Journal of Computer Applications
计算机应用,2013,33(8):2379 - 2382
ISSN 1001-9081
CODEN JYIIDU
2013-08-01
http://www.joca.cn

文章编号:1001-9081(2013)08-2379-04
doi:10.11772/j.issn.1001-9081.2013.08.2379

# BM 算法中函数 shift 的研究

韩光辉[1*],曾　诚[2,3]
(1.武汉商业服务学院 信息工程系,武汉 430056;　2.湖北大学 数学与计算机科学学院,武汉 430062;
3.软件工程国家重点实验室(武汉大学),武汉 430072)
(*通信作者电子邮箱 hgh.mail@163.com)

**摘　要**:建立 BM 算法中函数 shift 及其构造算法的严格的形式理论,对于 BM 算法及其各种变形的研究与改进是十分必要的。给出了 shift 的一个清晰的形式定义,引入模式串后缀的特征集及其最小值函数,通过特征集描述了 shift 的构造,从而严格建立了 shift 及其构造算法的理论基础。根据 shift 的构造定理与最小值函数的迭代计算方法,给出了 shift 的一个新的构造算法,证明了该算法具有线性的时间与空间复杂度。理论分析和计算结果表明,该算法比已有算法更简单,计算复杂度更低,因而更适合硬件实现。

**关键词**:串匹配;BM 算法;好后缀规则;shift 函数;复杂度分析
**中图分类号**:TP301.6　　**文献标志码**:A

## Research on function shift in Boyer-Moore algorithm

HAN Guanghui[1*], ZENG Cheng[2,3]
(1. *Information Engineering Department, Wuhan Commercial Service College, Wuhan Hubei* 430056, *China*;
2. *College of Mathematics and Computer Science, Hubei University, Wuhan Hubei* 430062, *China*;
3. *State Key Laboratory of Software Engineering (Wuhan University), Wuhan Hubei* 430072, *China*)

**Abstract:** For the research and improvement of Boyer-Moore (BM) algorithm and its variants, it is very necessary to establish strict formal theory of the function shift in Boyer-Moore's and its construction algorithm. A clear formal definition of shift was given. Then, characteristic sets of the pattern suffixes and its minimum value function were introduced, and the construction of shift was described by the characteristic sets, thus theoretical basis of shift and its construction algorithm were strictly established. Finally, a new construction algorithm of shift was presented based on the construction theorem of shift and iterative computing method of the minimum value function. It is proved that the algorithm has linear time and space complexity. The theoretical analysis and computing results show that the algorithm is simpler, and its complexity of computation is lower, so it is more suitable for hardware implementation compared with several existing algorithms.

**Key words:** string matching; Boyer-Moore (BM) algorithm; good-suffix rule; function shift; complexity analysis



## 0　引言

串匹配是指在文本串中查找模式串的第一次出现或所有出现。串匹配算法在文本检索、语言翻译、数据压缩、搜索引擎等应用中起着关键作用。近年来,在病毒检测、网络入侵检测、网络协议识别、计算生物等领域也都大量应用了串匹配技术。因此,串匹配算法一直是计算机科学领域的研究焦点之一。

根据扫描策略的不同,已有的精确串匹配算法大致可以分为三类:1)自左向右匹配窗口中文本串与模式串的最大前缀,具有线性的最差时间复杂度;2)自右向左匹配窗口中文本串与模式串的最大后缀,具有亚线性的平均时间复杂度;3)自右向左匹配窗口中文本串与模式串的最大子串,也具有亚线性的平均时间复杂度,甚至能达到最优平均时间复杂度。

BM(Boyer-Moore)算法[1]是最早的第二类算法。在 BM 算法的各种变形[2-5]中,Horspool 算法、Sunday 算法是 BM 算法的简化;Commentz-Walter 算法是 BM 算法在多模式匹配的推广,Set-Horspool 算法是 Horspool 算法在多模式匹配的推广;Wu-Manber 算法则是 Set-Horspool 算法的一个改进。

BM 算法及其各种变形一直被认为是实际应用中最有效的串匹配算法。例如,开源入侵检测系统 Snort 就是采用 BM 算法,在监听到的数据中匹配系统的安全规则,根据发现的入侵情况采取相应的安全措施。

