



Complexity of sequential pattern matching algorithms

Mireille Regnier, Wojciech Szpankowski

► To cite this version:

Mireille Regnier, Wojciech Szpankowski. Complexity of sequential pattern matching algorithms. [Research Report] RR-2549, INRIA. 1995. inria-00074130

HAL Id: inria-00074130

<https://inria.hal.science/inria-00074130>

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Complexity of sequential pattern matching algorithms

Mireille REGNIER - Wojciech SZPANKOWSKI

N° 2549

Avril 1995

PROGRAMME 2

*R*apport
de recherche

Les rapports de recherche de l'INRIA
sont disponibles en format postscript sous
ftp.inria.fr (192.93.2.54)

si vous n'avez pas d'accès ftp
la forme papier peut être commandée par mail :
e-mail : dif.gesdif@inria.fr
(n'oubliez pas de mentionner votre adresse postale).

par courrier :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

INRIA research reports
are available in postscript format
ftp.inria.fr (192.93.2.54)

if you haven't access by ftp
we recommend ordering them by e-mail :
e-mail : dif.gesdif@inria.fr
(don't forget to mention your postal address).

by mail :
Centre de Diffusion
INRIA
BP 105 - 78153 Le Chesnay Cedex (FRANCE)

COMPLEXITY OF SEQUENTIAL PATTERN MATCHING ALGORITHMS

April 26, 1995

Mireille Régnier*
INRIA
Rocquencourt
78153 Le Chesnay Cedex
France
Mireille.Regnier@inria.fr

Wojciech Szpankowski†
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A.
spa@cs.purdue.edu

Abstract

We formally define a class of sequential pattern matching algorithms that includes all variations of Morris-Pratt algorithm. For last twenty years it was known that complexity of such algorithms are bounded by a linear function of the text string length. Recently, substantial progress has been made in identifying lower bounds. However, it was not known whether really there exists a *linearity constant*. We prove this fact rigorously for the worst case and the average case using *Subadditive Ergodic Theorem*. Our results hold for any given pattern and text and for stationary ergodic pattern and text providing the length of the pattern is order of magnitude smaller than the square root of the text length. In the course of the proof, we also establish some structural property of Morris-Pratt-like algorithms. Namely, we prove the existence of “unavoidable positions” where the algorithm must stop to compare. This property seems to be uniquely reserved for Morris-Pratt type algorithms since as, we point out in our concluding remarks, a popular pattern matching algorithm proposed by Boyer and Moore does not possess this property.

Complexité des algorithmes de recherche de motifs séquentiels

Resumé

Nous définissons formellement une classe d’algorithmes de recherche de motifs, les algorithmes séquentiels, qui comprend toutes les variantes de l’algorithme original de Morris-Pratt. On sait depuis une vingtaine d’années que la complexité de ces algorithmes est linéairement bornée en la taille du texte. Récemment, des progrès importants ont été faits quant à la détermination des bornes inférieures. Néanmoins, l’existence d’une *constante de linéarité* n’était pas établie. Nous prouvons ce fait rigoureusement dans le cas le pire et en moyenne en utilisant le Théorème de Sous Additivité Ergodique. Nos résultats sont valables lorsque le texte et le motif sont donnés et lorsque les lois de probabilités sont stationnaires et ergodiques, pourvu que l’ordre de grandeur de la longueur du motif recherché soit inférieure à la racine carrée de la longueur du texte. Dans le courant de la preuve, nous établissons une propriété structurelle des variantes de Morris-Pratt. Plus précisément, nous prouvons l’existence de “positions inévitables” où l’algorithme doit effectuer des comparaisons. Cette propriété semble être propre aux algorithmes du type Morris-Pratt. Comme nous le montrons dans notre conclusion, l’algorithme de recherche de motifs proposé par Boyer-Moore ne possède pas cette propriété.

*This work was partially supported by the ESPRIT III Program No. 7141 ALCOM II.

†This research was supported by NSF Grants CCR-9201078 and NCR-9206315. The work was partially done while the author was visiting INRIA, Rocquencourt, France. The author wishes to thank INRIA (projects ALGO, MEVAL and REFLECS) for a generous support.

COMPLEXITY OF SEQUENTIAL PATTERN MATCHING ALGORITHMS

April 12, 1995

Mireille Régnier*
INRIA
Rocquencourt
78153 Le Chesnay Cedex
France
Mireille.Regnier@inria.fr

Wojciech Szpankowski†
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A.
spa@cs.purdue.edu

Abstract

We formally define a class of sequential pattern matching algorithms that includes all variations of Morris-Pratt algorithm. For last twenty years it was known that complexity of such algorithms are bounded by a linear function of the text string length. Recently, substantial progress has been made in identifying lower bounds. However, it was not known whether really there exists a *linearity constant*. We prove this fact rigorously for the worst case and the average case using *Subadditive Ergodic Theorem*. Our results hold for any given pattern and text and for stationary ergodic pattern and text providing the length of the pattern is order of magnitude smaller than the square root of the text length. In the course of the proof, we also establish some structural property of Morris-Pratt-like algorithms. Namely, we prove the existence of “unavoidable positions” where the algorithm must stop to compare. This property seems to be uniquely reserved for Morris-Pratt type algorithms since as, we point out in our concluding remarks, a popular pattern matching algorithm proposed by Boyer and Moore does not possess this property.

