基于AC算法的比特流频繁序列挖掘

雷 东 王 韬 马云飞
(军械工程学院信息工程系 石家庄 050003)

摘 要 为解决比特流频繁序列挖掘效率不高以及易受用户数据影响而导致准确率低的问题,首先从理论上论证了短频繁序列挖掘存在的局限性,根据不同长度的频繁序列挖掘时存在的特点,将其分为长频繁序列与短频繁序列,提出比特流协议头部字段定位算法;基于AC多模式匹配算法分别针对长、短频繁序列挖掘的不同特点,提出了相应的挖掘方法,提高了挖掘结果的准确性。最后通过实验验证了所提算法的有效性。

关键词 比特流,AC算法,长频繁序列挖掘,短频繁序列挖掘
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.01.025

Frequent Pattern Mining in Bit Stream Based on AC Algorithm

LEI Dong WANG Tao MA Yun-fei
(Department of Information Engineering, Ordnance Engineering College, Shijiazhuang 050003, China)

Abstract The existing method of frequent pattern mining in bit stream is inefficient and the precision of the results is low under the influence of redundant data. In order to solve the problem, it was proved that the mining of short frequent pattern has great limitations. According to the different features when the patterns are mined with different lengths, the frequent sequences are divided into two types: long frequent pattern and short frequent pattern. An algorithm of finding the header fields of the protocol in bit stream was proposed, and the efficient algorithm of mining the long frequent pattern and the short frequent pattern were proposed based on AC multi-pattern matching algorithm. Simulation results on the Ethernet show that the proposed algorithm is effective.

Keywords Bit stream, AC algorithm, Long frequent pattern mining, Short frequent pattern mining

1 引言

在日益严峻的网络安全形势及军事对抗需求的驱使下,通信网络的对抗技术成为各国争夺制信息权的主战场。在电子对抗环境下,从截获的通信比特流中进一步识别未知协议是一个重要的课题[1]。本文中的未知协议是指协议格式未公开,对于截获者而言是完全未知的协议。为了识别此类协议进而对截获的通信比特流加以利用,首先需要推断协议格式,而频繁序列挖掘是协议格式推断的重要前提[2]。

无论是格式已知还是未知的网络协议,都具有一些固定的协议格式,当同种协议的比特流数据大量汇聚时,便为频繁序列的挖掘提供了理论上的可能性。在数据挖掘领域,相关的研究已较为成熟,但在面向比特流的未知协议识别领域,仅有少数的研究。金陵等[3-6]基于AC算法[7]对比特流频繁序列进行挖掘,但由于AC算法的空间复杂度与频繁序列长度呈指数关系,且其所采用的剪枝算法不够优化,导致其只能挖掘长度较短的频繁序列;金陵[3]通过实验验证了其方法仅对长度为3~8bit的频繁序列有较好的效果。基于此,提出长频繁序列挖掘算法,并提出新的"剪枝"方法,结合新的频繁序列判别方法,使得长频繁序列挖掘的时、空复杂度都大大降低,且能保证挖掘结果的准确性。虽然王和洲等[4]都提出通过挖掘得到的短频繁序列进行拼接得到长频繁序列,陶术松等[8]通过揭示比特流序列挖掘具有的"向下闭包性",提出通过短频繁序列挖掘长频繁序列的方法,但此类方法都非常依赖短频繁序列的准确性,一旦挖掘到的短频繁序列有误,则由此得到的长频繁序列也必然有误,并且在由短频繁序列拼接长频繁序列的过程中会产生组合爆炸问题。实际情况是短频繁序列的挖掘结果极易受协议中用户数据的影响,宋疆等[3-6,9,10]采用ARP以及ICMP等包含用户数据较少的协议进行实验,金陵[3]通过实验验证指出,短频繁序列挖掘结果的准确率随着协议用户数据的增多急剧下降。针对短频繁序列受协议用户数据影响而导致挖掘准确率低的问题,提出一种能够定位协议头部字段的算法。所提算法在频繁序列挖掘过程中仅扫描协议头部字段,而跳过用户数据部分,从而避免了大量冗余数据对挖掘结果的影响,大大提高了短频繁序列挖掘的准确性。

2 相关概念与性质

数据挖掘领域对频繁模式挖掘的定义是频繁地出现在数据集中的模式(如项集、子序列或子结构)[11]。在比特流中进

到稿日期:2015-12-16 返修日期:2016-04-20 本文受军内科研基金资助项目(YJJXM12033)资助。
雷 东(1992-),男,硕士生,主要研究方向为网络协议识别,E-mail:ldd_lw@163.com;王 韬(1964-),男,教授,博士生导师,主要研究方向为网络安全、密码学;马云飞(1993-),男,硕士生,主要研究方向为网络协议识别。