BM 算法通过两个预先计算好的函数来确定下一个窗口的起始位置,本文中,这两个函数分别记为 skip 与 shift。其中,skip 十分简洁且容易计算,在大字母表、小模式串的情形具有很高的效率;然而,shift 的计算却远比 skip 困难得多。1977 年,Boyer 与 Moore 在 1975 年最初定义的基础上,给出了 shift 的一个改进了的定义,但没有给出构造算法。同年,Knuth 等[6]也给出了一个定义并描述了构造算法,但没有详述计算方法,其算法也不完全正确。1980 年,Rytter[7]完善了 Knuth 等的构造算法,但没有给出计算方法的严格的形式证明。因此,Hume 等[8]指出 shift 的构造算法难以理解,Horspool 甚至认为 shift 并不真正实用,这也正是 Horspool 算法中仅保留并改进了 skip 而去掉了 shift 的原因。近年来,对 BM 算法的各种改进也主要是针对 skip 的改进[9-11]。

其实,shift 保证了 BM 算法在最坏情况下是线性时间复杂度[6,12-13],这在小字母表情形(例如基因序列匹配和蛋白

**收稿日期**:2013-02-24;**修回日期**:2013-04-08。
**基金项目**:国家自然科学基金资助项目(60903034,61100018,61100025,61100026);湖北省自然科学基金资助项目(2011CDB069)。
**作者简介**:韩光辉(1956 - ),男,湖北武汉人,副教授,硕士,主要研究方向:计算理论、软件形式化;　曾诚(1976 - ),男,湖北武汉人,副教授,博士,主要研究方向:网络化软件工程。

质序列匹配)以及模式串与文本串相似度较大的情形[14]是非常重要的。随着串匹配技术应用领域的不断扩大和理论研究的不断深入,研究人员逐渐开始关注串匹配算法应用环境因素及其之间的关系,尤其是模式串与文本串之间的关系;在网络内容安全方面,也开始关注串匹配算法的抗攻击性能。现在看来,仅仅根据传统文本检索的实验结果,并不足以低估 shift 的作用。所以,建立 shift 及其构造算法的严格、清晰的形式理论,对于 BM 算法及其各种变形的研究与改进是十分必要的。

2009 年,Yang[15]试图通过一个新的辅助数组 suffixLength 计算 shift 并给出了形式证明,但辅助数组 suffixLength 既不如 Knuth 所引入的辅助数组直观,而且证明过程也非常冗长复杂。

本文给出了 shift 及其计算方法的严格、清晰的形式表述和数学证明,并在此基础上给出了 shift 的一个新的构造算法。

## 1 BM 算法的原理

设 $\Sigma$ 是一个字母表,$T,P\in\Sigma^+$,其中,文本串 $T=t_1t_2\cdots t_n$,模式串 $P=p_1p_2\cdots p_m$。

BM 算法的基本思想是:自右向左逐个比较文本字符与模式字符,若 $t_{i+1}\cdots t_{i+m-j}=p_{j+1}\cdots p_m(t_i\neq p_j)$,则下一轮比较从 $(t_{i+\Delta_{ij}},p_m)$ 开始,其中 $\Delta_{ij}=m-j+\delta_{ij}(\delta_{ij}>0)$。

确定增量 $\Delta_{ij}$ 的要素有两个,一个是文本字符在 $P$ 中的出现,另一个是 $P$ 的已匹配后缀。

根据文本字符在 $P$ 中的出现确定 $\Delta_{ij}$ 的思想,是由 Boyer 与 Moore 于 1974 年春提出的,Gosper 也独立地提出了这一思想。按照这一思想确定 $\Delta_{ij}$ 的方法,后来常称为坏字符规则(bad-character rule),其形式化是函数 $skip:\Sigma\to\{0,1,\cdots,m\}$。

**定义 1** (BM)函数 skip 定义如下:

$$skip(c)=\begin{cases}m-\max\{k\mid p_k=c\}, & c\in\{p_1,p_2,\cdots,p_m\}\\ m, & c\notin\{p_1,p_2,\cdots,p_m\}\end{cases};\quad c\in\Sigma$$

根据 $P$ 的已匹配后缀确定 $\Delta_{ij}$ 的最初方法,是由 Boyer 与 Moore 于 1975 年夏提出的,同年,Knipers 与 Knuth 都对其最初方法提出了改进建议。这一方法后来常称为好后缀规则(good-suffix rule),其形式化便是 $\{1,2,\cdots,m\}$ 上的函数 shift。

