第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)20—0048—03 文献标识码：A 中图分类号：TP391

基于过滤的中文多模式近似字符串匹配算法

范立新[1,2]，谢晓能[1,3]，吴 飞[1]
（1. 浙江大学计算机学院，杭州 310027；2. 绍兴文理学院计算机系，绍兴 312000；3. 杭州广播电视大学信息工程学院，杭州 310012）

摘 要：当前近似字符串匹配算法主要针对英文等中小字符集，该文针对汉字等大字符集的有效算法很少，尤其缺少适合汉字等大字符集的多模式近似匹配算法的情况，提出了一种适合汉字等大字符集的多模式近似匹配算法——MBPM-BM，通过实验证明了该算法的有效性。
关键词：近似字符串匹配；中文字符串匹配；多模式匹配；位并行运算；过滤

Algorithm of Multiple Approximate String for Chinese Characters Based on Filtering

FAN Lixin[1,2], XIE Xiaoneng[1,3], WU Fei[1]
(1. College of Computer, Zhejiang University, Hangzhou 310027; 2. Department of Computer, Shaoxing Arts and Science University, Shaoxing 312000; 3. College of Information Engineering, Hangzhou Radio & TV University, Hangzhou 310012)

【Abstract】 Most of the algorithms of approximate string match are designed for small or middle size of character set. Until now, people can't find any efficient algorithms for searching of multiple patterns of large size of character set. This paper presents an algorithm——MBPM-BM, which can be used for searching of multiple patterns. Experimental results show that MBPM-BM works well in practice especially in chinese characters match.
【Key words】 Approximate string match; Chinese string match; Multiple patterns match; Bit-parallel calculation; Filtering

近似字符串匹配在病毒检测、语音识别、文件检索、计算生物学、模式识别、OCR纠错等许多领域有广泛应用，是算法设计中的一个热点。可描述为：给定一个文本$T=T_{1...n}$、一个模式$P=P_{1...m}$以及允许的最大误差k (k<m)，找出满足$ed(s,P)\leq k$的所有子串s。其中ed(s,P)是s和P 2个字符串之间编辑距离的函数。当查找r个模式$P^1,\ldots,P^r$时，就扩展成多模式近似字符串匹配。即用户提交r个模式$P^1,\ldots,P^r$，在T中查找满足$ed(s,P^q)\leq k$的所有子串s，其中$1\leq q\leq r$。每个模式可有不同长度$m_1,\ldots,m_r$。

1 背景与相关研究

1.1 符号表示

用s代表任意字符串，a、b、c代表字符，Σ表示字符集，$s\in\Sigma^*$，|s|表示其长度。s_i代表s中第i个字符，其中$i\in\{1,\ldots,|s|\}$，用$s_{i..j}$表示$s_i\ldots s_j$。为简化描述，文中部分内容采用指数形式来表示位的二进制，如$0^3 1^2=00011$。

设Σ是全体字符的集合，其大小$|\Sigma|=\sigma$，匹配文本$T\in\Sigma^*$，长度是n，模式$P\in\Sigma^*$的长度是m，$k\in N$表示允许的最大误差，$\alpha=k/m$代表误差率。距离函数$ed:\Sigma^*\times\Sigma^*\rightarrow N$。近N似字符串匹配可定义为给定T、P、k、ed(.)，求解满足条件$ed(P,T_{i..j})\leq k$的所有字符串$T_{i,...,j}$，或求解所有满足条件的开始位置i和终止位置j。

1.2 相关算法介绍

(1)BPM算法[4,5]和MBPM算法[3]

BPM针对动态规划矩阵进行位并行运算，是目前理论上最快的算法(非过滤方法)。若计算机物理字长为w位，但BPM只用到了其中m位，文献[3]对BPM进行了扩充，若r个模式的总长不超过w，即$rm\leq w$，则同时搜索r个模式。若r个模式总长超过w，将模式分组，按每组$\lfloor w/m\rfloor$个模式处理。

(2)BPM-BM算法[2]

