Average running time of Boyer-Moore-Horspool algorithm

R.A. Baeza-Yates, Mireille Regnier

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: (1) 39 63 55 11

Rapports de Recherche

N° 1316

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

AVERAGE RUNNING TIME OF BOYER-MOORE-HORSPOOL ALGORITHM

Ricardo A. BAEZA-YATES
Mireille RÉGNIER

Octobre 1990

RR-1316

Average Running Time of Boyer-Moore-Horspool Algorithm

Ricardo A. Baeza-Yates

Depto. de Ciencias de la Computación, Universidad de Chile
Casilla 2777, Santiago, Chile

Mireille Régnier

INRIA-78 153 Le Chesnay, France *

Abstract

We study Boyer-Moore-type string searching algorithms. First, we analyze the Horspool's variant. The searching time is linear. An exact expression of the linearity constant is derived and is proven to be asymptotically α, $\frac{1}{c} \leq \alpha \leq \frac{2}{c+1}$, where c is the cardinality of the alphabet. We exhibit a stationary process and reduce the problem to a word enumeration. The same technique applies to other variants of the Boyer-Moore algorithm.

Coût moyen de l'algorithme de Boyer-Moore-Horspool

Résumé:

Nous étudions l'algorithme de recherche de motifs de Boyer-Moore. En premier lieu, nous analysons la variante de Horspool. Le temps de recherche est linéaire en la taille du texte. Une expression exacte de la constante de linéarité est calculée et on montre qu'elle vaut asymptotiquement α, $\frac{1}{c} \leq \alpha \leq \frac{2}{c+1}$, où c est la cardinalité de l'alphabet. Nous exhibons un processus stationnaire et nous ramenons à un problème d'énumération de mots. La même technique s'applique à d'autres variantes de l'algorithme de Boyer-Moore.

*The work of the first author was partially supported by the University of Waterloo and INRIA and the second author by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

Average Running Time of the Boyer-Moore-Horspool Algorithm

Ricardo A. Baeza-Yates

Depto. de Ciencias de la Computación, Universidad de Chile
Casilla 2777, Santiago, Chile

Mireille Régnier

INRIA-78 153 Le Chesnay, France *

Abstract

We study Boyer-Moore-type string searching algorithms. First, we analyze the Horspool's variant. The searching time is linear. An exact expression of the linearity constant is derived and is proven to be asymptotically $\alpha, \frac{1}{c} \leq \alpha \leq \frac{2}{c+1}$, where c is the cardinality of the alphabet. We exhibit a stationary process and reduce the problem to a word enumeration problem. The same technique applies to other variants of the Boyer-Moore algorithm.

1 Introduction

String searching is an important component of many problems, including text editing, data retrieval and symbol manipulation. The string matching problem consists in finding one or all occurrences of a pattern in a text, where the pattern and the text are strings over some alphabet. A good parameter to evaluate the complexity of string searching algorithms is the number of text-pattern comparisons of characters. The worst case is well known for most algorithms. Notably, for the Boyer-Moore algorithm studied here, the searching time is $O(n)$, for a pattern of length m and a text of length n, $n > m$. Moreover, at least $n - m + 1$ characters must be inspected in the worst case [Riv77].

The average complexity is also of great significance [Yao79, KMP77]. It is interesting to show (when possible!) that the number of comparisons, $\overline{C}_n$, is asymptotically $K \cdot n$; to derive the linearity constant K for different string searching algorithms and then compare them. One common characteristic of these algorithms is the dependence on history: the number of comparisons made on a given character depends on the result of comparisons on its neighbours. Hence, first attempts to derive asymptotics used Markov chains [Bar84, Sch88, BY89b, BY89c]. Unfortunately, this

*The work of the first author was partially supported by the University of Waterloo and INRIA and the second author by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

quickly leads to a combinatorial explosion when the size of the pattern increases. Recently, another algebraic approach, based on pattern enumeration and combinatorics on words allowed an analysis of the Knuth-Morris-Pratt algorithm [Reg89].

In this paper, we derive the analysis of the Boyer-Moore-Horspool or BMH algorithm [Hor80]. This algorithm, described below, proceeds from right to left, a (presumably) efficient method for large alphabets. The method is rather in the same vein as [Reg89] but the dependence on history is much tighter. The originality of our approach is the immediate reduction to a stationary process. The study of this stationary process, using algebraic tools and combinatorics on words, leads to the linearity constant K. It appears to be a simple function of the cardinality c of the alphabet: $K \sim 1/c + O(1/c^2)$.