行挖掘与在数据库中进行挖掘的不同点在于:数据库从"数据集合"中挖掘,而比特流是从"比特序列"中挖掘。根据其特点做出以下定义。

设 S 是一个包含 n 个比特的比特流,P 是 S 中一个包含 m 个比特的比特序列($m<n$)。

定义 1　序列包含的比特数即为序列的长度,如:P 的长度为 m。

定义 2　S 的长度为 n,那么 S 中包含的长度为 m 的序列个数记为 r,$r=n-m+1$,若 S 中序列 P 的出现频数为 k,那么 P 的支持度为 $\frac{k}{r}$,记为 $Supp(P)$。

定义 3　当 $Supp(P)$ 大于或等于预先定义好的最小支持度时,则认为序列 P 是频繁序列,否则为非频繁序列。

定义 4　若序列 S 为一个完全随机的比特序列,那么其中长度为 m 的序列期望的支持度为 $\frac{1}{2^m}$,期望出现的频数为:

$$E=\frac{n-m+1}{2^m}=\frac{r}{2^m}\cong\frac{n}{2^m}(n\geqslant m) \tag{1}$$

定义 5　若 S 由大量相同协议的比特流连接组成,每条比特流即为一个如图 1 所示的链路层数据帧,包含各层协议头部字段与用户数据字段。数据流的条数记为 Z,如果每条比特流中都包含某一序列 P,那么 S 中 P 的出现频数至少为 Z。由于物理特性的限制,不同协议都有其最大传输单元 MTU(以太网 MTU 为 1500 字节),记为 L 字节,即 $8\times L$bit,则序列 P 真实出现的频数至少为:

$$Z=\frac{n}{8\times L} \tag{2}$$

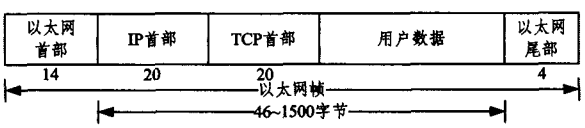

图 1　以太网帧结构示意图

性质 1　单一性:比特序列中元素非"1"即"0",相比于数据库事物简单得多。

性质 2　顺序性:比特序列中各比特之间的位置是既定的、有序的,即"01"不等同于"10"。

3　相关问题分析

3.1　现有挖掘方法的局限性

频繁序列挖掘具体可分为两个步骤:首先选取候选序列并统计其在比特流 S 中出现的频数,其中最重要的环节是多模式匹配[12],多数研究都采用经典的 AC 算法;然后根据出现的频数判定序列是否频繁。

3.1.1　候选序列的选取与频数统计

基于 AC 算法的候选序列频数统计,对于长度为 m 的候选序列,首先枚举出所有可能的 2^m 个候选序列,然后用其构建字典树以进行模式匹配,该树是一个完全二叉树,所需存储空间复杂度为 $O(2^m)$,需要 $2^{m+1}-1$ 个节点,因此无法对长度较长的频繁序列进行挖掘;而且当 m 较大时,枚举出所有的候选序列,其中有一部分可能不会出现在 S 中。虽然采用了"剪枝"算法控制存储空间,但依旧存在以下缺陷:采用的剪枝算法是在建立字典树之后,此时的剪枝对存储空间的减少并没有太大帮助,仅能稍微减少挖掘时间,而剪枝的代价是有可能造成真频繁序列丢失;此外,剪枝是在统计的过程中进行的,剪枝的时机很难把握,如果剪枝过早,真正的频繁序列可能过早被淘汰;而如果剪枝时间过晚,剪枝的效果又不明显。

3.1.2　候选序列是否频繁的判定

现有研究中多数采用定义 3 中提到的方法判定挖掘到的候选序列是否频繁。以序列 P 为例,当 $Supp(P)$ 不小于其期望的支持度,或 P 的出现频数不小于其期望的出现频数时,则判定 P 为频繁序列:

$$\frac{k}{r}\geqslant\alpha\cdot\frac{1}{2^m}\Leftrightarrow k\geqslant\alpha\cdot\frac{r}{2^m} \tag{3}$$

为了防止最小支持度过大造成频繁序列丢失,经常为其乘以一个阈值 $\alpha(0<\alpha<1)$。该方法存在以下缺陷:

1)对 m 较大的长频繁序列不适用。以挖掘长度为 48bit 的序列为例,由式(1)可知,比特流 S 的长度至少要达到 2^{48} 个 bit(约 2.8×1014)才能使得该长度候选序列期望出现的频数达到 1,此时挖掘得到的所有候选序列都满足频繁判别要求,因为序列只要在 S 中出现,其频数 k 至少为 1。要解决此问题,需要 S 具有相当长的长度,并且该长度随挖掘序列长度的增加呈指数增长。

