一种节约内存的中文多模式匹配算法*

侯整风,杨 波,朱晓玲
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:AC及其改进算法基于有限状态自动机,随着中文模式串数目增加,完全Hash表和状态表矩阵存储方式会导致存储空间快速膨胀,状态转移函数计算量大,Cache命中率下降,算法的时空性能急剧下降。提出以邻接链表方式存储有限状态自动机,并将状态"0"的链表转化为线性表,以提高算法的时空效率。在此基础上,设计了一种适合中文的多模式匹配算法,该算法所需存储空间仅为完全Hash表方式的10%,约为状态表矩阵方式的20%。

关键词:多模式匹配;AC算法;邻接链表;有限状态自动机

中图分类号:TP393.08 **文献标识码**:A **文章编号**:1674-7720(2013)13-0053-05

A memory-efficient multiple pattern algorithm for chinese

Hou Zhengfeng, Yang Bo, Zhu Xiaoling
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: AC and its improved algorithms are based on finite state automata. With the increasing in the number of Chinese mode string, the complete hash table and the state table matrix storage methods require a lot of storage space which may drop the cache hit rate and spend much time to calculate the goto function, and results in poor performance of the algorithms both in time and in space. To improve the performance of the space and time, a new method which uses adjacency linked list to store finite state automata is proposed, and the linked list of the state '0' is changed into a linear list. Based on this, a multi-pattern matching algorithm for Chinese is designed. The algorithm requires less than 10% of the storage space needed by complete Hash table, and about 20% of the storage space needed by the state table matrix.

Key words: multi-pattern matching; AC algorithm; adjacency linked list; finite state automata

模式匹配是计算机应用领域中的研究热点之一,广泛应用于搜索引擎[1]、网络入侵检测[2]等。1977年,KNUTH、MORRIS和PRATT提出了最早的单模式匹配算法——KMP算法[3]。该算法首次利用"部分匹配"的结果,将模式串向右移动若干位置后继续与文本串当前字符进行匹配。同年,BOYER R S和MOORE J S提出了一种更高效的模式匹配算法——BM算法[4],该算法运用坏字符规则(Bad Char)和好后缀规则(Good Suffix)来计算模式串右移的距离,使模式串跳跃式的移动,提高了匹配效率。随后人们对模式匹配算法进行了广泛深入的研究,提出了QS[5]及KR[6]等单模式匹配算法。1975年,AHO A V和CORASICK M J提出了一种多模式匹配算法——AC算法[7],该算法基于有限状态自动机,通过状态的转移,扫描一遍文本可匹配多个模式串。COMMENTZ-WALTER B对AC算法进行改进,提出了AC_BM算法[8]。该算法结合了AC算法与BM算法的优点,可跳跃式匹配,因而具有更高的匹配效率。

AC算法基于有限状态自动机,随着中文模式串数目的增加,有限状态自动机所需存储空间快速膨胀,导致状态转移函数计算量大,Cache命中率下降,算法的时空性能急剧下降,难以处理大数量级的模式串。王永成等[9]提出用完全Hash表存储组合状态自动机,所需存储空间快速膨胀的问题有所改善,但对大规模中文模式串匹配,存储空间仍非常大,算法效率不高。NORTON M[10]等提出了两种压缩的稀疏存储方式:带状行方式和稀疏行方式,在一定程度上降低了有限状态自动机的存储空

*基金项目:安徽省自然科学基金(090412051);广东省教育部产学研结合项目(2008B0905002400)

间。TUCK N[11]等和张元竞[12]等使用位图压缩来降低有限状态自动机的存储空间。但上述方法对中文模式串压缩率不高，没有从根本上解决存储空间快速膨胀问题。

本文对有限状态自动机存储结构进行研究，提出了一种适合中文的多模式匹配算法——AC_SC（AC Suit Chinese）。该算法采用邻接链表存储有限状态自动机，较好地解决了有限状态自动机存储空间快速膨胀问题，提高了算法的时空性能。

1 AC 算法

AC 算法将 KMP 算法与有限状态自动机理论相结合，根据模式串集合，构造有限状态自动机，并由跳转的状态决定是否匹配。AC 算法包括两个阶段：预处理阶段和匹配阶段。

