On the Suffix Automaton with Mismatches

4 authors:

Maxime Crochemore
King's College London
357 PUBLICATIONS   **6,079** CITATIONS

SEE PROFILE

Chiara Epifanio
Università degli Studi di Palermo
12 PUBLICATIONS   **100** CITATIONS

SEE PROFILE

Alessandra Gabriele
Università degli Studi di Palermo
10 PUBLICATIONS   **83** CITATIONS

SEE PROFILE

Filippo Mignosi
Università degli Studi dell'Aquila
83 PUBLICATIONS   **1,559** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

combinatorics View project

Motifs Bases View project

On the Suffix Automaton with mismatches [*]

Maxime Crochemore[1], Chiara Epifanio[2],
Alessandra Gabriele[2], Filippo Mignosi[3]

[1] Institut Gaspard-Monge, Université de Marne-la-Vallée, France and King's College
London, UK, mac@univ-mlv.fr
[2] Dipartimento di Matematica e Applicazioni, Università di Palermo, Italy
(epifanio,sandra)@math.unipa.it
[3] Dipartimento di Informatica, Università dell'Aquila, Italy mignosi@di.univaq.it

Abstract. In this paper we focus on the construction of the minimal
deterministic finite automaton S_k that recognizes the set of suffixes of a
word w up to k errors. We present an algorithm that makes use of the
automaton S_k in order to accept in an efficient way the language of all
suffixes of w up to k errors in every windows of size r, where r is the value
of the repetition index of w. Moreover, we give some experimental results
on some well-known words, like prefixes of Fibonacci and Thue-Morse
words, and we make a conjecture on the size of the suffix automaton
with mismatches.

Keywords: Combinatorics on words, suffix automata, languages with mismatches,
approximate string matching.

1 Introduction

One of the seminal results in string matching is that the size of the suffix au-
tomaton of a word, called also DAWG, is linear [4, 10]. In the particular case of
prefixes of the Fibonacci word, a result by Carpi and de Luca [6] implies that
the suffix automaton of any prefix v of the Fibonacci word f has $|v| + 1$ states.

These results are surprising as the maximal number of subwords that may
occur in a word is quadratic according to the length of the word. Suffix trees
are linear too, but they represent strings by pointers to the text, while DAWGs
work without the need of accessing it.

In this work we are interested in an extension of suffix automata, more pre-
cisely we consider the DAWG recognizing the set of occurrences of a word w up to
k errors. Literature on data structures recognizing languages with mismatches
involves many results, among the most recent ones [2, 5, 7, 8, 12, 13, 15, 18, 22].
Several of these papers deal with approximate string matching. In particular,
in [12, 13, 15] authors have considered some data structures recognizing words
occurring in a text w up to k errors in each substring of length r of the text.

The presence of a window in which allowing a fixed number of errors generalizes the classical k-mismatch problem and, at the same time, it allows more errors in all. Moreover, this approach has a specific interpretation in Molecular Biology, such as, for instance, the modeling of some evolutionary events.

In this paper we focus on the minimal deterministic finite automaton that recognizes the set of suffixes of a word w up to k errors, denoted by $S_{w,k}$, or simply by S_k if there are no risks of misunderstanding on w.

As first main result we give a characterization of the Nerode's right-invariant congruence relative to S_k. This result generalizes a result described in [4] (see also [9, 19]), where it was used in an efficient construction of the suffix automaton with no mismatches, that, up to the set of final states, is also called DAWG (directed acyclic word graph). We think that it is possible to define such an algorithm even when dealing with mismatches. It would be probably more complex than the classical one. It still remains an open problem how to define it.

As a second main result, we describe an algorithm that makes use of the automaton S_k in order to accept, in an efficient way, the language of all suffixes of w up to k errors in every windows of size r, for a specific integer r called repetition index.

We have constructed the suffix automaton with mismatches of a great number of words and we have considered overall its structure when the input word is well-known, such as the prefixes of Fibonacci and Thue-Morse words, as well as words of the form bba^n, $a, b \in \Sigma, n \geq 1$ and some random words generated by memoryless sources. We have studied how the number of states grows depending on the length of the input word. By the results of our experiments on these classes of words, we conjecture that the (compact) suffix automaton with k mismatches of any text w has size $O(|w| \cdot \log^k(|w|))$. Given a word v, Gad Landau wondered if a data structure having a size "close" to $|v|$ that allows approximate pattern matching in time proportional to the query plus the number of occurrences exists. This question is still open, even if recent results are getting closer to a positive answer. If our conjecture turns out to be true, it would settle Landau's question as discussed at the end of this paper.

The remainder of this paper is organized as follows. In the second section we give some basic definitions. In the third section we describe a characterization of the Nerode's right invariant congruence relative to S_k. The fourth section is devoted to describe an algorithm that makes use of the automaton S_k in order to accept in an efficient way the language of all suffixes of w up to k errors in every window of size r, where r is the value of the repetition index of w. The fifth section contains our conclusions and some conjectures on the size of the suffix automaton with mismatches based on our experimental results. Finally, appendix contains the proofs of some results.