2)阈值 α 的设置需要依赖人工经验,根据挖掘所得到结果的好坏对 α 进行调整。而在实际情况中,面对完全未知的协议,人工无法判断挖掘得到的候选序列是否真的频繁,人工能做的是根据得到的频繁序列的数量对 α 进行调整,如果得到的频繁序列数量太多,则调高 α;如果得到的频繁序列太少,则调低 α。

此外,对于长度较短的频繁序列挖掘,其挖掘结果极易受协议用户数据的影响,从而导致挖掘结果准确率低下,这是由其自身特性所决定的,下节将专门对其进行分析。

3.2　比特流频繁序列挖掘的特点

3.2.1　短频繁序列挖掘易受"冲突"影响

对于长度为 m 的序列,共有 2^m 种不同的候选序列。比特流 S 包含 r 个长度为 m 的序列,要从中挖掘长度为 m 的频繁序列,由抽屉原理可知,当 $r>2^m$ 时,必然至少存在两个序列与同一个候选序列匹配,如图 2 中的序列"11"。

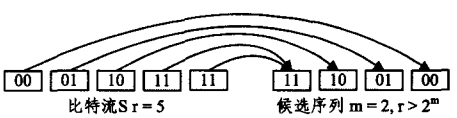

图 2　短频繁序列挖掘"冲突"现象

图 2 中的序列"11"可能来源于两个部分:1)作为真正的频繁序列,其在 S 中出现的频数不止一次,与候选序列多次匹配成功,这是挖掘者所希望得到的;2)来自于协议用户数据,这一现象称为"冲突"。"冲突"会造成频繁序列频数统计偏大,影响挖掘结果。为了更好地减少这种"冲突",现对其进行量化。

在比特流 S 中,相比于协议头部字段中的频繁序列,协议数据部分的序列可以认为是随机的比特序列,且在 S 中,协议数据部分往往占绝大多数(如已知的 TCP,UDP 协议),则认为"冲突"次数(记为 C)约等于定义 4 中所定义的序列期望出现的频数:

$$C=E=\frac{r}{2^m}\cong\frac{n}{2^m} \tag{4}$$

由式(4)可知,"冲突"次数 C 与比特流 S 的长度 n 成正

比，与候选序列 P 的长度 m 成反比。当比特流 S 的长度 n 保持不变时，序列 P 的长度 m 每增加一位，其平均出现的频数便会减半，发生“冲突”的几率也会减半。实验 1 将会分析频繁序列长度对挖掘结果的影响。

3.2.2 长、短频繁序列挖掘的区别与划分

由上述分析可知，候选序列 P 的长度越长，其受“冲突”的影响越小，P 的长度越短，其受“冲突”的影响越大。根据“冲突”对挖掘结果的影响，将频繁序列的挖掘分为长频繁序列挖掘与短频繁序列挖掘，具体的划分方法如下。

对于长度为 m 的候选序列 P，若“冲突”次数不大于频繁序列最小出现频数 Z 的 $\eta(\eta<1)$ 倍，则认为“冲突”对挖掘结果的影响不大。

$$\frac{n}{2^m}\leqslant\frac{n}{8\times M}\times\eta \tag{5}$$

由式(5)可求得长、短频繁序列的分界点 M。即当候选序列长度小于等于 Mbit 时，认为其为短频繁序列，易受协议用户数据部分“冲突”的影响，在挖掘过程中不可忽略；而对于长度大于 Mbit 的候选序列，判定其为长频繁序列，认为其在挖掘的过程中受协议用户数据“冲突”的影响较小，挖掘过程中可忽略。

4 基于 AC 算法的频繁序列挖掘

4.1 基本思想

现有的基于 AC 算法的挖掘方法的局限性在于其对 AC 算法的使用，而并不存在于 AC 算法本身。吴艳梅[6]通过实验对常用的模式匹配算法 AC，BM[13]，AC-BM[14,15] 进行比较，得知在候选序列数量较小的情况下，BM 算法速度非常快，AC-BM 较快，而 AC 相对较慢；但是 BM 算法的时间消耗与候选序列数量成正比，随着候选序列数量的增加，时间消耗也激增，在候选序列数量较大的情况下 BM 算法效率很低，而 AC 算法匹配速度较快。因此本文依旧采用 AC 算法进行模式匹配，基于上述对现有挖掘方法的局限性分析，根据长、短频繁序列挖掘在挖掘过程中存在的不同特点提出相应的长、短频繁序列挖掘方法，然后提出新的频繁序列判别方法。

4.2 AC 算法字典树的建立与剪枝

AC 算法兼具有限状态自动机和字典树的优点，使得其匹配的时间复杂度仅为 $O(n)$[16]。其匹配时间不取决于候选序列数量，因此非常适合于候选序列数量巨大的比特流频繁序列挖掘；而且 AC 算法可以在一次扫描过程中匹配不同长度的候选序列[17]。

