doi:10.11959/j.issn.1000-436x.2015148

HybridFA：一种基于统计的 AC 自动机空间优化技术

熊刚[1]，何慧敏[2]，于静[1]，刘燕兵[1]，郭莉[1]
（1. 中国科学院 信息工程研究所，北京 100093；2. 中国移动（深圳）有限公司，深圳 518031）

摘 要：针对高级 Aho-Corasick (AC)自动机为提高串匹配速度而造成的空间浪费问题，研究发现数据流对自动机节点的访问规律，据此提出基于数据访问特征的混合自动机构建算法 HybridFA。分别研究了基于访问频率、访问层次以及结合上述 2 种特征对 AC 自动机的部分节点实现完全化的算法。在 Snort、ClamAV、URL 等真实数据集上的实验结果表明，HybridFA 算法的存储空间低于高级 AC 自动机的 5%。此外，结合访问频率和访问层次的改进算法在保证匹配速度的同时具有更强的数据适应性。

关键词：多模式串匹配；空间优化；高级 AC 自动机；统计策略；节点完全化
中图分类号：TP301.1 **文献标识码**：A

HybridFA: a memory reduction technique for the AC automata based on statistics

XIONG Gang[1], HE Hui-min[2], YU Jing[1], LIU Yan-bing[1], GUO Li[1]
（1. Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China;
2. China Mobile Group, Guangdong Co. Ltd., Shenzhen Branch, Shenzhen 518031, China)

Abstract: Despite the fast speed in multiple string matching tasks, the advanced Aho-Corasick(AC) automata wastes storage memory to a great extent. Study indicated that the automata states have specific statistical access characteristics in practice. Accordingly, a series of algorithms based on statistical characteristics for building hybrid finite automata, named HybridFA, are proposed. This work completes partial states of the AC automata according to different features, including access frequency, state hierarchy, and combined characteristics respectively. Experimental results on the real-world datasets like Snort, ClamAV, and URL show that the storage space of HybridAC is reduced to less than 5% of the space cost by the advanced AC automata. Furthermore, HybridFA based on combined characteristics achieves the superior performance on matching speed and robustness comparing to other proposed algorithms.

Key words: multiple string matching; memory reduction; advanced AC automata; statistical strategy; state completing

1 引言

多模式串匹配是计算机科学领域最经典的问题之一，目的是在任意字符串文本中找出已知的一组特定字符串集合出现的所有位置。网络内容安全处理主要采用串匹配算法对网络分组的内容负载进行检测、分析和过滤。典型的网络内容安全应用软件的入侵检测/防御系统和反病毒反垃圾邮件检测系统都采用串匹配算法来检测恶意数据[1~3]。基于自动机的串匹配算法是多模式串匹配采用的主要研究方法之一，它以自动机作为数据结构，在搜索过程中通过搜索文本中自动机所能识别的语言来实现匹配，该类方法性能相对稳定，因而在实际系统中被广泛使用。但该类方法空间开销较大，匹

收稿日期：2014-07-08；**修回日期**：2014-10-15
基金项目：中国科学院战略性科技先导专项基金资助项目（XDA06030602）；国家高技术研究发展计划（“863”计划）基金资助项目（2011AA010703）；国家自然科学基金青年基金资助项目（61202477）
Foundation Items: The Strategic Priority Research Program of the Chinese Academy of Sciences (XDA06030602); The National High Technology Research and Development Program of China (863 Program) (2011AA010703); The National Natural Science Foundation of China (61202477)

配速度较慢。

Aho-Corasick (AC) 自动机[4]是最经典、实际应用最广的自动机之一，开源病毒检测系统 ClamAV 和开源入侵检测系统 Snort 均使用 AC 自动机作为其基本的数据结构。基于 AC 自动机的串匹配算法称为 AC 算法，其思想是使用 Trie 树存储所有的模式串，然后从 Trie 树的根节点开始层次遍历，构造节点间的后缀链（supply link）[5]。在匹配过程中，若当前状态无法识别输入字符时，就沿着后缀链回溯（也称为失效转移），直到能够识别该字符的状态为止。该算法可以在 $O(n)$时间复杂度内找到文本中的所有目标模式（n 为给定文本长度），而与模式集合的规模无关。但其突出的问题是数据流在匹配时往往需要进行多次回溯才能跳转到有效状态继续匹配，严重影响了匹配速度。

高级 AC 自动机是在 AC 自动机的基础上，将后缀链进行改进，直接指向当前状态的后继状态。也就是说，所有状态对于字母表的每一个字符都有相应的转移，称为一个完全的自动机。在匹配过程中无需沿着后缀链回溯，而是直接跳转到它的后继状态，因而匹配速度更快。但高级 AC 自动机的空间复杂度为 $O(|P||\Sigma|)$（其中 P 是模式串集合，Σ 是字符集），存在巨大的空间浪费。

为解决自动机因为存储空间引发的性能瓶颈，国内外学者提出了许多经典的压缩方案。1984 年 Dencker 等[6]最早提出了一种行压缩方法，使状态转移表中每一行只存储非空的转移边对应的读入字符和下一跳状态，但该方法仅对稀疏的状态转移表具有良好的压缩效果，当状态转移表相对稠密时，该方法的效果变差，甚至会超过压缩前的存储空间。文献[7~9]使用一个一维数组重叠排列状态转移表中所有的行，但必须保证各行的非空元素不得相互冲突，另外用 2 个数组分别存储每一行的偏移位置和一维数组中每个元素所属的行。该方法具有较好的压缩效果，并且可以达到 $O(1)$ 的状态转移速度，但处理过程中峰值内存太大，导致其不能处理大规模的串匹配问题。此外，刘燕兵等[10]利用 suffix-trie 和双数组相结合的数据结构来存储模式串，该方法在压缩存储空间的同时也保持了快速的访问速度。实验表明该方法比较适合关键词规模为 10 000~20 000 的应用环境。