若$M[T_j]=0$，称T_j无用字符；若$M[T_j]\neq 0$，称T_j有用字符。其中$1\leq j\leq m$，M[]的含义与BPM中模式匹配向量数组M[]相同。Bad(s)表示s中无用字符数，Good(s)表示s中有用字符数。S_j对应BPM中扫描至T_j后的Score值。一个有用字符最多使Score减1，一个无用字符不会使Score减1。若$S_j=m$且T_{j+1}是无用字符，则$S_{j+1}=S_j=m$。

引理1 若$S_j=m$且$Bad(T_{j+1..j+m})\geq k+1$，有$S_{j+i}\geq k+1$，其中$1\leq i\leq m$。

引理2 若$S_j=m$且$Bad(T_{j+1..j+m})\geq k+1$，$T_{j+i}$是从$T_{j+m}$开始从右往左数的第k+1个无用字符，则将$T_{j+1,...,}T_{j+i-1}$全改为无用字符，不影响P与T从$T_j$开始的后续匹配。

在引理2前提下，可直接令$S_{j+i}=m$。实际上若能记住T_{j+i}右边第1个有用字符T_{j+last}，必有$S_{j+last-1}=\ldots=S_{j+i}=m$，故可令$S_{j+last-1}=m$，然后继续扫描。

本文把区域$T_{j+1,...,}T_{j+m}$看作当前的过滤窗口。算法思想是：先确定过滤窗口，然后从窗口右端开始向左扫描，变量last记录最新遇到的有用字符位置，bad累计遇到的无用字符总数，一旦bad大于k，则当前窗口不满足匹配条件，将j+last作为新窗口起点，然后重新开始。若向左扫描到达窗口起始端，则用BPM进一步对该区域进行匹配。

2 多模式近似字符串匹配

设r个模式$P^1,\ldots,P^r$的长度分别为$m_1,\ldots,m_r$，则

作者简介：范立新(1968－)，男，硕士、讲师，主研方向：计算机算法，信息检索，数据库应用；谢晓能，硕士、讲师；吴 飞，博士、副教授
收稿日期：2006-04-17 **E-mail：**csclx@zscas.edu.cn

minL=min{m_q | 1≤q≤r}。假设计算机物理字长为w位，且r个模式总长不超过w，条件为

$$\sum_{q=1}^{r} m_q \le w \tag{1}$$

2.1 多模式跳跃引理

先作如下准备，$Bad_q(s)$：针对模式P^q，s中无用字符个数。IsGood(c)：若存在q、i（1≤q≤r，1≤i≤m_q），使c=P_i^q，则为真，否则为假。类似地，在引理 3 中，分别将满足IsGood(c)为真或假的字符c称为有用或无用字符。s_j^q：针对模式P^q，在BPM中扫描至T_j后的Score值。

引理 3 设r个模式P^1、…、P^r，对q∈{1,…,r}，均有s_j^q=m_q，且i是使得Bad_q ($T_{j+i..j+minL}$)≥k+1 的最小i，则将$T_{j+1..j+i-1}$ 全改为无用字符不影响P^1、…、P^r与T从T_j开始的后续匹配。实际上如能记住T_{j+i}右边的第一个有用字符T_{j+last}，则对q∈{1..r}，均有$S^q_{j+last-1}$=…=S^q_{j+i}=m_q，不必检测$T_{j+i..j+last-1}$。可跳至T_{j+last}处，并令$S^q_{j+last-1}$=m_q，继续扫描。

2.2 MBPM-BM 主要思想

(1)为使多个模式对齐，仅在满足多模式跳跃引理情况下才移动至新位置 j+last，函数 IsGood()作为记录和改变 last 值的判断条件；

(2)令过滤窗口大小为 minL，多个模式右对齐（忽略较长模式的左端部分），从窗口右端开始向左扫描；

(3)用last记录扫描过程中遇到的有用字符最新位置，状态变量S并行计算扫描过程中每个模式所对应的累计无用字符总数bad_1、…、bad_r。若bad_q（1≤q≤r）均大于k，将j+last作为新窗口起点；若向左扫描到达窗口起始端，则采用MBPM进一步对该区域进行匹配。状态变量S的具体格式后面详述。