算法的框架与文献[3-6]提出的针对 3～8bit 长度的模式序列挖掘算法类似，下面主要介绍改进之处。

4.2.1 字典树的建立

字典树是一种以空间换时间的数据结构，查找时间取决于候选序列的长度，与树中的节点多少无关[18]。首先建立一个大小为 K 的候选序列集合 T(T 中包含 K 个不同的候选序列)，$K=\min(2^m,2^M)$。即当挖掘短频繁序列($m\leqslant M$)时，枚举所有可能的 2^m 个候选序列构成集合 T。而当挖掘长频繁序列($m>M$)时，从数据 S 的起始部位依次取出 2^M 个不同的长度为 m 的候选序列，即将候选序列的数量控制为常数 2^M，以大大降低所需的存储空间，之后用集合 T 中的候选序列构建字典树。

4.2.2 剪枝

剪枝的目的是节省存储空间，提高挖掘速度，但同时剪枝可能会造成频繁序列的丢失，影响挖掘结果的准确性。对于长频繁序列挖掘，减小存储空间，完成频繁序列的挖掘是提高准确率的前提。因此在构建字典树时，从 S 中选择已经出现的 K 个序列作为候选序列，而不是盲目地枚举，该操作相当于一次剪枝操作，其构建的字典树不再是一个完全二叉树，此后的挖掘过程中不再进行剪枝。

对于短频繁序列，由于 m 较小，其建立字典树所需的存储空间是可以接受的，上文分析短频繁序列易受“冲突”影响，使得挖掘结果的准确性成为短频繁序列挖掘的主要问题。因此，在短频繁序列挖掘的过程中，为保证挖掘结果的准确性，不应进行剪枝操作。实验 2 将验证该论述的正确性。

4.3 短频繁序列挖掘

4.3.1 比特流中协议头部字段的定位方法

通过上述基本算法对短频繁序列进行挖掘，由于没有剪枝操作，可以保证得到的候选序列的统计频数都是准确的。但是由于短频繁序列极易受到来自协议用户数据“冲突”的影响，导致得到的统计频数偏大。为了减小这种影响，提高挖掘结果的准确性，提出在比特流 S 中定位协议头部字段的方法，从而在挖掘过程中仅扫描协议头部字段，跳过用户数据部分。该算法的描述如下。

输入：比特流序列 S，候选序列长度 m

输出：协议头部字段所在区域

步骤：

Step1 将 S 划分为长度为 U bit 的等长序列，分别由序列起始位置 $a_1,a_2,\cdots,a_n$ 表示。

Step2 从 $a_i(i=1,2,\cdots,n)$ 序列中寻找长度为 m 的所有不同序列作为候选序列，构建字典树。

Step3 根据建好的字典树从 a_i 起始向后匹配至少 10 个协议头部字段，即 10×8×L bit，统计 a_i 中候选序列在其后 10×8×L bit 内的平均出现频数 average。

Step4 若 average 高于设定阈值，说明 a_i 序列里包含频繁出现的协议头部字段，记录该序列位置；反之，说明 a_i 序列内无频繁出现的序列，即不包含协议头部字段。

Step5 跳转到 Step2，对下一序列 a_{i+1} 进行处理，直到扫描完最后一个序列 a_n。

Step6 输出记录的所有协议头部字段所在序列。

该算法的时间复杂度主要取决于 AC 算法的模式匹配过程，而 AC 算法的时间复杂度为 $O(n)$，本算法在执行过程中对部分 S 中的部分比特进行了多次扫描，其基本操作次数 $C(n)=C(\frac{n-80L}{U}\times 80L)$，当 L 与 U 都为常数时，其时间复杂度为 $O(n)$。

图 3 更直观地描述了该方法的过程，a_1，a_3 及 a_5 序列都包含了至少一个协议头部字段，其求得的 *average* 应该较高；而 a_2，a_4 及 a_6 区域里没有包含协议头部字段，求得的 *average* 应该较低。实验 3 验证了该方法的有效性。

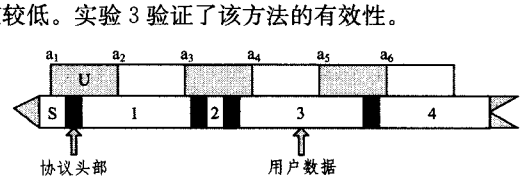

图 3 比特流协议头部字段定位

4.3.2　短频繁序列挖掘方法

得到协议头部字段存在的区域后，基于 4.2 节提出的方法建立字典树，仅对协议头部字段所在的区域进行扫描，跳过大量影响挖掘结果的用户数据，挖掘其短频繁序列。虽然跳过协议用户数据部分可以减少挖掘所用时间，但其主要时间还是消耗在协议头部字段的定位过程中，因此其时间复杂度主要取决于协议头部字段定位算法的复杂度。该算法的主要目的是提高挖掘结果的准确性。实验 4 验证了该方法的有效性。