The organization of the paper is as follows. Section 2 briefly presents the BMH algorithm. In Section 3 we reduce the analysis to the study of a stationary process. Section 4 addresses the average performance; notably, the expected number of comparisons, $\overline{C}_n \sim K_c n$, is derived. Asymptotic bounds on K_c are proven, and a conjecture is stated. All these results are in good agreement with experimental values. The last section is our conclusion. In a preliminary version of this paper [BYGR90] we also studied *Boyer-Moore automata*.

2 The Boyer-Moore-Horspool Algorithm

The Boyer-Moore or BM algorithm positions the pattern over the leftmost characters in the text and attempts to match it from right to left. If no mismatch occurs, then the pattern has been found. Otherwise, the algorithm computes a *shift*, that is an amount by which the pattern is moved to the right before a new matching attempt is undertaken. This shift can be computed with two heuristics: the match heuristic and the occurrence heuristic. In this paper we only consider the second one; it consists in aligning the last mismatching character in the text with the first character of the pattern matching it. A simplification was proposed in 1980 by Horspool [Hor80]. In that paper, it is pointed out that any character from the text read since the last shift can be used for the alignment. To maximize the average shift after a mismatch, the character compared with the last character of the pattern is chosen for the alignment. Empirical results show that this simpler version is as good as the original algorithm.

The code for the Boyer-Moore-Horspool algorithm is extremely simple and is presented in Figure 1.

For convenience for further analysis, we use a pattern of length $m+1$ instead of m. The occurrence heuristic table is computed associating a shift to any character in the alphabet. Formally

$$d[x] = \min\{s | s = m+1 \text{ or } (1 \leq s \leq m \text{ and } pattern[m+1-s] = x)\} .$$

Note that $d[x]$ is $m+1$ for any character not appearing in the m first characters of the pattern, and notably for the last one if it occurs only once. The shift is always greater than 0. For example, the d table for the pattern *abracadabra* is

```
d['a'] = 3,  d['b'] = 2,  d['c'] = 6,  d['d'] = 4,  d['r'] = 1,
```

and the value for any other character is 11.

Remark that this can be seen as a special automaton, following Knuth, Morris and Pratt [KMP77] (see also [BYGR90]).

```
bmhsearch( text, n, pat, m )   /* Search pat[1..m] in text[1..n] */
char text[], pat[];
int n, m;
{
    int d[ALPHABET_SIZE], i, j, k;

    for( j = 0; j < ALPHABET_SIZE; j++ ) d[j] = m; /* Preprocessing */
    for( j = 1; j < m; j++ ) d[pat[j]] = m - j;

    for( i = m; i <= n; i += d[text[i]] )          /* Search        */
    {
        k = i;
        for( j = m; j > 0 && text[k] == pat[j]; j-- ) k--;
        if( j == 0 ) Report_match_at_position( k + 1 );
    }
}
```

Figure 1: The Boyer-Moore-Horspool algorithm.

3 A Stationary Process

We turn now to the evaluation of the average performance. Note that, for a given pattern and a given text, the algorithm is fully deterministic. Nevertheless, for a given pattern and a random text, we shall point out in this section a stationary process. The next section will be devoted to the average performance, when both pattern and text are random.

We first state our probabilistic assumptions, i.e. the distribution of the characters appearing in the text or in the pattern (in the case of a random pattern).

Probability assumptions: The distribution of the characters occurring in the text or in the pattern is **uniform**. I.e. the random variable of the characters, X, ranging over the c-alphabet A, satisfies, for any a in A:

$$P(X = a) = \frac{1}{c}.$$

We first introduce the key notion of **head**. A head is a starting point in the text of a right to left comparison. It is always compared to the last character in the pattern.

Definition 3.1 *A character x in the text is a* **head** *iff it is read immediately after a shift.*

Theorem 3.1 *For a given fixed pattern p of length $m+1$, let $\mathcal{H}_k$ be the probability that the k-th character be a head. Then, $\mathcal{H}_k$ converges to a stationary probability $\mathcal{H}_p^\infty$ defined by:*

$$\mathcal{H}_p^\infty = \frac{1}{E_p[shift]},$$

where $E_p[shift]$ denotes the average shift when the aligned character ranges over the c values in the alphabet.