2 Basic definitions

Let Σ be a finite set of symbols, usually called *alphabet*. A *word* or *string* w is a finite sequence $w = a_1 a_2 \ldots a_n$ of characters in the alphabet Σ, its length (i.e.

the number of characters of the string) is defined to be n and it is denoted by $|w|$. The set of words built on Σ is denoted by Σ^* and the empty word by ϵ. We denote by Σ^+ the set $\Sigma^* \setminus \{\epsilon\}$.

A word $u \in \Sigma^*$ is a *factor* (resp. a *prefix*, resp. a *suffix*) of a word w if and only if there exist two words $x, y \in \Sigma^*$ such that $w = xuy$ (resp. $w = uy$, resp. $w = xu$). Notice that some authors call *substring* what we have defined as factor. We denote by *Fact(w)* (resp. *Pref(w)*, resp. *Suff(w)*) the set of all factors (resp. prefixes, resp. suffixes) of a word w. We denote an occurrence of a factor in a string $w = a_1 a_2 \ldots a_n$ at position i ending at position j by $w(i, j) = a_i \ldots a_j$, $1 \leq i \leq j \leq n$. The length of a factor $w(i, j)$ is the number of letters that compose it, i.e. $j - i + 1$. We say that u *occurs* in w at position i if $u = w(i, j)$, with $|u| = j - i + 1$.

In order to handle languages with errors, we need a notion of distance between words. In this work we consider the *Hamming distance*, that is defined between two words x and y of the same length as the minimal number of character substitutions that transform x into y.

In the field of approximate string matching, typical approaches for finding a string in a text consist in considering a percentage D of errors, or fixing the number k of them. Instead, we use an hybrid approach introduced in [15] that considers a new parameter r and allow at most k errors for any factor of length r of the text. We have the following definition.

Definition 1. *Let w be a string over the alphabet Σ, and let k, r be non negative integers such that $k \leq r$. A string u occurs in w at position l up to k errors in a window of size r or, simply, k_r-occurs in w at position l, if one of the following two conditions holds:*

- $|u| < r \Rightarrow d(u, w(l, l + |u| - 1)) \leq k$;
- $|u| \geq r \Rightarrow \forall i, 1 \leq i \leq |u| - r + 1, \quad d(u(i, i + r - 1), w(l + i - 1, l + i + r - 2)) \leq k$.

A string u satisfying the above property is a k_r-occurrence of w. A string u that k_r-occurs as a suffix of w is a k_r-suffix of w.

We suppose that the text is non-empty, $r \geq 2$ and $0 \leq k \leq r$, otherwise the above definition would have no meaning. We denote by $L(w, k, r)$ (resp. $Suff(w, k, r)$) the set of words (resp. suffixes) u that k_r-occur in w at position l, for some l, $1 \leq l \leq |w| - |u| + 1$. Notice that $L(w, k, r)$ is a *factorial language*, i.e. if $u \in L(w, k, r)$ then each factor (or substring) of u belongs to $L(w, k, r)$. Moreover, we denote by $Suff(w, k)$ the set of k_r-suffixes of w for $r = |w|$.

Remark 1. The condition $r = |w|$ is equivalent to the fact that we do not want to consider a window in which it is possible to allow errors. Indeed, when the size r of the window is equal to the size of the text w, then the problem of finding all k_r-occurrences of a string u in the text is equivalent to the k-mismatch problem, that consists in finding all occurrences of the string u in w with at most k errors (cf. [17]).

Example 1. Let $w = abaa$ be a string on the alphabet $\Sigma = \{a, b\}$. The set of words that k_r-occur in w, when $k = 1$ and $r = 2$, is $L(w, 1, 2) = \{a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaa, aaab, abaa, abab, abba, bbaa, bbab, bbba\}$. Notice that words $aab, aaab, bbab, bbba$ occur with one error every $r = 2$ symbols, but with two errors in the whole word. Hence, they belong to $L(w, 1, 2)$, but not to $L(w, 1, 4)$.

Moreover, $\mathit{Suff}(w, 1, 2) = \{a, b, aa, ab, ba, aaa, aab, baa, bab, bba, aaaa, aaab, abaa, abab, abba, bbaa, bbab, bbba\}$ and $\mathit{Suff}(w, 1) = \{a, b, aa, ab, ba, aaa, baa, bab, bba, aaaa, abaa, abab, abba, bbaa\}$.

Now we can recall the definition of the *repetition index*, denoted by $R(w, k, r)$, that plays an important role in the construction of an automaton recognizing the language $L(w, k, r)$ (cf.[15]).

Definition 2. *The* repetition index *of a string w, denoted by $R(w, k, r)$, is the smallest integer h such that all strings of this length k_r-occur at most once in the text w.*

The parameter $R(w, k, r)$ is well defined because the integer $h = |w|$ satisfies the condition. Moreover, it is easy to prove that if $\frac{k}{r} \geq \frac{1}{2}$ then $R(w, k, r) = |w|$ (cf. [12]).