上述自动机压缩方法的基本思想大都是针对自动机的状态转移表，删除以空转移为主的冗余状态转移边，只保存有用的信息，在搜索过程中通过计算来查找下一跳状态。此类方法都不能适用于无空转移状态的高级 AC 自动机的压缩。而高级 AC 自动机空间开销大的主要原因是它将每个节点都完全化，而在实际应用系统中并不是所有节点都具有较高的访问频率，将访问频率相对较低的节点完全化会占用大量存储空间，而不会带来匹配速度的显著提升。本文将充分分析待处理数据流量的特征，提出基于数据访问特征的混合自动机构建算法 HybridAC。根据不同的数据流特征，分别研究了按访问频率完全化和按层次完全化的 2 种混合自动机的构建算法 HybridAC_1 和 HybridAC_2，并进一步综合考虑数据流量的特征，结合上述 2 种方法的优点，提出改进算法 HybridAC_3。在 Snort、ClamAV、URL 等真实数据集上的实验表明，本算法在保证匹配速度的同时，大大降低了高级 AC 自动机的存储空间。

2 算法设计

2.1 算法提出的依据

AC 自动机作为最经典、实际应用最广的自动机之一，广泛应用于入侵检测/防御系统和反病毒反垃圾邮件检测系统。根据自动机构造的程度，AC 自动机可以分为基本的 AC 自动机和高级 AC 自动机（完全化的 AC 自动机）。基本的 AC 自动机只保存模式串的 Trie 结构和每个节点的回溯状态，在搜索过程中，如果状态转移不成功，则需要回溯；高级 AC 自动机对于字符集中的每个字符，所有的状态都有相应的转移，在搜索过程中不需要回溯，因而匹配速度更快。但在高级 AC 算法中，自动机的所有状态节点都被完全化，每个节点具有相同的地位。然而在实际应用中，自动机的每个节点的访问频率各不相同，把一个数据访问频率为零或者接近于零的节点完全化，存在很大的空间浪费。

数据流量对自动机不同节点的访问频率各不相同，下面以开源入侵检测系统 Snort 中的模式串规则为例，分析数据流量对自动机各状态节点的访问特征，本实验中待扫描的文本数据来自 MIT 公开的一组真实的网络数据，所采用的实验方法如下。

1) 按频率统计方法

统计高级 AC 自动机各个节点的访问频率，按照访问频率从大到小对自动机的各个节点排序，依次对访问频率累计求和，分别统计每个频率界限内

的节点数目。统计结果如表 1 所示，可见，前 408 个节点的累计访问频率已经达到 91%，节点数目所占比例不超过 1%；前 3% 的节点累计访问频率达到 98%。

表 1　高级 AC 自动机各节点的访问频率统计

节点数目	节点数目所占百分率	累计访问频率
408	0.86%	91%
465	0.98%	92%
531	1.12%	93%
613	1.29%	94%
715	1.51%	95%
850	1.79%	96%
1 057	2.23%	97%
1 404	2.96%	98%

2) 按层次统计方法

把高级 AC 自动机的 Trie 树结构图的根节点标记为第一层节点，按照广度优先搜索的方式，根节点的后续未标记节点为第 2 层节点，以此类推。分别统计各层节点的访问频率，实验结果如表 2 所示。可以看出，随着层次的增大，每层节点的访问频率逐渐降低。根据统计，前 3 层节点的累计访问频率已经超过 95%，第 8 层及以后的节点访问频率几乎为零。

表 2　高级 AC 自动机各层访问频率统计

访问层次	访问频率	累计访问频率
2	43.6%	86.3%
3	8.9%	95.2%
4	2.0%	97.2%
5	1.0%	98.2%
6	0.4%	98.6%
7	0.4%	99.0%
8	0.3%	99.3%
9	0.2%	99.5%

统计结果表明，大部分数据流只访问小部分节点，大多数节点几乎未被访问过；其次，数据流量倾向于访问靠近根节点的低层节点，离根节点较远的高层节点的访问频率接近于 0。根据数据流量的这些显著特征，本文将把访问频率高的节点和层次完全化，其余节点只保存基本的 Trie 树结构和回溯状态。根据表 1 和表 2 的统计分析，需要完全化的节点比例很小，所以该算法可以大幅度降低存储空间。

2.2　HybridFA 算法的基本思想

HybridFA 算法是根据数据流量的访问特征，分别按访问频率完全化（HybridFA_1 算法）和按层次完全化（HybridFA_2 算法）2 种方式来改善高级 AC 自动机的空间浪费问题。按访问频率完全化是根据大部分数据流量只访问小部分节点的特征，把访问频率高的节点完全化；按层次完全化则根据数据流量倾向于访问靠近根节点的低层节点的特征，把低层节点完全化。2 种算法的流程如图 1 所示。

