Wu-Manber 算法性能分析及其改进*)

陈 瑜 陈国龙
(福州大学数学与计算机科学学院 福州 350002)

摘 要 在模式匹配中,多模式匹配算法越来越受到人们的关注。本文首先介绍了一些著名的多模式匹配算法,重点介绍了 Wu-Manber 算法的基本概念及其实现原理,此算法在实践应用中是最有效的。然后提出了对 Wu-Manber 算法的改进,以解决多模式串长度很短时出现的性能问题。最后,实验数据表明,改进后的 Wu-Manber 算法,其性能远远优于传统的 Wu-Manber 算法。

关键词 Wu-Manber 算法,多模式匹配,性能分析

The Performance Analysis of Wu-Manber Algorithm and its Improvement

CHEN Yu CHEN Guo-Long
(Institute of Mathematics and Computer Science, Fuzhou University, Fuzhou 350002)

Abstract In the field of pattern matching, the multiple pattern matching algorithms attract more and more attentions. This paper firstly introduces some famous multiple pattern matching algorithms and puts emphasis on the basic idea and the implementation principle of the Wu-Manber algorithm which is the most efficient in practice. Then, an improvement to the Wu-Manber algorithm is provided to solve the problem of performance falling when the patterns are very short. At last, the experiment data show that the performance of the improved Wu-Manber algorithm is much better than the traditional Wu-Manber algorithm.

Keywords Wu-Manber algorithm, Mutiple pattern matching, Performance analysis

1 引言

近几年,在字符串的模式匹配领域,多模式匹配算法越来越多地受到人们的关注。字符串的多模式匹配问题的形式化定义如下:在字符集Σ上,给定一个长度为n的文本字符串$T[1\cdots n]$,以及k个长度为$m(m$可不同)的模式字符串$P_1\cdots P_k$。如果对于$1\leqslant i\leqslant n$,存在$T[i+1\cdots i+m]=P_j[1\cdots m]$,其中$1\leqslant j\leqslant k$,则模式串$P_j$在文本$T$的位置$i$处出现,即模式串与文本匹配。字符串的多模式匹配问题就是要寻找多个P在T中是否出现,以及出现的位置。

目前已出现多种多模式匹配算法。早在 1975 年,A. V. Aho 和 M. J. Corasick 就提出解决多模式匹配问题的 Aho-Corasick 算法[1],第一次在 O(n)时间复杂度上解决了这个问题,它是 KMP 算法在处理多模式匹配问题上的一个扩展算法。1979 年,Comments Walter 提出了 AC-BM 算法[2],这个算法将 AC 算法和 BM 算法的思想结合起来,获得了很高的执行效率。1994 年,Sun Wu 和 Udi Manber 提出第一个基于过滤思想的 Wu-Manber 算法[3],在实际的应用中获得了极其高的效率。1996 年,Robert Muth 和 Udi Manber 又提出一个快速的多模式模糊匹配算法[4],这个算法也是基于过滤的思想,同时采用了两级散列的技术,从而获得了很高的执行效率。1999 年,Crochemore 等人将 AC 算法和 DAWG 结合,获得了一种新的算法,称为 DAWG-MATCH[5]。他们的实验结果表明该算法比 AC-BM 算法更有效率。

在这些多模式匹配算法中,Wu-Manber 算法是最受人关注的一个算法。Sun Wu 和 Udi Manber[3] 的实验表明,在 Sun Sparc10 上,他们的算法可以于 10 秒内完成在 15.8M 的文本中搜索 10000 个模式的工作。在 WM 算法的基础上,Sun Wu 和 Udi Manber 实现了一个用于模糊匹配的工具 agrep[6] 和一个文本检索的工具 limpse[7],在实际的应用中都获得了良好的效率。但是,我们通过测试实验发现 Wu-Manber 算法在某些特定情况下性能并不是很好,针对这一问题我们展开讨论并对算法做了改进,提高了算法的性能。

2 Wu-Manber 算法概述

Wu-Manber 算法将过滤思想和 Boyer-Moore 算法[8] 思想结合起来。Boyer-Moore 算法中的不良字符转移机制记录了字符集中所有字符在模式串中出现的最右位置距离模式串串尾的距离。在算法匹配过程中,可以根据这个位置信息安全地移动而不用担心忽略任何可能出现的匹配。但是随着模式串个数的增加,各个字符出现在模式串尾端的概率也相应增加了,相应地与串尾的距离缩小,因而所能跳过的距离也同样变小,所以这种转移机制的效果在多模式串的情况下被极大地削弱了[9]。