1.1 预处理

（1）构造有限状态自动机

设模式串集合 $K=\{y_1, y_2, \cdots y_i \cdots y_n\}$。$K$ 中的每个模式串为 y_i，初始状态为 0。从状态 0 出发，逐个取出 y_i 中的每一个字符，如果从当前状态出发能够找到标注该字符的矢线，则将矢线所指的状态作为当前状态；否则，添加一个新状态，新状态为已有的最大状态加 1，并将新加入的状态作为当前状态，继续对 y_i 中的下一个字符进行处理。当模式串集合 K 中的所有模式串都处理完毕，再添加一条从状态 0 到状态 0 的自反线，表示保持状态 0 的字符集。如 $K=\{$he，she，his，hers$\}$，有限状态自动机如图 1 所示。

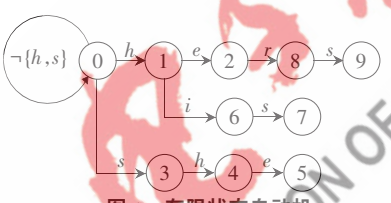

图 1 有限状态自动机

（2）goto 函数

goto 函数用来计算跳转的下一状态，计算过程为：如果从当前状态出发能够找到标注字符“a”的矢线，则 goto 函数返回矢线所指的状态，否则返回跳转失败状态 fail。图 1 所示的有限状态自动机，goto（1，e）=2，goto（1，i）=6，goto（1，b）=fail。

（3）failure 函数

failure 用来计算某个字符失配时应转移的状态。计算过程：

①若父状态为 0，则该状态的失效函数为 0。

②对于其他状态 m，若其父状态为 r，则该状态的失效函数 failure（m）=goto（failure（r），a）。

如图 1 所示，failure（1）=0，failure（6）=goto（0，i）=0，failure（4）=goto（0，h）=1。

（4）output 函数

output 用来输出已匹配的模式串，计算过程分为两步：

①在构造有限状态自动机的过程中，每处理完一个模式串，将该模式串加入到当前状态 s 的输出函数中，即 output（s）=y_i。

②在计算失效函数过程中，若 failure（s）=s'，则 output（s）=output（s）∪output（s'）。如图 1 所示，output（7）={his}，output（9）={she，he}。

1.2 匹配过程

（1）将当前状态 s 初始化为 0，文本串指针指向文本串的头。

（2）如果文本串的当前指针不为空，则取出所指的字符“a”；否则匹配过程结束。

（3）计算 s'=goto（s，a）。如果 $s' \neq$ fail，则 $s=s'$，向右移动文本串指针 1 位。如果 output（s）=NULL，转至步骤（2）；如果 output（s）≠NULL，则输出 output（s）的值 y_i，表示模式串 y_i 匹配成功，转至步骤（2）。

（4）如果 s'=fail，调用 failure 函数，即当前状态 s=failure（a），转至步骤（3）。

2 有限状态自动机的存储方式

有限状态自动机的存储方式对算法的时空效率的影响很大。目前常用的存储方式有两种：完全 Hash 表方式及状态表矩阵方式。

2.1 完全 Hash 表方式

完全 Hash 表存储方式为每一个状态 s_i 建立一个完全 Hash 表 h_i，h_i 的下标表示当前输入字符的编码，值表示跳转的状态。图 1 所示的有限状态自动机，其完全 Hash 表存储方式如图 2 所示。

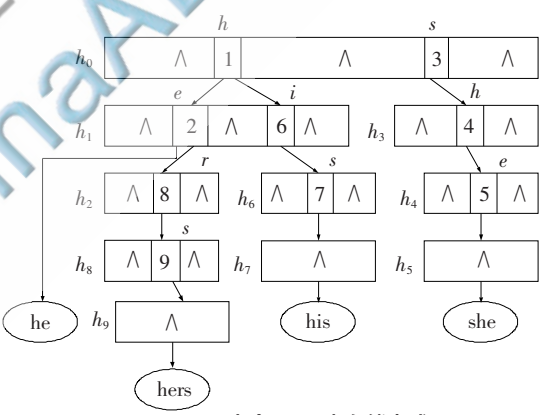

图 2 完全 Hash 表存储方式

该存储方式所需的存储空间为 statenum×maxcharset，其中 statenum 为有限状态自动机的状态数目，maxcharset 为字符集的最大字符数。对于英文模式串，maxcharset 为 256，而对于中文模式串，maxcharset 为 256×256（65 536），随着模式串数目的增大，状态数随之增大，存储空间快速膨胀，巨大的存储空间也会导致算法时间效率相应下降。

