第 6 卷第 8 期
1995 年 8 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 8
Aug. 1995

一种有效的并行汉字/字符串相似检索技术*

王素琴　　邹旭楷

(郑州大学计算机科学系，郑州 450052)

摘要　本文提出了一种有效的并行汉字/字符串相似检索技术.通过引入搜索状态向量及字符—模式匹配向量,该技术将字符串匹配比较转化为简单的整数字位运算,通过对字符串方向相反的搜索有效地实现了多处理机对汉字/字符串的并行相似检索.文中也给出了并行实现算法,同时分析了算法的复杂性.

关键词　并行算法,文本,模式,字符串检索,搜索状态向量,字符—模式匹配向量,允许错误的匹配,相似匹配,编辑距离.

字符串相似检索是指在文本 $Text=t_1\cdots t_n$ 中检索与子串 $Pat=p_1\cdots p_m$(称为模式)相似的所有子串.在顺序检索情况下,已提出了很多相似串匹配方法与技术[1-4],对字符串进行并行相似检索,也有两个算法.文献[5]提出了一种并行相似串检索算法,该算法只适合于允许替代错的相似匹配,使用 $n*k$ 个处理机,运行时间为 $O(\log(m/k)\log k/(\log\log m+\log\log k)+\log m)$.文献[6]提出了第一个允许插入/删除/替代的并行相似匹配算法,它使用 n 个处理机,运行时间为 $O(\log m+k)$.以上所有顺序/并行算法只能对单字节字符串检索.文献[7]提出了一种适合于字符串又适合于汉字串的快速相似检索技术.基于文献[7]中的算法,本文提出一种时空优化的汉字/字符串并行相似检索算法.该算法使用 n/m 个处理机,运行时间为 $O(km)$.与文献[5]中的算法相比,本算法可适合于允许替代错的相似匹配,但使用更少的操作(处理机个数 * 运行时间).与文献[6]中的算法相比,本算法在操作总数量上与文献[6]相同,但本算法的操作仅为简单的整数字位运算,因此在实际使用上,比文献[6]中的算法要快.另外,本算法需要附加空间 $O(64+m)$,而文献[6]中的算法需要附加空间 $O(kn)$;一般情况下,m 比 n 小得多,因此本算法需要附加空间比文献[6]的算法要小得多.在匹配过程中只对文本进行读操作,故本算法可运行于 CREW PRAM 或 CRCW PRAM并行处理机上.在实际应用中,m 与 n 相比,通常小到几乎可以忽略的地步,因此无论从处理机数、运行时间、总操作数及附加空间哪方面看,本并行算法都是很实用的.

本文第 1 部分简述文献[7]中提出的汉字/字符串相似检索的原理与方法,第 2 部分讨论并行匹配原理与方法,最后给出并行实现算法并分析了算法的复杂度.

* 本文 1993－09－10 收到,1994－03－16 定稿

作者**王素琴**,1964 年生,讲师,主要研究领域为算法与复杂性,软件工程.**邹旭楷**,1963 年生,副教授,主要研究领域为算法与复杂性,数据库与知识库.

本文通讯联系人:王素琴,郑州 450052,郑州大学计算机科学系

1 汉字/字符串相似匹配

设模式 $Pat=p_1\cdots p_m$(m 为长度),文本 $Text=t_1\cdots t_n$(n 为长度).其中:$p_i(i=1,\cdots,m)$,$t_j(j=1,\cdots,n)$都取自于某个字母表 Σ.假定编辑距离为 k 即允许最多 k 个(插入/删除/替代)错误的匹配.

令 S^0 是一个有 m 个分量的状态向量,S_j^0 表示在 $Text$ 的当前位置 j,其前的字符与 Pat 的任意前缀的匹配情况.记 $S_j^0[i]$为状态向量的第 i 个分量,则如果 $p_1\cdots p_i=t_{j-i+1}\cdots t_j$,有 $S_j^0[i]=0$,否则 $S_j^0[i]=1$,显然:$S_j^0[m]=0$ 表示存在一个确切匹配($t_{j-m+1}\cdots t_j$).

引入字符—模式匹配向量 MT:对任意 $a\in\Sigma$,定义 $MT(a)=(MT_1(a),\cdots,MT_i(a),\cdots,MT_m(a))$,其中:如果 $a=p_i$,则 $MT_i(a)=0$,否则 $MT_i(a)=1$.字符—模式匹配向量可预先根据 Σ 及 Pat 求出.