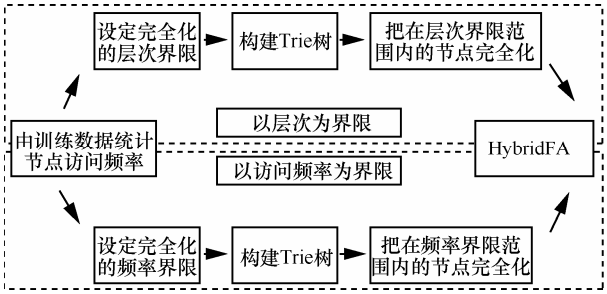


图 1　基于访问层次完全化和访问频率完全化构建 HybridFA 的流程

2.3　HybridFA 算法的设计与实现

HybridFA 算法在数据训练阶段统计各状态节点的访问频率，确定完全化的访问频率界限和层次界限，然后根据按访问频率完全化和按层次完全化 2 种算法分别构建由完全化节点和不完全节点混合构成的自动机 HybridFA，并实现数据扫描。

2.3.1　数据训练

数据训练阶段的主要任务是分析数据流量的访问特征，统计自动机各状态节点的访问频率，分别确定按访问频率完全化和按访问层次完全化 2 种方法所需要完全化的节点。具体步骤如下。

Step1　对模式串规则构造高级 AC 自动机。首先对模式串规则构建 Trie 树，然后为每个节点计算其回溯状态，并把所有节点都完全化，也就是使所有节点包含对任意字符的下一跳状态，至此完成对高级 AC 自动机的构建。

Step2　扫描训练数据，在高级 AC 自动机对数据进行匹配的过程中，统计自动机每个节点的访问频率。

Step3　按照访问频率的递减顺序，对自动机的每个节点排序，并对排序后节点的访问频率累计求和。选择合适的访问频率作为访问频率界限 fre，把在访问频率界限范围内的所有节点标记为 Hybrid_1 算法需要完全化的节点，其他节点仍按照 AC 自动机的构造方式保留失效转移。

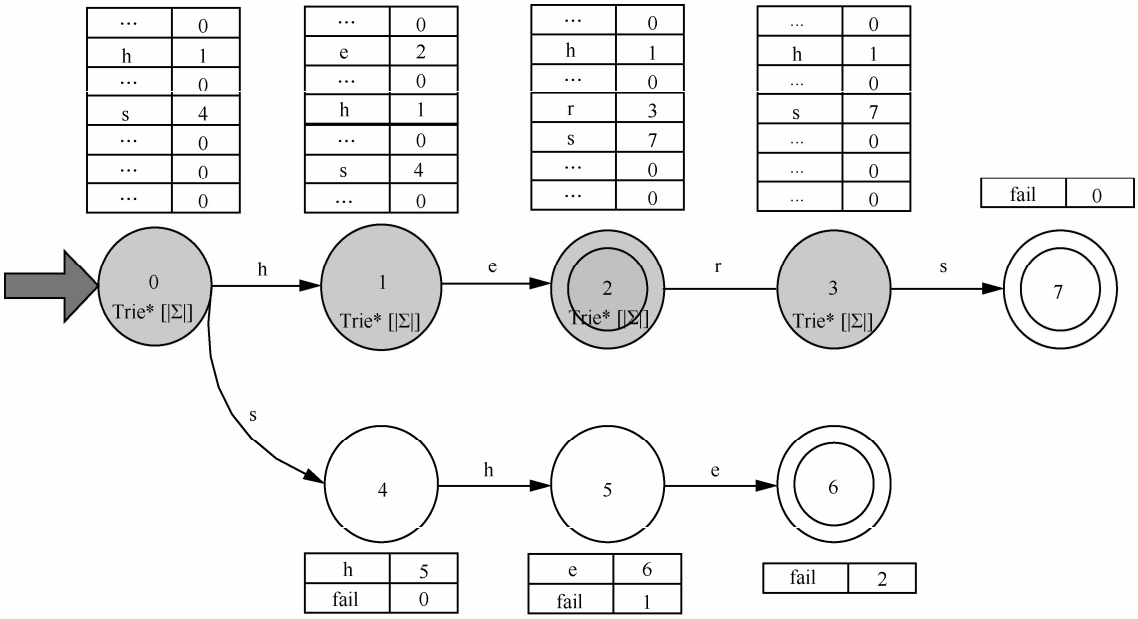


图 2　按照访问频率完全化构建 HybridFA 自动机示意

Step4　广度优先遍历高级 AC 自动机，从根节点开始，按层次递增顺序统计每层节点的累计访问频率，并计算累计层次的访问频率之和占所有节点访问频率之和的比值，选择合适的比值作为访问层次界限 level，把层次界限范围内的所有节点标记为 Hybrid_2 算法需要完全化的节点，其他节点仍保存可识别字符的跳转和失效转移。

2.3.2　按访问频率完全化算法

按访问频率完全化的方法在预处理过程中，首先对模式串规则构建 Trie 树，然后为每个节点计算其回溯状态，并把在访问频率界限 fre 范围内的节点完全化，即对任意字符，计算该节点的下一跳状态，称此算法为 HybridAC_1。图 2 是按照访问频率完全化构建的混合自动机示意。编号为{0，1，2，3}的灰色节点表示被完全化的节点，节点上的表格表示该节点存储的对任意字符的跳转状态；标号为{4，5，6，7}的白色节点表示未被完全化的节点，节点上的表格表示该节点存储的可识别字符的跳转和失效转移。图 3 是该自动机的构建过程的伪代码（其中 *Trie*(*P*) 采用文献[11]中的 *Trie*(*P*) 算法）。