针对这一问题，参考文献[11]提出了一种新型的组合状态自动机。该自动机将状态分为两类：一类为虚状态，负责匹配中文字符的高字节部分；另一类为实状态，负责匹配中文字符的低字节部分。这种基于组合状态机的完全 Hash 表存储方式所需存储空间为 statenum×256×

2，在一定程度上缓解了存储空间快速膨胀问题。由于中文模式串汉字之间的相互独立性，有限状态自动机的状态数目与模式串数目近似呈线性增长；对于大规模中文模式匹配，组合状态机所需存储空间仍然巨大。

2.2 状态表矩阵方式

状态表矩阵方式为每一个状态 s_i 建立一个数组 A_i [0…j]，下标表示当前输入的字符，值表示匹配成功时跳转的状态。每个 A_i 生成一个二维矩阵 $M[0\cdots i][0\cdots j]$，其中 $0\cdots i$ 表示状态 s_i，$0\cdots j$ 表示模式串中不重复的字符。图 1 所示的有限状态自动机的状态表矩阵如表 1 所示。

表 1 状态表矩阵

状态	字符				
	e	*h*	*i*	*r*	*s*
0	0	1	0	0	3
1	2	0	0	0	0
2	0	0	0	8	0
3	0	4	0	0	0
4	5	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	7
7	0	0	0	0	0
8	0	0	0	0	9
9	0	0	0	0	0

该存储方式所需的存储空间为 statenum×charnum，其中 charnum 为模式串集中不重复字符的个数。对于英文模式串，charnum 最大为 256，所需的存储空间同完全 Hash 表方式；一般情况下，charnum 远小于 256（约为 94 个，52 个大小写字母和 42 个特殊字符），存储空间开销不大。对中文模式串，随着模式串的增大，由于汉字之间的相互独立性，statenum 近似呈线性增大，charnum 也相应线性增大，导致所需存储空间迅速增加。

参考文献[10]提出了两种稀疏存储方式：带状行方式和稀疏行方式，其中带状行方式存储矩阵中第一个非 0 值到最后一个非 0 值之间的全部元素，而稀疏行方式则只存储矩阵中的非 0 值元素。对中文模式串，一个汉字需拆分成两个字节存储，状态矩阵不再是稀疏矩阵，压缩率不高，且会增加 goto 函数的计算量。因此，参考文献[10]提出的存储方式不适合中文模式匹配。而参考文献[11]及参考文献[12]提出的位图存储方式，同样存在压缩率不高的问题。

3 AC_SC 算法

随着中文模式串数目的增加，完全 Hash 表和状态表矩阵存储有限状态自动机所需存储空间会快速膨胀。针对这一问题，本文提出邻接链表方式存储有限状态自动机，并将扇出系数最大的状态 0 的链表转化为线性表，以加快计算 goto 函数的速度。在此基础上，设计了一种适合中文的多模式匹配算法。

3.1 邻接链表存储方式

为有限状态自动机中的每一个状态 s_i 建立一个单链表 V_i，V_i 中的结点表示可以从状态 s_i 跳转的状态。结点由 3 个域组成：字符域、状态域、链域。其中字符域存储从状态 s_i 出发跳转到其他状态时所经过的字符，状态域存储从状态 s_i 出发经过字符域中字符跳转的状态，链域存储从状态 s_i 出发可以跳转到的另一结点的地址。此外，设置一个顶点表，记录各个单链表的表头地址，形成一个邻接链表。对图 1 所示的有限状态自动机，其邻接链表存储方式如图 3 所示。