Wu-Manber 算法利用块字符扩展了不良字符的转移效果来解决这个问题,同时用散列表来筛选匹配阶段应进行匹配的模式串,减少算法匹配时间[9]。

在每一次对匹配情况的考察中,我们不再一个字符一个字符地进行考察,取而代之地,我们一次考察一"块",即考察B个字符[3]。根据这B个字符的匹配情况来决定模式串的

*)基金项目:福建省自然基金资助项目(A0410010);福建省科技三项资助项目(K03012);福建省教育厅资助项目(JA04155)。陈 瑜 硕士研究生,研究方向为计算机算法、计算机网络;陈国龙 博士,教授,研究方向为智能优化算法、计算机网络。

移动距离。当模式串个数较少时，发生散列冲突的可能性较小，通常取 $B=2$，否则取 $B=3$。

Wu-Manber 算法首先要对模式串的集合进行预处理，预处理阶段将建立三个表格：SHIFT 表，HASH 表和 PREFIX 表。SHIFT 表中存储字符集中所有块字符在文本中出现时的转移距离。HASH 表用来存储匹配窗口内尾块字符散列值相同的模式串。PREFIX 表用来存储匹配窗口内首块字符散列值相同的模式串。在对模式串进行匹配的时候就是利用这三个表完成文本的扫描和寻找匹配的过程。

假设 $B=2$，S 是我们当前正在处理的文本中的 2 个字符组成的字符串。并且 S 映射到 SHIFT 表的第 i 项，即 S 被散列为 i。考虑两种情况[3]：

(1)S 不在任何一个模式串中出现，我们可以将考察的位置向后移动 $m-B+1$ 个字符的距离，于是我们在 SHIFT $[i]$ 中存放 $m-B+1$。

(2)S 在某些模式串中出现，这种情况下，我们考察那些模式串中 S 出现的最右位置。假设，S 在 P_j 中的 q 位置出现，且在其他的出现 S 的模式串中 S 的位置都不大于 q。那么我们应该在 SHIFT $[i]$ 中存放 $m-q$。

下面描述算法匹配的主要过程：

(1)计算所有模式串中最短的模式串的长度，记为 m，并且我们只考虑每一个模式串的前 m 个字符，即 m 为匹配窗口的大小。

(2)根据文本当前正考察的 m 个字符计算其尾块字符散列值 h。(从 $T_{m-B+1}\cdots T_m$ 开始)

(3)检查 SHIFT $[h]$的值，如果 SHIFT $[h]>0$，那么将窗口向右移动 SHIFT $[h]$大小位置，返回第(2)步，否则，进入第(4)步。

(4)计算文本中对应窗口“前缀”的散列值，记为 text_prefix。

(5)对符合 HASH $[h]\leqslant p<$ HASH $[h+1]$的每一个 p 值，检验是否存在 PREFIX $[p]=$ text_prefix。如果相等，对文本和模式串进行完全匹配[3]。

例如：在文本串“All of the students are very cool in this school.”中匹配模式串 student、crude 和 school。假设 B=2。

(1)计算 $m=5$，即匹配窗口大小为 5，如图 1 所示。

All of the students are very cool in this school.

图 1　文本匹配窗口示意图

(2)计算 SHIFT 表

考虑每一个模式串的前 5 个字符，计算每个块字符与匹配窗口内模式串串尾的距离如图 2 所示，此即当该块字符在文本中出现时的转移距离。

student:

st	tu	ud	de
3	2	1	0

crude:

cr	ru	ud	de
3	2	1	0

school:

sc	ch	ho	oo
3	2	1	0

图 2　块字符转移距离示意图

合并后的 SHIFT 表如图 3 所示，其它未出现在匹配窗口内的块字符的 SHIFT 表值均为 $m-B+1=4$：

st	tu	ud	de	cr	ru	sc	ch	ho	oo	…
3	2	1	0	3	2	3	2	1	0	4

图 3　SHIFT 表示意图

(3)计算 HASH 表，如图 4 所示。

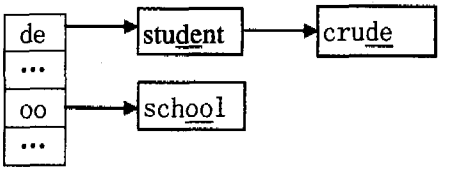

图 4　HASH 表示意图

(4)计算 PREFIX 表，如图 5 所示。

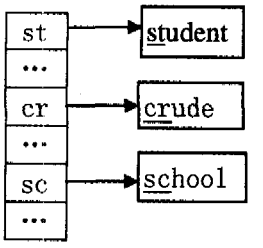

图 5　PREFIX 表示意图

(5)匹配过程。

All of the students are very cool in this school.