4.4　长频繁序列挖掘

4.2 节提出的针对长频繁序列挖掘的基础方法从 S 头部开始，取 2^M 个候选模式序列构建字典树，然后扫描 S 进行挖掘。该方法有一个缺陷：挖掘结果的准确性依赖于 S 头部所取的候选序列的准确性，如果 S 头部所取的候选序列在其附近的小范围内频繁但在整个 S 中并不频繁，则挖掘的结果必然造成严重的频繁序列的丢失。为了避免这一缺陷，对其加以改进，算法描述如下。

输入：比特流序列 S，候选模式序列长度 m

输出：频繁序列集合 Result

流程：

Step1　从 S 头部位置选取候选序列，构建字典树并扫描 S 进行挖掘，得到频繁序列集合 Result。

Step2　更改候选序列选取位置，再次挖掘得到新的频繁序列集合，与 Result 进行比较并合并得到新的 Result。

Step3　如果 Result 连续 2 次或累计 3 次没有更新，认定 Result 达到最佳，返回 Result，否则跳转到 Step2。

这将使得挖掘过程至少要对 S 扫描 3 次才能得到结果。因此，候选序列选取位置依次取 0，$\frac{1}{3}n$，$\frac{2}{3}n$，$\frac{1}{6}n$，$\frac{1}{2}n\cdots$，其中 n 为 S 的总长度，如图 4 所示。

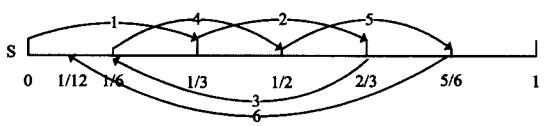

图 4　候选序列选取位置

这种位置的选取方法能够较为“公平”地将选取候选序列的位置均匀分布在 S 各处，使其快速得到最佳结果。实验 5 验证了该方法的有效性。

4.5　新的频繁序列判别方法

鉴于现有的基于支持度的频繁序列判别方法的局限性，提出一种新的频繁序列判别方法：对于挖掘长度为 m 的候选序列，将得到的序列按照其出现频数从大到小进行排序，认定其出现频数最大的前 $\min(Q,2^m)$ 个序列是频繁的，即当 m 较小时认为其挖掘得到的所有候选序列都是频繁的。该方法可以有效降低现有频繁序列判别方法中对人工经验的依赖性，使其更加适用于未知协议的环境。

5　实验结果与分析

本节使用实际捕获的数据对所提方法进行测试。5.1 节对实验所需的数据与参数进行了解释；5.2 节分析了频繁序列的长度 m 对挖掘结果的影响；5.3 节验证了短频繁序列挖掘不应剪枝的论述；5.4 节验证了比特流协议头部字段定位方法的有效性；5.5节验证了短频繁序列挖掘方法的有效性；5.6 节验证了长频繁序列挖掘方法的有效性。

5.1　实验设置

实验所使用的数据均来自互联网，由 Wireshark 软件捕获某一主机上一段时间内同种协议的交互数据包，其经过处理后转换成连续的比特流数据，其中 $S2$ 与 $S3$ 的数据量都足够大，如表 1 所列。

表 1　实验所用数据详情

序号	协议	数据包数(个)	比特流长度(bit)
S1	UDP	1000	4235920
S2	UDP	10000	32984040
S3	TCP	10000	76100120

本文中提出的算法有很多参数，实验中对其赋值，如表 2 所列，并给出了合理解释。

表 2　实验所用参数设置

L	η	M	K	U	Q
1500	0.1	16	$\min(2^m,65536)$	1000	100

实验数据来自以太网，由其物理特性可知其每个数据包的最大长度 L 为 1500byte[19]，即 12000bit；长、短频繁序列划分中“冲突”次数占实际频繁序列最小出现频数 Z 的比例 η 的取值越小，“冲突”对挖掘结果的影响就越小，挖掘结果越精确，且实际情况中捕获到的数据长度大部分小于其最大值 L，使得 Z 的实际值远小于估计值，因此取 $\eta=0.1$ 可以满足要求；L 与 η 确定后由式(5)求得长、短频繁序列划分的临界点 $M=16$bit；构建字典树时，候选序列数量 $K=\min(2^m,2^{16})$；定位协议头部字段的算法中，U 不能太小，应保证该区域至少能覆盖一个协议头部字段，U 也不能太大，否则会包含太多用户数据，失去挖掘的意义，本实验取 $U=1000$。对现有以太网下的已知协议进行分析统计，得知其协议头部字段固定部分的频繁序列数量不超过 100 个，因此提出的新频繁序列判别方法中 $Q=100$。

5.2　候选序列长度对挖掘结果的影响

3.2 节从理论上分析了长、短频繁序列挖掘各自的特点，通过实验 1 分析了候选序列长度 m 对挖掘结果的影响。实验从比特流数据 $S1$ 中挖掘不同长度的频繁序列，统计其挖掘时间以及所得频繁序列的平均出现频数，结果如图 5 所示。