Proof: Position k in a text is a head iff some position $k-j$ is a head with an associated shift j. As such events are not independent, we consider the equivalent expression:

$$\{t[k] \neq head\} = \cup_{j=1}^{m} \{t[l-k] = head \text{ and } shift > j\}$$

Note that if in position $k-j$ we had a shift of less than j, say i, that case is considered now in position $k-j+i$ (that is, a different value of j). Thus, we obtain the following linear recurrence

$$\mathcal{H}_k = 1 - \sum_{j=1}^{m} Pr\{shift > j\}\mathcal{H}_{k-j} ,$$

with initial conditions $\mathcal{H}_{m+1} = 1$ and $\mathcal{H}_k = 0$, for $k \leq m$. As $Pr(shift = 1) \neq 0$, it converges to $1/\sum_{j=1}^{m} Pr\{shift > j\}$ which can be rewritten as $1/\sum_{j=1}^{m+1} jPr\{shift = j\}$ (see Figure 2). ■

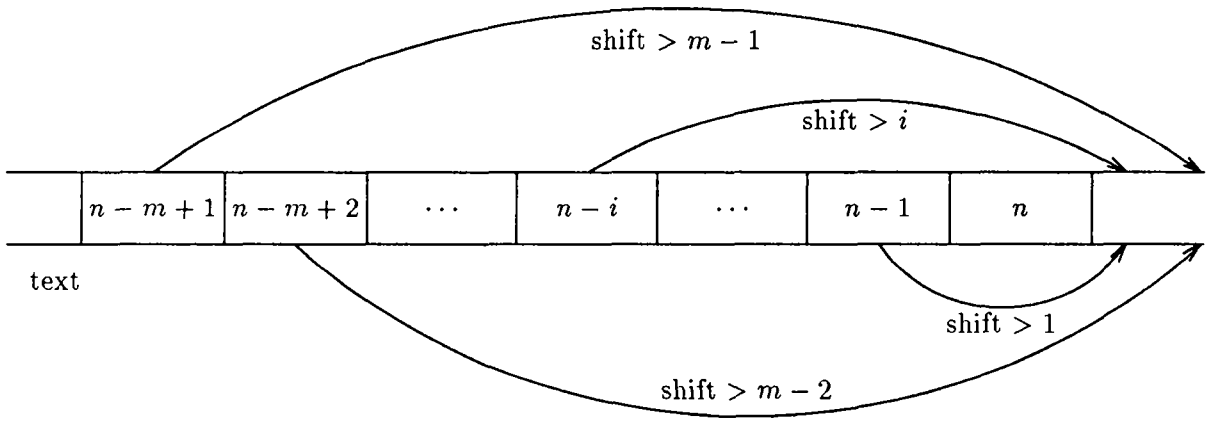

Figure 2: Possible events such that the l-th position is not a head.

Remark: The convergence of such a linear equation is exponential.

In the following proposition, we state a general expression for $E_p[shift]$ as a function of p and the distribution of characters in the text.

Proposition 3.1 *Let $p = p'x$ be a pattern. There exists a sequence $(a_1, \ldots, a_j)$ of characters and a unique sequence $(w_1, \ldots, w_j)$ of words such that:*

$$\begin{aligned} p &= w_j \ldots w_1 x \\ w_i &\in \{a_1, \ldots, a_i\}^*.\{a_i\}. \end{aligned}$$

Let us denote $|w_i|$ by k_i. Then, for a uniform character distribution in the text:

$$c\ E_p[shift] = j + \sum_{i=1}^{j-1}(j-i)k_i + (c-j)(m+1) .$$

If $j = c$ this simplifies to

$$c\ E_p[shift] \ = \ c + \sum_{i=1}^{c-1}(c-i)k_i \ .$$

Proof: In the BMH algorithm the value of the last letter of the pattern is not used to compute the next shift; hence, we only consider the prefix p' of length m. If the last head is the i-th character y_i of p', it is aligned with the first occurrence of y_i in p'; hence the shift is $s_i = 1 + k_1 + \ldots + k_{i-1}$. In other cases, the shift is $m + 1$. Each case happens with probability $1/c$. Hence:

$$\begin{aligned} c\ E_p[shift] &= \sum_{i=1}^{j} s_i + (c-j)(m+1) \\ &= j + \sum_{i=1}^{j-1}(j-i)k_i + (c-j)(m+1) \end{aligned}$$

■

Example: Consider the pattern $abcbcbabaax = abcbc.bab.aaa.x$. Here, $k_1 = 2$ (b), and $k_2 = 3$ (c). If the last head was a, we shift the pattern in one position. Similarly, if it was b (resp. c), we shift three (resp. six) positions. Then,

$$E_p[shift] = \frac{10 + (c-3)(m+1)}{c}$$

We are now ready to derive the expected number of comparisons.

