Theoretical Computer Science 92 (1992) 19–31
Elsevier

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

Baeza-Yates, R.A. and M. Régnier, Average running time of the Boyer–Moore–Horspool algorithm, Theoretical Computer Science 92 (1992) 19–31.

We study Boyer–Moore-type string searching algorithms. 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, $1/c \leqslant \alpha \leqslant 2/(c+1)$, 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 of 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 [11].

The average complexity is also important [14, 9]. It is interesting to show (when possible!) that the expected number of comparisons $\bar{C}_n$ is asymptotically $K \cdot n$; and

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

compare the linearity constant K for different string searching algorithms. One common characteristic of these algorithms is the dependence on history: the number of comparisons made to a given character depends on the result of comparisons to its neighbors. Hence, first attempts to derive asymptotics used Markov chains [6, 12, 2, 3]. Unfortunately, this quickly leads to a combinatorial explosion as the size of the pattern increases. Recently another algebraic approach, based on pattern enumeration and combinatorics on words, facilitated an analysis of the Knuth–Morris–Pratt algorithm [10].

In this paper we derive the analysis of the Boyer–Moore–Horspool or BMH algorithm [8]. 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 [10] 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 tool and combinatorics on words, leads to the linearity constant K. It appears to be a simple function of the cardinality c of the alphabet: $K_c \sim 1/c + O(1/c^2)$.

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 $\bar{C}_n \sim K_c n$ is derived. Asymptotic bounds on K_c are proven, and a conjecture is stated. All these results agree well with experimental values. The last section is our conclusion. In a preliminary version of this paper [4] we also studied *Boyer–Moore automata.*

2. The Boyer–Moore–Horspool algorithm

The Boyer–Moore (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*, the 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 [8]. In that paper it was 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. This also implies that the comparisons can be done in any order (left-to-right, right-to-left, random, etc.) [3]. 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 Fig. 1.

```
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; /* Preprocess */
    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 );
    }
}
```

Fig. 1. The Boyer–Moore–Horspool algorithm.

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

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

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

$$d['a'] = 3, \quad d['b'] = 2, \quad d['c'] = 6, \quad d['d'] = 4, \quad d['r'] = 1,$$

and the value for any other character is 11.

Note that this can be seen as a special automaton, following Knuth, Morris and Pratt [9] (see also [4]).

3. A stationary process

We turn now to the evaluation of 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, a stationary stochastic process serves as a good model, as will be developed in this section. The next section will be devoted to the average performance when both pattern and text are random.

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

Probability assumptions. The distribution of characters occurring in the text or in the pattern is *uniform*. That is, given the random variable X, whose value may be any character from the c-alphabet A, for any character a in A:

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

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 3.2. *For a given fixed pattern p of length $m+1$, let $\mathscr{H}_k$ be the probability that the kth text character be a head. Then, $\mathscr{H}_k$ converges to a stationary probability $\mathscr{H}_p^x$ defined by*

$$\mathscr{H}_p^x=\frac{1}{E_p[shift]},$$

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:

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

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

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

with initial conditions $\mathscr{H}_{m+1}=1$ and $\mathscr{H}_k=0$, for $k\leqslant m$. As $Pr(shift=1)\neq 0$, the recurrence converges to $1/\sum_{j=1}^{m} Pr\{shift>j\}$ which can be rewritten as $1/\sum_{j=1}^{m+1} j\,Pr\{shift=j\}$ (see Fig. 2). □

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.

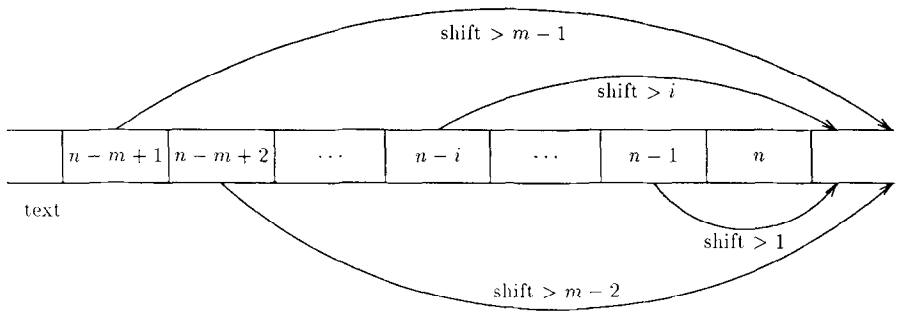

Fig. 2. Possible events such that the nth position is not a head.

