Journal of Computer Applications
计算机应用,2012,32(11):3082-3084,3088
ISSN 1001-9081
CODEN JYIIDU
2012-11-01
http://www.joca.cn

文章编号:1001-9081(2012)11-3082-03
doi:10.3724/SP.J.1087.2012.03082

Sunday 算法效率分析

潘冠桦*,张兴忠
(太原理工大学 计算机科学与技术学院,太原 030024)
(*通信作者电子邮箱 P130979@sina.com)

摘　要:针对 Sunday 算法的过程比较复杂,难以构建马尔可夫链的问题,提出一种新的根据算法的匹配次数差求平均效率的方法。首先选定初等算法作为效率分析的基准算法,使用马尔可夫链得出初等算法比较精确的平均效率估计公式;然后根据相应的概率公式计算出初等算法和 Sunday 算法匹配过程的差值;将两者结合,得出 Sunday 算法平均效率估计公式。实验结果表明,由此公式计算的估计值可以代表实际匹配次数的平均值。

关键词:Sunday 算法;算法效率;马尔可夫链;初等算法;平均匹配次数
中图分类号:TP301.6　　**文献标志码**:A

Study on efficiency of Sunday algorithm

PAN Guan-hua*, ZHANG Xing-zhong
(*College of Computer Science and Technology, Taiyuan university of Technology, Taiyuan Shanxi 030024, China*)

Abstract: Given the characteristic that the Sunday algorithm is too complex to construct Markov chains, a new method according to the difference of matching number in the algorithms was proposed to compute the average efficiency. Firstly, the naive algorithm was chosen as the foundation, and its accurate average efficiency was computed by Markov chains. Secondly, the difference of the two algorithms was computed by the corresponding knowledge in probability theory. The two results were combined to get the equation which represented the average efficiency of Sunday algorithm. The experimental results show that the estimated value computed by the equation is the average number of the matching times.

Key words: Sunday algorithm; algorithm efficiency; Markov chain; naive algorithm; average matching number

0　引言

字符串匹配在防火墙技术、入侵检测技术、文档编辑、内容管理、计算机病毒检测等多个应用领域中都有重要应用。单字符串匹配的算法分为基于前缀搜索方法、基于后缀的搜索方法和基于字串的搜索方法。其中,基于前缀搜索的方法有 KMP(Knuth-Morris-Pratt)算法、Shit-And/Shift-Or 算法等;基于后缀搜索的方法有 BM(Boyer-Moore)算法、Horspool 算法等;基于字串搜索的方法有 BDM(Backward Dawg Matching)算法、BNDM(Backward Nondeterministic Dawg Matching)算法和 BOM(Backward Oracle Matching)算法等[1]。

有大量针对算法效率改进的研究,也有很多文献对算法的效率进行了分析,为以后的算法改进工作提供鉴定标准。文献[2-3]分别针对模式串和文本使用马尔可夫链对初等算法和 KMP 算法的平均效率进行了计算,并对结果进行了分析,最后都得到独立同分布条件下的 KMP 和初等算法的效率之比,且两者结论一致;文献[4]计算出了当文本随机时 KMP 算法和 BMH 算法平均值的上下界;文献[5]使用中心极限定理计算出 BM 算法在随机文本情况下查找给定模式串的匹配次数;文献[6]对 BNDM 算法的参数进行了改进,使得算法更加高效;文献[7]对初等算法、KMP 算法和 BM 算法等字符串匹配算法地匹配效率进行了综述;文献[8]提出了一种能够简单高效的求出和文本串具有双射关系的模式串的算法;文献[9]提出了一种计算两个字符串参数化匹配的算法;文献[10]提出了三种精确的多模式字符串匹配的算法;文献[11]提出了一种能够分析基于子串匹配的算法平均复杂度的框架;文献[12]提出了一种高效,通用,可扩展的多模式匹配技术;文献[13]提出了一种在线性时间内完成匹配的树形模型;文献[14]提出了一种 Horspool 算法的改进算法,使用相异符号的概率加速匹配。