图 4 为 HybridFA 算法扫描过程的伪代码，在扫描过程中，逐个读入文本字符，此时可能出现 2 种情况。

1) 如果当前状态 *Current* 为完全化的状态，则根据当前读入字符 c，直接从状态转移表中查找其对应的下一跳状态 $GoTo(Current,c)$，并令 $Current \leftarrow GoTo(Current,c)$。

2) 如果当前状态 *Current* 为未完全化的状态，首先查找当前状态的状态转移表，查看 $GoTo(Current, c)$ 是否为 NULL，如果 $GoTo(Current, c) \neq$ NULL，则令 $Current \leftarrow GoTo(Current,c)$，如果 $GoTo(Current, c) =$ NULL，则根据其回溯状态回溯，直到找到 *Current* 使 $GoTo(Current,c) \neq$ NULL，令 $Current \leftarrow GoTo(Current,c)$。

```
Build_Hybridfa(P={p^1, p^2,⋯, p^l }=flag)
1)  HFA_trie ←  Trie(P)
2)      GoTo  is transition function
3)      Supply  is supply function
4)  initial ← root of HFA_trie
5)  Create a new non terminal state head
6)  For each c in Σ   DO GoTo(head,c) ← initial
7)  Supply(initial)  ← head
8)  Qucue ← empty
9)  Qucue · push(initial)
10) While Queue ≠ empty
11)     parent ← Queue.front()
12)     For each c in ∑
13)         Current ← GoTo(parent,c)
14)         If Current ≠ θ Then  Queue·push(Current)
15)         If flag(parent) ← true OR Current ≠ θ Then
16)             Down ← Supply(parent)
17)             While GoTo(Down,c)= θ Do Down ← supply(Down)
18)             If Current= θ Then
19)                 GoTo(Parent,c) ← GoTo(Down,c)
20)             Else
21)                 Supply(Current) ← GoTo(Down,c)
22)                 If Supply(Current) is terminal state Then
23)                     F(Current) ← F(Current)∪F(Supply(Current))
24)                 End  of if
25)             End  of if
26)         End  of if
27)     End for
28)     Q.pop()
29) End of while
```

图 3　HybridFA 自动机构建过程伪代码

```
HybirdFA(P=p^1,p^2…p^r,T=t_1,t_2…t_n,flag)
1)  Preprocessing
2)      HFA ← Build_HybridFA(P,flag)
3)  Searching
4)      Current ← Initial state of the automaton HFA
5)      For i ∈ 1…n Do
6)          While flag(Current) = false AND GoTo_HFA(Current,t_i)=θ
7)              AND Supply_HFA(Current)≠θ
8)              Current ← Supply_HFA(Current)
9)          End of while
10)         If GoTo_HFA(Current,t_i)≠ θ Then
11)             Current ← GoTo_HFA(Current,t_i)
12)         Else Current ← initial state of HFA
13)         End of if
14)         If Current is terminal Then
15)             Mark all the occurrences(F(Current),i)
16)         End of if
17)     End of for
```

图 4　HybridFA 算法扫描过程伪代码

处理完当前读入字符，判断当前状态 *Current* 是否为终止状态，如果为终止状态则输出所有的匹配结果，否则，读入下一个文本字符，继续对文本进行扫描。

2.3.3　按访问层次完全化算法

按访问层次完全化的算法在预处理过程中，首先对模式串规则构建 Trie 树，然后为每个节点计算其回溯状态，然后把在层次界限 level 范围内所有层的节点完全化，即对任意字符，计算该节点的下一跳状态。其中，根节点的 level 为 0，level 值随着树的深度变化逐层增加 1，称此算法为 HybridAC_2。图 5 是按照访问层次完全化生成的 HybridFA 自动机的示意。其中自动机的第 1、2、3 层包含的全部节点被完全化，其他层的节点仅保留可识别字符的跳转和失效转移。HybridAC_2 算法在构建自动机过程中，除了选择完全化的节点不同，其他过程与 HybridFA_1 算法相同，而数据扫描过程与 HybridFA_1 算法一致。

2.3.4　综合考虑数据流量特征的完全化算法

根据 2.1 节对数据流量的统计分析，得到数据流量的 2 种访问特征：大部分的数据只访问小部分的节点；数据倾向于访问靠近根节点的低层节点。据此，上文分别按访问频率完全化和按层次完全化 2 种方法构建符合数据流量访问特征的自动机。按频率完全化的方法，完全化的最小单位为节点，对数据流量的统计更精确，所以需要完全化的节点相对较少，自动机所需要存储空间越少；但是当数据流量有较大变化时，当前访问频率高的节点可能变成未完全化的节点，从而会导致匹配速度下降。按层次完全化的方法，完全化靠近根节点的所有低层节点，当数据流量发生变化时，尽管访问频率高的节点可能发生变化，但大部分高访问频率节点仍偏向集中在低层节点，所以对匹配速度的影响相对较小；但是当部分访问频率高的节点出现在较高层次时，由于完全化的最小单位为一层，若想进一步提高匹配速度，需增多完全化的层数。因此，在提高相同匹配速度时，基于访问层次完全化算法增加的存储开销将超过基于访问频率完全化算法。