Proposition 3.3. *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*

$$p=w_j\ldots w_1 x$$

$$w_i\in\{a_1,\ldots,a_i\}^*.\{a_i\}.$$

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

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

If $j=c$ this simplifies to

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

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 ith character y_i of p', it is aligned with the first occurrence of y_i in p'; hence, the shift is $s_i=1+k_1+\cdots+k_{i-1}$. In other cases, the shift is $m+1$. Each case happens with probability $1/c$. Hence,

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

□

Example. Consider the pattern $abcbcbabaax=abcbc.bab.aa.x$. Here, $k_1=2(b)$, and $k_2=3(c)$. If the last head were a, we shift the pattern by one position. Similarly, if it were b we shift three positions; for c, shift 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 3.4. *Let $\bar{C}_n(p)$ be the expected number of text-pattern comparisons for a given pattern p and a random text t. Then*

$$\frac{\bar{C}_n(p)}{n} = H_p^{\infty}\left(\frac{c}{c-1} + E_p\left[\frac{1}{c^{shift}}\right] + O\left(\frac{1}{c^3}\right)\right) \quad m \geqslant 3,$$

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

$$\frac{\bar{C}_n(p)}{n} = H_p^{\infty}\left(1 + \frac{1}{c}\right) \quad m = 1.$$

When m tends to ∞

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

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 [1]

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

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

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

$$= \frac{c}{c-1}\left(1 - \frac{1}{c^m}\right) + \frac{1}{c^{shift}}$$

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

This sequence of approximations easily generalizes. The kth approximation will yield a correcting term

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

provided that $s_1 + \cdots + s_k \leqslant m$. Noticing that

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

the result stated then 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 $(c-1)/c$ (or $1/c$). Hence, $S_p(l) = H_p^{\infty}(1+1/c)$. For 3-patterns, i.e. when $m=2$, one has a correcting term iff $shift+1 \leqslant m$, or $shift=1$, which occurs with probability $1/c$. Hence, the result $1+1/c+2/c^2-1/c^3$. Notice that this also is: $c/(c-1)+E_p[1/c^{shift}] + O(1/c^2)$. □

4. Average performance

4.1. Some formalism

We need to introduce some notation. From Proposition 3.3, it appears that we are led to enumerate patterns associated with 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

$$S(z) = \sum_{n \geqslant 0} s_n z^n.$$

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

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

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 [7], the generating function for words w_i is $z_i(1/(1-iz_i))$. 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 $k_1 + \cdots + k_c = m$ 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 $F(z,\ldots,z)$, namely, $F_m(1,\ldots,1)$ or c^m. For example, for $c=2$ (binary alphabet) we have

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

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 behavior when the alphabet size c tends to ∞.

Theorem 4.2. *The probability of being a head, when p ranges over all patterns of length $m+1$ on a c-ary alphabet is*

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

Moreover,

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

Corollary 4.3. *For a binary alphabet, the result is*

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

Proof.

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

As $[z_1^{k_1}\ldots z_j^{k_j}]\mathscr{H}(c,m)=c^j/j!\prod_{i=1}^{j-1}(i/j)^{k_i}j^m$ and $1/E_p[shift]=1/(c+\sum_{i=1}^{c-1}(c+i)k_i)$, the expression of $\mathscr{H}(c,m)$ 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 $E_p[shift]=(2+k_1)/2$. Then,

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

Table 1 gives some exact values for this probability, from which it seems that $H(c, m)$ quickly converges.

Theorem 4.4. *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* $m\to 1$ *on a c-ary alphabet. Then*

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

Conjecture. *When* $c\to\infty$, *then* $Ph(c)\to 1/c$.

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

$$cE_p[shift] \geqslant 1+2+\cdots+c = \frac{c(c+1)}{2} \quad c \leqslant m.$$

If $c>m$, one gets the tighter bound: $1+\cdots+m+(c-m)(m+1)$. The lower bound is a direct consequence of Jensen's inequality [15], that can be expressed as: $E(1/s) \geqslant 1/E(s)$. □

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.