Keywords: *String searching, pattern matching, analysis of algorithms, automata, complexity, combinatorics on words convergence of processes, Subadditive Ergodic Theorem.*

*This work was partially supported by the ESPRIT III Program No. 7141 ALCOM II.

†This research was supported by NSF Grants CCR-9201078 and NCR-9206315. The work was partially done while the author was visiting INRIA, Rocquencourt, France. The author wishes to thank INRIA (projects ALGO, MEVAL and REFLECS) for a generous support.

1 INTRODUCTION

The complexity of string searching algorithms has been discussed in various papers (cf. [1, 7, 8, 6, 11, 16]). It is well known that most pattern matching algorithms perform linearly in the worst case as well as “on average”. Several attempts have been made to provide tight bounds on the so-called “linearity constant”. Nevertheless, the existence of such a constant has never been proved. The only exception is the average case of Morris-Pratt-like algorithms [16] for the symmetric Bernoulli model (independent generation of symbols with each symbol occurring with the same probability) where the constant was also explicitly computed.

In this paper we investigate a fairly general class of algorithms, called *sequential algorithms*, for which the existence of the linearity constant is proved for the worst and average case. Sequential algorithms include the naïve one and several variants of Morris-Pratt algorithm [15]. These algorithms never go backward, and are easy to implement. They perform better than Boyer-Moore like algorithms in numerous cases, e.g., for binary alphabet [2], when character distributions are strongly biased, and when the pattern and text distributions are correlated. Thus, even from practical point of view these algorithms are worth studying.

In this paper we analyze sequential algorithms under a general probabilistic model that only assumes stationarity and ergodicity of the text and pattern sequences. It relies on the Subadditive Ergodic Theorem [10]. The “average case” analysis is also understood in the strongest possible sense, that is, we establish asymptotic complexity that is true for all but finite number of strings (i.e., in almost sure sense).

The literature on worst case as well average case on Knuth-Morris-Pratt type algorithms is rather scanty. For almost twenty years the upper bound was known [15], and no progress has been reported on a lower bound or a tight bound. This was partially rectified by Colussi *et al.* [8] and Cole *et al.* [7] who established several lower bounds for the so called “on-line” sequential algorithms. However, the existence of the linearity constant was not established yet. In this paper we prove this fact rigorously. In the course of proving it, we construct the so called *unavoidable positions* where the algorithm must stop to compare. The existence of these positions is crucial to establish subadditivity of complexity for the Morris-Pratt type algorithms, and hence their linearity. This property seems to be restricted to Morris-Pratt type algorithms since we shall present an example of a text and a pattern for which the Boyer-Moore algorithm does not possess any unavoidable position.

The paper is organized as follows. In the next section we present a general definition of sequential algorithms, and formulate our main results. Section 3 contains all proofs. In concluding remarks we discuss possible extensions of our approach to other classes of algorithms,

notably Boyer-Moore like [5].

2 SEQUENTIAL ALGORITHMS

In this section, we first present a general definition of sequential algorithms (i.e., algorithms that work like Morris-Pratt). Then, we formulate our main results and discuss some consequences.

2.1 Basic Definitions

Throughout we write \mathbf{p} and \mathbf{t} for the pattern and the text which are of lengths m and n , respectively. The i th character of the pattern \mathbf{p} (text \mathbf{t}) is denoted as $\mathbf{p}[i]$ ($\mathbf{t}[i]$), and by \mathbf{t}_i^j we define substring of \mathbf{t} starting at position i and ending at position j , that is $\mathbf{t}_i^j = \mathbf{t}[i]\mathbf{t}[i+1] \cdots \mathbf{t}[j]$. We also assume that for a given pattern \mathbf{p} its length m does not vary with the text length n .

Our prime goal is to investigate complexity of string matching algorithms. We define it formally as follows.

Definition 1 (i) For any string matching algorithm that runs on a given text \mathbf{t} and a given pattern \mathbf{p} , let $M(l, k) = 1$ if the l th symbol $\mathbf{t}[l]$ of the text is compared by the algorithm to the k th symbol $\mathbf{p}[k]$ of the pattern. We assume in the following that this comparison is performed at most once.

(ii) For a given pattern matching algorithm partial complexity function $c_{r,n}$ is defined as

$$c_{r,n}(\mathbf{t}, \mathbf{p}) = \sum_{l \in [r,n], k \in [1,m]} M[l, k] \quad (1)$$

where $1 \leq r \leq n$. For $r = 1$ the function $c_{1,n} := c_n$ is simply called the **complexity** of the algorithm. If either the pattern or the text is a realization of a random sequence, then we denote the complexity by a capital letter, that is, we write C_n instead of c_n .

Our goal is to find an asymptotic expression for c_n and C_n for large n under deterministic and stochastic assumptions regarding the strings \mathbf{p} and \mathbf{t} . However, for simplicity of notation we often write c_n instead of $c_n(\mathbf{t}, \mathbf{p})$. In order to accomplish this, we need some further definitions that will lead to a formal description of sequential algorithms.

We start with a definition of an *alignment position*.

Definition 2 Given a string searching algorithm, a text \mathbf{t} and a pattern \mathbf{p} , a position AP in the text \mathbf{t} satisfying for some k ($1 \leq k \leq m$)

$$M[AP + (k - 1), k] = 1$$

is said to be an alignment position.

