第 35 卷 第 21 期 计 算 机 工 程 2009 年 11 月
Vol.35 No.21 Computer Engineering November 2009
·软件技术与数据库· 文章编号：1000—3428(2009)21—0039—03 文献标识码：A 中图分类号：TP311.12

串匹配算法中的自动机紧缩存储技术

杨毅夫[1,2]，刘燕兵[1,2]，刘 萍[1]，郭 莉[1]
(1. 中国科学院计算技术研究所，北京 100190；2. 中国科学院研究生院，北京 100039)

摘 要：自动机是串匹配算法中常用的数据结构，对自动机实现紧缩存储可以节省算法空间。总结常用自动机紧缩存储方法，分析其原理、时间效率、空间效率和优缺点，给出各种方法与数据稀疏性之间的关系。运用紧缩存储方法实现基本 AC 算法，对随机数据和真实数据的实验结果证明该算法有效。

关键词：紧缩存储；自动机；串匹配

Automaton Compact Representation Technology in String Matching Algorithm

YANG Yi-fu[1,2], LIU Yan-bing[1,2], LIU Ping[1], GUO Li[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】 Automaton is one kind of data structure often being used in string matching algorithms. By realizing compact representation of automaton, the algorithm space can be decreased. This paper summarizes several frequently used compact representations of automaton, analyzes their principles, time efficiencies, space efficiencies, merits and demerits, and gives relationships between above methods and sparsity character. It implements the basic AC algorithm with compact representation method. Experimental results of random and real data demonstrate the efficiency of this algorithm.

【Key words】 compact representation; automaton; string matching

1 概述

字符串匹配被应用于入侵检测、病毒检测、文本检索、搜索引擎和生物计算等诸多领域。由于基于自动机的字符串匹配算法稳定性较高，因此得到广泛采用，尤其在网络数据处理中。例如，病毒检测系统 ClamAV 采用 AC 算法[1]，该算法是基于自动机的典型算法；内容过滤系统 DansGuardian 采用 Hospool DFA 算法，它是一种效果较好的自动机算法；SNORT 系统针对其自身应用特点，采用 5 种改进的 AC 算法。随着应用需求的不断增长，系统中的规则数量急剧增加，例如，Snort 的规则数量从 2004 年的 2 500 条增长到 2008 年的 15 000 条，ClamAV 系统的规则数量从 0.83 版本的 25 000 条增长到了 0.93 版本的 80 000 条。此时，基于自动机的串匹配算法由于其存储空间的急速膨胀，使得算法局部性变差，降低了匹配算法的性能，导致整个系统的吞吐率受到严重影响。因此，高效存储自动机技术日益受到研究者的关注。

2 紧缩存储方法

本节阐述全矩阵方法、行压缩方法、Banded-Row 方法、位图方法、行偏移方法和哈希方法的核心思想与紧缩存储过程，并分析每种方法的时间和空间复杂度。以模式串集合 $P=\{abe, cab\}$及其对应的基本 AC 自动机(图 1)为例，假定每个状态用 k Byte 存储。

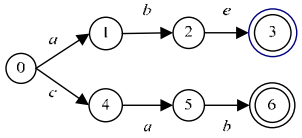

图 1 $P=\{abe, cab\}$的基本 AC 自动机

2.1 全矩阵方法

全矩阵方法[2]的核心思想如下：用矩阵 $\boldsymbol{T}_{|S||\sum|}$存储自动机，其中，$|S|$是自动机的状态数；$|\sum|$是字母表大小。矩阵元素 $T[q, s]$存储在状态 q 下，输入字符 s 时转移到的下一个状态。

定理 1 全矩阵方法存储自动机需要 $k|S||\sum|$ Byte 空间。

全矩阵方法通常用于模式串规模和字母表规模较小的情况。其状态转移只需要一次访存：当前状态为 q，输入字符为 s，则 $nextstate(q, s)=T[q, s]$。

2.2 行压缩方法

行压缩[2]的核心思想如下：把矩阵的每一行表示成$(r, s_1,s_2,\cdots,s_r, q_1,q_2,\cdots,q_r)$，其中，$r$ 为非空元素的个数；$s_1,s_2,\cdots,s_r$ 为非空元素对应的字符；$q_1,q_2,\cdots,q_r$ 为非空元素的值(即下一个状态)。

定理 2 自动机边数为 e，状态数为$|S|$，则行压缩方法需 $|S|+(k+1)e$ Byte 存储空间。

当自动机边数很少时，行压缩方法能有效紧缩存储空间，行压缩方法的状态转移过程如下：

```
nextstate(q, s)
  If T[q][0] > 0 Then
      For i∈1…T[q][0] Do
          If T[q][i] = s Then
              Return T[q][T[q][0]+i]
```