**定义 2** (BM)函数 shift 定义如下:

$$shift(j)=m+1-rpr(j);\ j=1,2,\cdots,m$$

其中:

$$rpr(j)=\max\{k\mid k\leqslant m\wedge p_k\cdots p_{k+m-j-1}=p_{j+1}\cdots p_m\wedge(k\leqslant 1\vee p_{k-1}\neq p_j)\}$$

**定义 3** (Knuth)函数 shift 定义如下:

$$shift(j)=\min\{s+m-j\mid s\geqslant 1\wedge(s\geqslant j\vee p_{j-s}\neq p_j)\wedge((s\geqslant i\vee p_{i-s}=p_i),j<i\leqslant m)\};\quad j=1,2,\cdots,m$$

增量 $\Delta_{ij}$ 由 $skip(t_i)$ 与 $shift(j)$ 共同确定,即 $\Delta_{ij}=\max\{skip(t_i),shift(j)\}$。

## 2 函数 shift 的定义

### 2.1 shift 的定义

分析定义 2 与定义 3,不难发现两个定义都存在一定的缺陷:

1)当定义中的集合为空集时,两个定义均未明确规定 shift 的函数值。

2)当 $j=m$ 时,按定义 2:

$$shift(m)=m+1-\max\{k\mid k\leqslant m\wedge(k\leqslant 1\vee p_{k-1}\neq p_m)\}$$

按定义 3:

$$shift(m)=\min\{s\mid s\geqslant 1\wedge p_{m-s}\neq p_m)\}$$

直观地说,是将 $P$ 中右起第一个不等于 $p_m$ 的那个字符与 $t_i$ 对齐;然而,只须注意到 $t_i\neq p_m$,这时必有 $skip(t_i)\geqslant shift(m)$,因此,按定义计算 $shift(m)$ 是没有任何意义的。

下面给出一个更加清晰、准确的 shift 的定义。记

$$B_j=B_j^{(1)}\cup B_j^{(2)};\ j=1,2,\cdots,m-1$$

其中:

$$B_j^{(1)}=\{d\mid 0<d<j\wedge p_{j+1-d}\cdots p_{m-d}=p_{j+1}\cdots p_m\wedge p_{j-d}\neq p_j\}$$

$$B_j^{(2)}=\{k\mid j\leqslant k<m\wedge p_1\cdots p_{m-k}=p_{k+1}\cdots p_m\}$$

**定义 4** 本文的函数 shift 定义如下:

$$shift(j)=m-j+\delta(j);\ j=1,2,\cdots,m$$

其中 $\delta(j)$ 是 $P$ 的右移量,有

$$\delta(j)=\begin{cases}\min B_j, & B_j\neq\varnothing\\ m, & B_j=\varnothing\\ 1, & j=m\end{cases}$$

### 2.2 特征集 $A_s$

为了描述 shift 的构造,引入刻画 $B_j^{(1)}$ 与 $B_j^{(2)}$ 共同特征的集合 $A_s$:

$$A_s=\{k\mid s<k<m\wedge p_{s+1}\cdots p_{s+m-k}=p_{k+1}\cdots p_m\};\quad s=0,1,\cdots,m-1$$

下面的定理 1 是本文的主要结果之一,该定理通过 $A_s$ 描述了 shift 的构造。

**定理 1**

$$shift(j)=\begin{cases}m-\max\{s\mid s>0\wedge j\in A_s\wedge p_s\neq p_j\}, & B_j^{(1)}\neq\varnothing\\ m-j+\min\{k\mid k\in A_0\wedge k\geqslant j\}, & B_j^{(1)}=\varnothing\wedge B_j^{(2)}\neq\varnothing\\ 2m-j, & B_j=\varnothing\\ 1, & j=m\end{cases}$$

**证明** 易知:$B_j^{(1)}=\{j-s\mid s>\wedge j\in A_s\wedge p_s\neq p_j\}$,$B_j^{(2)}=\{k\mid k\in A_0\wedge k\geqslant j\}$,故:$\min B_j^{(1)}=j-\max\{s\mid s>0\wedge j\in A_s\wedge p_s\neq p_j\}$,$\min B_j^{(2)}=\min\{k\mid k\in A_0\wedge k\geqslant j\}$,由定义 4 便知结论成立。