Intuitively, at some step of the algorithm, an alignment of pattern \mathbf{p} at position AP is considered, and a comparison made with character $\mathbf{p}[k]$ of the pattern.

Finally, we are ready to define *sequential algorithms*. Sequentiality refers to a special structure of a sequence of positions that pattern and text visit during a string matching algorithm. Throughout, we shall denote these sequence as (l_i, k_i) where l_i refers to position visited during the i th comparison by the text while k_i refers to a position of the pattern when the pattern is aligned at position $l_i - k_i + 1$.

Definition 3 *A string searching algorithm is said:*

- (i) **semi-sequential** if the text is scanned from left to right;
- (ii) **strongly semi-sequential** if the order of text-pattern comparisons actually performed by the algorithm defines a non-decreasing sequence of text positions (l_i) and if the sequence of alignment positions is non-decreasing.
- (iii) **sequential** (respectively **strongly sequential**) if they satisfy, additionally for any $k > 1$

$$M[l, k] = 1 \Rightarrow \mathbf{t}_{l-(k-1)}^{l-1} = \mathbf{p}_1^{k-1} \quad (2)$$

In passing, we point out that condition (i) means that the text is read from left to right. Note that non-decreasing text positions implies (i). Furthermore, non-decreasing alignment positions implies that all occurrences of the pattern before this alignment position were detected before this choice. Nevertheless, these constraints on the sequence of text-pattern comparisons (l_i, k_i) are not enough to prevent the algorithm to “fool around”, and to guarantee a general tight bound on the complexity. Although (2) is not a logical consequence of semi-sequentiality, it represents a natural way of using the available information for semi-sequential algorithms. In that case, subpattern $\mathbf{t}_{l-(k-1)}^{l-1}$ is known when $\mathbf{t}[l]$ is read. There is no need to compare $\mathbf{p}[k]$ with $\mathbf{t}[l]$ if $\mathbf{t}_{l-(k-1)}^{l-1}$ is not the prefix of \mathbf{p} of size $k - 1$, i.e if $AP = l - (k - 1)$ has already been disregarded.

We now illustrate our definition on several examples.

Example 1: *Naive or brute force algorithm*

The simplest string searching algorithm is the naive one. All text positions are alignment positions. For a given one, say AP , text is scanned until the pattern is found or a mismatch occurs. Then, $AP + 1$ is chosen as the next alignment position and the process is repeated.

This algorithm is sequential but not strongly sequential. Condition in (ii) is violated after any mismatch on a alignment position l with parameter $k \geq 3$, as comparison $(l+1, 1)$ occurs after $(l+1, 2)$ and $(l+2, 3)$.

Example 2: *Morris-Pratt-like algorithms* [15].

It was already noted [15] that after a mismatch occurs when comparing $\mathbf{t}[l]$ with $\mathbf{p}[k]$, some alignment positions in $[l+1, \dots, l+k-1]$ can be disregarded without further text-pattern comparisons. Namely, the ones that satisfy $\mathbf{t}_{l+i}^{l+k-1} \neq \mathbf{p}_1^{k-i}$. Or, equivalently, $\mathbf{p}_{1+i}^k \neq \mathbf{p}_1^{k-i}$, and the set of such i can be known by a preprocessing of \mathbf{p} . Other i define the “surviving candidates”, and choosing the next alignment position among the surviving candidates is enough to *ensure* that condition (ii) in Definition 3 holds. Different choices lead to different variants of the classic Morris-Pratt algorithm [15]. They differ by the use of the information obtained from the mismatching position. We formally define three main variants, and provide an example. One defines a shift function S to be used after any mismatch as:

Morris-Pratt variant:

$$S = \min\{k-1; \min\{s > 0 : \mathbf{p}_{1+s}^{k-1} = \mathbf{p}_1^{k-1-s}\}\} ;$$

Knuth-Morris-Pratt variant:

$$S = \min\{k; \min\{s : \mathbf{p}_{1+s}^{k-1} = \mathbf{p}_1^{k-1-s} \text{ and } \mathbf{p}_k^k \neq \mathbf{p}_{k-s}^{k-s}\}\} ;$$

Simon variant:

$$\begin{aligned} K &= \max\{k : M(l, k) = 1\} ; \\ B &= \{s : \mathbf{p}_{1+s}^{K-1} = \mathbf{p}_1^{K-1-s} \text{ and } 0 \leq s \leq K-k\} ; \\ S &= \min\{d > 0 : \mathbf{p}_{1+d}^{k-1} = \mathbf{p}_1^{k-1-d} \text{ and } (\mathbf{p}_{k-d}^{k-d} \neq \mathbf{p}_{K-s}^{K-s}, s \in B)\} \end{aligned}$$

Figure 1 shows a generic program for all three variants of the algorithm.

Example 3: *Illustration to Definition 3.*

Let $\mathbf{p} = abacabacabab$ and $\mathbf{t} = abacabacabaaa$. The first mismatch occurs for $M(12, 12)$. The comparisons performed from that point are:

1. **Morris-Pratt variant:**

$$(12, 12); (12, 8); (12, 4); (12, 2); (12, 1); (13, 2); (13, 1) ,$$

```

{
    if t[l] = p[k] then
    {
        if k ≠ m then M(l + 1, k + 1)
        else
        {
            s := min{s; pl+sk = plk-s};
            M(l + 1, m - s + 1)
        }
    }
    else
    {
        Compute(S);
        if S = 0 then M(l + 1, 1)
        else M(l, k - S)
    }
}

```

Figure 1: Morris-Pratt-like Algorithms

where the text character is compared in turn with pattern characters (b, c, c, b, a, b, a) with the alignment positions $(1, 5, 9, 11, 12, 12, 13)$.