Table 1
Exact values for $\mathscr{H}(c,m)$

	c			
$m+1$	2	3	4	5
2	0.666667	0.600000	0.571429	0.555556
3	0.583333	0.476190	0.433333	0.410256
4	0.558333	0.421958	0.368094	0.339860
5	0.550000	0.395198	0.332003	0.299440
6	0.547024	0.381249	0.310381	0.273988
7	0.545908	0.373737	0.296842	0.257047
8	0.545474	0.369597	0.288135	0.245365
9	0.545300	0.367275	0.282438	0.237120
10	0.545229	0.365954	0.278663	0.231206
15	0.545178	0.364246	0.271961	0.218487
20	0.545177	0.364118	0.270950	0.215601
25	0.545177	0.364108	0.270783	0.214899
30	0.545177	0.364107	0.270754	0.214722

4.3. Average number of comparisons

Theorem 4.5. *Let $\bar{C}_{n,m}$ be the expected number of text-pattern comparisons for random texts of size n and random patterns of length $m+1$. Then,*

$$\frac{\bar{C}_{n,m}}{n} = \mathscr{H}(c, m)\left(1 + \frac{1}{c} + \frac{2}{c^2} + O\left(\frac{1}{c^3}\right)\right)$$

or, for large patterns,

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

Corollary 4.6. *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. It is desirable to derive

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

The rightmost character contributes $1/c$ to $E_p[1/c^{shift}]$ and is found with probability $1/c$. Other characters contribute at most: $(1/c^2 + 1/c^3 + \cdots + 1/c^c)1/c = O(1/c^3)$. Now

Table 2
Expected number of comparisons per character

	c			
$m+1$	2	3	4	5
2	0.916667	0.711111	0.633929	0.595556
3	1.07813	0.707819	0.577734	0.513477
4	1.16927	0.671381	0.512026	0.437875
5	1.22044	0.643041	0.466753	0.388051
6	1.24812	0.625051	0.437543	0.355491
7	1.26270	0.614318	0.418757	0.333594
8	1.27025	0.608056	0.406556	0.318453
9	1.27412	0.604426	0.398543	0.307757
10	1.27609	0.602321	0.393227	0.300084
15	1.27804	0.599555	0.383782	0.283581
20	1.27810	0.599347	0.382357	0.279837
25	1.27811	0.599329	0.382122	0.278927
30	1.27811	0.599328	0.382081	0.278696

summing over all patterns yields the correction

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

Table 2 gives some values for the second order approximation of $\bar{C}_{n,m}/n$ for different values of 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 50,000 characters.

5. Concluding remarks

In this paper we have realized an extensive study of a Boyer–Moore-type string searching algorithm. We first derived an average analysis of the Boyer–Moore–Horspool algorithm. The expected number of text-pattern comparisons $\bar{C}_n$ is linear in the size of the text, and we derived the linearity constant $K=\bar{C}_n/n$ when n goes to