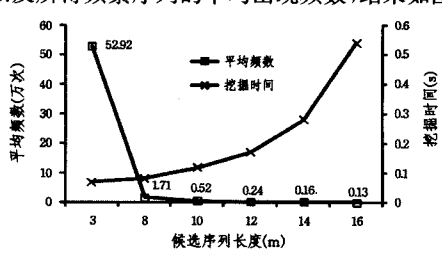

图 5　候选序列长度对挖掘结果的影响

由图 5 可知：随着候选序列长度 m 的增加，挖掘所需的时间逐渐增多，虽然 AC 算法的时间复杂度为 $O(n)$，但挖掘过程还包括字典树的构建以及频繁序列的判别选取等过程，该过程增大了总体时间，但是总体挖掘时间较短，可见 AC 算法的高效性。此外，随着候选序列长度 m 的增加，挖掘所得频繁序列的平均出现频数急剧降低，且逐渐接近于其目标频数 1000。可见 m 越长，挖掘结果“冲突”次数越少，使得挖掘结果的准确性越高。

5.3　短频繁序列挖掘不应剪枝

4.2 节提出对短频繁序列进行挖掘，不应进行剪枝操作。

为证明该论述，进行实验 2。实验从比特数据流 $S1$，$S2$ 和 $S3$ 中挖掘长度最长的短频繁序模式列(16bit)，所需时间随候选模式数量 K 的变化趋势如图 6 所示。

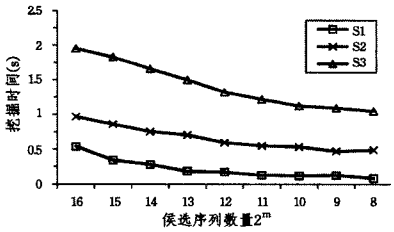

图 6 候选序列数量对短序列挖掘的影响

由图 6 可知：当候选序列数量 K 从 2^{16} 指数下降到 2^8，即剪枝幅度越来越大时，挖掘的时间降低得很少，且对于数据量巨大的 $S3$，其挖掘仅耗时 2s 多，而影响挖掘速度的主要因素是比特流数据 S 的大小 n。实验证明，对于短频繁序列，其需要的存储空间并不大，因此不需要对其进行剪枝操作以保证挖掘结果的准确性，该方法避免了现有文献中所提剪枝操作中阈值的选择问题，同时可以降低其对挖掘结果准确率的影响。

剪枝操作对短频繁序列的挖掘速度提升很小，其益处在于可以节省存储空间，但是在存储空间可以满足的情况下，保证挖掘结果的准确率才是主要目标。

5.4 比特流协议头部字段定位

实验 3 以 $S1$ 为实验数据，验证比特流协议头部字段定位方法。分别令 $m=32$ 与 $m=48$，对 $S1$ 中局部协议头部字段进行定位，理论上无论 m 取何值，对于同一个比特流 S，得到的协议头部字段位置是相同的，m 之所以取较大的值，是为了减小“冲突”的影响，提高准确率，结果如图 7 所示。

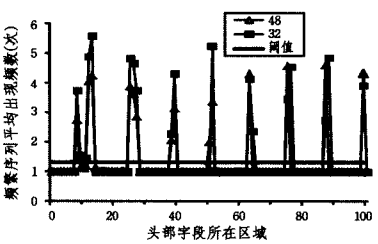

图 7 比特流协议头部字段定位结果

图 7 中横坐标上的点代表 $S1$ 中一段长度为 100bit 的区域。由图 5 可知：大多数点出现频数的平均值 *average* 都约等于 1，说明其中不包含协议头部字段；而只有在少数点处的 *average* 突然增加，且明显高于周围的区域，说明这些峰值点所对应的区域中包含协议头部字段。此外，$m=48$ 与 $m=32$ 时所对应图中的峰值点几乎重叠，也验证了求得的协议头部字段的位置是正确的。

5.5 短频繁序列挖掘

实验 4 使用基于协议头部字段定位的短频繁序列挖掘方法从 $S1$ 中挖掘长度为 8 的短频繁序列，与现有文献的基本挖掘方法进行比较。

所得结果如表 3 所列，相比基本方法，改进后的方法在扫描比特数、频繁序列平均出现频数方面都有大幅减少；而对于关键的挖掘结果的准确性，在挖掘得到的 100 个频繁序列中，相比于基本方法，改进方法新增了 18 个新的频繁序列，根据出现频数进行排序的排位上升超过 20 位的频繁序列有 20 个，由此可知改进方法得到的频繁序列的准确率提升了 38%。