2. **Knuth-Morris-Pratt variant:**

$(12, 12); (12, 8); (12, 2); (12, 1); (13, 2); (13, 1)$,

where the text character is compared in turn with pattern characters (b, c, b, a, b, a) with the alignment positions $(1, 5, 11, 12, 12, 13)$.

3. **Simon variant:**

$(12, 12); (12, 8); (12, 1); (13, 2); (13, 1)$,

where the text character is compared in turn with pattern characters (b, c, a, b, a) with the alignment positions $(1, 5, 12, 12, 13)$.

In passing we note that several alignment positions can be consistent at a given step of the algorithm. Morris-Pratt variant considers one alignment position at a time, while the optimal

sequential algorithm, that of Simon, considers several alignment positions at the same time, and may disregard several of them simultaneously (e.g., in Example 3 positions 1 and 9 at the first step and 5 and 11 at the second step). It is interesting to observe that the subset $\{1, 5, 12\}$ appears in all variants. We will see that they share a common property of “unavoidability” explored below.

Our definition of semi-sequentiality is very close to the definition of sequentiality given in [12]. We do not use the “on-line” concept of [6]. Their efficient on-line algorithms are very close to our strongly sequential ones. Also, while condition (2) is a natural optimization for semi-sequential algorithms, it seems not to be true for other efficient algorithms discussed in [8].

Finally, in the course of proving our main result we discover an interesting structural property of sequential algorithms. Namely, that when the algorithm is run on a substring of the text, say \mathbf{t}_r^n , then there are some positions $i \geq r$ that are *unavoidable* alignment positions, that is, the algorithm *must* align at this positions at some step (e.g., see positions $\{1, 5, 12\}$). More formally:

Definition 4 *For a given a pattern \mathbf{p} , a position i in the text \mathbf{t}_1^n is an **unavoidable alignment position** for an algorithm if for any r, l such that $r \leq i$ and $l \geq i + m$, the position i is an alignment position when the algorithm is run on \mathbf{t}_r^l .*

Having in mind the above definitions we can describe our last class of sequential algorithms for which we formulate our main results.

Definition 5 *An algorithm is said to be l -convergent if, for any text \mathbf{t} and pattern \mathbf{p} , there exists an increasing sequence $(U)_i$ of unavoidable alignment positions satisfying $U_{i+1} - U_i \leq l$ where $U_0 = 0$ and $n - \max_i U_i \leq l$.*

In passing we note that clearly the naive pattern matching algorithm (cf. Ex. 1) is 1-convergent. We also prove that all strongly sequential algorithms (i.e., all Morris-Pratt-like algorithms) are m -convergent which will further imply several interesting and useful properties of these algorithms (e.g., linear complexity).

2.2 Main Results

In this section we formulate our main results. Before, however, we must describe modeling assumptions concerning the strings (cf. [17]). We adopt one of the following assumptions:

(A) WORST-CASE (DETERMINISTIC) MODEL

Both strings \mathbf{p} and \mathbf{t} are non random (deterministic) and \mathbf{p} is given.

(B) SEMI-RANDOM MODEL

The text string \mathbf{t} is a realization of a stationary and ergodic sequence while the pattern string \mathbf{p} is given.

(C) STATIONARY MODEL

Both strings \mathbf{p} and \mathbf{t} are random realization of a *stationary* and *ergodic* sequence (cf. [3]). (Roughly speaking, a sequence, say \mathbf{t}_1^n , is stationary if probability associated with any substrings of equal sizes is the same, that is, probability of, say \mathbf{t}_i^{i+k} is the same as the probability of \mathbf{t}_j^{j+k} for $1 \leq i < j \leq n$.)

Formulation of our results depends on the model we work with. So, in the deterministic model we interpret the complexity $c_n(\mathbf{t}, \mathbf{p})$ as the worst case complexity (i.e., we maximize the complexity over all texts). In the journal version of this paper, under assumption (C) we consider the strongest possible convergence of random variable C_n , namely *almost sure* (a.s.) convergence. More formally, we write $C_n/a_n \rightarrow \alpha$ (a.s.) where a_n is a deterministic sequence and α is a constant if $\lim_{n \rightarrow \infty} \Pr\{\sup_{k \geq n} |C_k/a_k - \alpha| > \varepsilon\} = 0$ for any $\varepsilon > 0$ (cf. [3]). Finally, in the semi-random model we use standard average case complexity, that is, EC_n .

Now we are ready to formulate our main results.

Theorem 2.1 *Consider an $l \leq m$ convergent sequential string matching algorithm. Let \mathbf{p} be a given pattern of length m .*

(i) *Under assumption (A) the following holds*

$$\lim_{n \rightarrow \infty} \frac{\max_{\mathbf{t}} c_n(\mathbf{t}, \mathbf{p})}{n} = \alpha_1(\mathbf{p}) \quad (3)$$

where $\alpha_1(\mathbf{p}) \geq 1$ is a constant.