图 1 给出了对 2 个模式进行并行过滤的过程。

其中，√和×分别表示多模式情况下的有用和无用字符，为方便起见，图中只有 2 个模式 P1、P2，长度为别为 m1、m2，且 m1>m2。

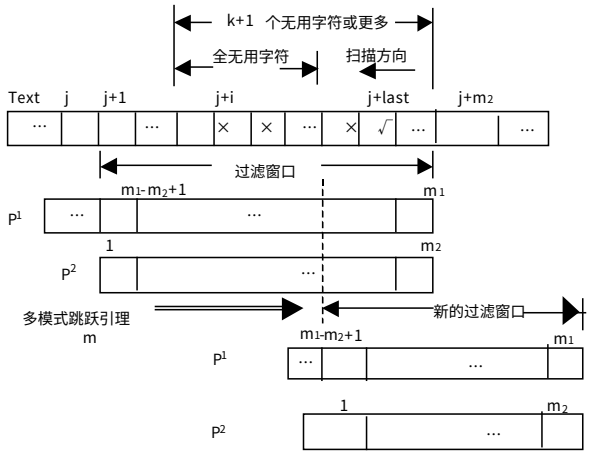

图 1 多模式跳跃引理

2.3 并行记录多个 bad 值

由于统一采用 MinL 作为过滤窗口的标准长度，因此每个模式所对应的 bads 满足 0≤bad≤MinL，但为了判断处理的方便，调整了 bads 值的设置方式，初值设为 k，然后每次遇到一个 bad 字符就减 1，实际单元的取值范围为[k- MinL, k]。保存每个模式所对应的 bads 值的相应单元的长度为

$$L = \lceil \log_2 MinL \rceil + 1$$

最后的加 1 既可保证单元的足够长度，又可作为相应模式的 bads 值是否已经超过 k 的判断标志。于是，利用一个物理字进行存储的所需条件为

$$r \times L \le W \tag{2}$$

显然当 minL=1、2、3，且所有模式长度 mq 均相等，才有

$$\sum_{q=1}^{r} m_q = r \times (\lceil \log_2^{\min L} \rceil + 1)$$

当模式长度m_q不全相等或minL≥4 时，有

$$\sum_{q=1}^{r} m_q > r \times (\lceil \log_2^{\min L} \rceil + 1)$$

因此任何时候均有

$$\sum_{q=1}^{r} m_q \ge r \times (\lceil \log_2^{\min L} \rceil + 1)$$

其中，$r \times L \le w$弱于前面$\sum_{q=1}^{r} m_q \le w$，只需保证$\sum_{q=1}^{r} m_q \le w$即可。当然，实际中若超出计算机字宽 w 的话，可串接多个字，本文为方便说明才提出上面假设。

为并行计算，S中r个单元的初值均为k，分割符为 1，也就是S= $(10^L+k)^r$。为并行更新S值，单独创建一个数组B[]，用来并行记录不同模式对应的好字符、坏字符，定义为

$$B[x] = \sum_{q=1}^{r} \sum_{i=1}^{m_q} \delta(x \ne p_i^q) 2^{(q-1)\times L}$$

其中，x∈Σ，函数δ(C)=1 当条件C为真，否则为 0。例如，当P^1=aabab，P^2=caaac，minL=5，L=4，则B[a]=[0000,0000]，B[b]=[0000，0001]，B[c]=[0001,0000]，对字符x∉{a,b,c}，则B[x]=[0001,0001]。

若 k=2，则 S 的初值=[1010,1010]。

当扫描至位置 i 时，状态变量 S 的更新公式为

S=S-B[T_{j+i}]