In [12] it is proved that $R(w, k, r)$ is a non-increasing function of r and a non-decreasing function of k and that the equation $r = R(w, k, r)$ admits an unique solution. Authors also give an evaluation of the repetition index. More precisely, they prove that, under some hypothesis, $R(w, k, r)$ has a logarithmic upper bound in the size of the text w almost surely.

Remark 2. In [15] authors gave an algorithm for building a deterministic finite automaton (DFA) recognizing the language $L(w, k, r)$ of all words that k_r-occur in the string w. They proved that the size of such automaton $\mathcal{A}(w, k, r)$ is bounded by a function that depends on the length $|w|$ of the text w, the repetition index $R(w, k, r)$ and the number k of errors allowed in a window of size $r = R(w, k, r)$, that is $|\mathcal{A}(w, k, r)| = O(|w| \cdot (R(w, k, r))^{k+1})$. In the worst case, when both $R(w, k, r)$ and k are proportional to $|w|$, the size of the automaton $\mathcal{A}(w, k, r)$ is exponential. But, under the hypothesis that w is a sequence generated by a memoryless source with identical symbol probabilities and the number k of errors is fixed for any window of size $r = R(w, k, r)$, the size of this automaton is $O(|w| \cdot \log^{k+1}(|w|))$ almost surely.

Starting from the automaton $\mathcal{A}(w, k, r)$, an automaton recognizing the language $\mathit{Suff}(w, k, r)$ can be simply deduced from a procedure that first builds the automaton $\mathcal{A}(w\$, k, r)$, extending the alphabet Σ by letter $\$$, then sets as terminal states only those states from which an edge by letter $\$$ outgoes, and finally removes all edges labeled $\$$ and the state they reach [9].

In this paper we focus on the minimal automaton recognizing $\mathit{Suff}(w, k)$. It is therefore natural to study the Nerode's congruence corresponding to it.

3 On the Nerode's congruence

In this section, we introduce a right-invariant congruence relation on Σ^* used to define the suffix automaton of a word up to mismatches and we prove some properties of it. In particular we give a characterization of the Nerode's congruence relative to S_k. This result generalizes a classical result described in [4] (see also [9, 19]), where it was used in an efficient construction of the suffix automaton with no mismatches, that is also called DAWG (directed acyclic word graph), up to the set of final states. We think that it is possible to define such an algorithm even when dealing with mismatches. It would be probably more complex than the classical one. It still remains an open problem how to define it. Let us start by introducing the following definition, that is a generalization of the one given in [4].

Definition 3. *Let $w = a_1 \ldots a_n$ be a word in Σ^*. For any nonempty word $y \in \Sigma^*$, the end-set of y in w up to k mismatches, denoted by end-set$_w(y,k)$, is the set of all final positions in which y k-occurs in w, i.e. end-set$_w(y,k) = \{i \mid y$ k-occurs in w with final position $i\}$. Notice that end-set$_w(\epsilon, k) = \{0, 1, \ldots, n\}$.*

By using Definition 3 it is possible to define a equivalence relation between words on Σ^*.

Definition 4. *Two words x and y in Σ^* are end$_k$-equivalent, or $\equiv_{w,k}$, on w if the following two conditions hold.*

1. *end-set$_w(x,k) =$ end-set$_w(y,k)$;*
2. *for any position $i \in$ end-set$_w(x,k) =$ end-set$_w(y,k)$, the number of errors available in the suffix of w having $i+1$ as first position is the same after the reading of x and of y, i.e. $\min\{|w|-i, k-err_i(x)\} = \min\{|w|-i, k-err_i(y)\}$, where $err_i(u)$ is the number of mismatches of the word u that k_r-occurs in w with final position i.*

We denote by $[x]_{w,k}$ the equivalence class of x with respect to $\equiv_{w,k}$. The degenerate class is the equivalence class of words that are not k-occurrences of w (i.e., words with empty end-set in w up to k mismatches).

In other words, two words x and y in Σ^* are end$_k$-equivalent if, besides having the same end-set in w up to k mismatches as in the exact case [4], the number of errors available in the suffix of w after the reading of x and of y is the same. The definition includes two cases depending on the considered final position $i \in$ end-set$_w(x,k) =$ end-set$_w(y,k)$ of x and y in w:

2.a) if this position is sufficiently "far from" the end of the word, which means that $|w| - i \geq \max\{k - err_i(x), k - err_i(y)\}$, then the number of errors available after this position is the same in both cases, i.e. $k - err_i(x) = k - err_i(y)$, which implies that $err_i(x) = err_i(y)$. In this case

$$\min\{|w|-i, k-err_i(x)\} = k-err_i(x) = k-err_i(y) = \min\{|w|-i, k-err_i(y)\}.$$

2.b) otherwise, if this position is sufficiently "near" the end of the word, which means that $|w| - i \leq \min\{k - err_i(x), k - err_i(y)\}$, then it is possible to have mismatches in any position of the suffix of w having length $|w| - i$. This does not necessarily imply that $err_i(x) = err_i(y)$. Therefore

$$\min\{|w| - i, k - err_i(x)\} = |w| - i = \min\{|w| - i, k - err_i(y)\}.$$

Example 2. Let us consider the prefix of length 10 of the Fibonacci word, $w = abaababaab$, and let us suppose that the number k of errors allowed in any factor is 2.