(ii) *Under assumption (B) one finds*

$$\lim_{n \rightarrow \infty} \frac{E_{\mathbf{t}} C_n(\mathbf{p})}{n} = \alpha_2(\mathbf{p}) \quad (4)$$

where $\alpha_2(\mathbf{p}) \geq 1$ is a constant, and $E_{\mathbf{t}}$ denotes the the average over all text strings.

Theorem 2.2 *Consider an l -convergent sequential string matching algorithm. Under assumption (C) we have*

$$\lim_{n \rightarrow \infty} \frac{E_{\mathbf{t}, \mathbf{p}} C_n}{n} = \alpha_3 \quad (5)$$

provided $m = o(\sqrt{n})$, where $\alpha_3 \geq 1$ is a constant and $E_{\mathbf{t}, \mathbf{p}}$ denotes the average over all text strings and patterns.

Remark. Our proof presented in the next section will allow in fact an extension to almost sure convergence. We shall deal with this in the journal version of the paper.

Finally, with respect to our main class of algorithms, namely, Morris-Pratt like (i.e., sequential) we shall prove in the next section the following results concerning the existence of unavoidable positions.

Theorem 2.3 *Given a pattern \mathbf{p} and a text \mathbf{t} , all strongly sequential algorithms have the same set of unavoidable alignment positions $U = \bigcup_{l=1}^n \{U_l\}$, where*

$$U_l = \min\left\{\min_{1 \leq k \leq l} \{\mathbf{t}_k^l \preceq \mathbf{p}\}, l+1\right\} \quad (6)$$

and $\mathbf{t}_k^l \preceq \mathbf{p}$ means that the substring \mathbf{t}_k^l is a prefix of the pattern \mathbf{p} .

Theorem 2.4 *A strongly sequential algorithm is m -convergent and (3)-(5) hold.*

In summary, the above says that there exists a constant α such that $c_n = \alpha n + o(n)$ and/or $EC'_n = \alpha n + o(n)$. All previous results have been able only to show that $c_n = \Theta(n)$ but they did not excluded some bounded fluctuation of the coefficient at n . We should point out that in the analysis of algorithms on words such a fluctuation can occur in some problems involving suffix trees (cf. [4, 13]). For example, in the worst case model we proved here that for any given pattern \mathbf{p} , any $\epsilon > 0$ and any $n \geq n_\epsilon$, one can find a text \mathbf{t}_1^n such that $|\frac{c_{1,n}}{n} - \alpha_1(p)| \leq \epsilon$.

3 ANALYSIS

In this section we prove Theorems 2.1–2.4. The idea of the proof is quite simple. We shall show that a function of the complexity (i.e., $c'_n = c_n + f(m)$ where $f(m)$ is a function of the length m of the pattern \mathbf{p}) is subadditive. In the “average case analysis” we indicate that under assumption (C) the average complexity C'_n is a stationary and ergodic sequence. Then, direct application of an extension of Kingman’s *Subadditive Ergodic Theorem* due to Derriennic [9] will do the job of proving our results. In passing, we point out that the most challenging is establishing the subadditivity property to which most of this section is devoted. We observe, however, that subadditivity of the Morris-Pratt type algorithms is a consequence of the existence of unavoidable positions.

For the reader’s convenience we start this section with a brief review of the subadditive ergodic theorem (cf. [10, 14]).

Theorem 3.1 (Subadditive Sequence). (i) Let for a (deterministic) nonnegative sequence $\{x_n\}_{n=0}^\infty$ the following property, called subadditivity, holds

$$x_{m+n} \leq x_n + x_m. \quad (7)$$

Then

$$\lim_{n \rightarrow \infty} \frac{x_n}{n} = \inf_{m \geq 1} \frac{x_m}{m} = \alpha \quad (8)$$

for some constant α .

(ii) (Subadditive Ergodic Theorem [14]). Let $X_{m,n}$ ($m < n$) be a sequence of nonnegative random variables satisfying the following three properties

(a) $X_{0,n} \leq X_{0,m} + X_{m,n}$ (subadditivity);

(b) $X_{m,n}$ is stationary (i.e., the joint distributions of $X_{m,n}$ are the same as $X_{m+1,n+1}$) and ergodic (cf. [3]);

(c) $EX_{0,1} < \infty$.

Then,

$$\lim_{n \rightarrow \infty} \frac{EX_{0,n}}{n} = \gamma \quad \text{and} \quad \lim_{n \rightarrow \infty} \frac{X_{0,n}}{n} = \gamma \quad (\text{a.s.}) \quad (9)$$

for some constant γ .

(iii) (Almost Subadditive Ergodic Theorem [9]). If the subadditivity inequality is replaced by

$$X_{0,n} \leq X_{0,m} + X_{m,n} + A_n \quad (10)$$

such that $\lim_{n \rightarrow \infty} EA_n/n = 0$, then (9) holds, too. ■

Thus, to prove our main results we need to establish the subadditivity property for the complexity $c_n(\mathbf{t}, \mathbf{p})$ (for all texts \mathbf{t} and patterns \mathbf{p}). The next lemma proves such a result for l -convergent sequential algorithms.

Lemma 3.1 An l -convergent semi-sequential (or strongly semi-sequential) algorithm satisfies the basic inequality for all r such that $1 \leq r \leq n$:

$$|c_{1,n} - (c_{1,r} + c_{r,n})| \leq n^2 + lm, \quad (11)$$

provided any comparison is done only once.