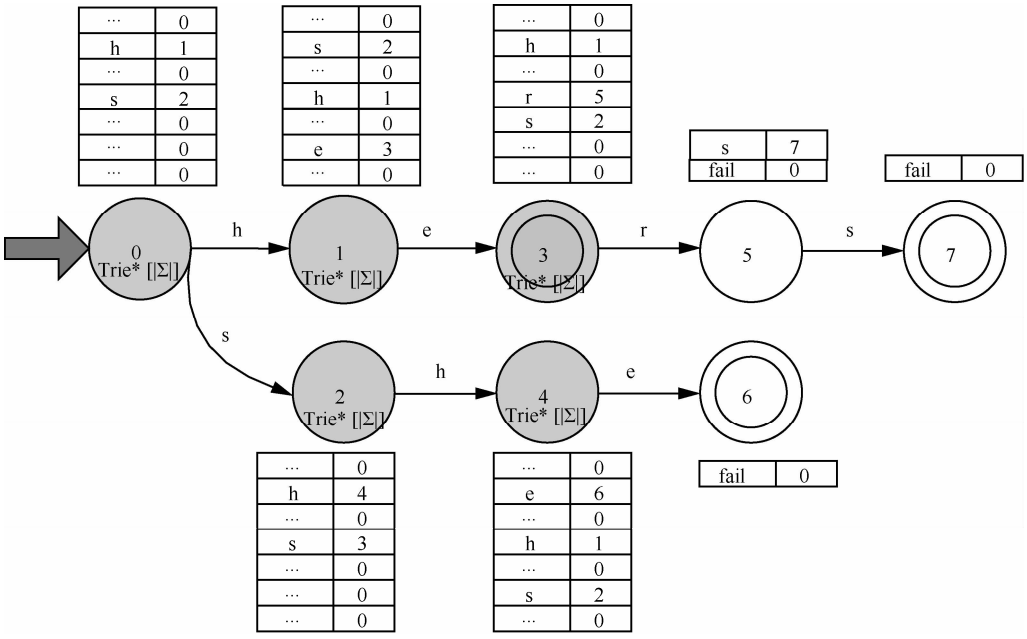


图 5　按照层次完全化构建 HybridFA 自动机示意

综合考虑数据流量的特征，结合按访问频率完全化和按层次完全化 2 种方法的特点，本节提出进一步的改进方案。首先将靠近根节点的低层节点完全化，再把层次相对较高且访问频率较高的节点也完全化，如此便不会陷入因某条模式串的命中率较高而使完全化的层次增多或者因数据流量发生变化而使匹配速度大幅度下降的问题，称这种构建自动机的算法为 HybridAC_3。构建 HybridAC_3 的具体步骤如下。

Step1 通过扫描训练数据并统计自动机中每个节点的访问频率，确定需要完全化的访问层次界限 level 和访问频率界限 fre，并把在 level 范围内所有层次的节点标记为需要完全化的节点，另外标记在 fre 范围内而不在 level 范围内的节点为需要完全化的节点。

Step2 确定完全化节点后，自动机构建过程和数据扫描过程与 HybridFA_1 算法相同。

2.4 空间复杂度分析

自动机存储空间的大小与状态转移边的数目成正比。由于按频率完全化和按照层次完全化的方法根据数据流量的特点，仅需要完全化一部分状态节点，所以与高级 AC 算法相比，二者的状态转移边更少，存储空间更小。具体存储空间的大小与完全化的节点数目相关。下面分析完全化的节点比率与所需存储空间的关系。

已知有限字符集合记作 Σ（本文数据集中字符集大小 $|\Sigma|$ 均为 256），构建的自动机中所有状态集合记作 M，被完全化的节点（状态）占所有节点（状态）的比率记作 k $(0<k<1)$。当完全化比率为 k 时，所构建自动机存储空间的上限占高级 AC 自动机存储空间的比率为

$$R=\frac{k|\Sigma||M|+(1-k)2|M|}{|\Sigma||M|} \tag{1}$$

其中，$k|\Sigma||M|$ 表示被完全化节点所产生的状态转移边数目；$(1-k)2|M|$ 表示未被完全化节点所产生的状态转移边数的上限，系数 2 指每个节点存在一条有效状态转移边和一条失效转移边（在实际 AC 自动机中，一般只有部分节点存在失效转移边，此处计算的为状态转移边上限）。若压缩比率期望达到 10%或更小，由式(1)计算可得，完全化的节点比率 k 需不大于 9.29%。在实际应用中，首先需要根据 2.1 节的方法统计数据流中不同访问频率和不同访问层次内节点所占的比率，再根据实际需要的压缩比率和式(1)中的计算方法，确定访问层次界限和访问频率界限。

3 实验结果与分析

本节将通过一系列实验对本文提出的 HybridFA 算法的性能进行系统验证，3 种 HybridFA 算法的存储空间（MB）和匹配速度（Mbit/s）将与高级 AC 算法在 3 个真实数据集上进行比较。

1) ClamAV 规则+MIT 数据

Clam Antivirus[12]是一个开源的防病毒检测软件，病毒库特征不断更新，抽取了 ClamWin Free AntiVirus 0.90.1 版本中的部分精确模式串作为本实验的规则。待扫描的文本数据来自 MIT 公开的一组真实的网络数据[13]，它的原始目的是用来对网络入侵检测系统进行评估，使用 mit_1999_training_week1_friday_inside.dat（约 62.7 MB）作为待扫描文本，选取部分（约 32.4 MB）作为训练数据，剩余部分作为测试数据。

2) Snort 规则+MIT 数据

Snort [14]是一套开源代码的网络入侵预防软件与网络入侵检测软件，规则库定期发布。抽取了 2009.06 版本中的部分精确模式串作为本实验的规则。待扫描的文本数据与 ClamAV 相同。