1　相关内容

1.1　Sunday 算法

Sunday 算法是 Sunday[15] 于 1990 年提出的一种比 BM 算法搜索速度更快的算法。其核心思想是:在匹配过程中,模式串并不被要求一定要从左向右进行比较还是从右向左进行匹配,它在发现不匹配时,算法能跳过尽可能多的字符以进行下一步的匹配,从而提高了匹配效率。Sunday 算法思想跟 BM 算法很相似,在匹配失败时关注的是文本串中参加匹配的最末位字符的下一位字符。如果该字符没有在匹配串中出现则直接跳过,即移动步长=匹配串长度+1;否则,同 BM 算法一样其移动步长=匹配串中最右端的该字符到末尾的距离+1。所提到的匹配串中最右端的该字符到末尾的距离也就是算法中该字符在 next 数组中对应的值。next 数组存储了模式串中出现的每一个字符,并为其赋值,这个值等于该字符在匹配串中出现的最右端的位置到串尾的距离,对于没有在模式串中出现的字符,其对应的 next 数组中的值等于模式串的长度。Sunday 算法在最坏情况下的计算复杂度为 $O(mn)$。

1.2　马尔可夫链

马尔可夫过程是在已知时刻 t_0 系统所处状态的条件下,在 t_0 时刻以后系统到达的情况与时刻 t_0 以前系统所处的状态无关,完全取决于时刻 t_0 系统所处的状态。这个特性称为无后

收稿日期:2012-05-03;修回日期:2012-06-22。

作者简介:潘冠桦(1987-),男,山西运城人,硕士,主要研究方向:嵌入式系统、字符串匹配;　张兴忠(1964-),男,山西汾阳人,副教授,主要研究方向:嵌入式系统、软件工程。

效性,也称为“马尔可夫性”。

马尔可夫链是一个随机过程. 在任一时刻处于某个有限状态集 $S = \{S_1, S_2, \cdots, S_r\}$ 的随机过程被称作有限马尔可夫链。

定义 1　一个状态如果到达后不会再离开被称作是稳定的。

定义 2　马尔可夫链是稳定的当且仅当它至少包含一个稳定状态而且从任一个非稳定状态出发都会最终达到稳定状态。

定理 1　矩阵 $\boldsymbol{F}$ 的元素 f_{ij},等于从状态 s_i 出发到达稳定时经过 s_j 的次数的期望值。

定理 2　马尔可夫过程中,从非稳定状态出发,到达稳定状态时的步数期望值等于矩阵 $\boldsymbol{F}$ 第 i 行元素的和。

2　Sunday 算法的效率分析

因为使用马尔可夫链直接对 Sunday 算法构建状态机比较复杂,本文根据算法之间的关系,通过分析 Sunday 算法和初等算法的匹配过程,根据减少的匹配次数差得出 Sunday 算法的平均匹配效率。

为使用马尔可夫链对算法进行分析。首先进行如下定义:模式串长为 m,$P:p_1p_2\cdots p_m$,$p_i \in c(1 \leqslant i \leqslant m)$;文本串长度为 n。文本串是由 c 个字符的字母表生成的随机序列,而模式串是由 $w(w \leqslant c)$ 个字符的字母表生成的随机序列。

Sunday 算法与初等算法相比,减少了下列两种情况下的匹配次数:

1) 模式串尾字符所对应文本的字符的下一个字符属于模式串时减少的平均匹配次数等于 S_1;

2) 尾字符所对应文本的字符的下一个字符不属于模式串时减少的平均匹配次数等于 S_2。

设定初等算法的平均匹配次数等于 I,Sunday 算法的平均匹配次数为 $I - S_1 - S_2$。

设定 p_i 匹配 t_j 的概率为 a,不匹配的概率为 b,在文本和模式串均是随机字符的情况下 $a = 1/c, b = 1 - 1/c$。

2.1　初等算法的平均匹配次数

最早的初等算法查找字符串平均次数的证明由 Barth 于 1984 年提出,构建模式串的马尔可夫链,假定状态间转移的可能性仅由上一个状态决定,其结果为:

$$I = \sum_{j=1}^{m} f_{ij} = \frac{c^{m+1}}{c-1} - \frac{c}{c-1} \tag{1}$$

随后在文献[4]中 Ricardo 等证明平均每个字符的匹配次数为:

$$\overline{c_m} = \sum_{i=0}^{m-1} \frac{1}{c^i} = \frac{c}{c-1}\left(1 - \frac{1}{c^m}\right) \tag{2}$$

王洪波使用文本串构建的马尔可夫链得出的结果为:

$$\frac{\overline{c_{NAIVE}}}{n} = \left(1 - \frac{1}{c-1} + \frac{1}{c-1}\cdot\frac{1}{c^{m-1}} + \frac{m-1}{c^m}\right)^{-1} \approx \left(1 - \frac{1}{c-1}\right)^{-1} \tag{3}$$

使用概率证明:

设每次比较过程中,模式串比较到第 γ 个字符时,发现不匹配。

$$E(\gamma) = \frac{1 - (m+2)a^{m+1} + (m+1)a^{m+2}}{1-a} \approx \frac{1}{1-a}; a \to 0$$

共匹配 $n - m + 1$ 次,由此可得初等算法的平均匹配次数为:

$$\frac{1}{1-a}(n - m + 1) \tag{4}$$

式(4) 结果和式(2) 接近,但此结果和实验结果相比仍存在误差。

使用马尔可夫链证明:由于 Barth 在构建马尔可夫链的状态转换图时,没有将匹配次数和文本串长联系起来,因此式(1) 的偏差较大;而王洪波对文本串构建马尔可夫链的计算过程过于复杂。

本文对模式串构建了初等算法的马尔可夫链的状态转换图,如图 1 所示,它没有吸收态。因此,当 n 较大时每个状态的转换概率都会收敛为一个稳定值,而且 $\overline{C_n}$ 的下界为 n。

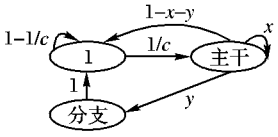

图 1　初等算法状态转换

主干状态表示正常的匹配过程,分支表示发现不匹配的字符。P_{outside} 表示访问分支状态概率的稳定值。每次位于分支状态时,都会进行一次多余的匹配,因此初等算法的平均匹配次数为:

$$\frac{\overline{c_n}}{n} = \frac{1}{1 - P_{\text{outside}}} + O(1/n)$$

由图 1 构建的马尔可夫链可以得到:

$$P_{\text{outside}} = \frac{y}{c + 1 + y - xc}$$

由于 $0 \leqslant y \leqslant 1 - 1/c, 0 \leqslant x \leqslant 1/c$ 因此有:

$$0 \leqslant P_{\text{outside}} \leqslant \frac{c-1}{c^2 + c - 1}$$

可得初等算法的平均匹配次数为:

$$\frac{\overline{c_n}}{n} \leqslant \left(1 - \frac{c-1}{c^2 + c - 1}\right)^{-1} \tag{5}$$

由图 2 可知,式(2)、(4) 的估计值和实际仍存在较大误差,而式(3)、(5) 可以得到比较精确的估计结果。

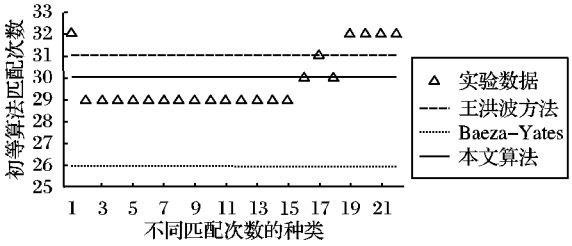

图 2　$m = 5, c = 13$ 时的实验数据和 3 种算法估计值

2.2　两种算法的匹配差

设在移动过程中恰好位于文本对应模式串尾字符位置的下个字符的个数为 μ,如图 3 所示 t_{i+m} 的位置,设此类字符中属于模式串的字符数目为 α,不属于模式串的个数为 β。

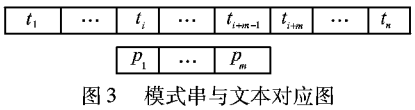

图 3　模式串与文本对应图

$$S_1 = E(\gamma)\cdot\alpha\cdot\left(\sum_{u=0}^{m-1} u\cdot\frac{1}{c}\right) \approx \frac{(m-1)\cdot m}{2(c-1)}\cdot\alpha;$$

$$0 \leqslant \alpha \leqslant n - m$$

因为文本是随机字符,且独立同分布,符合二项分布特性。

$$\alpha \to c_\mu^\gamma\left(\frac{w}{c}\right)\left(1 - \frac{w}{c}\right)^{\mu-\gamma}; E(\alpha) = \mu\cdot\frac{w}{c} \tag{6}$$

当 $c = w$ 时,即文本中不出现不属于模式串的字符,位于 t_{i+m} 处的字符均属于模式串。Sunday 算法的平均匹配次数为:

$$I - S_1 = n\cdot\left(1 - \frac{c-1}{c^2 + c - 1}\right)^{-1} - \frac{\mu}{2(c-1)}\left((m-1)\cdot m\cdot\right.$$

$$\left[1-(m+2)\left(\frac{1}{c}\right)^{m+1}+(m+1)\left(\frac{1}{c}\right)^{m+2}\right]\approx$$

$$n\cdot\left(1-\frac{c-1}{c^2+c-1}\right)^{-1}-\frac{(m-1)\cdot m}{2(c-1)}\cdot\mu$$