- If we consider words $x = baba$ and $y = babb$, one has that they have the same *end-set*, that is $end\text{-}set_w(baba, 2) = \{5, 6, 8, 10\} = end\text{-}set_w(babb, 2)$, but the two words are not end_k-equivalent because it is not true that for any position $i \in end\text{-}set_w(baba, 2) = end\text{-}set_w(babb, 2)$, the number of errors available in the suffix of w having $i+1$ as first position is the same after the reading of x and of y. In fact, if we consider $i = 5$, $err_5(baba) = 2$ and $err_5(babb) = 1$ and then $\min\{|w| - 5, 2 - err_5(baba)\} = 0 \neq 1 = \min\{|w| - 5, 2 - err_5(babb)\}$.
- If we consider words $x = abaababa$ and $y = baababa$, one has that they are trivially end_k-equivalent because they have the same *end-set*, that is $end\text{-}set_w(abaababa, 2) = \{8\} = end\text{-}set_w(baababa, 2)$, and for $i = 8$ the number of errors available in the suffix of w having $i+1$ as first position is the same after the reading of x and of y. In fact, if we consider $i = 8$, $err_8(abaababa) = 0$ and $err_8(baababa) = 0$ and then $\min\{|w| - 8, 2 - err_8(abaababa)\} = 2 = \min\{|w| - 8, 2 - err_8(baababa)\}$.
- If we consider words $x = abaababaa$ and $y = baababab$, one has that they have the same *end set*, that is $end\text{-}set_w(abaababaa, 2) = \{9\} = end\text{-}set_w(baababab, 2)$, and for $i = 9$ the number of errors available in the suffix of w having $i + 1$ as first position is the same after the reading of x and of y, even if $err_9(abaababaa) = 0$ and $err_9(baababab) = 1$. In fact, one has that $\min\{|w| - 9, 2 - err_9(abaababaa)\} = 1 = \min\{|w| - 9, 2 - err_9(baababab)\}$, and then x and y are end_k-equivalent.

The following lemma and theorem summarize some properties of end_k-equivalence.

Lemma 1. *(i)* $\equiv_{w,k}$ *is a right-invariant equivalence relation on* Σ^*.
(ii) If x and y are end_k-equivalent, then one is a suffix of the other up to $2k$ errors.
(iii) Words xy and y are end_k-equivalent if and only if for any $i \in end\text{-}set_w(xy, k) = end\text{-}set_w(y, k)$, the k-occurrence of y with final position i is immediately preceded by a t-occurrence of x, where $t = \max\{(k - err_i(y)) - (|w| - i), 0\}$.

Theorem 1. *Words x and y are end_k-equivalent if and only if they have the same future in w, i.e. for any $z \in \Sigma^*$, xz is a k-suffix of w if and only if yz is a k-suffix of w.*

In what follows we use the term *partial* DFA (with respect to the alphabet Σ) for a deterministic finite automaton in which each state has not necessarily a transition for every letter of Σ. The smallest partial DFA for a given language

is the partial DFA that recognizes the language and has the smallest number of states. It is called the minimal DFA recognizing the language. Uniqueness follows from Nerode's Theorem [20] of the right invariant equivalence relation. As usual, an equivalence relation $\equiv$ on Σ^* is *right invariant* if, for any $x, y, z \in \Sigma^*$, $x \equiv y$ implies that $xz \equiv yz$.

By using Nerode's theorem and by Theorem 1 we have the following result.

Corollary 1. *For any word $w \in \Sigma^*$, the (partial) deterministic finite automaton having input alphabet Σ, state set $\{[x]_{w,k} \mid x \text{ is a } k \text{ occurrence of } w\}$, initial state $[\epsilon]_{w,k}$, accepting states those equivalence classes that include the k-suffixes of w (i.e., whose end-sets include the position $|w|$) and transitions $\{[x]_{w,k} \xrightarrow{a} [xa]_{w,k} \mid x \text{ and } xa \text{ are } k\text{-occurrences of } w\}$, is the minimal deterministic finite automaton, denoted by $\mathcal{S}_{w,k}$ (or simply by S_k if there are no risks of misunderstanding on w), which recognizes the set $Suff(w, k)$.*

Remark 3. We note that $Suff(w,k)=Suff(w,k,r)$ with $r = |w|$ (which is equivalent to saying that there are at most k errors in the entire word without window) and that $Suff(w,k,r) \subseteq L(w, k, r)$.

4 Allowing more mismatches

In this section we present the second main result of the paper. More precisely, we describe an algorithm that makes use of the automaton S_k in order to accept, in an efficient way, the language $Suff(w, k, r)$ of all suffixes of w up to k errors in every window of size $r = R(w, k, r)$.