图 6　匹配过程

如图 6 所示：从右向左扫描前 5 个字符，o 在 SHIFT 表中值为 4，可以将考察的位置向后移动 4 个字符的距离。th 在 SHIFT 表中值也为 4，所以也将考察的位置向后移动 4 个字符的距离。st 在 SHIFT 表中值为 3，所以将考察的位置向后移动 3 个字符的距离。de 在 SHIFT 值为 0，转入 HASH 表，在 HASH 表中对应的模式串有 student 和 crude。然后计算当前文本窗口 stu de 的 text_prefix，即 st 的散列值，转入 PREFIX 表对应的模式串有 student，最后进行完全匹配。剩余文本匹配过程类似。

3 算法测试及性能分析

(1)测试环境

方正个人电脑，配置如下：CPU 是 Intel Pentinum IV 2.4GHz，内存 512M，硬盘 80G，操作系统 Windows 2000 Server，算法实现环境是 Visual C++ 6.0。

测试 Aho-Corasick 和 Wu-Manber 两种算法，所使用的中文语料长度为 5,797,998 字节；英文语料长度为 4,296,532 字节。

(2)模式串个数对算法性能的影响

使用英文语料作为测试文本，测试当模式串个数分别为 1,100,500,1000,5000 时的算法性能。测试结果如表 1 所示。

表 1 测试结果(单位：毫秒)

模式串个数	算法名称	
	Aho-Corasick	Wu-Manber
1	278	10
100	1593	38
1000	2391	56
5000	2372	1337

从表 1 可以看出，Wu-Manber 算法的匹配速度明显要快于 Aho-Corasick 算法，最好情况下快了将近 40 倍左右。对于 Wu-Manber 算法，当模式串增加到 5000 个时，hash 值相同的模式串个数大量增加，导致进入前缀匹配的模式串的数目增加，最终进入完全匹配的模式串数目增加，因此匹配时间急剧上升。

(3)模式串长度对算法性能的影响

使用中文语料作为测试文本，测试当模式串个数为 10，模式串长度分别为 1,10,100,500 个汉字时的算法性能。测试结果如表 2 所示。

表 2 测试结果(单位：毫秒)

模式串长度(汉字)	算法名称	
	Aho-Corasick	Wu-Manber
1	256	28
10	294	6
100	362	3
500	416	6

从表 2 可以看出，Wu-Manber 算法不仅匹配速度快，而且匹配时间不随模式串长度的增加而有明显的增长，性能很稳定。一般情况下，模式串长度越大，算法的匹配速度越快。

4 Wu-Manber 算法改进及性能分析

在测试中，我们还发现，当模式串长度为一个字节时，Wu-Manber 算法匹配所用时间是其它情况下的 7～10 倍，如图 7 所示。当模式串长度为一个字时，Wu-Manber 算法匹配所用时间是其它情况下的 4～9 倍，如图 8 所示。

经研究发现，这主要是由于算法在匹配过程中的最大“跳跃”距离是由所有模式串中最短的模式串长度 m 决定的。所以，当出现单字节模式串时，$m=1$，每次考察的字符的个数只能为 1，即 $B=1$。同时也决定了不匹配时最大的移动距离只能为 $m-B+1=1$。也就是说每次匹配所能移动的最大距离为 1，这将大大降低算法的性能。并且，这种情况对于中文匹配也是一样的，只不过中文是 2 字节编码的。当出现单个字模式串时，$m=2$，每次考察的字的个数只能为 1，如果不匹配则最多只能跳过一个字的距离。

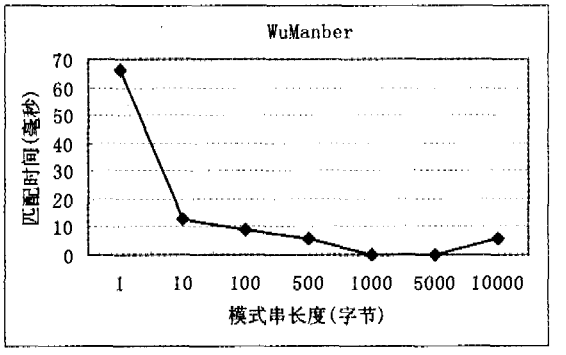

图 7 不同模式串长度的测试结果图(英文)