表 3 短频繁序列挖掘结果

比较项目		现有方法	改进后方法
扫描比特数		4235920	940000
频繁序列平均出现频数		18013	3490
准确性	增加数	—	18
	排位上升数	—	20
	总计	—	38

这种定位方法虽然不能精确定位头部字段具体的开始与结束位置，并且在挖掘过程中依旧能扫描到一小部分协议用户数据，但是与跳过的大量的协议用户数据相比，该方法对短频繁序列挖掘结果的准确性已有较大提升。实验证明，基于协议头部字段定位的短频繁序列挖掘方法解决了文献[3-6]中所提的短频繁序列挖掘结果的准确率随着协议用户数据的增多而急剧下降的问题。

5.6 长频繁序列挖掘

实验 5 使用提出的长频繁序列挖掘方法对数据量最大的 $S3$ 进行 32bit 及 48bit 的长频繁序列挖掘，结果如表 4 所列。

表 4 长频繁序列挖掘结果

候选序列长度	扫描次数	起始位置	Result 更新个数	累计用时(s)
32	1	0	—	5.84
	2	1/3	9	11.72
	3	2/3	0	17.52
	4	1/6	0	23.29
48	1	0	—	7.67
	2	1/3	1	15.16
	3	2/3	0	22.66
	4	1/6	0	30.18

由表 4 可知，通过更换选取候选序列的位置对 S 进行多次扫描，不同程度地修正了挖掘结果有，提高了准确率。而且在长度为 48bit 的频繁序列挖掘实验中，第二次扫描得到的一个新的频繁序列是整个结果中出现频数最高的。实验证明，该算法解决了现有文献在对长频繁序列挖掘时所遇到的构建字典树所需空间随候选序列长度呈指数增长的问题，同时通过合理的剪枝方法保证了挖掘结果的准确性。

对于长频繁序列挖掘是否可以采用协议头部字段定位方法仅扫描协议头部字段来提升挖掘准确率的问题，实验 5 对数据 $S1$ 中长度为 48bit 的频繁序列进行挖掘比较，结果如表 5 所列。

表 5 基于协议头部字段定位的长频繁序列挖掘结果

比较项目		长频繁序列挖掘方法	协议头部定位挖掘方法
扫描比特数		4235920	940000
频繁序列平均出现频数		575	547
准确性	增加数	—	1
	排位上升数	—	0
	总计	—	1

由表 5 可知，对于长频繁序列挖掘，采用定位协议头部字段的方法跳过对协议用户数据进行扫描得到的结果，与本节提出的方法所得到的结果极其接近，这也证明了长频繁序列挖掘的结果不易受协议用户数据的影响。此外，本实验也证明了提出的议头部字段定位方法的准确性。

结束语 针对短频繁序列挖掘结果的准确性易受大量协议用户数据的影响而大大降低的问题，提出比特流协议头部字段定位方法，该方法跳过对协议用户数据的挖掘，大大提高

了短频繁序列挖掘结果的准确性。针对长频繁序列挖掘需要大量存储空间而无法挖掘的问题，提出改进的候选序列选取方法，将其数量控制为一个常数，大大缩减了存储空间。同时，为了解决这种方法带来的准确率问题，提出更改候选序列选取位置进行多次挖掘的方法，确保了其挖掘结果的准确性。

本文所提方法不仅能挖掘到短频繁序列，还能直接挖掘长频繁序列，在此基础上，在时间和空间允许的情况下，以提高挖掘结果的准确率为主要目标。最后通过实验验证了所提方法的有效性，对后续未知协议格式推断等研究打下了良好基础。

参 考 文 献

[1] 爱德华·华兹. 信息战原理与实战[M]. 吴汉平，译. 北京：电子工业出版社，2003：1-20.

[2] LI Fen, LI Tong, ZHANG Chun-rui, et al. Length identification of unknown data frame[C]// 2012 Eighth International Conference on Computational Intelligence and Security, Washington, DC, USA, IEEE Computer Society. 2012：674-677.

[3] JIN Ling. Study on bit stream oriented unknown frame head identification[J]. Shanghai：Shanghai Jiaotong University, 2011：29-39. (in Chinese)
金陵. 面向比特流的未知帧头识别技术研究[D]. 上海：上海交通大学，2011：29-39.

[4] WANG He-zhou, XUE Kai-ping, HONG Pei-lin, et al. An unknown link protocol bit stream segmentation algorithm based on frequent statistics and association rules[J]. Journal of University of Science and Technology of China, 2013, 43(7)：554-560. (in Chinese)
王和洲，薛开平，洪佩琳，等. 基于频繁统计和关联规则的未知链路协议比特流切割算法[J]. 中国科技大学学报，2013，43(7)：554-560.

[5] SONG Jiang. Unknown protocol identification in wireless environment[D]. Chengdu：University of Electronic Science and Technology, 2013：23-27. (in Chinese)
宋疆. 无线网络环境下未知协议发现探索研究[D]. 成都：电子科技大学，2013：23-27.

