应的是图左边的主坐标轴。由图 9 可以看出，在相同的事务数量下，Cofi 算法和改进的 Cofi 算法的时间开销远小于 Apriori 算法的时间开销。3 种算法中，改进的 Cofi 算法有效缩短了处理时间，在不同支持度下的运算速度最快。最小支持度越小，改进 Cofi 算法的效率提升效果越明显，其越领先于其他两种算法，且稳定性良好。表 2 为 3 种算法在不同事务数量下最小支持度为 50%的最大频繁项集，括号内为项的支持度。由比较结果可知，Apriori 算法、Cofi 算法和改进的 Cofi 算法的结果几乎相同。

表 2 最大频繁项集的比较

Table 2 Comparison of maximum frequent itemsets

事务数量	Apriori	Cofi	改进 Cofi
2000	0-1-1032-255-SF-ecr_i-icmp-smurf. [1163]	0-1-1032-255-SF-ecr_i-icmp-smurf. [1163]	0-1-1032-255-SF-ecr_i-icmp-smurf. [1163]
5000	0-1-1032-255-SF-ecr_i-icmp-smurf. [4163]	0-1-1032-255-SF-ecr_i-icmp-smurf. [4163]	0-1-1032-255-SF-ecr_i-icmp-smurf. [4163]
8000	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [4972]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [4972]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [4972]
10000	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [5561]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [5561]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [5561]
20000	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [10097]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [10097]	0-1-1032-255-511-SF-ecr_i-icmp-smurf. [10097]

结束语 本文主要改进了一种运用在 RSS 系统上的关联分析算法，该算法能够快速有效地筛选大规模网络告警信息，大幅提高自适应告警关联系统的效率。相对于以往低效的关联挖掘算法，本文从节点寻找方式与频繁项处理方式入手，提出了一种新的基于头表节点改进和 SD 结构处理频繁项的 COFI 算法。实验表明，改进后的 COFI 算法在时间性能与空间性能两个方面较 COFI 传统算法均有所提升，增强了网络安全防御系统分析处理大规模告警信息的能力。

在处理超大规模网络安全告警信息时，本文提出的改进 COFI 算法仍有改进空间。下一步工作着重考虑算法的并行性，将分布式计算思想引入 COFI 算法中，例如 spark 框架、hadoop 框架等，并在分布式计算框架下运用 COFI 算法对大规模数据进行关联挖掘，以最大化提升算法处理超大规模告警信息的能力。

参 考 文 献

[1] LIU X R，LI B S，CHANGA N Q，et al. The Current Network Security Situation and Emergency Network Response[J]. Engineering Sciences，2016，18(6)：83-87. (in Chinese)
刘欣然，李柏松，常安琪，等. 当前网络安全形势与应急响应[J]. 中国工程科学，2016，18(6)：83-87.

[2] HOFMANN A，SICK B. Online intrusion alert aggregation with generative data stream modeling[J]. IEEE Transactions on Dependable and Secure Computing，2011，8(2)：282-294.

[3] GANAPATHI REDDY K L，SDNIVAS K. GDS an efficient approach for online intrusion alert aggregation[J]. International Journal of Computer Application，2012，2(1)：13-139.

[4] 单莘. 一种网络告警的增量式情景规则挖掘方法[C]// 中国通信学会学术年会. 2008.

[5] TIAN Z H，ZHANG Y Z，ZHANG W Z. An Adaptive Alert Correlation Method Based on Pattern Mining and Clustering Analysis[J]. Journal of Computer Research and Development，2009，46(8)：1304-1315. (in Chinese)
田志宏，张永铮，张伟哲. 基于模式挖掘和聚类分析的自适应告警关联[J]. 计算机研究与发展，2009，46(8)：1304-1315.

[6] ZHENG Z Y，LIU Y. High performance information filtering system for large-scale alarm data[J]. Computer Engineering and Design，2014，35(2)：436-439. (in Chinese)
郑哲渊，刘渊. 面向大规模告警数据的高性能信息筛选系统[J]. 计算机工程与设计，2014，35(2)：436-439.