Theorem 3.2 *Let $C_n(p)$ be the expected number of text-pattern comparisons for a given pattern p and a random text t. Then:*

$$\frac{C_n(p)}{n} = H_p^{\infty}\left(\frac{c}{c-1} + E_p[\frac{1}{c^{shift}}] + O\left(\frac{1}{c^3}\right)\right) \ , \ m \geq 3 \ ,$$

$$\frac{C_n(p)}{n} = H_p^{\infty}\left(1 + \frac{1}{c} + \frac{2}{c^2} - \frac{1}{c^3}\right) \ , \ m = 2 \ ,$$

$$\frac{C_n(p)}{n} = H_p^{\infty}\left(1 + \frac{1}{c}\right) \ , \ m = 1 \ .$$

When m tends to ∞:

$$\frac{C_n(p)}{n} = H_p^{\infty}\left(\frac{c}{c-1} + \frac{E_p[\frac{1}{c^{shift}}]}{1 - cE_p[\frac{1}{c^{shift}}]} + O\left(\frac{1}{c^m}\right)\right) \ .$$

Proof: Let us count the number of right to left comparisons performed from position l. We compute $S_p(l)$, its average value for a given p and random text. Here, this number depends on history, but we can get very good approximations. A first approximation is [BY89a]:

$$\frac{S_p(l)}{H_p(l)} = 1 + \frac{1}{c} + \cdots + \frac{1}{c^m} = \frac{c}{c-1}\left(1 - \frac{1}{c^{m+1}}\right)$$

This assumes no knowledge on left neighbours: comparisons are random. But if the last head position is attained in a backward reading, a match certainly occurs, and the left neighbour will also be read. Hence, a second approximation is:

$$\frac{S_p(l)}{H_p(l-shift)} = 1 + \frac{1}{c} + \cdots + \frac{1}{c^{shift}} + \frac{1}{c^{shift}} + \frac{1}{c^{shift+1}} + \ldots + \frac{1}{c^{m-1}} = \frac{c}{c-1}\left(1 - \frac{1}{c^m}\right) + \frac{1}{c^{shift}}$$

which gives the correcting term: $E_p[\frac{1}{c^{shift}}] - \frac{1}{c^m}$.

This sequence of approximations easily generalizes. The k-th approximation will yield a correcting term:

$$\frac{1}{c^{s_1+\ldots+s_k-(k-1)}} + O\left(\frac{1}{c^{m-(k-1)}}\right) ,$$

provided that $s_1 + \ldots + s_k \leq m$. Noticing that:

$$E_p\left[\frac{1}{c^{s_1+\ldots+s_k-(k-1)}}\right] = c^{k-1} E_p[\frac{1}{c^{shift}}]^k ,$$

the result stated follows.

Let us turn now to small patterns. For 2-patterns, i.e. when $m = 1$, the right to left comparisons always stop at step 1 (or 2) with probability $\frac{c-1}{c}$ (or $\frac{1}{c}$). Hence, $S_p(l) = H_p^{\infty}(1 + \frac{1}{c})$. For 3-patterns, i.e. when $m = 2$, one has a correcting term iff $shift + 1 \leq m$, or $shift = 1$, which occurs with probability $\frac{1}{c}$. Hence, the result: $1 + \frac{1}{c} + \frac{2}{c^2} - \frac{1}{c^3}$. Notice that this also is: $\frac{c}{c-1} + E_p[\frac{1}{c^{shift}}] + O(\frac{1}{c^2})$. ■