基金项目：国家“973”计划基金资助项目(2007CB311100)
作者简介：杨毅夫(1982－)，男，硕士，主研方向：网络信息安全；刘燕兵、刘 萍，助理研究员；郭 莉，高级工程师
收稿日期：2009-05-10 **E-mail：**yangyifu@software.ict.ac.cn

End of for

End of if

2.3 Banded-Row 方法

Banded-Row[3]的核心思想：对存储矩阵的每一行，只存储第 1 个非空元素到最后一个非空元素之间的元素。若用数组 V 表示，则 $V[0]$中存储第 1 个非空元素的位置，$V[1]$存储带宽 *bandwidth*(即第 1 个非空元素和最后一个非空元素之间的元素个数)，$V[2,3,\cdots,1+bandwidth]$存储第 1 个非空元素到最后一个非空元素之间的所有元素。

定理 3 自动机状态数为$|S|$，则 Banded-Row 方法需 $2|S|+k\sum_{i=1}^{|S|}bandwidth_i$ Byte 存储空间。

当非空元素集中在某一范围时，该方法有很好的压缩效果，并能在 $O(1)$的时间内转移到下一个状态，具体如下：

```
nextstate(q, s)
  offset = s-q->V[0]
  If 0≤offset< q->V[1] Then
        Return q->V[2+offset]
  End of if
```

2.4 位图表示方法

位图方法[4]是用硬件高效实现 AC 算法的方法。其核心思想如下：先用大小为$|\sum|$的比特向量 ***bitmap*** 表示一行的存储情况，如果第 i 列元素非空，则 ***bitmap***$[i]=1$，否则为 0。然后用向量 ***V*** 存储该行所有非空元素。

定理 4 设自动机边数为 e，状态数为$|S|$，字母表大小为$|\sum|$，则位图表示法需$|S||\sum|/8+ke$ Byte 存储空间。

该方法有很好的压缩效果，但其状态转移过程需要硬件支持，其状态转移过程如下：

```
nextstate(q, s)
  If q->bitmap[s]!=0 Then
        temp=(q->bitmap) & 1^{j-1}0^{|Σ|-j+1}
        k=PopCount(temp)
        Return q->V[k]
```

2.5 行偏移方法

行偏移方法[5]的核心思想如下：用 3 个数组表示重叠，*base*[*s*]表示 s 行在数组 *next* 中的偏移量；数组 *next* 重叠排列所有行，使各行的非空元素互不冲突；数组 *check* 与数组 *next* 等长，*check*[*t*]=*s* 表示 *next*[*t*]存放的数据属于 s 行。

定理 5 设自动机状态数为$|S|$，边数为 e，*check* 和 *next* 碎片率为 r，则行偏移方法占用 $k|S|+2ke(1+r)$ Byte 空间。

行偏移方法支持随机存取，其状态转移过程如下：

```
nextstate(q, s)
  t = base[q] + s
  If check[t] = q Then
        Return next[t]
```

文献[6]针对 trie 结构的特点对上述方法进行简化，去除 *next* 数组，提出双数组结构。双数组结构由 2 个等长的整数数组组成，即 *base* 和 *check*。每次状态转移需要 2 次访存，其状态转移过程如下：

```
nextstate(q, s)
  t = base[q] + s
  If check[t] = q Then
        Return t
  Return fail
```

2.6 哈希方法

哈希方法[7]的核心思想如下：用尽可能小的存储空间组织存储矩阵，在尽可能短的时间内(用探测次数度量)找到元素 q 的位置。

文献[7]给出基于完美散列的 $O(n)$空间复杂度和 $O(1)$最坏时间复杂度的解决办法。主要使用 $f(x)=(kx \bmod p)\bmod q$ 函数类对 T 进行随机化划分和散列。它先用函数 $f(x)$将 T 划分为 n 组，然后对各分组 T_j 选择适当的完美散列函数 $f_j(x)$，使 S 的大小不超过 $6n$。

文献[8]提出基于随机化和行偏移的方法。它先使用 $f(x)=(kx \bmod p)\bmod n^2$ 函数类对 T 进行随机化划分和散列，然后寻找一组行偏移 $rd[0,1,\cdots,n-1]$，使函数 $h(x)=(rd[i]+j) \bmod n$ 是 T 上的单射。

定理 6 设自动机边数为 e，则基于随机化和行偏移的哈希方法的存储空间复杂度为 $O(e)$。

由于该方法构建的哈希函数是最小完备哈希函数，因此有很好的紧缩效果。状态转移过程即计算哈希值的过程，当前状态为 q，输入字符为 s，先通过 $f=(ks \bmod p)\bmod n^2$ 计算随机化后的值，然后通过计算$(rd[f/n]+f \bmod n) \bmod n$ 得到下一个状态。