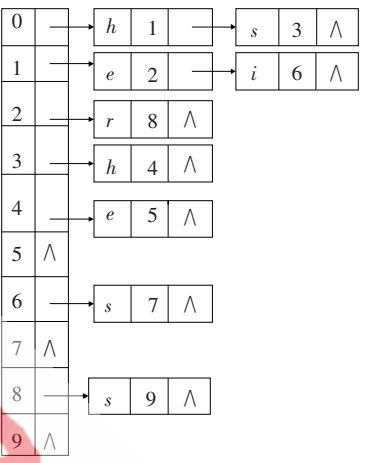

图 3 邻接链表存储方式

对于当前状态 s_i，计算 goto 函数需搜索对应的单链表 V_i。如果模式串字符之间的独立性不高（如英文，仅由 26 个字母组成），各状态的扇出系数平均分布，即各状态对应的单链表长度相当，搜索效率较高。若模式串字符之间的独立性较高（如中文，常用的汉字有 3 500 个），则某些状态（如状态 0）的扇出系数较大，而其他状态的扇出系数较小，搜索扇出系数高的单链表将花费大量的时间。设模式串集 $K=\{a_0,a_1,a_2,\cdots,a_k\}$。极端情况下，$a_0\neq a_1\cdots\neq a_k$，则状态 0 的扇出系数为 k。当 k 较大时，计算 goto 函数的时间开销较大。

因此，对中文模式集，本文将状态 0 的单链表 V_0 转化为线性表 L_0，即保留字符域和状态域，去掉链域，释放单链表 V_0 的存储空间，并对线性表 L_0 的字符域进行排序。计算 goto 函数时，如果当前状态为 0，则对线性表 L_0 二分查找；否则搜索对应状态单链表。该方法在不增加额外存储空间的前提下，减少了 goto 函数的计算时间，提高了算法的效率。

例如，模式串集={升职，时尚白领，中国，外企，生存}，有限状态自动机的存储结构如图 4 所示。

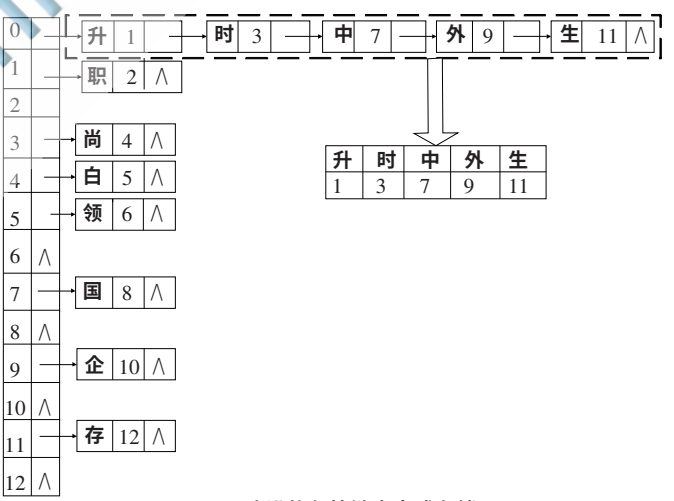

图 4 改进的邻接链表方式存储

3.2 AC_SC 算法描述

（1）预处理阶段

①建立有限状态自动机。建立顶点表 statetable，用于记录单链表表头地址。对于模式串集 K 中的每一个模式串 k_i，初始状态为状态 0。逐个取出模式串中的每一

个字符，搜索 statetable[0]后面的单链表，若存在文本域为该字符的结点，则将结点的状态域的值 i 作为当前状态，并跳转到 statetable[i]；否则，添加一个新结点，新结点的状态域为已有的最大状态加 1，文本域为当前读入的模式串的字符，继续对 y_i 中的下一个字符进行处理。

②对状态 0 的单链表处理。定义一个线性表，将单链表中的字符域和状态域存储在该线性表中，对其字符域排序，释放状态 0 单链表的存储空间。

③计算 failure 函数。父状态为 0 的状态的 failure 函数为 0。对于其他状态 m，若其父状态为 r，则 failure(m)=goto(failure(r),a)。计算 goto 函数时，如果当前状态 i 为 0，则对线性表二分查找，否则直接搜索 statetable[i]后的单链表。

④计算 output 函数分为两步：一是构造有限状态自动机的过程中，每处理完一个模式串，将该模式串加入到当前状态 s 的输出函数中；二是若 failure (s)=s'，则 output(s)=output(s) ∪ output(s')。

(2)匹配阶段

①将当前状态 s 初始化为 0，文本串指针指向文本串的头。

②如文本串指针不为空，则取出所指的字符“a”；否则匹配过程结束。

③调用 goto 函数，计算 s'=goto(s,a)。若 s=0，则二分查线性表 L_0，否则直接搜索状态 s 对应的单链表。

④如果 s'==fail，调用 failure 函数，即当前状态 s=failure(s)。如果 $s\neq 0$，转至步骤③；

⑤若 $s'\neq$fail，则 $s=s'$。若 output(s)=NULL，转到步骤②；如果 output(s)≠NULL，输出 output(s)的值 y_i，表示模式串 y_i 匹配成功，转至步骤②。