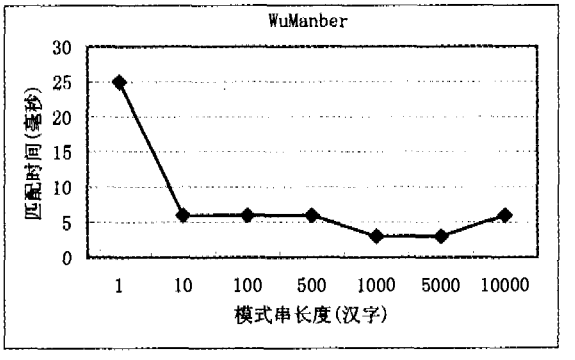

图 8 不同模式串长度的测试结果图(中文)

4.1 改进的 Wu-Manber 算法

针对上述问题，我们对 Wu-Manber 算法进行了改进。改进的思路：将模式串分成两组，用 Wu-Manber 算法对文本进行两次匹配，降低短字符串对其它字符串的影响。

下面描述改进算法的实现过程：

(1)对于 k 个模式串，如果当前第 i 个模式串有 $i<k$，那么进入第(2)步。否则，进入第(4)步。

(2)计算该模式串的长度 length(pat[i])。如果长度小于等于 2，则将该模式串存入模式串数组 shortPat。否则将该模式串存入模式串数组 longPat。

(3)i 增加 1，进入第(1)步。

(4)对于数组 shortPat 中的模式串，用 Wu-Manber 算法对文本进行匹配。

(5)对于数组 longPat 中的模式串，用 Wu-Manber 算法对文本进行匹配。

改进的 Wu-Manber 算法一方面降低了当多模式串匹配中出现单字节或单个汉字模式串时对其它模式串匹配速度的影响。另一方面，从匹配的可能性来讲，文本中的两个字节(汉字)与某模式串的最后两个字节(汉字)匹配的可能性将比文本中的一个字节(汉字)与某模式串的最后一个字节(汉字)匹配的可能性要小得多。

例如，在文本串“I am a student.”中匹配模式串“a”和“school”。算法改进之前，因为 $m=1$，每次检查我们只检查一个字符。当我们检测到文本中的“s”时，首先检查 HASH

(下转第 209 页)

过程占用的内存空间与处理时间，提高了频繁项集的挖掘效率。实验结果表明 FPRSG 算法优于 FP-growth 算法。

FPRSG 算法是从物理存储方面对 FP-growth 算法改进，完全可以将其与算法逻辑方面的改进（如对生成条件模式基必要性的判断和裁剪）结合起来，进一步提高频繁模式挖掘的效率。

参 考 文 献

1 Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. 1994 Int. Conf. Very Large Data Bases(VLDB'94)[C], Santiago, 1994. 487～499

2 Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Proc. of 2000 ACM SIGMOD Int. Conf. on management of data [C], Dallas, USA, 2000. 1～12

3 Han J W, Pei J, Yin Y. Mining partial periodicity using frequent pattern trees [R]. Canada: Simon Fraser University: [Computing Science Technical Report: TR-99-10]. 1999

4 Mining generalized association rules [A]. In: Proc. of the 21st Int Conf. on Very Large DataBase [C]. Zurich, Switzerland, 1995. 407～419

5 王晓峰，王天然，赵越. 一种自顶向下挖掘长频繁项的有效方法. 计算机研究与发展，2004，41(1)：148～155

6 闫炜，崔杜武，付长龙. 基于幂集的关联规则挖掘算法研究. 计算机工程与应用，2004(1)：192～194

7 孟祥萍，王华金，王贤勇，任纪川，鞠传香. 基于改进 FP-树的最大模式挖掘算法. 计算机工程与应用，2005，14：179～181

8 陈安龙，唐常杰，陶宏才，元昌安，谢方军. 基于极大团和 FP-Tree 的挖掘关联规则的改进算法. 软件学报，2004(8)：1198～1207

(上接第 205 页)

表，发现有"school"可能匹配，然后查到 PREFIX 表时才知道不匹配。算法改进之后，在第二次匹配过程中，$m > 2$，每次至少检查两个字符，发现"*st*"与"*sc*"不匹配，我们不会转而去查 HASH 表和 PREFIX 表。这样就减少了很多没有意义的比较，大大降低算法匹配的时间。