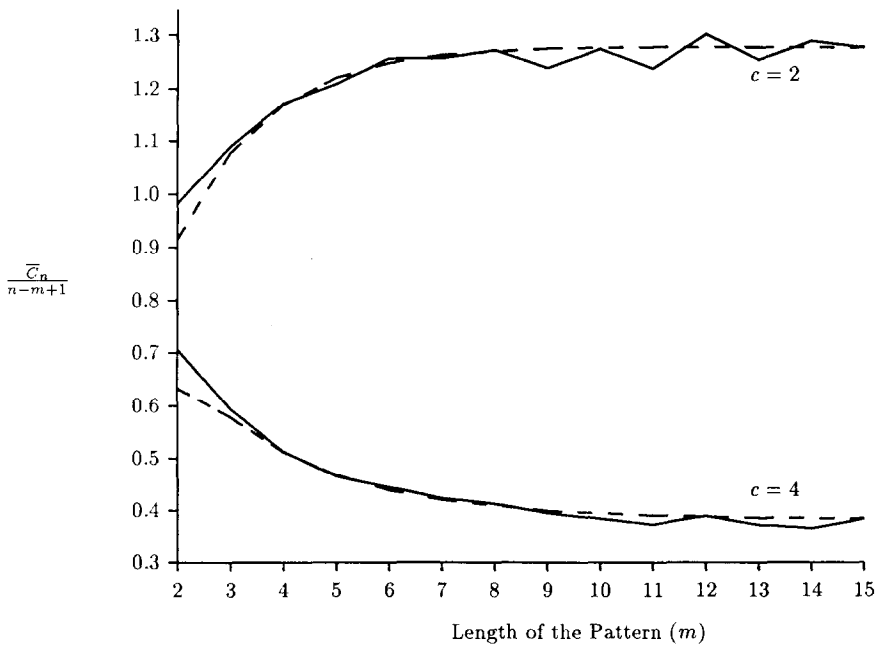

Fig. 3. Experimental versus theoretical values for $\bar{C}_{n,m}/n$.

infinity. The case of a given pattern was addressed first. Then, averaging over all patterns, we derived K. Finally, we pointed out a tight asymptotic result, namely, $K \sim 1/c$, where c is the cardinality of the alphabet.

The approach combines two different tools. First, probability theory is used to establish 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 facilitates the analysis. Probabilities also provide an asymptotic development of the linearity constant. Second, the analysis reduces 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 [1]. Moreover, their convergence to the asymptotic results is very fast. Our results also prove that as c increases, Boyer–Moore performs better (as expected!).

Recently, Sunday [13] suggested using 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 is applicable, 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 nonuniform distributions when one considers a given pattern. Averaging over all patterns is more intricate and is the object of current research. Also, we are extending this kind of analysis to new multiple string searching and two dimensional pattern matching algorithms [5].

References

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

[2] R.A. Baeza-Yates, Efficient text searching, Ph.D. Thesis, Dept. of Computer Science, University of Waterloo, 1989; also as Research Report CS-89-17.

[3] R.A. Baeza-Yates, String searching algorithms revisited, in: F. Dehne, J.-R. Sack and N. Santoro, eds., *Workshop in Algorithms and Data Structures*, Ottawa, Canada (1989) Lecture Notes in Computer Science, Vol. 382 (Springer, Berlin, 1989) 75–96.

[4] R.A. Baeza-Yates, G. Gonnet and M. Regnier, Analysis of Boyer–Moore-type string searching algorithms, in: *Proc. 1st ACM-SIAM Symposium on Discrete Algorithms*, San Francisco (1990) 328–343.

[5] R.A. Baeza-Yates and M. Regnier, Fast algorithms for two dimensional and multiple pattern matching, in: R. Karlsson and J. Gilbert, eds., *Proc. 2nd Scandinavian Workshop in Algorithmic Theory, SWAT '90*, Bergen, Norway (1990) Lecture Notes in Computer Science, Vol. 447 (Springer, Berlin, 1990) 332–347.

[6] G. Barth, An analytical comparison of two string searching algorithms, *Inform. Process. Lett.* **18** (1984) 249–256.

[7] P. Flajolet, Mathematical methods in the analysis of algorithms and data structures, in: Egon Börger, ed., *Trends in Theoretical Computer Science*, Chapter 6 (Computer Science Press, Rockville, Maryland, 1988) 225–304.

[8] R.N. Horspool, Practical fast searching in strings, *Software–Practice and Experience*, **10** (1980) 501–506.

[9] D.E. Knuth, J. Morris and V. Pratt, Fast pattern matching in strings, *SIAM J. Comput.* **6** (1977) 323–350.

[10] M. Regnier, Knuth–Morris–Pratt algorithm: An analysis, in: *MFCS '89*, Porabka, Poland (1989) Lecture Notes in Computer Science, Vol. 379 (Springer, Berlin, 1989) 431–444; also as INRIA Report 966, 1989.

[11] R. Rivest, On the worst-case behavior of string-searching algorithms, *SIAM J. Comput.* **6** (1977) 669–674.

[12] R. Schaback, On the expected sublinearity of the Boyer–Moore algorithm, *SIAM J. Comput.* **17** (1988) 548–658.

[13] D.M. Sunday, A very fast substring search algorithm, *Comm. ACM*, **33** (8) (1990) 132–142.

[14] A.C. Yao, The complexity of pattern matching for a random string, *SIAM J. Comput.* **8** (1979) 368–387.

[15] W. Feller, *An Introduction to Probability Theory and its Applications, Vol. 1* (Wiley, New York, 1968).