[7] YIN Z H，ZHANG D P，TAN M，et al. Improved Algorithm for Efficiently Mining Maximum Frequent Itemsets Based on Frequent Pattern Tree[J]. Journal of University of Jinan(Science and Technology)，2017，31(2)：111-117. (in Chinese)
尹治华，张大鹏，谭明，等. 一种改进的基于 FP-Tree 的高效挖掘最大频繁项目集算法[J]. 济南大学学报(自然科学版)，2017，31(2)：111-117.

[8] LIU L J. Research and application of improved Apriorialgorithm [J]. Computer Engineering and Design，2017，38(12)：3324-3328. (in Chinese)
刘丽娟. 改进的 Apriori 算法的研究及应用[J]. 计算机工程与设计，2017，38(12)：3324-3328.

[9] MIAO S Q，ZHENG X S. Research and Implementation of Association Analysis[J]. Intelligent Computer and Applications，2018，8(2)：138-139. (in Chinese)
苗世强，郑晓势. 关联分类算法的研究与实现[J]. 智能计算机与应用，2018，8(2)：138-139.

[10] PASQUIER N，BASTIDE Y，TAOUIL R，et al. Discovering frequent closed itemsets for association rules[J]. Lecture Notes in Computer Science，1999，1540：398-416.

[11] NIU X Z，SHE K. Mining Maximal Frequent Item Sets with Improved Algorithm of FPMAX[J]. Computer Science，2013，40(12)：223-227. (in Chinese)
牛新征，余堃. 基于 FPMAX 的最大频繁项目集挖掘改进算法[J]. 计算机科学，2013，40(12)：223-227.

[12] WA′EL H，ABURUB F，ALHAWARI S. A new fast associative classification algorithm for detecting phishing websites[J]. Applied Soft Computing，2016，48：729-734.

[13] WANG J M，YUAN W. Improved FP-Growth algorithm based on node table[J]. Computer Engineering and Design，2018，39(1)：140-145. (in Chinese)
王建明，袁伟. 基于节点表的 FP-Growth 算法改进[J]. 计算机工程与设计，2018，39(1)：140-145.

[14] SHRIVASTAVA V K，KUMAR P，PARDASANI K R. Fp-tree and cofi based approach for mining of multiple level association tules in large databases[J]. International Journal of Computer Science & Information Security，2010，7(2)：248-225.

[15] WANG L，FAN X J，LIU X L，et al. Mining data association based on a revised FP-growth algorithm[C]// International Conference on Machine Learning and Cybernetics. IEEE，2012：91-95.

[16] NGUYEN T，HA Q T. Novel Operations for FP-Tree Data Structure and Their Applications[M]. Cham：Springer，2014.

[17] TANG W，MA J，ZENG G P. Analysis of Sample Database for Intelligence Intrusion Detection Evaluation[J]. Journal of South-Central University for Nationalities(Natural Science Edition)，2010，29(2)：84-87. (in Chinese)
唐菀，马杰，曾广平. 评测智能化入侵检测方法的样本库分析[J]. 中南民族大学学报(自然科学版)，2010，29(2)：84-87.

[18] ZHANG X Y，ZENG H S，JIA L. Research of intrusion detection system dataset-KDD CUP99[J]. Computer Engineering and Design，2010，31(22)：4809-4812. (in Chinese)
张新有，曾华燊，贾磊. 入侵检测数据集 KDD CUP99 研究[J]. 计算机工程与设计，2010，31(22)：4809-4812.

[19] LI F W，ZHENG B，ZHU J，et al. A method of network security situation prediction based on AC-RBF neural network[J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)，2014，26(5)：576-581. (in Chinses)
李方伟，郑波，朱江，等. 一种基于 AC－RBF 神经网络的网络安全态势预测方法[J]. 重庆邮电大学学报(自然科学版)，2014，26(5)：576-581.