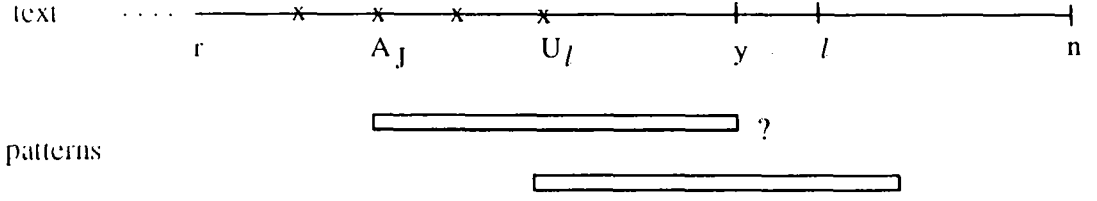


Figure 2: Illustration to the proof of Lemma 3.1: Contribution S_1 (cf. Figure 3(a)), and contribution S_3 (cf. Figure 3(b)).

Proof: Let U_r be the smallest unavoidable position greater than r . We evaluate in turn $c_{1,n} - (c_{1,r} + c_{U_r,n})$ and $c_{r,n} - c_{U_r,n}$ (cf. Figure 2).

We start our analysis with considering $c_{1,n} - (c_{1,r} + c_{U_r,n})$ (see dotted part in Figure 3(a)). This contribution consists of two parts. First, we must include those comparisons that are performed after position r for alignment positions before r . Observe that those comparisons contribute to $c_{1,n}$ but not to $c_{1,r}$. To avoid counting the last character r twice, we must subtract one comparison. Thus, this contribution, which we call S_1 , can be computed as follows (cf. Figure 3(a))

$$S_1 = \sum_{AP < r} \sum_{i \geq r} M(i, i - AP + 1) - 1 .$$

The second part of the contribution $c_{1,n} - (c_{1,r} + c_{U_r,n})$ accounts for alignments AP satisfying $r \leq AP \leq U_r$ that only contribute to $c_{1,n}$. It is easy to see that this part adds the following

$$S_2 = \sum_{AP=r}^{U_r-1} \sum_{i \leq m} M(AP + (i - 1), i) .$$

Observe now that the alignment positions after U_r on the text $\mathbf{t}_{U_r}^n$ and \mathbf{t}_1^n are the same. Thus, the only difference in contribution comes from the amount of information saved from previous comparisons done on \mathbf{t}_1^n . This is clearly bound by

$$|c_{1,n} - (c_{1,r} + c_{U_r,n} + S_1 + S_2)| \leq m .$$

Now, we evaluate $c_{r,n} - c_{U_r,n}$ (see Figure 3(b)). We assume that the algorithm runs on \mathbf{t}_r^n and let AP be any alignment position satisfying $r \leq AP < U_r$. Then the contribution S_3 :

$$S_3 = \sum_{AP=r}^{U_r-1} \sum_i M(AP + (i - 1), i)$$

counts for the number of comparisons associated to these positions. This sum is the same as S_2 but the associated alignment positions and searched text positions $AP + k - 1$ may be

different. Additionally, at alignment position U_r , no more than m comparisons can be saved from previous comparisons. Hence, we get:

$$|c_{r,m} - c_{U_r,m} - S_3| \leq m.$$

Finally, it remains to find upper bounds on S_1 , S_2 , and S_3 . For $l \geq U_r - r$ we easily see that S_2 and S_3 are smaller than lm . So is their difference. With respect to S_1 , for a given alignment position AP we have $|i - AP| \leq m$. This implies that $|r - AP| \leq m$, and for any AP the index i has at most m different values. Thus, $S_1 \leq m^2$. This completes the proof. ■

Now we are ready to prove l -convergence for strongly sequential algorithms, i.e. Theorem 2.4. It relies on Theorem 2.3 which we prove first. Let l be a text position such that $1 \leq l \leq n$, and r be any text position satisfying $r \leq U_l$. Let $\{A_i\}$ be the set of alignment positions defined by a strongly sequential algorithm that runs on \mathbf{t}_r^n . As it contains r , we may define (cf. Figure 3).

$$A_J = \max\{A_i : A_i < U_l\}$$

Hence, we have $A_{J+1} \geq U_l$. Using an adversary argument, we shall prove that $A_{J+1} > U_l$ cannot be true, thus showing that $A_{J+1} = U_l$. Let $y = \max\{k : M(A_J + (k - 1), k) = 1\}$, that is, y is the rightest point we can go starting from A_J . We observe that we have $y \leq l$. Otherwise, according to condition (2), we would have $\mathbf{t}_{A_J}^l \preceq \mathbf{p}$, which contradicts the definition of U_l . Also, semi-sequentiality implies that $A_{J+1} \leq y + 1 \leq l + 1$. Hence $U_l = l + 1$ contradicts the assumption $A_{J+1} > U_l$ and we may assume $U_l \leq l$. In that case, $p_{U_l}^l \preceq \mathbf{p}$ and an occurrence of \mathbf{p} at position U_l is consistent with the available information. Let the adversary chose that \mathbf{p} does occur. As sequence (A_i) is non-decreasing and A_{J+1} has been chosen greater than U_l , this occurrence will not be detected by the algorithm: a contradiction. This completes the proof.