3.3 AC_SC 算法分析

(1)空间复杂度

完全 Hash 表方式存储有限状态自动机，对于中文模式串，每一状态所需的存储空间为 256×256(=65 536)，算法的空间复杂度为 $O(n\times 65\ 536)$，其中 n 为状态数。

状态矩阵存储方式对模式串集建立状态矩阵，算法空间复杂度为 $O(n\times m)$，其中 m 为模式串中不重复的字符个数。由于常用的汉字个数约为 3 500 且相互独立，对于大规模中文模式串集，算法的空间复杂度为 $O(n\times 3\ 500)$。

带跳转距离的邻接链表有 n 个结点(除顶点表)，每一结点所需存储空间为 1，顶点表所需的存储空间为 $2n$，算法的空间复杂度为 $O(2n+n\times 1)(=O(3n))$，故带跳转距离的邻接链表方式的空间性能优于 Hash 表方式和状态矩阵方式。

(2)时间复杂度分析

计算 goto 函数时，实际上就是根据当前状态和当前输入的字符在有限状态自动机上查找下一状态。完全 Hash 表具有直接存取的特性，因此，计算 goto 函数的时间复杂度为 $O(1)$。

状态矩阵存储方式计算 goto 函数时，二维数组具有直接存取的特性，计算 goto 函数的时间复杂度为 $O(1)$。

带跳转距离的邻接链表方式时间复杂度分 3 种情况讨论。

设模式串集合 $K=\{a_0,a_1,a_2,\cdots,a_n\}$，单链表长度集合 $P=\{p_0,p_1,\cdots,p_n\}$，$p_0+p_1+\cdots+p_n=n$，$P'=\{p_1,\cdots,p_n\}$。

①最好的情况，即 $a_0\in a_1\cdots\in a_n$，所有状态的单链表长度都为 1，时间复杂度为 $O(1)$；

②最坏的情况，即 $a_0\neq a_1\cdots\neq a_n$，状态 0 的单链表长度为 n，其他状态的单链表长度为 1，时间复杂度为 $O\left(\frac{1b\ \ n+\sum_{i=1}^{n}1}{n}\right)$。

③一般情况，时间复杂度为 $O\left(\frac{1b\ \ p_0+\sum_{i=1}^{n}p_i}{n}\right)$，由于汉字之间的相互独立性，$P'$ 中元素的值较小，$p_1+\cdots+p_n$ 远远小于 n，时间性能比完全 Hash 表方式和状态矩阵稍差。

4 测试结果与分析

测试环境：CPU 为 Intel(R)Pentium(R)4 2.40 GHz，内存为 1 GB，操作系统为 Microsoft Windows XP Professional Service Pack 2，VC++6.0 语言实现。

模式串集为百度过滤关键词集，每次测试从中取出一定数量的中文模式串。测试文本为 5 个不同的中文文本(大小约为 12 M)。每个文本测试 10 次，取其平均值。

(1)空间性能

如图 5 所示，完全 Hash 表存储方式所需的存储空间较大，且随着模式串数量的增加，所需存储空间近似呈几何增加。当模式串数目大于 1 500 时，完全 Hash 表存储方式所需内存超出了系统所能分配的最大内存，因此没有实验数据 (下同)。状态表矩阵存储方式及位图方式，随着模式串的增加，所需的存储空间近似呈线性增加。邻接链表存储方式的空间性能最优，其存储空间不超过完全 Hash 表的 10%，约为状态表矩阵的 20%。

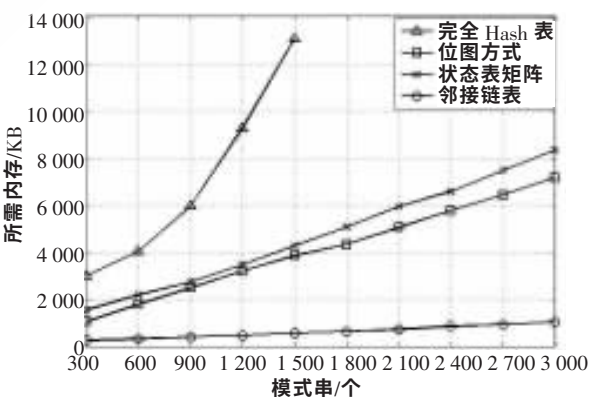

图 5 4 种存储方式的空间性能

(2)时间性能

如图 6 所示，完全 Hash 表存储方式的时间性能最佳，但当模式串数量超过 1 500 时，由于内存消耗过大而无法运行；位图方式时间性能最差。尽管前面理论分析(4.3 节)表明，状态表矩阵方式的时间复杂度略低于