## 3 特征集 $A_s$ 的性质及其最小值函数

### 3.1 $A_s$ 的性质

**引理 1** 设 $k\in A_s$,则:1)$A_k\subset A_s$;2)若 $k'\in A_s,k'>k$,则 $k'\in A_k$;3)若 $k=\min A_s$,则 $A_s=\{k\}\cup A_k$。

**证明**

1)不妨设 $A_k\neq\varnothing,k'\in A_k$,则因 $p_{k+1}\cdots p_{k+m-k'}=p_{k'+1}\cdots p_m,p_{s+1}\cdots p_{s+m-k}=p_{k+1}\cdots p_m$,故 $p_{s+1}\cdots p_{s+m-k'}=p_{k+1}\cdots p_{k+m-k'}=p_{k'+1}\cdots p_m$,于是 $k'\in A_s$,显然 $A_s-A_k\neq\varnothing$,所以 $A_k\subset A_s$。

2)由于 $p_{s+1}\cdots p_{s+m-k'}=p_{k'+1}\cdots p_m,p_{s+1}\cdots p_{s+m-k}=p_{k+1}\cdots p_m$,又因 $k'>k$,故 $p_{k+1}\cdots p_{k+m-k'}=p_{s+1}\cdots p_{s+m-k'}=p_{k'+1}\cdots p_m$,于是 $k'\in A_k$。

3)设 $k'\in A_s$,不妨设 $k'>k$,由 2)可知 $k'\in A_k$,故 $A_s\subseteq$

$\{k\} \cup A_k$;另一方面,由 1),$\{k\} \cup A_k \subseteq A_s$,所以 $A_s = \{k\} \cup A_k$。

**引理 2**　设 $A_s' = \{k \mid k \in A_s \wedge p_s = p_k\}(s = 1,2,\cdots,m-1)$,则 $A_{s-1} = \{k-1 \mid k \in A_s'\} \cup \{m-1 \mid p_s = p_m\}$。

证明　事实上,

$$\begin{aligned}k' \in A_{s-1} &\Leftrightarrow s-1 < k' < m \wedge p_s \cdots p_{s+m-k'-1} = p_{k'+1} \cdots p_m \Leftrightarrow\\ &(s-1 < k' < m-1 \wedge p_s \cdots p_{s+m-k'-1} = p_{k'+1} \cdots p_m) \vee\\ &(k' = m-1 \wedge p_s = p_m) \Leftrightarrow (k' = k-1 \wedge\\ &s < k < m \wedge p_s = p_k \wedge p_{s+1} \cdots p_{s+m-k} = p_{k+1} \cdots p_m) \vee\\ &(k' = m-1 \wedge p_s = p_m) \Leftrightarrow (k' = k-1 \wedge k \in A_s \wedge\\ &p_s = p_k) \vee (k' = m-1 \wedge p_s = p_m) \Leftrightarrow\\ &(k' = k-1 \wedge k \in A_s') \vee (k' = m-1 \wedge p_s = p_m) \Leftrightarrow\\ &k' \in \{k-1 \mid k \in A_s'\} \cup \{m-1 \mid p_s = p_m\}\end{aligned}$$

### 3.2　$A_s$ 的最小值函数

由引理 1 不难看出,可以通过 $\min A_s$ 求出 $A_s$ 的全部元素。为此,定义函数 $f:\{0,1,\cdots,m\} \to \{1,2,\cdots,m+1\}$,

$$f(s) = \begin{cases}\min A_s, & A_s \neq \varnothing\\ m, & A_s = \varnothing\\ m+1, & s = m\end{cases}$$

由 $f$ 的定义,$f(m) = m+1$,$f(m-1) = m$,并且由引理 2 易知:

$$f(s-1) = \begin{cases}\min A_s' - 1, & A_s' \neq \varnothing\\ m-1, & A_s' = \varnothing \wedge p_s = p_m\\ m, & A_s' = \varnothing \wedge p_s \neq p_m\end{cases} \quad (1)$$

下面的定理 2 是本文的另一个主要结果,该定理利用 $f$ 给出了 $A_s$ 的构造。

**定理 2**　设 $A_s \neq \varnothing$,则存在 $k_r$ 使得 $f(s) = k_1$,$f(k_1) = k_2,\cdots,f(k_r) = m$,并且 $A_s = \{k_1,k_2,\cdots,k_r\}$。