[6] WU Yan-mei. The frame location and protocol feature analysis from the bit-stream in the wireless network[D]. Chengdu：University of Electronic Science and Technology, 2014：10-11. (in Chinese)
吴艳梅. 无线环境下比特流协议帧定位与特征分析[D]. 成都：电子科技大学，2014：10-11.

[7] AHO A V, Corasick M J. Efficient String Matching：An Aid to Bibliographic Search[J]. Communications of the ACM, 1975, 18(6)：330-340.

[8] TAO Shu-song, CHEN Xing-shu, YIN Xue-yuan. Identification of Unknown Frame Structure Based on Data Mining[J]. Journal of Sichuan University (Engineering Science Edition), 2014, 46(suppl)：155-159. (in Chinese)
陶术松，陈兴蜀，尹学渊. 基于数据挖掘的未知帧结果识别[J]. 四川大学学报(工程科学版)，2014，46(增刊)：155-159.

[9] WANG Yong, WU Yan-mei, LI Fen, et al. Protocol identification association analysis in mobile network environment[J]. Application Research of Computers, 2015, 32(1)：243-248. (in Chinese)
王勇，吴艳梅，李芬，等. 面向比特流数据的未知协议关联分析与识别[J]. 计算机应用研究，2015，32(1)：243-248.

[10] JU Yu-jian, XIE Shao-bin, ZHANG Wei. Research and simulation of optimization process for network protocol frame segmentation[J]. Computer Simulation, 2015, 32(1)：318-321. (in Chinese)
琚玉建，谢绍斌，张薇. 基于自适应权值的数据报指纹特征识别与发现[J]. 计算机仿真，2015，32(1)：318-321.

[11] 韩家炜，裴健. 数据挖掘概念与技术[M]. 范明，孟小峰，译. 北京：机械工业出版社，2012：157-169.

[12] ZHENG Q. An improved multiple patterns matching algorithm for intrusion detection[C]// International Conference on Intelligent Computing and Intelligent Systems. Xiamen, China：IEEE Press, 2010：124-127.

[13] BOYER R S, MOORE J S. A fast string searching algorithm [J]. Communications of the Association for Computing Machinery, 1977, 20(10)：762-772.

[14] FAN Jang-jong, SU K. An efficient algorithm for matching multiple pattems[J]. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(2)：339-351.

[15] COIT C J, STANIFORD S, MCALERNY J. Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort[C]// Proc 2nd DARPA Information Survivability Conference & Exposition II. IEEE CS, 2001：367-373.

[16] SONG Hua, DAI Yi-qi. A new fast string matching algorithm for content filtering and detection[J]. Journal of Computer Research and Development, 2004, 41(6)：940-945. (in Chinese)
宋华，戴一奇. 一种用于内容过滤和检测的快速多关键字识别算法[J]. 计算机研究与发展，2004，41(6)：940-945.

[17] SOURDIS I, PNEVMATIKATOS D. Pre-decoded CAMs for efficient and high-speed NIDS pattern matching[C]// Field-Programmable Custom Computing Machines. 2004：258-267.

[18] KNUTH D. The Art of Computer Programming Volume 3：Sorting and Searching(2th ed)[M]. Boston：Addison-Wesley Press, 1997：492.

[19] Tanenbaum A S, Wetherall D J. 计算机网络(5 版)[M]. 严伟，潘爱民，译. 北京：清华大学出版社，2012：151-161.

(上接第 102 页)

廖律超，蒋新华，邹复民，等. 一种支持轨迹大数据潜在语义相关性挖掘的谱聚类方法[J]. 电子学报，2015，43(5)：956-963.

[7] DING S F, JIA H J, SHI Z Z. Spectral clustering algorithm based on adaptive Nyström sampling for big data analysis[J]. Journal of Software, 2014, 25(9)：2037-2049 (in Chinese).

[8] NG A Y, JORDAN M I, WEISS Y. On spectral clustering：Analysis and an algorithm[C]// Proceedings of Advances in Neural Information Processing Systems, 2002, 14：849-856.

[9] ZHANG Yan, HUAN Fei. Colour image segmentation method using gentic algorithm[J]. Computer Applications and Software, 2011, 38(3)：237-240. (in Chinese)
张艳，宦飞. 一种应用遗传算法的彩色图像分割方法[J]. 计算机应用与软件，2011，38(3)：237-240.

[10] WANG Li-guo, WEI Fang-jie. Band selection for hyperspectral imagery based on combination of genetic algorithm and ant colony algorithm[J]. Journal of Image and Graphics, 2013, 18(2)：235-242. (in Chinese)
王立国，魏芳洁. 结合遗传算法和蚁群算法的高光谱图像波段选择[J]. 中国图像图形学报，2013，18(2)：235-242.