■

Finally, we turn to the proof of Theorem 2.4. Let AP be some alignment position and define $l = AP + m$. As $|\mathbf{p}| = m$, one has $l - (m - 1) \leq U_l \leq l$. Hence, $U_l - AP \leq m$ which establishes the m -convergence.

Remark: We use the strong monotonicity assumption only in the establishment of unavoidable positions. This property may not be satisfied by other algorithms such as efficient “on-line” algorithms in [6] that still one may intuitively call sequential. Nevertheless, it is easy to extend our proof of subadditivity when we remove this condition. Notably, unavoidability still holds for non-periodic patterns \mathbf{p} , by a witness argument.

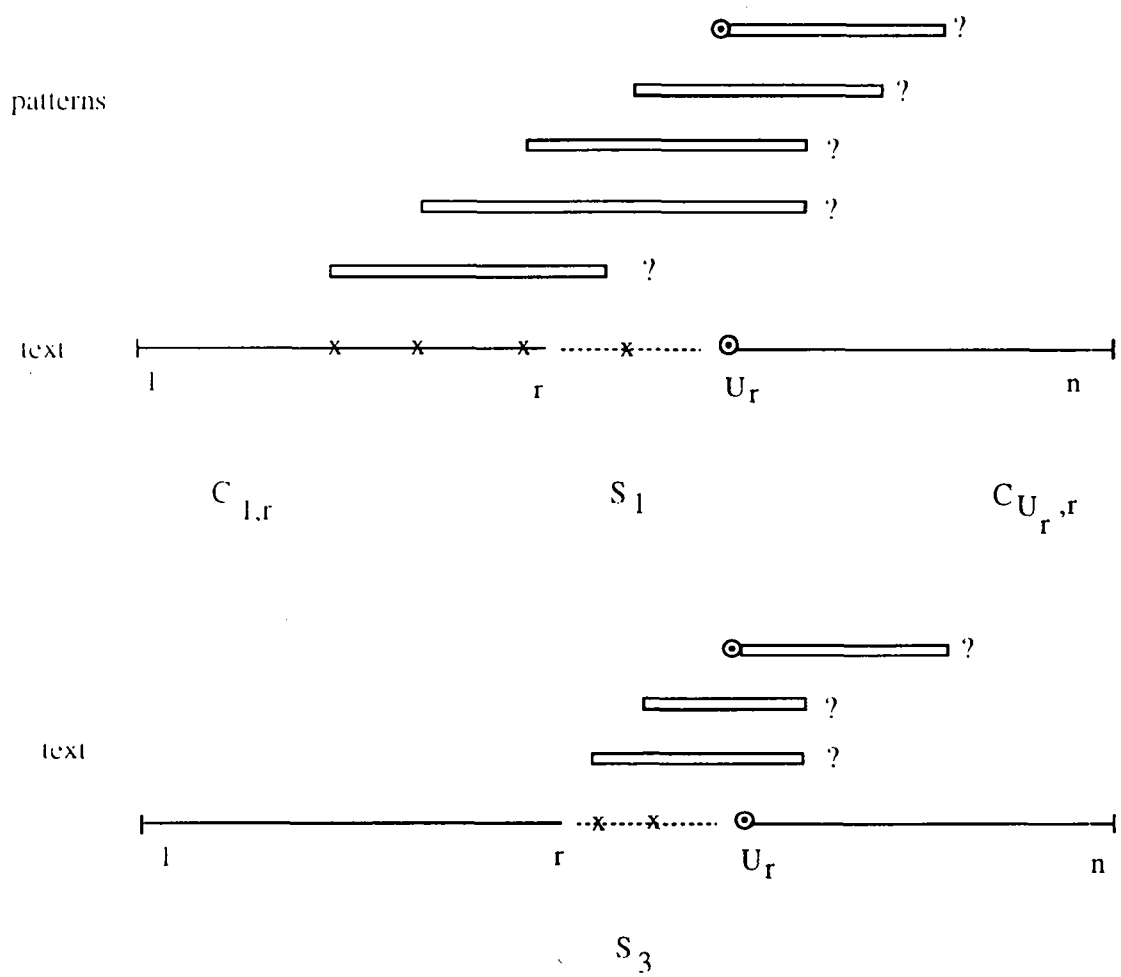


Figure 3: Illustration to the proof of Theorem 2.4.

We now apply Theorem 3.1 to derive Theorems 2.1 and 2.2. After substituting $x_n = c_n + 1.5m^2 + lm$, we get subadditivity for any given \mathbf{p} and deterministic \mathbf{t} by Theorem 3.1(iii). Now, let \mathbf{t}_1^n range over the set of texts of size n , \mathbf{t}_1^r and \mathbf{t}_r^n range over the sets of texts of size r and $n - r$. Then, the subadditivity holds for the worst case complexity since

$$\max_{|\mathbf{t}|=n} c_{1,n} \leq \max_{|\mathbf{t}|=r} c_{r,n} + \max_{|\mathbf{t}|=n-r} c_{r,n}$$

and for the average complexity as well:

$$EC_{1,n} \leq EC_{1,r} + EC_{r,n} ,$$

and this completes the proof. ■

4 CONCLUDING REMARKS

We consider here sequential algorithms that are variants of classical Morris-Pratt algorithms. In order to speed up the search, Boyer and Moore introduced in [5] a quite different algorithm. Given an alignment position AP , matching against \mathbf{p} are checked from right to left; i.e. k is decreasing. Several variants have been proposed that differ by the amount of information saved to compute the next alignment position.