First of all we recall that if $r = R(w, k, r)$ then for any value $x \geq r$ one has that $r = R(w, k, x)$, i.e. the repetition index gets constant. And this is also valid when the parameter x is such that $x = |w|$, which implies that if $r = R(w, k, r)$ then $r = R(w, k, |w|) = R(w, k)$ (cf. [12–14]). These two extremal cases are the two cases we are considering. From now on we denote this value simply by r. This fact implies that any word u of length $|u| = r$ has the following property: if u k_r-*occurs* or k-*occurs in* w, then, in both cases, it occurs only once in the text $|w|$.

Before describing our algorithm, we give a preliminary result that is important both for the following and in itself.

Lemma 2. *Given the automaton S_k, there exists a linear time algorithm that returns the repetition index r such that $r = R(w, k, r)$.*

Remark 4. As a side effect of this construction, each state of the automaton S_k is equipped with an integer that represents a distance from this state to the end. For this purpose, it is sufficient to make a linear time visit of the automaton.

Now we can describe the algorithm that accepts the language $Suff(w, k, r)$. We can distinguish two cases.

i) If a word u has length less than or equal to $r = R(w, k, r)$, then we check if the word u is accepted by the automaton S_k. If u is accepted by this automaton, then it is in the language $Suff(w, k, r)$, otherwise it does not belong to the language.

ii) If a word u has a length greater than or equal to $r = R(w, k, r)$, then we consider its prefix u' of length $r = R(w, k, r)$. Let q be the state that is reached after reading u' and let i be the integer associated to this state (cf. Remark 4). We have that $|w| - i - r + 1$ is the unique possible initial position of u. Given a position, checking whether a word k_r-occurs at that position in the text w can be done in linear time.

5   Conclusions and experimental results

We have constructed the suffix automaton with mismatches of a great number of words and we have considered overall its structure when the input word is well-known, such as the prefixes of Fibonacci and Thue-Morse words, as well as words of the form bba^n, $a, b \in \Sigma, n \geq 1$ and some random words. We have studied how the number of states grows depending on the length of the input word. In the case of the prefixes of length n of the Fibonacci word, our experimental results have led us to the following sequence $\{a_n\}_n$, representing the number of states of the suffix automaton with one mismatch: $\{a_n\}_n = 2, 4, 6, 11, 15, 18, 23, 28, 33, 36, 39, 45, 50, 56, 61, 64, 67, 70, 73, 79, 84, 90,$ $96, 102, 107, 110, 113, 116, 119, 122, 125, 128, 134, 139, 145, 151, 157, 163, 169, 175,$ $180, 183, 186, 189, 192, 195, 198, 201, 204, 207, 210, 213, 216, 222, 227, 233, 239, 245,$ $251, 257, 263, 269, \dots$ This means that the sequence of differences between two consecutive terms is: $\{a_{n+1} - a_n\}_n = 2, 2, 5, 4, 3, 5, 5, 5, 3, 3, 6, 5, 6, 5, 3, 3, 3, 3, 6, 5,$ $6, 6, 6, 5, 3, 3, 3, 3, 3, 3, 3, 6, 5, 6, 6, 6, 6, 6, 6, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 6, 5, 6, 6, 6,$ $6, 6, 6, 6, 6, 6, 6, 6, 5, 3, 3, 3, 3, 3, 3, 3, 3, \dots$

If we examine the last sequence, we note that after an initial part, there is one 6, one 5, $(fib_{i-1} - 2)$ consecutive 6s, one 5, $(fib_i - 1)$ consecutive 3s, etc, where fib_i denotes the i-th Fibonacci number, $i = 4, 5, 6 \dots$. This leads to the following recursive formula: $a_{fib_n} = a_{fib_{n-1}} + 3(a_{fib_{n-3}} - 1) + 10 + 6(a_{fib_{n-4}} - 1)$. From this recursion an explicit formula is easy to find. We did not prove the rule that describes the growth of the suffix automaton with one mismatch, but we checked that this rule holds true up to prefixes of length 2000 of the Fibonacci word f.

Conjecture 1. The size of the suffix automaton with one mismatch of the prefixes of the Fibonacci word grows according to the recursive formula $a_{fib_n} = a_{fib_{n-1}} + 3(a_{fib_{n-3}} - 1) + 10 + 6(a_{fib_{n-4}} - 1)$.

Given a word v, Gad Landau wondered if a data structure having a size "close" to $|v|$ and that allows approximate pattern matching in time proportional to the query plus the number of occurrences exists. In the non approximate case, suffix trees and compact suffix automata do the job (cf. [11, 19]). Let us see the approximate case. In [12, 14, 15, 21, 22] it is proved that for a random text w, the size of its compact suffix automaton with k mismatches is linear times a

polylog of the size of w, i.e. $O(|w| \cdot \log^k |w|)$. By using this data structure, the time for finding the list $occ(x)$ of all occurrences of any word x in the text w up to k mismatches is proportional to $|x| + |occ(x)|$. Therefore, for random texts the open problem of Landau has a positive answer. For prefixes of Fibonacci word our previous conjecture tells us that suffix automata do the same.