4 Average Performance

4.1 Some formalism

Here, we introduce some notations. From Proposition 3.1, it appears that we are led to enumerate patterns associated to sequences (k_i). We do so using generating functions. Let W be a set of words, and $|w|$ the size of a word $w \in W$. Let s_n be the number of words w of length n. The generating function enumerating words of W is:

$$S(z) = \sum_n s_n z^n .$$

Proposition 4.1 *We denote by $D_j(z_1, \ldots, z_j)$ the generating function of words with exactly $j \leq c$ different characters, and by $F(z_1, \ldots, z_c)$ the generating function of words over a c-alphabet. They satisfy*

$$D_j(z_1, \ldots, z_j) = \sum_{p=w_j \ldots w_1 . x} z_1^{w_1} \ldots z_j^{w_j} = c^{\underline{j}} \frac{z_1}{1-z_1} \cdots \frac{z_j}{1-z_j} ,$$

and

$$F(z_1, \ldots, z_c) = \sum_{j=1}^{c} D_j(z_1, \ldots, z_j) = \frac{cz_1}{1-z_1}\left(1 + \frac{(c-1)z_2}{1-2z_2}\left(\cdots\left(1 + \frac{z_c}{1-cz_c}\right)\right)\right) .$$

Proof: Applying classical rules [Fla88], the generating function for words w_i is $z_i \frac{1}{1-tz_i}$. Concatenation translates to a product, and we have $c(c-1)\ldots(c-j+1) = c^{\underline{j}}$ choices for the sequence $(a_1, \ldots, a_j)$. Note that the generating function of all strings of length m, $F_m(z_1, \ldots, z_c)$ is the restriction to $k_1 + \ldots + k_c = m$ of $F(z_1, \ldots, z_c)$, where k_i is the degree of z_i in F_m. ■

Notably, all possible patterns of length m are given by the coefficient of degree m in $F(z, ..., z)$, namely $F_m(1, \ldots, 1)$ or c^m. For example, for $c = 2$ (binary alphabet) we have

$$F_m(z_1, z_2) = 2z_1^m + \frac{2z_2z_1}{z_1 - 2z_2}\left(z_1^{m-1} - (2z_2)^{m-1}\right) = 2z_1^m + 2z_2z_1 \sum_{j\geq 0}^{m-2} z_1^j (2z_2)^{m-2-j}$$

The total number of patterns is $F_m(1,1) = 2^m$.

4.2 Average number of heads

We now assume that both text and pattern are random. We first study the average number of heads for patterns of length $m+1$. Then we derive an asymptotic expression when m tends to ∞ and study its asymptotic behaviour when the alphabet size, c, tends to ∞.

Theorem 4.1 *The probability of being a head, when p ranges over all patterns of length $m+1$ on a c-ary alphabet is:*

$$\mathcal{H}(c,m) = c\sum_{j=1}^{c}\binom{c}{j} \sum_{\substack{k_i\geq 1 \\ k_1+\cdots+k_{j-1}<m}} \frac{\prod_{i=1}^{j-1}\left(\frac{i}{j}\right)^{k_i}\left(\frac{j}{c}\right)^m}{j + \sum_{i=1}^{j-1}(j-i)k_i + (c-j)(m+1)} .$$

Moreover:

$$Ph(c) = \lim_{m\to\infty}\mathcal{H}(c,m) = c \sum_{\substack{k_i\geq 1 \\ i=1,\ldots,c-1}} \frac{\prod_{i=1}^{c-1}\left(\frac{i}{c}\right)^{k_i}}{c + \sum_{i=1}^{c-1}(c-i)k_i} + O((1-1/c)^m/m) .$$

Corollary 4.1 *For a binary alphabet, one has:*

$$Ph(c) = 8\ln 2 - 5 \sim 0.5452 .$$

Proof:

$$\mathcal{H}(c,m) = \frac{1}{c^m}\sum_j \sum_{k_1,\ldots,k_j} \frac{q}{E_{pattern}[shift]}[z_1^{k_1}\ldots z_j^{k_j}]D_j(z_1,\ldots,z_j) .$$

As $[z_1^{k_1}\ldots z_j^{k_j}]\mathcal{H}(c,m) = \frac{c^j}{j!}\prod_{i=1}^{j-1}(\frac{i}{j})^{k_i} j^m$ and $1/E_p[shift] = 1/(c + \sum_{i=1}^{c-1}(c-i)k_i)$, the expression of $\mathcal{H}(c,m)$ follows. For large patterns, only the last term does not converge to 0 when m goes to infinity, and $Ph(c)$ follows.