邻接链表方式，但由于状态表矩阵所需存储空间较大，Cache 命中率下降，频繁进行缺页中断操作，导致时间性能下降。因此，实际测试中，状态表矩阵方式的时间性能低于邻接链表方式。

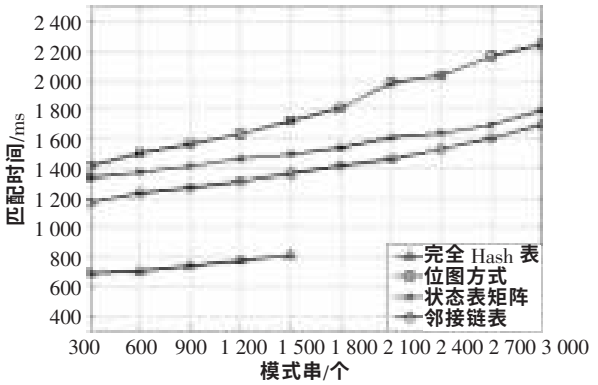

图 6　4 种存储方式的时间性能

本文提出了一种适合中文的多模式匹配算法。该算法采用邻接链表方式存储有限状态自动机，减少了算法的空间开销，同时将扇出系数较大的状态“0”的单链表转化为线性表，以便匹配过程中二分快速查找，提高算法的时间性能。最后对算法空间和时间性能进行了测试，结果表明，本文提出的有限状态自动机的存储方式——改进的邻接链表方式，其时间性能高于状态矩阵存储方式，略低于完全 Hash 表存储方式，而其空间性能远优于完全 Hash 表和状态表矩阵方式，适合大规模中文模式匹配。

参考文献

[1] 王继成，萧嵘，孙正兴，等.Web 信息检索研究进展[J].计算机研究与发展，2001，38(2)：187-193.

[2] 王慧强，杜晔，庞永刚.入侵检测技术研究[J].计算机应用研究，2003(10)：90-115.

[3] KNUTH D E, MORRIS J H, PRATT V R. Fast patter matching in strings [J]. SIAM Journal Computer, 1977,6(2):323-350.

[4] BOYER R S, MOORE J SA fast string searching algorithm[J]. Communications of the ACM, 1977,20(10):762-772.

[5] SUNDAY D M. A very fast substring search algorithm[J]. Communication of ACM,1990, 33(8):132-142.

[6] KARP R M, RABIN M O. Efficient randomized pattern-matching algorithms [J]. IBM J.RES.DEVELOP, 1987, 31(2):249-260.

[7] AHO A V, CORASICK M J. Efficient string matching[J]. An Aid to Biographic Search Communications of the ACM, 1975, 18(6):333-340.

[8] COMMENTZ-WALTER B. A string matching algorithm fast on the average [C].Proceedings of 6th ICALP, 1979(71):118-132.

[9] 王永成，沈州，许一震.改进的多模式匹配算法[J].计算机研究与发展，2002，39(1)：55-60.

[10] NORTON M.Optimizing pattern matching for intrusion detection [EB/OL]. [2006-05-11]. http://does.id-sreseareh.org/OptimizingPatternMatching-ForIDS.pdf.

[11] TUCK N, SHERWOOD T, CALDER B, et al. Deterministic memory efficient string matching algorithms for intrusion detection[C]. IEEE Iafocom,2004:333-340.

[12] 张元竞，张伟哲.一种基于位图的多模式匹配算法[J].哈尔滨工业大学学报，2010，42(2)：277-280.

（收稿日期：2013-03-11）

作者简介：

侯整风，男，1958 年生，教授，主要研究方向：计算机网络，信息安全。

杨波，男，1986 年生，硕士研究生，主要研究方向：计算机网络，信息安全。

朱晓玲，女，1974 年生，讲师，主要研究方向：信息安全。