In the case of words of the form bba^n, $a, b \in \Sigma, n \geq 1$, our experimental results have led us to the following formula describing the behaviour of the sequence of differences between two consecutive terms involving words having n greater than or equal to 4: $\{a_{n+1} - a_n\} = 19 + 6 * (n - 4)$.

We have experimented also on prefixes of Thue-Morse words and, even if we have not obtained a well-formed formula, we have tested that the size of the compact suffix automata with 1 mismatch obtained is less than or equal to $2 \cdot |w| \cdot \log(|w|)$.

Moreover, the result is true even in the case of periodic words. So, we can state the following conjecture.

Conjecture 2. The (compact) suffix automaton with k mismatches of any text w has size $O(|w| \cdot \log^k(|w|))$.

The minimal deterministic finite automaton S_k that recognizes the set of suffixes of a word w up to k errors can be useful for solving the problem of approximate indexing and some applications of it. Classically, an index (cf. [10]) over a fixed text w is an abstract data type based on the set $Fact(w)$. Such data type is equipped with some operations that allow it to answer to the following queries. 1) Given a word x, say whether it belongs to $Fact(w)$ or not. If not, an index can optionally give the longest prefix of x that belongs to $Fact(w)$. 2) Given $x \in Fact(w)$, find the first (respectively the last) occurrence of x in w. 3) Given $x \in Fact(w)$, find the number of occurrences of x in w. 4) Given $x \in Fact(w)$, find the list of all occurrences of x in w. In the case of exact string matching, there exist classical data structures for indexing such as suffix trees, suffix arrays, DAWGs, factor automata or their compacted versions (cf. [10]). The algorithms that use them run in a time usually independent from the size of the text or at least substantially smaller than it. The last property is required by some authors to be an essential part in the definition of an index (cf. [1]).

All the operations defined for an index can easily be extended to the approximate case. But in the case of approximate string matching the problem is somehow different. We refer to [3, 16, 17, 23] and to the references therein for a *panorama* on this subject and on approximate string matching in general. The minimal deterministic finite automaton S_k introduced in this paper can be useful for solving the problem of approximate indexing. More precisely, it is easy to answer queries 1) and 2), but the other questions are more complex and they can be solved by using techniques analogous to those in [15].

Moreover, if the Conjecture 2 is true and constants involved in O-notation are small, our data structure is useful for some classical applications of approximate indexing, such as recovering the original signals after their transmission over noisy channels, finding DNA subsequences after possible mutations, text

searching where there are typing or spelling errors, retrieving musical passages, A.I. techniques in feature vector, and so on. It is important even in other applications, like in the field of Web search tools when we deal with *agglutinative languages*, i.e. languages that mainly resort to suffixes and declinations such as many Uralic languages (like Hungarian, Finnish, and Estonian), or in the case of real-time proposal of alternative internet URL in Domain Name Servers, or for deeper analysis of biological sequences (such as for finding long repeated factor with errors and their applications in prediction, in detecting horizontal gene transfer, in alignments, etc).

Finally, we think that it is possible to connect the suffix automaton S_k of the language $Suff(w, k, |w|)$ (without window) to the suffix automaton $S_{k,r}$ of the language $Suff(w, k, r)$ with $r = R(w, k, r)$. More precisely, our experimental results lead us to conjecture that this last automaton could have at most the same number of states and an additional number of edges that could be proportional to the size of the alphabet Σ with respect to S_k .

Conjecture 3. Let S_k and $S_{k,r}$ be the suffix automata of the languages $Suff(w, k)$ (without window) and $Suff(w, k, r)$ with $r = R(w, k, r)$, respectively. Then $|S_{k,r}| = O(|S_k|)$.

References

1. A. Amir, D. Keselman, G. M. Landau, M. Lewenstein, N. Lewenstein, and M. Rodeh. Indexing and dictionary matching with one error. *LNCS*, 1663:181–190, 1999.
2. A. Amir, D. Keselman, G. M. Landau, M. Lewenstein, N. Lewenstein, and M. Rodeh. Indexing and dictionary matching with one error. *Journal of Algorithms*, 37:309–325, 2000.
3. R. Baeza-Yates, G. Navarro, E. Sutinen, and J. Tarhio. Indexing methods for approximate string matching. *IEEE Data Engineering Bulletin*, 24(4):19–27, 2001. Special issue on Managing Text Natively and in DBMSs. Invited paper.
4. A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M.T. Chen, and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science*, 40:31–55, 1985.
5. A. L. Buchsbaum, M. T. Goodrich, and J. Westbrook. Range searching over tree cross products. In *ESA 2000*, volume 1879, pages 120–131, 2000.
6. A. Carpi and A. de Luca. Words and special factors. *Theoretical Computer Science*, 259:145–182, 2001.
7. E. Chávez and G. Navarro. A metric index for approximate string matching. In *LATIN 2002: Theoretical Informatics: 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002. Proceedings*, volume 2286 of *LNCS*, pages 181–195, 2002.
8. R. Cole, L. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of Annual ACM Symposium on Theory of Computing (STOC 2004)*, 2004.
9. M. Crochemore and C. Hancart. *Automata for Matching Patterns*, volume 2 of *G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Linear Modeling: Background and Application*, chapter 9, pages 399–462. Springer-Verlag.
10. M. Crochemore, C. Hancart, and T. Lecroq. *Algorithmique du texte*. Vuibert, 2001.