For a binary alphabet ($c = 2$), the expected shift is $E_p[shift] = \frac{2+k_1}{2}$. Then:

$$\mathcal{H}(2,m) = \frac{2}{2^m}\left(\frac{2}{(m+2)} + 2\sum_{j=0}^{m-2}\frac{2^{m-j-2}}{j+3}\right) = 8\ln 2 - 5 + O\left(\frac{1}{m2^m}\right)$$

	c			
$m+1$	2	3	4	5
2	.666667	.600000	.571429	.555556
3	.583333	.476190	.433333	.410256
4	.558333	.421958	.368094	.339860
5	.550000	.395198	.332003	.299440
6	.547024	.381249	.310381	.273988
7	.545908	.373737	.296842	.257047
8	.545474	.369597	.288135	.245365
9	.545300	.367275	.282438	.237120
10	.545229	.365954	.278663	.231206
15	.545178	.364246	.271961	.218487
20	.545177	.364118	.270950	.215601
25	.545177	.364108	.270783	.214899
30	.545177	.364107	.270754	.214722

Table 1: Exact values for $\mathcal{H}(c,m)$.

■

Table 1 gives some exact values for this probability.

From Table 1, it seems that $H(c,m)$ quickly converges. We will prove a theorem and set a conjecture:

Theorem 4.2 *Let $Ph(c)$ be $\lim_{m\to\infty} H(c,m)$, where $H(c,m)$ is the probability of being a head, when p ranges over all possible patterns of length $m+1$ on a c-ary alphabet. Then:*

$$\frac{1}{c} \leq Ph(c) \leq \frac{2}{c+1} .$$

Conjecture: *When $c \to \infty$, then $Ph(c) \to \frac{1}{c}$.*

Proof: For any pattern, the shift on the i-th different character is greater than or equal to i. Hence:

$$cE_p[shift] \geq 1+2+\ldots+c = \frac{c(c+1)}{2} , \; c \leq m .$$

If $c > m$, one gets the tighter bound: $1+\ldots+m+(c-m)(m+1)$. The lower bound is a direct consequence of Jensen's inequality [Fel68], that can be expressed as: $E\left(\frac{1}{s}\right) \geq \frac{1}{E(s)}$. ■

Practically, the computations in Table 1 show that the lower bound is very tight. We are currently working to exhibit the underlying process and the random variables involved. We conjecture that some variant of the Central Limit Theorem should apply.

4.3 Average number of comparisons

We prove here:

Theorem 4.3 *Let $C_{n,m}$ be the expected number of text-pattern comparisons for random texts of size n and random patterns of length $m+1$. Then:*

$$\frac{C_{n,m}}{n} = \mathcal{H}(c,m)\left(1+\frac{1}{c}+\frac{2}{c^2}+O\left(\frac{1}{c^3}\right)\right)$$

or, for large patterns:

$$\frac{C_n}{n} = Ph(c)\left(1+\frac{1}{c}+\frac{2}{c^2}+O\left(\frac{1}{c^3}\right)\right).$$

Corollary 4.2 *For a binary alphabet, the average number of comparisons is very close to:*

$$\left(26\ln 2 \;-\; 8\ln 3 \;-\frac{17}{2}\right) n \approx 1.2782n$$

with a difference not exceeding $0.02n$.

Proof: One wants to derive:

$$E_{all\ patterns}[H_p^{\infty}(\frac{c}{c-1}+E_p[\frac{1}{c^{shift}}]+O(\frac{1}{c^3}))] \ .$$

Now, the rightmost character contributes by $\frac{1}{c}$ to $E_p[\frac{1}{c^{shift}}]$ and is found with probability $\frac{1}{c}$. Other characters contribute by at most: $(\frac{1}{c^2}+\frac{1}{c^3}+\ldots+\frac{1}{c^c})\frac{1}{c} = O(\frac{1}{c^3})$. Now, summing over all patterns yields the correction:

$$E_{all\ patterns}\left[H_p^{\infty}\left(\frac{1}{c^2}+O(c^{-3})\right)\right] = \mathcal{H}(c,m)\left(\frac{1}{c^2}+O(c^{-3})\right).$$

■

Table 2 gives some values for the second order approximation of $C_{n,m}/n$ for differents c and m. Note that only for $c=2$ the expected number of comparisons increases with m.