因为S中每个单元初值均为k，所以只需判断每个单元标志位。定义变量high=$(10^{L-1})^r$，形式化定义为

$$high = \sum_{q=1}^{r} 2^{q \times l-1}$$

于是可利用条件(S&high)=0 来判断 S 中是否还有不是 0 的标志位，若 S 中标志位全为 0，则满足多模式跳跃引理，将位置 j+last 作为新过滤窗口起始位置。若向左扫描到达窗口起始端，则采用 MBPM 进行匹配。

2.4 更新 last 值

当前字符为有用字符时就更新last。定义变量ones=$(0^{L-1}1)^r$，形式化定义为

$$ones = \sum_{q=1}^{r} 2^{(q-1)\times L+1}$$

当条件(B[T_{j+i}]&ones)≠ones时，更新last=i。

2.5 多模式应用 BPM

多模式应用 BPM，可支持不同模式长度的 MBPM。算法还用到模式匹配向量数组 PM[]，思路类似 MBPM，但文献[3]只给出相等模式长度下的示例，把它扩充到不同模式长度，定义为

$$PM[x] = \sum_{q=1}^{r} \sum_{i=1}^{m_q} \delta(x \ne p_i^q) 2^{(m_q-i)+\sum_{j=1}^{q-1} m_j}$$

其中，x∈Σ，δ 定义如前。例如：当P^1=aabab，P^2=caaac，则PM[a]=[01110,110100]，PM[b]=[00000,001011]，PM[c]=[10001,000000]，对于字符x∉{a,b,c}，PM[x]=[00000,000000]。

为运算方便还定义了数组

$$MX[q]=\sum_{j=1}^{q} m_j$$

其中，q∈{1..r}。另外，文献[3]中的 ZM、EM、VN、VP 也需作相应修正，详见后面代码。

2.6 MBPM-BM 伪代码

MBPM-BM 伪代码如下：

```
MPrePro(P^1..P^r, k)  //预处理
    minL←min{m_q, q∈1..r}
L←⌈log_2 MinL⌉+1
high←(10^(L-1))^r,ones←(0^(L-1)1)^r,S←(10^L+k)^r
For c∈Σ  Do B[c]←(0^L)^r
For q∈1..r Do
For i∈1..m_s Do B[p_i^q]←B[p_i^q] | 1<<L×(q-1)
For c∈Σ  Do B[c]←B[c]^ones
MX[1]←m_1
For q∈2..r Do MX[q]←MX[q-1]+m_q
For c∈Σ  Do PM[c]←0^m
EM←0, ZM←0,initMC←0
For q∈1..r Do
    EM←EM<<m_q | 1<<(m_q-1)
    ZM←ZM<<m_q | (1<<(m_q-1))-1
    initMC←initMC<<m_q | (1<<(m_q-1))-m_q+k)
For q∈1..r Do
For i∈1..m_q Do PM[p_i^q]←PM[p_i^q] | 1<<(MX[q]-m_q+i)

MBPM-BM(P^1..P^r,T,k)
    MPrePro(P^1..P^r , k)
    j←0
    While j≤n- (minL-k) Do
        S←initS, last←minL+1
        For i=minL-1 Downto 0 Then
If B[T_j+i]&ones≠ones Then last←i
        S←S-B[T_j+i]
                If S & high=0 Then Break
//满足多模式跳跃引理
        End For
        If i≥1 Then j←j+last
        Else
VN←0, VP←1^MX[r], MC←initMC
        Repeat
    MStep( j )
UpdateMC(P^1..P^r,MC)
            If MC&EM≠0 Then MReport(j,MC)
j←j+1
Until MC=initMC or j>n
        End If
End While

UpdateMC(P^1..P^r, MC)    //模式长度不同情况
HNX←0,HPX←0
For q∈1..r Do
HNX←HNX | ((HN&(1<<(MX[q] -1)))>>( m_q -1))
HPX←HPX | ((HP&(1<<(MX[q] -1)))>>( m_q -1))
    End For
MC←MC+HNX-HPX
MReport(j,MC)
For q∈1..r Do
If (MC>>(MX[q] -m_q))&(1<<(m_q -1))≠0 Then P^q满足匹配(结束位置是T_j )
```