3) URL 规则+URL 数据

从骨干路由器上采集了约 100 GB 的 URL 数据，对采集数据去重后，得到 2.60 GB、约 2 000 万条 URL。从中随机抽取了 50 000 条 URL（URL 长度大于 4）作为规则，抽取了约 2.3 GB 作为待扫描文本，其中 1.09 GB 作为训练数据，剩余部分作为测试数据。

3.1 实验结果与分析

3.1.1 数据流量访问特征

按照 2.1 节中数据流量的统计方法，分别统计 Snort 规则、ClamAV 规则和 URL 规则上高级 AC 自动机不同访问频率下包含的节点数目所占的百分率（如图 6 所示）和各层节点下的累计访问频率（如图 7 所示）。由图 6 可见，在 98%的访问频率界限下，数据流仅访问了基于 Snort 规则的高级 AC 自动机不足 2.5%的节点，ClamAV 规则对应节点的累计访问比率不足 1%，而 URL 规则上的相应比例仅为 0.51%。其次，数据流量倾向于访问靠近根节点的低层节点，Snort 规则前 3 层的节点累计访问频率为 88.9%，ClamAV 规则为 89.3%，URL 规则为 86.5%。

由此，在真实数据集上，按照访问频率完全化和按照层次完全化的方法符合数据流量的访问特征。

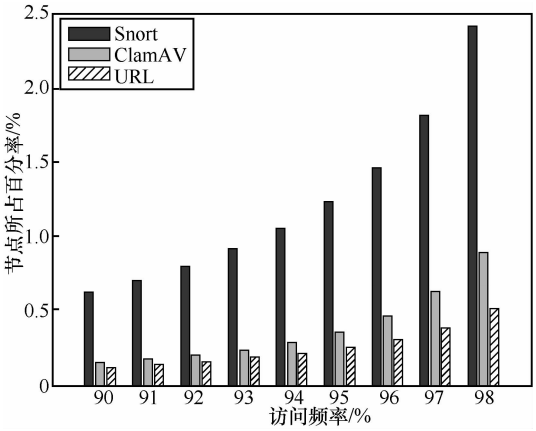


图 6　高级 AC 自动机以频率为界限的数据流量统计结果

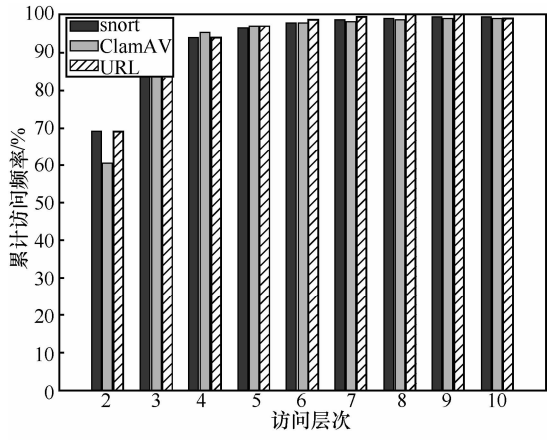


图 7　高级 AC 自动机以层次为界限的数据流量统计结果

3.1.2　存储空间和匹配速度测试

1) 按访问频率完全化和层次完全化 2 种算法

2 种完全化算法在 3 个数据集上的性能将分别与高级 AC 算法进行对比，高级 AC 算法在相应数据集上的存储空间与匹配速度如表 3 所示。

本节首先测试访问频率界限在 90%~98%变化时（变化间隔为 1%），算法 HybridFA_1 的存储空间和匹配速度的变化情况，实验结果如图 8 中相应曲线所示。可见随着访问频率界限的增高，在 3 种真实数据集上 HybridFA_1 算法的存储空间均逐渐增大，这是由于访问频率界限越高，需要完全化的节点越多，从而使存储空间增大；在匹配速度方面，随着访问频率界限的提高，在 3 种真实数据集上 HybridFA_1 算法的匹配速度也逐渐提高，这是由于完全化的节点数目越多，在匹配过程中状态需要回溯的次数越少，因而匹配速度越快。当访问频率界限为 98%时，HybridFA_1 算法在 Snort、ClamAV 和 URL 上的存储空间分别为 2.22 MB、10.25 MB 和 8.60 MB，仅占高级 AC 自动机存储空间的 3.96%、1.86%和 4.58%，而匹配速度分别是高级 AC 自动机的 82.78%、106.06%和 79.98%。可见，该算法在保证匹配速度的同时，大大降低了存储空间。