4.2 算法性能分析

下面，我们对 Wu-Manber 算法及其改进算法进行测试比较。

首先，我们从英文语料中取 100 个长度小于 6 个字符的模式串，测试这 100 个模式串当中出现不同个数的单字节字符串时所用时间，如图 9 所示。

其次，我们从中文语料中取 100 个长度小于 6 个汉字的模式串，测试这 100 个模式串当中出现不同个数的单汉字字符串时所用时间，如图 10 所示。

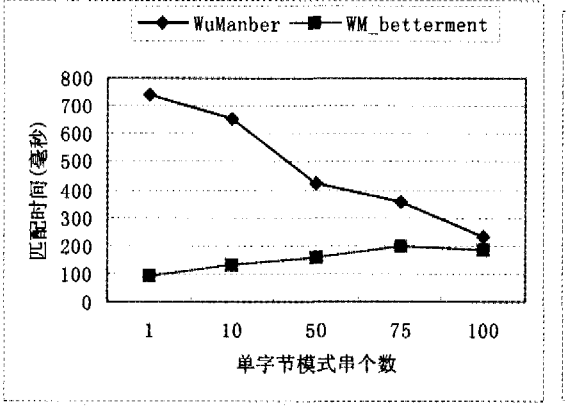

图 9 比较结果图(英文)

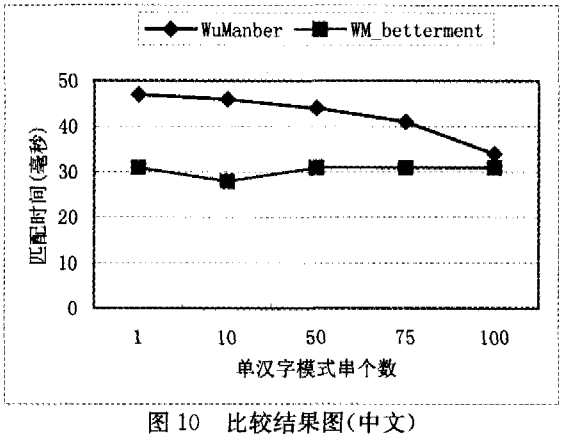

图 10 比较结果图(中文)

从图 9、图 10 可以看出，改进后的 Wu-Manber 算法性能有明显提高，匹配速度快于传统 Wu-Manber 算法。特别是对于英文匹配，在单字节模式串出现个数较少时，匹配速度较原来提高了 5～8 倍。考虑到现实中，单字节(单汉字)模式串出现个数较少，所以 Wu-Manber 改进算法还是具有一定的实用价值。

结束语 本文研究并测试了 Wu-Manber 算法，发现在大多数情况下，算法都具有良好的性能。针对匹配过程中出现单字节(单汉字)模式串时会大大降低 Wu-Manber 算法速度的情况，在不影响算法总体性能的基础上对算法进行了改进，使得算法性能有了进一步的提高。

参 考 文 献

1 Aho A V, Corasick M J. Efficient string matching: an aid to bibliographic search. Communications of ACM, 1975, 18(6): 333～340

2 Commentz-Walter B. A string matching algorithm fast on the average: [Technical Report]. The University of Heidelberg: IBM Heidelberg Scientific Center, Sep. 1979

3 Wu Sun, Manber U. A Fast Algorithm for Multi-pattern Searching: [Technical Report]. The University of Arizona: The Computer Science Department, May1994

4 Muth R, Manber U. Approximate multiple string search. In: Proc. 7th Combinatorial Pattern Matching (CPM'96). LNCS 1075. 1996. 75～86

5 Crochemore M, Czumaj A, Gasieniec L, Lecroq T, Plandowski W, Rytter W. Faster practical multi-pattern matching. Inf. Process. Leu, 1999, 71(3-4): 107～113

6 Wu Sun, Manber U. Agrep: A Fast Approximate Patternmatching Tool. Usenix Winter Technical Conference, San Francisco, 1992

7 Wu Sun, Manber U. GLIMPSE: A Tool to Search Through Entire FileSystem. Usenix Winter Technical Conference, San Francisco, 1994

8 Boyer R S, Moore J S. A fast string searching algorithm. Communications of ACM, 1977, 20(10): 762～772

9 谭建龙. 串匹配算法及其在网络内容分析中的应用：[学位论文]. 中国科学院计算技术研究所，2003