函数 MStep()具体代码参看文献[3]。

2.7 MBPM-BM 复杂性分析

为分析方便，假设所有模式长度为 m，且满足

$$\sum_{q=1}^{r} m_q \le w$$

其中，rm≤w。显然空间复杂性为 O(σ+rm)。

时间复杂度从最佳和最坏两个方面进行考虑。

预处理阶段时间复杂度为 O(σ+r+rm)。过滤匹配阶段，考虑最坏情况，每次需比较至过滤窗口最左端，则过滤匹配阶段时间复杂度为 O(nm)。最坏情况下总时间复杂度为 O(σ+r+rm+nm)，考虑通常情况下 n 要远大于 r、m 长度，约等于 O(σ+nm)。考虑最佳情况，假设每次只需比较窗口最右边 k+1 个字符就可移过当前过滤窗口，则过滤匹配阶段时间复杂度为 O(kn/m)，最佳情况下总时间复杂度为 O(σ+r+rm+kn/m)，考虑到 n 远大于 r、m，约等于 O(σ+kn/m)。

若不满足条件 rm≤w，就需对 r 个模式进行分组，则每组可容纳⌊w/m⌋个模式，即 r 个模式分成⌈r/⌊w/m⌋⌉组。则空间复杂性不变，仍为 O(σ+rm)。最坏情况下总时间复杂度约等于 O(σ+⌈r/⌊w/m⌋⌉nm)，最佳情况下总时间复杂度约等于 O(σ+⌈r/⌊w/m⌋⌉kn/m)。

3 对比实验结果

选择基于Counting过滤技术的多模式算法MCounting[1]与本算法和MBPM-BM在过滤阶段处理思路较接近。MBPM-BM采用从右向左扫描的BM策略，而MCounting则选择从左向右扫描的思路，类似于KMP算法。MBPM[3]使该算法时间复杂度最小（非过滤方式下）。字符集σ=26、σ=65 536分别表示不区分大小写的英文字符集和汉语等大字符集语言。将图书《浙江简志之九（浙江地名简志）》进行OCR处理后的文本作为大字符集语言测试源，而英文文本则由电脑随机生成。实验在P4 2.4GHz/512MB/Win2000 的个人计算机上实现。3 种算法进行多模式近似匹配的平均时间如图 2 所示。

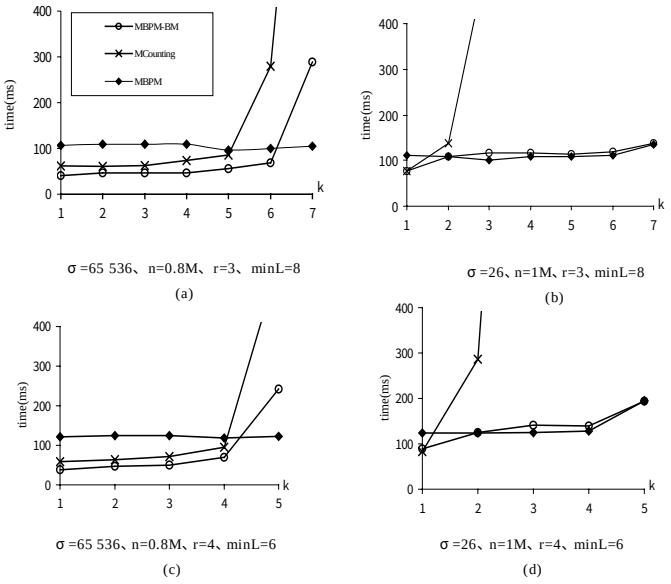

图 2 算法进行多模式近似匹配的平均时间

在图 2 中，σ=65 536 情况下，MBPM-BM 总体上明显占优，除 k=minL-1 的极端情况外，其他均占优势，即使 k 增加，时间增加依然非常缓慢。位居其次是 MCounting，最后是 MBPM。因为实际 k 几乎不可能接近 m，所以 MBPM-BM 在大字符集环境下性能非常优秀。在σ=26 的情况下，MBPM 总

(下转第 58 页)