Figure 3 shows the theoretical results compared with experimental results for $c=2$ and $c=4$. The experimental results are the average of 100 trials for searching 100 random patterns in a random text of 50000 characters.

5 Concluding Remarks

In this paper, we realized an extensive study of a Boyer-Moore-type string searching algorithm. We derived an average analysis of the Boyer-Moore-Horspool algorithm. The expected number of text-pattern comparisons, C_n, is linear in the size of the text, and we derived the linearity constant $K = \frac{C_n}{n}$ when n goes to infinity. We first addressed the case of a given pattern. Then, averaging over all patterns, we derived K. Finally, we pointed out a tight asymptotic development, namely $K \sim \frac{1}{c}$, where c is the cardinality of the alphabet.

The approach combines two different tools. First, probability theory is used to point out a stationary process. This avoids combinatorial explosion which limited other Markov-type analyses, due to the variety of searched patterns to be considered; hence, this approach allows to achieve the

	c			
$m+1$	2	3	4	5
2	.916667	.711111	.633929	.595556
3	1.07813	.707819	.577734	.513477
4	1.16927	.671381	.512026	.437875
5	1.22044	.643041	.466753	.388051
6	1.24812	.625051	.437543	.355491
7	1.26270	.614318	.418757	.333594
8	1.27025	.608056	.406556	.318453
9	1.27412	.604426	.398543	.307757
10	1.27609	.602321	.393227	.300084
15	1.27804	.599555	.383782	.283581
20	1.27810	.599347	.382357	.279837
25	1.27811	.599329	.382122	.278927
30	1.27811	.599328	.382081	.278696

Table 2: Expected number of comparisons per character

analysis. Probabilities also provide an asymptotic development of the linearity constant. Second, the analysis reduced to a word enumeration problem and algebraic tools such as generating functions appear powerful. These theoretical results appear to be very close to experimental results obtained by simulation [BY89a]. Moreover, their convergence to the asymptotic results is very fast. Our results also prove that the bigger c is, the better Boyer-Moore perfoms (as expected!).

Recently, Sunday [Sun90] suggested to use the character of the text after the character corresponding to the last position of the pattern to address the d table. The analysis presented here applies for this case considering a pattern of length $m+1$ for the head probability, and a pattern of length m for the expected number of comparisons.

Our analytic results easily generalize to non-uniform distributions, when one considers a given pattern. Averaging over all patterns is more intricate and is the object of a current work. Also, we are extending this kind of analysis to new multiple string searching and two dimensional pattern matching algorithms [BYR90].

References

[Bar84] G. Barth. An analytical comparison of two string searching algorithms. *Inf. Proc. Letters*, 18:249–256, 1984.

[BY89a] R. Baeza-Yates. Improved string searching. *Software-Practice and Experience*, 19(3):257–271, 1989.

[BY89b] R.A. Baeza-Yates. *Efficient Text Searching.* PhD thesis, Dept. of Computer Science, University of Waterloo, May 1989. Also as Research Report CS-89-17.