11. Maxime Crochemore. Reducing space for index implementation. *Theoretical Computer Science*, 292(1):185–197, 2003.
12. C. Epifanio, A. Gabriele, and F. Mignosi. Languages with mismatches and an application to approximate indexing. In *Proceedings of the 9th International Conference Developments in Language Theory (DLT05)*, LNCS 3572, pages 224–235, 2005.
13. C. Epifanio, A. Gabriele, F. Mignosi, A. Restivo, and M. Sciortino. Languages with mismatches. *Theoretical Computer Science*. To appear.
14. A. Gabriele. *Combinatorics on words with mismatches, algorithms and data structures for approximate indexing with applications*. PhD thesis, University of Palermo, 2004.
15. A. Gabriele, F. Mignosi, A. Restivo, and M. Sciortino. Indexing structure for approximate string matching. In *Proc. of CIAC'03*, volume 2653 of *LNCS*, pages 140–151, 2003.
16. Z. Galil and R. Giancarlo. Data structures and algorithms for approximate string matching. *Journal of Complexity*, 24:33–72, 1988.
17. D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
18. T. N. D. Huynh, W. K. Hon, T. W. Lam, and W. K. Sung. Approximate string matching using compressed suffix arrays. In *Proc. of CPM 2004*, volume 3109 of *LNCS*, pages 434–444.
19. S. Inenaga, H. Hoshino, A. Shinohara, M. Takeda, S. Arikawa, G. Mauri, and G. Pavesi. On-line construction of compact directed acyclic word graphs. *Discrete Applied Mathematics*, 146(2):156–179, 2005.
20. M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics*. Cambridge University Press, 1983.
21. M. G. Maass and J. Nowak. A new method for approximate indexing and dictionary lookup with one error. *Information Processing Letters*, 96(issue 5):185–191, December 2005.
22. M. G. Maass and J. Nowak. Text indexing with errors. In *Proceedings of the 16th Annual Symposium on Combinatorial Pattern Matching (CPM 2005)*, LNCS 3537, pages 21–32, 2005.
23. G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

Appendix

Proof of Lemma 1

Proof. (i) $\equiv_{w,k}$ is an equivalence relation. Indeed it is obviously reflexive, symmetric and transitive.

Moreover this relation is a right-invariant equivalence. For any $x, y \in \Sigma^*$, if $x \equiv_{w,k} y$, then $end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$ and for any position $i \in end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$, $\min\{|w| - i, k - err_i(x)\} = \min\{|w| - i, k - err_i(y)\}$. Since the number of errors available in the suffix of w having i as first position is the same after the reading of x and of y, then for any $z \in \Sigma^*$ xz is a k-occurrence of w if and only if yz is a k-occurrence of w. Hence, $end\text{-}set_w(xz,k) = end\text{-}set_w(yz,k)$ and for any position $j \in end\text{-}set_w(xz,k) = end\text{-}set_w(yz,k)$ the number of errors available in the suffix of w having j as first position is the same after the reading of xz and of yz.

(ii) By definition x and y are such that $end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$. Therefore, for any $i \in end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$, both x and y k-occur in w with final position i and $d(x,y) \le d(x,w) + d(w,y) \le 2k$. Hence x and y are one a $2k$-suffix of the other.

(iii) Let us suppose, by hypothesis, that $xy \equiv_{w,k} y$. Therefore, $end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$ and for any position $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$, $\min\{|w| - i, k - err_i(xy)\} = \min\{|w| - i, k - err_i(y)\}$. Since $err_i(y) \le err_i(xy)$, then $k - err_i(xy) \le k - err_i(y)$. For any $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$, we can distinguish the following cases.

1. Let $\min\{|w| - i, k - err_i(y)\} = k - err_i(y)$. Since $k - err_i(xy) \le k - err_i(y) \le |w| - i$, then $\min\{|w| - i, k - err_i(xy)\} = k - err_i(xy)$. Since $\min\{|w| - i, k - err_i(xy)\} = \min\{|w| - i, k - err_i(y)\}$, then $k - err_i(xy) = k - err_i(y)$ and all the $err_i(xy)$ errors are in y and x occurs exactly in w.

2. Let $\min\{|w| - i, k - err_i(y)\} = |w| - i$. Since $\min\{|w| - i, k - err_i(xy)\} = \min\{|w| - i, k - err_i(y)\}$, one has that the number of errors available in the suffix of w having $i + 1$ as first position is $\min\{|w| - i, k - err_i(xy)\} = |w| - i$. Therefore the maximal allowed number of errors in x is $k - [err_i(y) + (|w| - i)] \ge 0$.