当 $w<c$ 时，文本中存在不属于模式串的字符，即文本中位于 t_{i+m} 处的字符可能存在不属于模式串的字符。模式串将跳过不属于模式串的字符，直到字符属于模式串，进行匹配，设此类字符的数目为 β，有 $0\leqslant\beta\leqslant n-m$。

若文本对应模式串尾字符位置后存在连续 X 个不属于模式串的字符，则每次不匹配时模式串减少的平均匹配次数为：

$$\frac{1}{c}\sum_{i=0}^{m-1}(i+x)$$

由上式可得此种情况下减少的总的匹配次数为：

$$S_2=E(\gamma)\cdot\beta\cdot\frac{1}{c}\cdot\sum_{i=0}^{m-1}(i+x)\approx\frac{m(m+2x-1)}{2(c-1)}\cdot\beta;\quad 0\leqslant\beta\leqslant n-m$$

因为文本是随机字符，且独立同分布，符合二项分布特性。

$$\beta\rightarrow c_\mu^\gamma\left(1-\frac{w}{c}\right)^\gamma\left(\frac{w}{c}\right)^{\mu-\gamma};E(\beta)=\mu\cdot\left(1-\frac{w}{c}\right)\qquad(7)$$

若文本对应模式串尾字符位置后有连续 X 个不属于模式串的字符，平均每次的跳跃距离：

$$d=\left(\frac{1+m}{2}\right)\cdot m\cdot\frac{1}{c}+\left(\frac{m+2x+1}{2}\right)\cdot m\cdot\frac{1}{w}\left(1-\frac{w}{c}\right)^x$$

X 的期望为：

$$E(X)=\sum_{l=1}^{n}l\cdot\frac{1}{w}\cdot\left(1-\frac{w}{c}\right)^i\cong\frac{c^2}{w^3}-\frac{c}{w^2};(n\rightarrow\infty)$$

在最差情况下，即文本不包含模式串或者模式串位于文本末尾，此时有 $E(\mu)=\frac{n-m}{d}$，再由式(6)、(7)即可求得 $E(\alpha)$，$E(\beta)$。算法的平均匹配次数为：

$$I-S_1-S_2\approx n\cdot\left(1-\frac{c-1}{c^2+c-1}\right)^{-1}-\frac{(m-1)\cdot m}{2(c-1)}\cdot\alpha-\frac{(m+1)m}{2(c-1)}\cdot\beta$$

$$E(\alpha)=\mu\cdot\frac{w}{c}$$

$$E(\beta)=\mu\left(1-\frac{w}{c}\right)$$

$$E(\alpha)+E(\beta)=1\qquad(8)$$

此处求出的 Sunday 算法的平均匹配效率是在文本不包含模式串或者模式串位于文本末尾时，即在最差情况下的平均匹配次数。

2.3 实验及分析

为使实验文本串的大小和字母表容量严格相同，并在最差情况下匹配成功。本文将文本串的字符随机交换位置后形成新的文本，将新文本和模式串进行匹配，如此不断循环，直到不产生新的匹配次数。最后删去匹配次数相同和不在最差情况下匹配的文本串。将满足条件的文本串的匹配次数作为纵坐标，实验结果和理论估计值比较如图 4 所示。

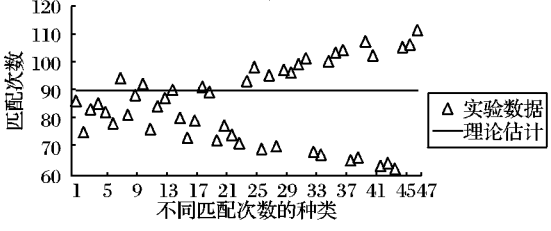

图 4 估计值和实验数据图

当 $m=w=5$，$n=105$，$c=25$ 时，由式(8)可得理论估计值为 90。

引理 1 求当 $x=1$，即不属于模式串的字符在文本中不连续时的算法效率。

每次的跳跃距离为：

$$d=\frac{m}{2}\left[\frac{1+m}{c}+\frac{3+m}{w}\left(1-\frac{w}{c}\right)\right]$$