We point out here that Boyer-Moore like algorithms do not satisfy unavailability property. We provide an example for the Horspool variant: given an alignment position AP , the next alignment position is computed by aligning the text character $\mathbf{t}[AP + m]$ with $\mathbf{t}[AP + j]$ where

$$m - j = \min\{\max\{k : \mathbf{p}[k] = \mathbf{t}[AP + m]\}, m\}$$

Let us now consider as an example $\mathbf{p} = x^4ax^2bx^2a$, $x \neq a, b$. When $\mathbf{t}[AP + m]$ is a (resp. b or x) the next alignment position is chosen to be $AP + 6$ (resp. $AP + 3$ or $AP + 1$). When $\mathbf{t}[AP + m] \notin \{a, b, x\}$, one shifts the alignment position by m . Assume now that $\mathbf{t} = y^{10}az^4(bazbz^2)^n$ with $y \neq x$ and natural n . If the Boyer-Moore-Horspool algorithm starts with $AP = 1$, a mismatch occurs on the second comparison between $\mathbf{t}[10]$ and $\mathbf{p}[10]$ with AP shifted by 6. The same event occurs then and we eventually get the sequence $AP_i = 1 + 6i$. Assume now that we split the text at $r = 6$. As $\mathbf{t}[16]$ is b , one shifts by 3 and b is found again. Finally, one gets sequence $AP'_i = 6 + 3i$. As $\gcd(6, 3)$ does not divide 5, these two sequences are disjoint and there is no unavoidable position.

In summary, we conclude that subadditivity and unavailability cannot be used to prove linearity of Boyer-Moore algorithms. Nevertheless, another tool from stochastic analysis, namely renewal theory, should provide a solution. We plan to investigate it in a future paper.

References

- [1] A. Apostolico and R. Giancarlo, The Boyer-Moore-Galil String Searching Strategies Revisited, *SIAM J. Comput.*, 15, 98-105, 1986.
- [2] R. Baeza-Yates and M. Régnier, Average Running Time of Boyer-Moore-Horspool Algorithm, *Theoretical Computer Science*, 92, 19-31, 1992.
- [3] P. Billingsley, *Convergence of Probability Measures*, John Wiley & Sons, New York, 1968.
- [4] A. Blumer, A. Ehrenfeucht and D. Haussler, Average Size of Suffix Trees and DAWGS, *Discrete Applied Mathematics*, 24, 37-45 (1989).
- [5] R. Boyer and J. Moore, A fast String Searching Algorithm, *Comm. of the ACM*, 20, 762-772, 1977.
- [6] D. Breslauer, L. Colussi, and L. Toniolo, Tight Comparison Bounds for the String Prefix-Matching Problem, *Proc. 4-th Symposium on Combinatorial Pattern Matching*, Padova, Italy, 11-19, Springer-Verlag, 1993.
- [7] R. Cole, R. Hariharan, M. Paterson, and U. Zwick, Tighter Lower Bounds on the Exact Complexity of String Matching, *SIAM J. Comp.*, 24, 30-45, 1995.
- [8] L. Colussi, Z. Galil, and R. Giancarlo, On the Exact Complexity of String Matching, *Proc. 31-st Annual IEEE Symposium on the Foundations of Computer Science*, 135-143, IEEE, 1990.
- [9] Y. Derriennic, Une Théorème Ergodique Presque Sous Additif, *Ann. Probab.*, 11, 669-677, 1983.
- [10] R. Durrett, *Probability: Theory and Examples*, Wadsworth & Brooks/Cole Books, Pacific Grove, California, 1991.
- [11] L. Guibas and A. Odlyzko, A New Proof of the Linearity of the Boyer-Moore String Matching Algorithm, *SIAM J. Comput.*, 9, 672-682, 1980.
- [12] C. Hancart, *Analyse Exacte et en Moyenne d'Algorithmes de Recherche d'un Motif dans un Texte*, These, l'Université Paris 7, 1993.
- [13] P. Jacquet and W. Szpankowski, Autocorrelation on Words and Its Applications. Analysis of Suffix Tree by String-Ruler Approach, *J. Combinatorial Theory. Ser. A*, 66, 237-269, 1991.

- [14] J.F.C. Kingman, *Subadditive Processes*, in Ecole d'Eté de Probabilités de Saint-Flour V-1975, Lecture Notes in Mathematics, 539, Springer-Verlag, Berlin 1976.
- [15] D.E. Knuth, J. Morris and V. Pratt, Fast Pattern Matching in Strings, *SIAM J. Comput.*, 6, 189-195, 1977.
- [16] M. Régnier, Knuth-Morris-Pratt Algorithm: An Analysis, *Proc. Mathematical Foundations for Computer Science 89*, Porubka, Poland, *Lecture Notes in Computer Science*, vol. 379, 431-441, Springer-Verlag, 1989.
- [17] W. Szpankowski, On The Height of Digital Trees and Related Problems, *Algorithmica*, 6, 256-277, 1991.



Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine - Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes - IRISA, Campus universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
Unité de recherche INRIA Sophia Antipolis - 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249 - 6399



★ R R . 2 5 4 9 ★