证明　因为 $k_1 = f(s) = \min A_s$,由引理 1 知 $A_s = \{k_1\} \cup A_{k_1}$。若 $A_{k_1} \neq \varnothing$,又因 $k_2 = f(k_1) = \min A_{k_1}$,故 $A_{k_1} = \{k_2\} \cup A_{k_2}$,从而 $A_s = \{k_1,k_2\} \cup A_{k_2}$。如此下去,必有 $k_r$ 使得 $A_s = \{k_1,k_2,\cdots,k_r\} \cup A_{k_r}(A_{k_r} = \varnothing)$。由 $f$ 的定义,$f(k_r) = m$,并且 $A_s = \{k_1,k_2,\cdots,k_r\}$。

由定理 2 与式(1),可以从 $f(m),\cdots,f(s)$ 递推计算 $f(s-1)$。计算方法是:迭代计算

$f(k_v) = k_{v+1}$;$v = 0,1,\cdots,r,k_0 = s,k_{r+1} = m$

1) 若 $p_s \neq p_{k_1},\cdots,p_s \neq p_{k_{v-1}},p_s = p_{k_v}(v \leqslant r)$,则 $f(s-1) = f(k_{v-1}) - 1 = k_v - 1$;

2) 若 $p_s \neq p_{k_1},\cdots,p_s \neq p_{k_r},p_s = p_{k_{r+1}}$,则 $f(s-1) = f(k_r) - 1 = k_{r+1} - 1 = m-1$;

3) 若 $p_s \neq p_{k_1},\cdots,p_s \neq p_{k_r},p_s \neq p_{k_{r+1}}$,则 $f(s-1) = f(k_{r+1}) - 1 = m+1-1 = m$。

## 4　函数 shift 的构造算法

根据定理 1 与 $f$ 的迭代计算方法,可以利用 $f$ 计算 shift。计算方法如下:

① 初始化 $shift(1) = \cdots = shift(m-1) = 0$,$shift(m) = 1$;

② 依次对 $s = m-1,m-2,\cdots,1$,迭代计算 $f(k_v) = k_{v+1}$ $(v = 0,1,\cdots,r,k_0 = s,k_{r+1} = m)$,当恰有 $p_s \neq p_{k_v}$ 且 $shift(k_v) = 0$ 时,$shift(k_v) = m-s$;

③ 如果存在 $shift(j) = 0$,迭代计算 $f(k_v) = k_{v+1}(v = 0,1,\cdots,r,k_0 = 0,k_{r+1} = m)$,当恰有 $k_{v+1} \geqslant j$ 时,$shift(j) = m - j + k_{v+1}$。

下面给出 shift 的构造算法。算法中,步骤 1) ~3)初始化 shift;步骤 4) ~10)计算 $f$ 的函数值,同时按 ② 计算 $shift(j)$;步骤 11) ~ 16) 按 ③ 计算余下的 $shift(j)$。

算法　计算 shift 函数值。

输入:模式串 $P = p_1p_2\cdots p_m$;

输出:$shift(j)(j = 1,2,\cdots,m)$。

```
1)  for j ← 1 to m − 1 do
2)    shift(j) ← 0
    end
3)  shift(m) ← 1
4)  f(m) ← m + 1, f(m − 1) ← m
5)  for s ← m − 1 to 1 do
6)    k ← f(s)
7)    while k ⩽ m and p_s ≠ p_k do
8)      if shift(k) = 0 then shift(k) ← m − s
9)        k ← f(k)
        end
10)     f(s − 1) ← k − 1
    end
11) for j ← 1 to m − 1 do
12)   if shift(j) = 0 then do
13)     k ← f(0)
14)     while k < j do
15)       k ← f(k)
        end
16)   shift(j) ← m − j + k
    end
  end
```

## 5　复杂度分析与计算结果比较

### 5.1　复杂度分析

考虑循环步骤 5) ~10),其循环次数为 $m-1$。显然,结果为 $p_s = p_k$ 的比较次数 $\leqslant m-1$。对于结果为 $p_s \neq p_k$ 的比较,观察变量 $k$:由步骤 9) 可知 $p_s \neq p_k$ 的每次比较导致 $k$ 增大;而步骤 6) 的第一次执行使得 $k$ 的初值为 $m$,其后的每次执行导致 $k$ 减小 1,步骤 6) 共执行了 $m-1$ 次,注意到 $k \leqslant m$,故 $p_s \neq p_k$ 的比较次数 $\leqslant m-1$,从而字符比较次数 $\leqslant 2(m-1)$。因此,算法的时间复杂度为 $O(m)$。