状态向量 S_j^0 可表示为共有 m 位的 0,1 位串,进而该位串可看作是一个整数.如果 S_j^0 的最低(右)位为 0,则找到一个匹配.同理字符—模式匹配向量 MT 可表示为整数,则 S_j^0 到 S_{j+1}^0的转换可表示为:

$$S_{j+1}^0=S_j^0\gg 1\,|\,MT(t_{j+1}) \tag{1}$$

其中$\gg$1 表示右移 1 位,|表示按位或.

如果是汉字串则由 S_j^0 到 S_{j+2}^0的转换表示为:

$$S_{j+2}^0=(S_j^0\gg 1\,|\,MT(t_{j+1}))\gg 1\,|\,MT(t_{j+2}) \tag{1'}$$

又设 $S^1,\cdots,S^k$ 分别表示允许最多有 $1,\cdots,k$ 个错误的匹配状态向量,$S_j^d(1\leqslant d\leqslant k)$表示到 t_j 允许最多有 d 个错误的匹配状态,即如果 $p_1\cdots p_i$ 与 t_j 向前有最多 d 个错误的匹配,则 $S_j^d[i]=0$,否则 $S_j^d[i]=1$,显然 $S_j^d[m]=0$ 表示存在一个最多 d 个错误的匹配(当 $d=k$ 时表示找到一个相似匹配).

从 S_j^d 到 S_{j+1}^d的转换可表示为:

$$S_{j+1}^d=(S_j^d\gg 1\,|\,MT(t_{j+1}))\&S_j^{d-1}\gg 1\&S_j^{d-1}\&S_{j+1}^{d-1}\gg 1 \tag{2}$$

当为汉字串时,S_j^d 到 S_{j+2}^d的转换可表示为:

$$S_{j+2}^d=((S_j^d\gg 1\,|\,MT(t_{j+1}))\gg 1\,|\,MT(t_{j+2}))\&S_j^{d-1}\gg 2\&S_j^{d-1}\&S_{j+2}^{d-1}\gg 2 \tag{2'}$$

初值:

$$S_0^d=0\cdots 01\cdots 1(0\leqslant d\leqslant k,d\text{ 个 }0,m-d\text{ 个 }1) \tag{3}$$

2 并行匹配原理

2.1 并行匹配机制

上面的字符串相似检索是从头到尾对文本进行扫描,与其对应,我们可以从尾到头对文本进行扫描,此时需要修改上述方法:重新定义 $S^d(0\leqslant d\leqslant k)$如下,$S_j^d$ 表示在 $Text$ 的当前位置 j,其后的字符与 Pat 的任意后缀最多允许 d 个错误的匹配情况,即如果 $p_i\cdots p_m$ 与 t_j 向后有最多 d 个错误的匹配,则 $S_j^d[i]=0$,否则 $S_j^d[i]=1$,显然 $S_j^d[1]=0$ 表示存在一个最多 d 个错误的匹配(当 $d=0$ 时为确切匹配,$d=k$ 时为相似匹配).

令字符—模式匹配向量如上;则由位置 j 后退一个字符到 $j-1$,计算 S_{j-1}^d如下:

当 $d=0$ 时,

$$S_{j-1}^0=S_j^0\ll 1\,|\,MT(t_{j-1}) \tag{4}$$

当 $1\leqslant d\leqslant k$ 时,

$$S_{j-1}^d=(S_j^d\ll 1\,|\,MT(t_{j-1}))\&S_j^{d-1}\ll 1\&S_j^{d-1}\&S_{j-1}^{d-1}\ll 1 \tag{5}$$

当为汉字串时，　$S^0_{j-2}=(S^0_j\ll 1\,|\,MT(t_{j-1}))\ll 1\,|\,MT(t_{j-2})$　(4')

当 $1\leqslant d\leqslant k$ 时，　$S^d_{j-2}=((S^d_j\ll 1\,|\,MT(t_{j-1}))\ll 1\,|\,MT(t_{j-2}))\&S^{d-1}_j\ll 2\&S^{d-1}_{j-1}\&S^{d-1}_{j-2}\ll 2$　(5')

初值：　$S^d_{n+1}=1\cdots 10\cdots 0$ ($0\leqslant d\leqslant k$, d 个 0, $m-d$ 个 1)　(6)

现假定从头到尾及从尾到头同时搜索(当从尾到头搜索时记 S^d 为 R^d)，则当双向搜索相遇时，S^d 与 R^{k-d} 的结合，刚好反映了最后的匹配情况. 分析如下：