访问层次界限在 2~10 层变化时（变化间隔为 1 层），算法 HybridFA_2 的存储空间和匹配速度随层次界限的变化结果如图 8 中相应曲线所示。

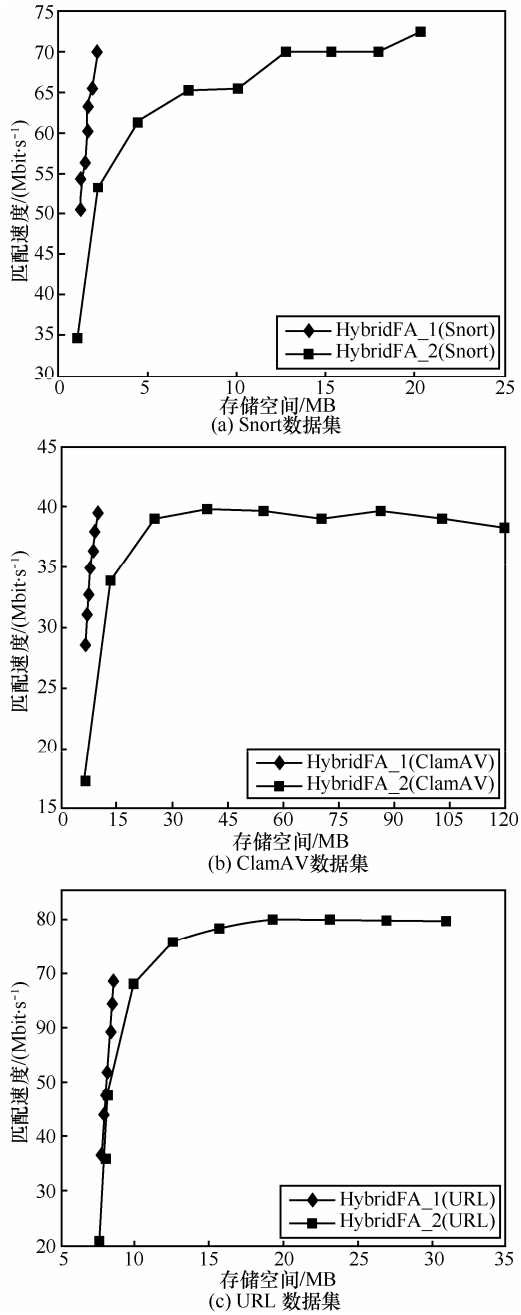


图 8　HybridFA_1 算法，HybridFA_2 算法在 Snort、ClamAV、URL 3 个真实数据集上匹配速度与存储空间的关系

在 3 种数据集上，当访问层次界限提高时，HybridFA_2 算法的存储空间均逐渐增大，相应的匹配速度也逐渐增快，与 HybridFA_1 具有相似的实验结果。但随着完全化的层次界限的提高，HybridFA_2 算法的匹配速度增长速度逐渐减慢，而存储空间却增长迅速，尤其当访问层次界限设为 3 层以上时，这种变化趋势更加明显。当访问层次界限设为 3 层时，HybridFA_2 算法在 Snort、ClamAV 和 URL 上的存储空间分别为 2.22 MB、13.29 MB 和 8.22 MB，仅分别占高级 AC 自动机存储空间的 3.95%、2.42%和 4.38%，而匹配速度分别是高级 AC 自动机匹配速度的 62.13%、90%和 56.22%。另外，与 HybridFA_1 算法相比，在存储空间相近时，HybridFA_1 算法均具有更快的匹配速度。

综合上述分析，从存储空间和匹配速度的测试结果表明，当访问频率界限为 98%时，HybridFA_1 算法在 3 种数据集上存储空间的平均值仅为 AC 自动机存储空间的 3.47%，且最高比率为 4.58%，而匹配速度的平均值为 AC 自动机匹配速度的 89.61%，且最低比率为 79.98%；当访问层次界限为 3 时，HybridFA_2 算法的在 3 种数据集上的存储空间的平均值仅为 AC 自动机存储空间的 3.58%，且最高比率为 4.38%，而匹配速度的平均值为 AC 自动机匹配速度的 69.45%，且最低比率为 56.22%。可见，2 种算法在匹配速度不低于高级 AC 的 50%时，存储空间已降低到高级 AC 的 5%以下。

2) 综合考虑数据流量特征的完全化算法

由图 8 可知，当完全化的访问频率界限设为 95%或更大时，HybridFA_1 算法存储空间的增长速度逐渐增快，本文将 HybridFA_3 算法的频率界限设为 98%；另外，随着完全化层次界限的增大，HybridFA_2 算法在匹配速度增长的同时，存储空间也增长迅速，而当层次界限设为 3 层时，存储空间小于高级 AC 的 5%，且匹配速度均超过高级 AC 算法的 50%，所以将 HybridFA_3 算法的层次界限设为 3。表 3 显示了 HybridFA_1 算法、HybridFA_2 算法、HybridFA_3 算法、高级 AC 算法在 3 个真实数据集上存储空间与匹配速度的对比结果。

由表 3 分析可见，HybridFA_3 算法的存储空间略大于 HybridFA_1 算法和 HybridFA_2 算法，但匹配速度都快于二者。同时，HybridFA_3 算法在 Snort、ClamAV 和 URL 这 3 个数据集上的存储空间分别为 2.75 MB、14.55 MB 和 8.70 MB，仅占高级 AC 自动机存储空间的 4.90%、2.65%和 4.63%，而匹配速度分别是高级 AC 自动机的 85.80%、120.05%和 83.81%。可见，存储空间均小于高级 AC 算法的 5%；匹配速度在 Snort 规则和 URL 规则上与高级 AC 算法相当，在 ClamAV 规则上快于高级 AC 算法。

实验结果表明，按访问频率完全化和按访问层次完全化 2 种方法在构建 HybridFA 时都具有较好的压缩效果。对比而言，按频率完全化的方法，完全化的最小单位为节点，对数据流量的统计更精确，所以需要完全化的节点相对较少，自动机所需要存储空间也更少，但是对数据流量变化适应性不强。按层次完全化的方法，完全化靠近根节点的所有低层节点，当数据流量发生变化时，访问频率高的节点可能发生变化，但大部分高访问频率的节点仍集中在低层节点中，对数据流量的变化适应性相对较强。在实际应用中，随着完全化层次的增加，存储空间也相应迅速增长，因此需要根据具体需求权衡存储空间与匹配速度的关系。最后提出的结合数据流访问频率和访问层次特征的混合自动机构建算法在存储空间、匹配速度和数据适应性等综合性能上获得了最好效果。