3 实验

运用上述 6 种紧缩存储方法实现基本 AC 算法，并通过实验说明紧缩存储方法在串匹配中的有效性。通过构造不同稀疏性的矩阵，给出 6 种紧缩存储方法的空间紧缩效果与稀疏性的关系。

3.1 串匹配算法的存储空间

随机数据和真实数据的存储空间如图 2 和表 1 所示。

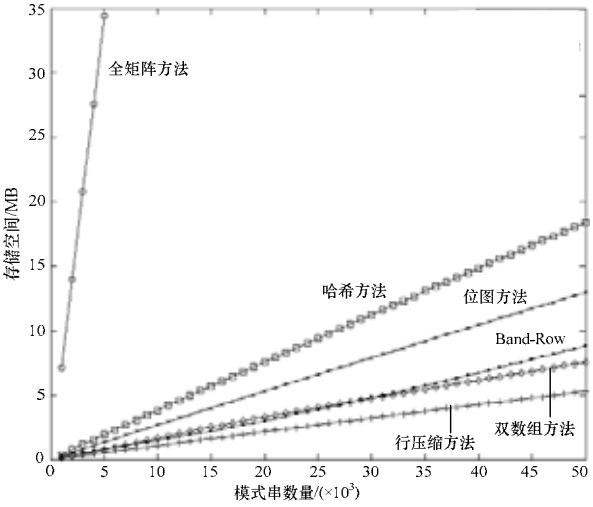

图 2 采用不同方法时随机数据的存储空间

表 1 采用不同方法时真实数据的存储空间 MB

方法	中文数据	英文数据	Snort	ClamAV
全矩阵	170.720	10.684	57.974	302.339
行压缩	2.981	0.601	0.954	4.736
Banded-Row	5.262	0.672	1.141	5.550
位图	6.960	0.841	2.307	11.795
双数组	4.867	0.339	1.309	6.056
哈希	9.001	0.986	3.346	17.524

实验结果表明，全矩阵方法占用的存储空间远大于其他方法。在多数情况下，行压缩方法占用的存储空间最小，空间压缩率(与全矩阵方法比较)达到 95%以上，其次是双数组方法、Banded-Row 方法和哈希方法。

3.2 串匹配算法的匹配时间

随机数据和真实数据的匹配时间如图 3 和表 2 所示。

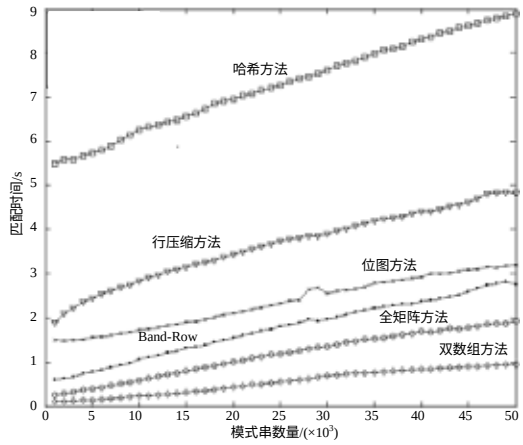

图 3　采用不同方法时随机数据的匹配时间

表 2　采用不同方法时真实数据的匹配时间　s

方法	中文数据	英文数据	Snort	ClamAV
全矩阵	3.867 2	2.898 4	2.665 7	3.676 9
行压缩	7.748 5	3.203 3	11.959 3	15.040 4
Banded-Row	5.656 4	2.728 1	4.826 1	6.179 8
位图	19.869 0	4.213 9	27.934 7	24.371 8
双数组	2.674 9	2.587 5	1.653 3	1.234 5
哈希	18.898 4	5.534 6	34.790 9	37.637 2

实验结果表明，双数组方法具有最优的匹配时间，全矩阵方法和 Banded-Row 方法次之，行压缩方法较慢，位图方法和哈希方法最慢。双数组方法优于全矩阵方法的一个可能原因是全矩阵方法占用的存储空间远大于双数组方法，因此，其 cache 失效率远大于双数组方法。