Hence, for any $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$, the k-occurrence of y with final position i is immediately preceded by a t-occurrence of x, where $t = \max\{(k - err_i(y) - (|w| - i), 0)\}$.

Let us suppose, now, that for any $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$, the k-occurrence of y with final position i is immediately preceded by a t-occurrence of x, where $t = \max\{(k - err_i(y)) - (|w| - i), 0)\}$. By hypothesis, $end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$. Let us distinguish two cases.

1. Let us consider positions $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$ such that $t = 0$. In this case all the $err_i(xy)$ are in y and the number of errors available in the suffix of w having $i + 1$ as first position is the same after the reading of xy and of y.

2. Let us, now, consider positions $i \in end\text{-}set_w(xy,k) = end\text{-}set_w(y,k)$ such that $t = (k - err_i(y)) - (|w| - i)$. In this case $k - err_i(y) \geq |w| - i$ and then $\min\{|w| - i, k - err_i(y)\} = |w| - i$. By hypothesis, this k-occurrence of y is immediately preceded by an occurrence of x up to $t = (k - err_i(y)) - (|w| - i)$ errors. Therefore, $k - err_i(xy) \geq k - [k - (err_i(y) + |w| - i) + err_i(y)] = |w| - i$ and $\min\{|w| - i, k - err_i(xy)\} = |w| - i$. $\qquad\square$

Proof of Theorem 1

Proof. By Lemma 1(i), if x and y are end_k-equivalent, then for any $z \in \Sigma^*$ xz and yz are end_k-equivalent and then xz is a k-suffix if and only if yz is a k-suffix.

Let us suppose, now, that for any $z \in \Sigma^*$, xz is a k-suffix if and only if yz is a k-suffix. Therefore $end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$. Moreover, for any z such that xz and yz are suffixes of w, the ending position i of x and y is such that $|w| - i = |z|$. By hypothesis, we can have two cases depending on $|z|$.

- Let $z \in \Sigma^*$ be such that $|z| \leq \min\{k - err_i(x), k - err_i(y)\}$. For such z one has that $\min\{|w| - i, k - err_i(x)\} = |w| - i$ and $\min\{|w| - i, k - err_i(y)\} = |w| - i$ and the thesis is proved.
- Let $z \in \Sigma^*$ be such that $|z| \geq \max\{k - err_i(x), k - err_i(y)\}$. For such z one has that $\min\{|w| - i, k - err_i(x)\} = k - err_i(x)$ and $\min\{|w| - i, k - err_i(y)\} = k - err_i(y)$. By hypothesis, for any position $i \in end\text{-}set_w(x,k) = end\text{-}set_w(y,k)$, any word $z \in \Sigma^*$ having $i+1$ as first position is such that xz is a k-suffix of w if and only if yz is a k-suffix of w and then $k - err_i(x) = k - err_i(v)$ and the thesis is proved. $\qquad\square$

Proof of Corollary 1

Proof. Since the union of the equivalence classes that form the accepting states of $\mathcal{S}_k$ is exactly the set of all k-suffixes of w, by Nerode's Theorem one has that $\mathcal{S}_k$ recognizes the language of all k-suffixes of w, i.e. the set $Suff(w,k)$. The minimality follows by Lemma 1. $\qquad\square$

Proof of Lemma 2

Proof. In this proof we consider w to be fixed. Let q_0 be the initial state of S_k and let δ be its transition function. By abuse of notation, we keep calling δ its extension to words.

If u k-occurs twice in w, then from state $\delta(q_0, u)$ there are two paths having different lengths to a final state. Therefore $r - 1$ will be the greatest length of all words that reach a state from which there are two paths having different lengths to a final state.

We firstly find all such states. Since the graph G underlying S_k is a DAG (its language is finite), the same happens to the inverse $\hat{G}$ of G that is the graph where all arcs are inverted. We can perform a visit of $\hat{G}$ by adding to each node a field distance d and a flag information that can be white, green or red. The flag white means that the node has not yet been met during the visit, the green one

means that, up that moment during the visit, the node has been encountered at last once during the visit and that all paths in G to a final state have same length. The flag red means that there are at last two paths in G to a final state having different lengths. In order to simplify the algorithm we add an initial state to $\hat{G}$ that goes with one arc to each final state of S_k. This initial state is set to be green and having distance -1 while all other nodes are set to be white and distance equal to $+\infty$. If a node with white flag is reached starting from a green node, then its flag is set to green and its distance becomes the distance from the initial state, i.e. the distance of previous node plus one. If a node with green flag is reached starting from a green node and if its distance is equal to be the distance of previous node plus one, then the node is not enqueued again, while otherwise it is set to red. Red flags propagate. Details are left to the reader. Nodes with red flags are the ones we were looking for.

At this point we perform a visit on G starting from the initial state (using a topological order) in order to obtain the greatest length of all words that reach a state with a red flag. The repetition index r is the greatest of such values plus one. $\qquad\square$