假定到位置 t_j，S 与 R 相遇，根据定义：

如果 $p_1\cdots p_i$ 与 t_j 向前有最多 d 个错误的匹配，则 $S^d_j[i]=0$；

如果 $p_{i+1}\cdots p_m$ 与 t_{j+1} 向后有最多 $k-d$ 个错误的匹配，则 $R^{k-d}_{j+1}[i+1]=0$；两者结合有：$p_1\cdots p_ip_{i+1}\cdots p_m$ 与 $t_{j-i+1}\cdots t_jt_{j+1}\cdots t_{j+m-i}$ 有最多 $d+k-d=k$ 个错误的匹配，即 $S^d_j[i]\,|\,R^{k-d}_{j+1}[i+1]=0$. 因此当 S 与 R 相遇时，对所有的 $d(0\leqslant d\leqslant k)$ 计算 $S^d\,|\,R^{k-d}\ll 1$，即计算

$$S=(S^0\,|\,R^k\ll 1)\&(S^1\,|\,R^{k-1}\ll 1)\&\cdots\&(S^k\,|\,R^0\ll 1)\qquad(7)$$

检查 S 的每一位，如果该位为 0，则存在一个匹配.

2.2 并行匹配原理

假定有 $p(p\leqslant n/m)$ 个处理机，可将文本 $Text$ 分成长度为 $h=n/p$(当为汉字串时，如果 h 为奇数，则加 1 变为偶数)的 p 份，即：

$$t_1\cdots t_h\cdots\underbrace{t_{(i-1)*h}\cdots t_{i*h}}_{p_i}t_{i*h+1}\cdots t_{(i+1)*h}\quad\underbrace{t_{(i+1)*h+1}\cdots t_{(i+2)*h}}_{p_{i+1}}\cdots t_n$$

对处理机 $i(i=1,\cdots,p)$，它搜索 $t_{(i-1)*h+1}\cdots t_{i*h}$，如果 i 为奇数，从左到右搜索，反之从右到左搜索，当处理机 i(i 为奇数)与处理机 $i+1$ 在 t_{i*h} 相遇时，处理机 i 的 S 与处理机 $i+1$ 的 S 相结合则得到其中所有相似匹配；但是处理机 $i+1$ 与 $i+2$ 的搜索是互相背离的，字符串 $t_{(i+1)*h-m+2}\cdots t_{(i+1)*h}t_{(i+1)*h+1}\cdots t_{(i+1)*h+m-1}$ 中的匹配却不能搜索出，当为汉字串时，$t_{(i+1)*h-m+3}\cdots t_{(i+1)*h}t_{(i+1)*h+1}\cdots t_{(i+1)*h+m-2}$ 中的匹配却不能求出. 解决办法如下：让处理机 $i+1$ 从 $t_{(i+1)*h+m/2-1}$ 开始向左搜索，处理机 $i+2$ 从 $t_{(i+1)*h-m/2+1}$ 处开始向右搜索；当为汉字时，如果 $m/2$ 为奇数，处理机 $i+1$ 从 $t_{(i+1)*h+m/2-1}$ 开始向左搜索，处理机 $i+2$ 从 $t_{(i+1)*h-m/2+2}$ 开始向右搜索，如 $m/2$ 为偶数，处理机 $i+1$ 从 $t_{(i+1)*h+m/2-2}$ 开始向左搜索，处理机 $i+2$ 从 $t_{(i+1)*h-m/2+1}$ 开始向右搜索. 显然对处理机 1，其前要初始化上 $m/2$ 个符号，对处理机 p，如果 p 为偶数，其尾部要加 $m/2-1$ 个字符. 当为汉字串时，如 $m/2$ 为奇数，处理机 1 前面初始化上 $m/2-1$ 个字符，如 $m/2$ 为偶数，则处理机 p 后要增加 $m/2-2$ 个字符.(说明：以上假定 m 为偶数，如为奇数，$m/2$ 应代之于大于或等于 $m/2$ 的最小整数或者小于或等于 $m/2$ 的最大整数).