3.3　紧缩存储方法与数据稀疏率的关系

以实验方式描述上述方法与数据稀疏率之间的关系。为了构造具有不同稀疏率的数据，实验采用行数为 1 000、列数为 256 的矩阵。实验结果如图 4 所示。

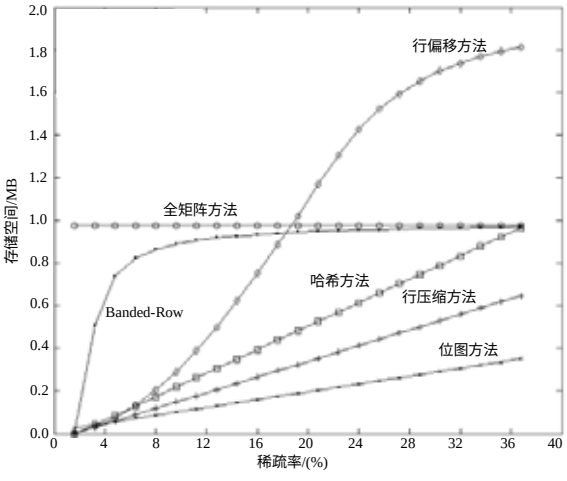

图 4　不同紧缩存储方法与数据稀疏率的关系

实验结果表明，全矩阵方法不受数据稀疏率的影响。位图方法、行压缩方法和哈希方法的存储空间随稀疏率的增大而线性增加。在稀疏率小于 5%时，Banded-Row 方法的存储空间上升较快，其后随稀疏率的增加逐渐接近全矩阵方法的存储空间。行偏移方法随稀疏率的变化很大，在稀疏率为 15%左右时，其存储空间超过全矩阵方法。

4　结束语

本文总结全矩阵方法、行压缩方法、Banded-Row 方法、位图方法、行偏移方法和哈希方法，并以基本 AC 自动机为例，分析比较上述方法的性能。本文工作对所有可以应用紧缩存储技术的领域具有参考价值。

在不同应用领域中，存在各种降低自动机空间复杂度的方法。文献[9]阐述了面向串匹配领域的几种通用方法，文献[10]通过增加 default 边减少自动机的边数量。上述方法通常被称为自动机压缩技术，而本文提到的各种紧缩存储技术能在压缩之后的自动机上使用。对匹配自动机使用压缩技术或紧缩存储技术的最终目标是提高串匹配算法的匹配速度。

参考文献

[1] Aho A V, Corasick M J. Efficient String Matching an Aid to Bibliographic Search[J]. Communications of the ACM, 1975, 18(6): 333-340.

[2] Dencker P, Dorre K. Optimization of Parser Tables for Portable Compilers[J]. ACM Transactions on Programming Languages and Systems, 1984, 6(4): 546-572.

[3] Norton M. Optimizing Pattern Matching for Intrusion Detection[Z]. [2008-12-05]. http://www.idsresearch.org.

[4] Tuck N, Sherwood T, Calder B, et al. Deterministic Memory-efficient String Matching Algorithms for Intrusion Detection[C]// Proc. of IEEE INFOCOM'04. Hong Kong, China: [s. n.], 2004: 2628-2639.

[5] Aho V, Sethi R, Ullman J D. Compilers: Principles, Techniques, and Tools[M]. [S. l.]: Addison-Wesley, 1985.

[6] Aoe J, Morimoto K, Sato T. An Efficient Implementation of Trie Structures[J]. Software-practice & Experience, 1992, 22(9): 695-721.

[7] Fredman M, Koml´os J, Szemer´edi E. Storing a Sparse Table with *O*(1) Worst Case Access Time[J]. Journal of the ACM, 1984, 31(3): 538-544.

[8] Galli N, Seybold B, Simon K. Tetris-hashing or Optimal Table Compression[J]. Discrete Applied Mathematics, 2001, 110(1): 41- 58.

[9] Nieminen J, Kilpel P. Efficient Implementation of Aho-Corasick Pattern Matching Automata Using Unicode[J]. Software Practice and Experience, 2007, 37(6): 669-690.

[10] Kumar S, Dharmapurikar S, Yu Fang, et al. Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection[C]//Proc. of SIGCOMM'06. New York, USA: [s. n.], 2006: 339-350.

编辑　陈　晖

（上接第 38 页）

参考文献

[1] 吴　松, 金　海. 存储虚拟化研究[J]. 小型微型计算机系统, 2003, 24(4): 728-732.

[2] 张　磊. 虚拟磁带库在灾备系统中的应用研究[J]. 小型微型计算机系统, 2007, 28(6): 1149-1152.

[3] Russinovich M E, Solomon D A. 深入解析 Windows 操作系统[M]. 潘爱民, 译. 北京: 电子工业出版社, 2007-06.

[4] 刘朝斌. 虚拟网络存储系统关键技术研究及其性能评价[D]. 武汉: 华中科技大学, 2003.

[5] 吴　涛. 虚拟化存储技术研究[D]. 武汉: 华中科技大学, 2004.

编辑　顾姣健