4 结束语

本文根据数据流对自动机节点的访问特征，提出基于访问频率完全化和基于访问层次完全化的 2 种构建混合自动机的算法，并综合分析以上 2 种算法的优势，进一步提出改进的自动机完全化算法。

表 3 HybridFA_1、HybridFA_2、HybridFA_3 和高级 AC 算法在 3 组真实数据集上存储空间和匹配速度对比

算法	频率界限	层次界限	Snort 数据集		ClamAV 数据集		URL 数据集	
			存储空间/MB	匹配速度/(Mbit·s^{-1})	存储空间/MB	匹配速度/(Mbit·s^{-1})	存储空间/MB	匹配速度/(Mbit·s^{-1})
高级 AC	—	—	56.16	84.39	549.89	37.49	187.74	85.35
HybridFA_1	98%	—	2.22	69.86	10.25	39.77	8.60	68.26
HybridFA_2	—	3	2.22	53.27	13.29	33.74	8.22	47.98
HybridFA_3	98%	3	2.75	72.41	14.55	45.01	8.70	71.53

真实数据集上的实验结果表明，与高级 AC 算法相比，3 种算法在保证匹配速度的同时，存储空间降低到高级 AC 算法的 5%左右。同时，结合数据流访问频率和访问层次特征的自动机完全化算法在匹配速度和数据适应性方面都获得了最好效果。未来工作将考虑对访问频率和访问层次这 2 个重要参数的选择进行优化。

参考文献：

[1] ME L, HEYE L, KURI J, *et al*. A pattern matching based filter for audit reduction and fast detection of potential intrusions[A]. The 3rd International Workshop on the Recent Advances in Intrusion Detection[C]. Toulouse, France, 2000.17-27.

[2] NAVARRO G, KURI J. Fast Multipattern Search Algorithms for Intrusion Detection[R]. Technical Report TR/DCC-99-11, Dept. of Computer Science, Univ of Chile.1999.

[3] VARGHESE G, FIST M. Applying Fast String Matching to Intrusion Detection[R]. Technical Report In preparation, successor to UCSD TR CS2001-0670. University of California, San Diego. 2002.

[4] AHO A V, CORASICK M J. Efficient string matching: an aid to bibliographic search[J]. Communications of the ACM, 1975, 18(6): 333-340.

[5] NAVARRO G, RAFFINOT M. Flexible Pattern Matching in Strings: Practical On-line Search Algorithms for Texts and Biological Sequences[M]. SIAM Publishing, 2002.

[6] DENCKER P, DORRE K, HEUFT J. Optimization of parser tables for portable compilers[J]. ACM Transactions on Programming Languages and Systems, 1984, 6(4):546-572.

[7] AHO A V, SETHI R, ULLMAN J D. Compilers: Principles, Techniques, and Tools[M]. Addison-Wesley Publishing, 2006.

[8] ZIEGLER S. Smaller faster table driven parser[EB/OL]. http://www.researchgate.net/publication/242522203_Smaller_faster_table_driven_parser. Unpublished manuscript. Madison Academic Computing Center, 1977.

[9] AOE J, MORIMOTO0 K, SATO T. An efficient implementation of trie structures[J]. Software—Practice and Experience, 1992, 22(9):695-721.

[10] 刘燕兵, 刘萍, 谭建龙等. 基于存储优化的多模式串匹配算法[J]. 计算机研究与发展, 2009, 46(10): 1768-1776.
LIU Y B, LIU P, TAN J L, *et al*. A multiple string matching algorithm based on memory optimization[J].Journal of Computer Research and Development,2009,46(10):1768-1776.

[11] NAVARRO G, RAFFINOT M. Fast and flexible string matching by combining bit-parallelism and suffix automata[J]. Experimental Algorithmics, 2000, 5(4): DOI:10.1145/351827.384246.

[12] Clam Antivirus, an anti virus detection software[EB/OL]. http://www.clamav.net/binary.html#pagestar.

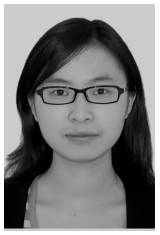
[13] Free real network data for string matching test released by MIT[EB/OL]. http://www.ll.mit.edu/IST/ideval/

[14] Snort, a free and open source network intrusion detection and prevention system[EB/OL]. http://www.snort.org/snort-rules/?#rules

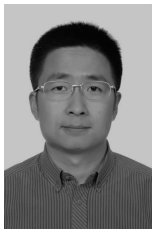
作者简介：

熊刚（1977-），男，湖北汉川人，中国科学院信息工程研究所高级工程师，主要研究方向为网络测量与行为分析、信息对抗、信息安全。

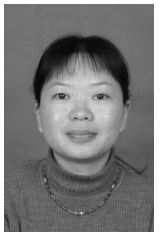
何慧敏（1987-），女，山东菏泽人，中国移动（深圳）有限公司工程师，主要研究方向为串匹配算法。



于静（1989-），女，河北承德人，中国科学院信息工程研究所实习研究员，主要研究方向为模式匹配算法、机器学习、图像处理。



刘燕兵（1981-），男，湖北麻城人，中国科学院信息工程研究所副研究员，主要研究方向为模式匹配算法、图数据计算、信息内容安全。



郭莉（1969-），女，湖南湘潭人，中国科学院信息工程研究所教授级高级工程师，主要研究方向为信息安全、数据流处理。