3 实现算法与复杂性分析

3.1 并行实现算法

Step 1. 依定义计算 MT；

Step 2. For $i=1$ to p do_parallel /* 并行执行 */

　　if i 为奇数 then

Step 2.1. 依式(3)初始化 $S^d(0\leqslant d\leqslant k)$

Step 2.2. 从 $(i-1)*h-m/2+1$ 向右扫描到 $i*h$

Step 2.2.1. 依式(1)或式(1')计算 S^0

Step 2.2.2. 依式(2)或式(2')计算 $S^d(1\leqslant d\leqslant k)$

Step 2.2.3. if S^k 最低位为 0 then 找到一个相似匹配

else

Step 2.1. 依式(6)初始化 $S^d(0\leqslant d\leqslant k)$

Step 2.2. 从 $i*h+m/2-1$ 向左扫描到$(i-1)*h+1$

Step 2.2.1. 依式(4)或式(4')计算 S^0

Step 2.2.2. 依式(5)或式(5')计算 $S^d(1\leqslant d\leqslant k)$

Step 2.2.3. if S^k 最高位为 0 then 找到一个相似匹配

Step 3. For $i=1$ to p do_parallel /* 并行执行 */

if i 为奇数 then

Step 3.1. 依式(7)计算 S

Step 3.2. 检查 S 的每一位,如果为 0,则找到一个相似匹配

说明:当为汉字串检索时,如 $m/2$ 为奇数,则 Step 2.2 中$(i-1)*h-m/2+1$ 应为$(i-1)*h-m/2+2$,如 $m/2$ 为偶数,则 Step 2.2 中 $i*h+m/2-1$ 应为 $i*h+m/2-2$.

3.2 复杂性分析

算法预先要根据模式 Pat 及字母表 Σ 生成字符—模式匹配向量 MT(Step 1),其处理时间为 $O(m+|\Sigma|)$; Step 2,对每个处理机,扫描的字符个数为 $n/p+m/2$,每次计算S^0-S^k需 $k+1$ 次,此检索时间为$(n/p+m/2)*(k+1)$. 最后 S 与 R 合并后作单纯检查(Step 3),其时间可忽略不计,因此算法的并行运行时间为 $O((n/p+m)*k)$. 为了存放 MT,需$|\Sigma|$个字的附加空间. 当$|\Sigma|>m$ 时,可用 $m+1$ 个字来存放 MT,另外引入字符数组 $inter\_mt[|\Sigma|]$,将 Pat 中的 $p_1,\cdots,p_m$ 影射到 $1,\cdots,m$,而将非 Pat 中的所有字符影射到 0. $inter\_mt$ 占的空间为$|\Sigma|/4=256/4=64$(假定 Σ 包括所有普通字符),因此总的附加空间为 $64+m$ 个字.

当 $p=n/m$ 时,运行时间 $t=1.5km$,$p*t=(n/m)*1.5km=O(kn)$,因此本并行算法是优化的. 通常 m 与 n 相比,小的几乎可以忽略,因此本算法实际上是很有效的.

4 总 结

上面给出了一种实际有效的并行相似汉字/字符串检索技术及实现算法,与其它并行算法相比,本算法的实现简单,时空优化,可实现字符串检索又可实现汉字串检索,具有较强的理论意义与实际使用价值.

参考文献

1 Chang W I, Lawler E L. Approximate string matching in sublinear expected time. FOCS, 1990. 116—124.

2 Galil Z, Park K. An improved algorithm for approximate string matching. SIAM J. Comput., 1990, **19(12)**: 989—999.

3 Crochemore M, Perrin D. Two way string matching. J. Assoc. Comput., Mach. 1991, **38(3)**: 651—675.

4 Wu S, Manber U. Fast text searching allowing errors. Comm. ACM, 1992, **35(10)**: 83—91.

5 Galil Z, Giancarlo R. Parallel string matching with k mismatches. Theoret. Comput. Sci., 1987,**51(2)**:341—348.

6 Landau G M, Vishkin U. Fast parallel and serial approximate string matching. J. Algorithms, 1989,**10(2)**:157—169.

7 邹旭楷,王素琴.允许错误的(汉字)字符串快速检索技术.软件学报,1994,**5(10)**:55—59.

AN EFFECTIVE APPROACH TO PARALLEL APPROXIMATE CHAR/CHINESE CHARACTER STRING SEARCHING

Wang Suqin　Zou Xukai

(*Department of Computer Science, Zhengzhou University, Zhengzhou* 450052)

Abstract　This paper offers an effective approach to parallel approximate string searching. By using searching state vector and char_pattern matching vector, this approach changes text_pattern matching from comparison to simple integer bit operation and by searching string from the two ends to the middle. It implements effectively the parallel approximate Char/Chinese character string searching on multiprocessors. The parallel implementation algorithm and the analysis of the algorithm are also provided.

Key words　Parallel algorithms, text, pattern, string searching, searching state vector, char_ pattern matching vector, matching allowing errors, approximate matching, edit—distance.