由 $f$ 的定义可知,算法的空间复杂度为 $O(m)$。

同理可证,文献[7]给出的算法中,字符比较次数 $\leqslant 2m$,最小值计算次数 $\leqslant 5m$,乘法次数为 $m$,空间复杂度为 $O(m)$。文献[15]给出的算法中,字符比较次数 $\leqslant 8m$,乘法次数为 $m-1$,空间复杂度为 $O(2m)$。

### 5.2　计算结果比较

设模式串 $P$ = abdbacbaaaa,几种算法的计算结果如表 1 所示。

计算结果的差异是:文献[7,15]给出的算法是按定义 2 或定义 3 计算得到 $shift(11) = 4$,而本文算法则是按定义 4 取 $shift(11) = 1$。

以文本串 $T$ = abdbacbaaadabdbacbacbaabdbacbaaab 为例,$T[11]$ = d ≠ a = $P[11]$,$skip$(d) = 8。按文献[7,15]给出的算法,$shift(11) = 4 < 8$,故 $\Delta_{ij} = skip$(d) = 8;按本文算法,$shift(11) = 1 < 8$,仍有 $\Delta_{ij} = skip$(d) = 8。

根据复杂度分析与计算结果对比,本文算法不仅是正确的,而且简化了计算,降低了计算复杂度。

表 1 shift 函数值

| $j$ | $P$ | skip | shift(文献[7,15]算法) | shift(本文算法) |
|---|---|---|---|---|
| 1 | a | 0 | 20 | 20 |
| 2 | b | 4 | 19 | 19 |
| 3 | d | 8 | 18 | 18 |
| 4 | b | 4 | 17 | 17 |
| 5 | a | 0 | 16 | 16 |
| 6 | c | 5 | 15 | 15 |
| 7 | b | 4 | 14 | 14 |
| 8 | a | 0 | 4 | 4 |
| 9 | a | 0 | 4 | 4 |
| 10 | a | 0 | 4 | 4 |
| 11 | a | 0 | 4 | 1 |

## 6 结语

本文首先分析了 BM 算法中函数 shift 的经典定义,给出了 shift 及其计算方法的严格、清晰的形式表述和数学证明;在此基础上,给出了 shift 的一个新的构造算法,证明了其时间复杂度与空间复杂度均为 $O(m)$。

值得注意的是,随着串匹配技术应用领域的不断扩大,基于软件的实现已经不能充分满足对匹配速度日益增长的需求,因此,基于硬件的实现逐渐出现,并成为当前的研究热点。理论分析与计算结果表明,本文给出的 shift 构造算法比已有的算法更简单,计算复杂度更低,从而更适合硬件实现。

BM 算法及其各种变形的平均时间复杂度仍然是一个尚未完全解决的问题。目前的研究结果主要集中在没有 shift 的 Horspool 算法,最近的结果有:在文本字符相互独立且同分布的假设下,Mahmoud 等[16]证明了 Horspool 算法的平均时间复杂度是渐进正态的,Smythe[17]应用 Markov 链理论也得到了这一结果。至于 BM 算法,shift 使得算法分析变得更加复杂,虽然 Boyer 与 Moore 证明了是亚线性的,但其概率模型有待商榷。因此,BM 类算法的平均时间复杂度有待进一步研究。

**参考文献:**


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

[2] HORSPOOL R N. Practical fast searching in strings [J]. Software Practice and Experience, 1980, 10(6): 501 – 506.

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

[4] COMMENTZ-WALTER B. A string matching algorithm fast on the average [C]// Proceedings of the 6th Colloquium, on Automata, Languages and Programming, LNCS 71. Berlin: Springer-Verlag, 1979: 118 – 132.

[5] WU S, MANBER U. A fast algorithm for multi-pattern searching, TR-94-17 [R]. Tucson: University of Arizona, 1994.