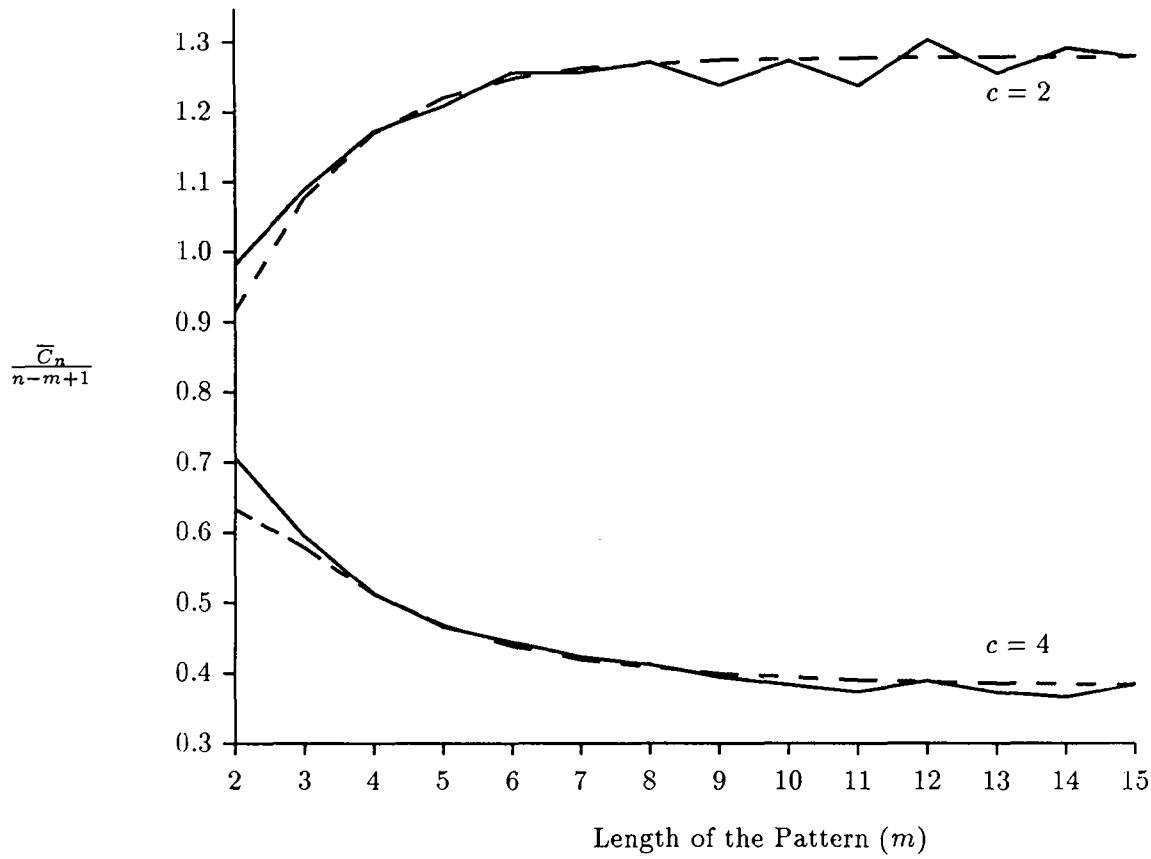

Figure 3: Experimental vs. theoretical values for $C_{n,m}/n$.

[BY89c] R.A. Baeza-Yates. String searching algorithms revisited. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Workshop in Algorithms and Data Structures*, pages 75–96, Ottawa, Canada, August 1989. Springer Verlag Lecture Notes on Computer Science 382.

[BYGR90] R. Baeza-Yates, G. Gonnet, and M. Regnier. Analysis of Boyer-Moore-type string searching algorithms. In *1st ACM-SIAM Symposium on Discrete Algorithms*, pages 328–343, San Francisco, January 1990.

[BYR90] R. Baeza-Yates and M. Regnier. Fast algorithms for two dimensional and multiple pattern matching. In R. Karlsson and J. Gilbert, editors, *2nd Scandinavian Workshop in Algorithmic Theory, SWAT'90*, Lecture Notes in Computer Science 447, pages 332–347, Bergen, Norway, July 1990. Springer-Verlag.

[Fla88] P. Flajolet. Mathematical methods in the analysis of algorithms and data structures. In Egon Börger, editor, *Trends in Theoretical Computer Science*, chapter 6, pages 225–304. Computer Science Press, Rockville, Maryland, 1988. (Lecture Notes for *A Graduate Course in Computation Theory*, Udine, 1984).

[Hor80] R. N. Horspool. Practical fast searching in strings. *Software - Practice and Experience*, 10:501–506, 1980.

[KMP77] D.E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM J on Computing*, 6:323–350, 1977.

[Reg89] M. Regnier. Knuth-Morris-Pratt algorithm: An analysis. In *MFCS'89, Lecture Notes in Computer Science 379*, pages 431–444, Porabka, Poland, August 1989. Springer-Verlag. Also as INRIA Report 966, 1989.

[Riv77] R. Rivest. On the worst-case behavior of string-searching algorithms. *SIAM J on Computing*, 6:669–674, 1977.

[Sch88] R. Schaback. On the expected sublinearity of the Boyer-Moore algorithm. *SIAM J on Computing*, 17:548–658, 1988.

[Sun90] D.M. Sunday. A very fast substring search algorithm. *Communications of the ACM*, 33(8):132–142, Aug 1990.

[Yao79] A.C. Yao. The complexity of pattern matching for a random string. *SIAM J on Computing*, 8:368–387, 1979.

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique .

ISSN 0249-6399