在最差情况下的平均匹配次数：

$$E(\mu)=\frac{2(n-m)\cdot c}{m\left[1+m+\frac{m+3}{w}(c-w)\right]}$$

$$E(\alpha)=\frac{2(n-m)\cdot w}{m\left[1+m+\frac{m+3}{w}(c-w)\right]}$$

$$E(\beta)=\frac{2(n-m)\cdot(c-w)}{m\left[1+m+\frac{m+3}{w}(c-w)\right]}$$

将以上结果代入式(8)，即可求得。

推论 1 Sunday 算法在最差情况下的匹配效率等于初等算法的效率。

证明 在最差情况下，$w=c$，即 c 不包含不属于 w 的字符，文本中不含有不属于模式串的字符，此时 $\beta=0$，$S_2=0$，且文本中尾字符所对应字符的下一个字符和模式串尾字符相等，即 $S_1=m\cdot\alpha\cdot 0$，此时 Sunday 算法不会进行跳跃。$I-S_1-S_2=I_n$。

Sunday 算法毫无疑问在一般情况下，比初等算法的效率要高很多。

推论 2 Sunday 算法与初等算法匹配相比较所减少的平均匹配次数为 S_1+S_2。

证明 将 Sunday 算法和初等算法的匹配次数的差值作为纵坐标。实验结果如图 5 所示，当 $m=w=5$，$n=28$，$c=13$，$x\approx 1$ 时，计算得出 $S_1\approx 3$，$S_2\approx 5$，$S_1+S_2=8$。

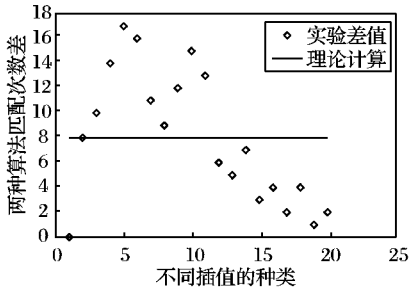

图 5 对比图 1

当 $m=w=5$，$n=52$，$c=16$，$x\approx 2$ 时，求得 $S_1+S_2=16$。实验结果如图 6 所示。

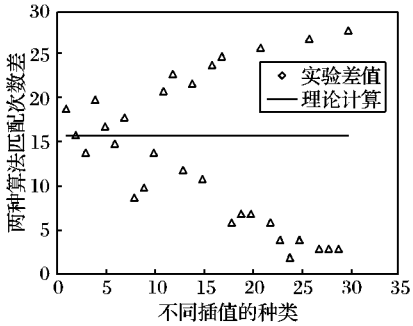

图 6 对比图 2

图 5、6 表明代表两种算法理论计算的平均匹配次数差值的直线穿过了实际实验数据差值点的中部，即 S_1+S_2 为两种算法的平均匹配次数差。

（下转第 3088 页）

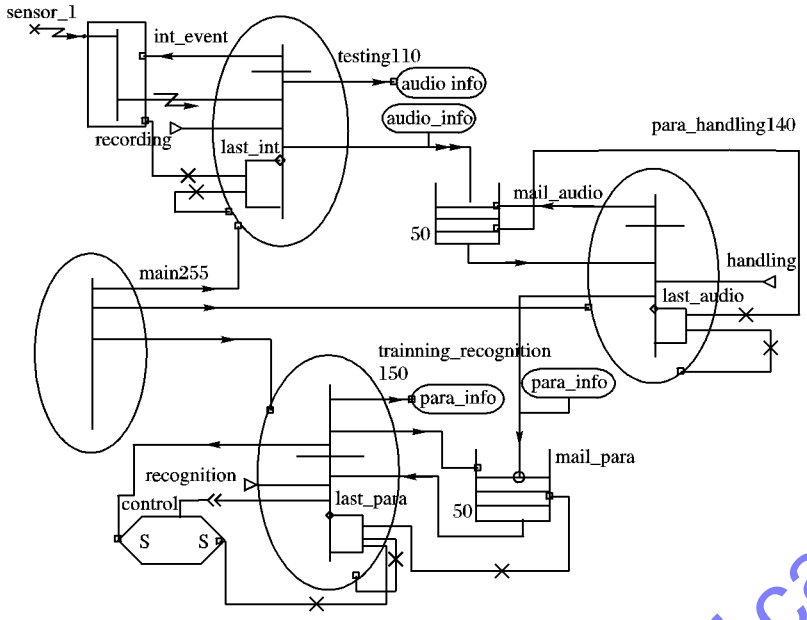

图 7 嵌入式语音控制系统源模型

[5] 黄罡, 王千祥, 曹东刚, 等. PKUAS: 一种面向领域的构件运行支撑平台[J]. 电子学报, 2002, 30(12A): 1938 – 1942.