[6] KNUTH D E, MORRIS J H, PRATT V R. Fast pattern matching in strings [J]. SIAM Journal of Computing, 1977, 6(2): 323 – 350.

[7] RYTTER W. A correct preprocessing algorithm for Boyer-Moore string searching [J]. SIAM Journal of Computing, 1980, 9(3): 509 – 512.

[8] HUME A, SUNDAY D M. Fast string searching [J]. Software Practice and Experience, 1991, 21(11): 1221 – 1248.

[9] 张红梅, 范明钰. 模式匹配 BM 算法改进[J]. 计算机应用研究, 2009, 26(9): 3249 – 3252.

[10] 董明明, 巩青歌, 张琦. 入侵检测系统中模式匹配算法的改进[J]. 计算机应用与软件, 2011, 28(5): 272 – 274.

[11] 王浩, 张霖. 基于坏字符序检测的快速模式匹配算法[J]. 计算机应用与软件, 2012, 29(5): 114 – 116.

[12] GUIBAS L J, ODLYZKO A M. A new proof of the linearity of the Boyer-Moore string searching algorithm [J]. SIAM Journal of Computing, 1980, 9(4): 672 – 682.

[13] COLE R. Tight bounds on the complexity of the Boyer-Moore algorithm [J]. SIAM Journal of Computing, 1994, 23(5): 1075 – 1091.

[14] 刘萍, 刘燕兵, 郭莉, 等. 串匹配算法中模式串与文本之间关系的研究[J]. 软件学报, 2010, 21(7): 1503 – 1514.

[15] YANG W. On the shift-table in Boyer-Moore's string matching algorithm [J]. International Journal of Digital Content Technology and its Applications, 2009, 3(4): 10 – 20

[16] MAHMOUD H M, SMYTHE R T, REGNIER M. Analysis of Boyer-Moore-Horspool string-matching heuristic [J]. Random Structures Algorithms, 1997, 10(1/2): 169 – 186.

[17] SMYTHE R T. The Boyer-Moore-Horspool heuristic with Markovian input [J]. Random Structures Algorithms, 2001, 18(2): 153 – 163.


(上接第 2374 页)


[7] JIANG L, TAN J L, LIU Y B. ClusterFA: a memory-efficient DFA structure for network intrusion detection [C]// ASIACCS '12: Proceedings of the 7th ACM Symposium on Computer and Communications Security Information. New York: ACM, 2012: 65 – 66.

[8] LIU T W, YANG Y F, LIU Y B, *et al.* An efficient regular expressions compression algorithm from a new perspective [C]// INFOCOM 2011: Proceedings of 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies. Washington, DC: IEEE Computer and Communications Societies, 2011: 2129 – 2137.

[9] YANG Y-H E, PRASANNA V K. Space-time tradeoff in regular expression matching with semi-deterministic finite automata [C]// INFOCOM 2011: Proceedings of 30th IEEE International Conference on Computer Communications. Washington, DC: IEEE Computer and Communications Societies, 2011: 1853 – 1861.

[10] QI Y X, WANG K, FONG J, *et al.* FEACAN: front-end acceleration for content-aware network processing [C]// INFOCOM 2011: Proceedings of 30th IEEE International Conference on Computer Communications. Washington, DC: IEEE Computer and Communications Societies, 2011: 2114 – 2122.

[11] CLARK C R, SCHIMMEL D E. Scalable pattern matching for high speed networks [C]// FCCM 2004: Proceedings of 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. Piscataway: IEEE, 2004: 249 – 257.

[12] SUTTON P. Partial character decoding for improved regular expression matching in FPGAs [C]// FTP 2004: Proceedings of 2004 IEEE International Conference on Field-Programmable Technology. Piscataway: IEEE, 2004: 25 – 32.

[13] YAMAGAKI N, SIDHU R, KAMIYA S. High-speed regular expression matching engine using multi-character NFA [C]// FPL 2008: Proceedings of the 2008 International Conference on Field Programmable Logic and Applications. Piscataway: IEEE, 2008: 131 – 136.

[14] Sourcefire, SNORT [DB/OL]. (2012 – 10 – 09) [2012 – 12 – 18]. http://www.snort.org/snort-downloads.

[15] International Computer Science Institute, Bro Intrusion Detection System [DB/OL]. (2012 – 08 – 29) [2012 – 12 – 25]. http://bro-ids.org/download/index.html.