[6] 胡博. 基于模型驱动的建模环境——Smart Desginer3. 5[D]. 杭州: 浙江大学, 2008.

[7] 徐唤杰. 面向特定领域的可视化建模技术研究[D]. 哈尔滨: 哈尔滨工程大学, 2009.

[8] 王彬. 基于实时语义模型的模型转换及语义一致性研究[D]. 昆明: 昆明理工大学, 2011.

[9] 何啸, 麻志毅, 邵维忠. 一种面向图形化建模语言表示法的元模型[J]. 软件学报, 2008, 19(8): 1867 – 1880.

[10] 赵海滨, 王宏. 嵌入式语音识别控制机械手的系统设计[J]. 仪器仪表学报, 2004, 25(4): 621 – 622.

[11] 王文慧. 基于 ARM 的嵌入式语音识别系统研究[D]. 天津: 天津大学, 2008.

[12] WANG BIN, ZHANG YUNSHENG, XIONG XIN, *et al*. A semantic-based model-driven design method for real-time control software [C]// Proceedings of the 29th Chinese Control Conference. Washington, DC: IEEE Computer Society, 2010: 4257 – 4262.

[13] WANG BIN, ZHANG YUNSHENG, XIANG FENGHONG, *et al*. Reconfigurable real-time design with port-based actor [C]// The 2nd International Conference on Computer Engineering and Technology. Washington, DC: IEEE Computer Society, 2010, 4: 242 – 245.

[14] 王彬, 张云生, 王剑平, 等. 基于 WSAN 的 AGVS 控制系统模型研究[J]. 计算机工程与应用, 2010, 46(11): 18 – 21.

[15] 王彬, 张云生, 熊新, 等. 工业硬实时嵌入式控制软件设计中的时间触发构架[C]// 第二十六届中国控制会议论文集. 北京: 北京航空航天大学出版社, 2007.

(上接第 3084 页)

3 结语

本文在使用马尔可夫链计算初等算法平均匹配次数的基础上, 对 Sunday 算法的平均匹配次数进行了计算, 实验结果证实式(8)可以用来估计 Sunday 算法的平均匹配效率。

参考文献:

[1] NAVARRO G, RAFFINOT M. 柔性字符串匹配[M]. 中国科学院计算所网络信息安全研究组, 译. 北京: 电子工业出版社, 2007.

[2] 王洪波. 基于马尔可夫链的算法复杂度分析[D]. 大连: 大连理工大学, 2007.

[3] BARTH G. An analytical comparison of two string searching algorithms[J]. Information Processing Letters, 1984, 18(5): 249 – 256.

[4] RICARDO A, BAEZA-YATE S. String searching algorithms revisited [C]// Proceedings of the Workshop on Algorithms and Data Structure. London: Springer-Verlag, 1989: 75 – 96.

[5] TSUNG-HSI T. Average case analysis of the Boyer-Moore algorithm [J]. Random Structures & Algorithms, 2006, 28(4): 481 – 498.

[6] BRURIAN B, HOLUB J, PELTOLA H. Improving practical exact string matching[J]. Information Processing Letters, 2010, 110(4): 148 – 152.

[7] BAEZA-YATES R A. Algorithms for string searching: a survey[J]. ACM SIGIR Forum, 1989, 23(3/4): 34 – 58.

[8] FREDRIKSSON K, MOZGOVOY M. Efficient parameterized string matching[J]. Information Processing Letters, 2006, 100(3): 91 – 96.

[9] SALMELA L, TARHIO J. Fast parameterized matching with Q-grams [J]. Journal of Discrete Algorithms, 2008, 6(3): 408 – 419.

[10] SALMELA L, TARHIO J, KYTÖJOKI J. Multipattern string matching with q-grams[J]. Journal of Experimental Algorithmics, 2006, 11: 1 – 19.

[11] SALMELA L. Average complexity of backward q-gram string matching algorithms [J]. Information Processing Letters, 2012, 112 (11): 433 – 437.

[12] RAMAKRISHNAN K, NIKHIL T, JIGNESH M. SigMatch: fast and scalable multi-pattern matching[J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 1173 – 1184.

[13] FOGLA P, LEE W. q-Gram matching using tree models[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(4): 433 – 447.

[14] MARKUS E. Nebel, fast string matching by using probabilities: on an optimal mismatch variant of Horspool's algorithm[J]. Theoretical Computer Science, 2006, 359(1): 329 – 343.

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