# Searching All Seeds of Strings with Hamming Distance using Finite Automata

Ondřej Guth, Bořivoj Melichar *

*Abstract*—**Seed is a type of a regularity of strings. A restricted approximate seed $w$ of string $T$ is a factor of $T$ such that $w$ covers a superstring of $T$ under some distance rule. In this paper, the problem of *all restricted seeds with the smallest Hamming distance* is studied and a polynomial time and space algorithm for solving the problem is presented. It searches for all restricted approximate seeds of a string with given limited approximation using Hamming distance and it computes the smallest distance for each found seed. The solution is based on a finite (suffix) automata approach that provides a straightforward way to design algorithms to many problems in stringology. Therefore, it is shown that the set of problems solvable using finite automata includes the one studied in this paper.**

***Keywords: approximate seed, suffix automaton, Hamming distance, stringology***

## 1 Introduction

Searching regularities of strings is used in a wide area of applications like molecular biology, computer–assisted music analysis, or data compression. By regularities, repeated strings are meant. Examples of regularities include repetitions, borders, periods, covers, and seeds.

The algorithm for computing all exact seeds of a string was introduced by Iliopoulos, Moore, and Park [1]. The first algorithm for searching all seeds using finite automata was introduced by Voráček and Melichar [2].

Finding exact regularities is not always sufficient and thus some kind of approximation is used. An algorithm for searching approximate periods, covers, and seeds under Hamming, Levenshtein (also called edit), and weighted Levenshtein distance was presented by Christodoulakis, Iliopoulos, Park, and Sim [3]. The algorithm for computing approximate seeds was originally introduced by these authors in [4]. An algorithm for searching all covers under Hamming, Levenshtein, and Damerau distance using finite automata was introduced by Guth [5], optimized algorithm for computing all covers with smallest Hamming distance was presented by Guth, Melichar, and Balík [6].

*Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science and Engineering. Email: {gutho1,melichar}@fel.cvut.cz

Finite automata provide common formalism for many algorithms in the area of text processing (stringology), involving forward exact and approximate pattern matching and searching for borders, periods, and repetitions [7], backward pattern matching [8], pattern matching in a compressed text [9], the longest common subsequence [10], exact and approximate 2D pattern matching [11], and already mentioned computing approximate covers [5, 6] and exact covers [12] and seeds [2] in generalized strings. Therefore, we would like to further extend the set of problems solved using finite automata. Such a problem is studied in this paper.

Finite automaton as a data structure may be easily implemented. Therefore, using it as a base for similar approach to many algorithms is not only theoretical problem, as it may make development of software related with above mentioned areas easier, faster, and cost-reduced.

This paper is organized as follows: in Section 2, basic definitions and previous works overview are placed. In Section 3, the algorithm for the problem being studied is presented. Its theoretical time and space complexity is derived in Section 4 and experimental results are shown in Section 5.

## 2 Preliminaries

Some notions and notations used in this paper are commonly used, thus they are not defined here. Exact definitions of such notions and notations could be found in [6].

A symbol of an alphabet is denoted by $a$. Having string $T = a_1, a_2, \ldots, a_{|T|}$, *reversed string* $T$ is denoted by $T^R$ and it is equal to $T^R = a_{|T|}, a_{|T|-1}, \ldots, a_1$. An *effective alphabet* of a string $T$ is denoted by $A_T$. Only effective alphabet is considered in this paper.

Suppose $p, s, u, w, p', T \in A^*$. $T$ is a *superstring* of $w$ if $T = pws$. $p'$ is an *approximate prefix* of $T$ with maximum Hamming distance $k$ if $T = pu$ and $D_H(p, p') \leq k$. Set of all approximate prefixes of $T$ with maximum Hamming distance $k$ is denoted by $\mathit{Pref}^k(T)$. An *approximate suffix* and set of all approximate suffixes of $T$ with respect to $k$ is defined by analogy and is denoted by $\mathit{Suff}^k(T)$.

We say that string $w$ *occurs* in string $T$ if $w \in \mathit{Fact}(T)$.

Factor $w$ *occurs at position* (or end-position) $i$ in string $T$ if $\forall j \in \{1, \ldots, |w|\} : w[j] = T[i - |w| + j]$. An *end-set* is a set of all $i$ such that $w$ occurs at position $i$ in $T$. String $w$ occurs approximately with maximum Hamming distance $k$ at position $i$ in string $T$ (or $w$ has approximate occurrence at position $i$ in $T$) if there exists factor $x$ that occurs at position $i$ in $T$ and $D_H(x, w) \leq k$.

String $w$ is a *restricted approximate seed* of $T$ with maximum Hamming distance $k$ if $w$ is a factor of $T$ and $w$ is a restricted approximate cover of some superstring of $T$ with maximum Hamming distance $k$.

The *smallest Hamming distance* of a restricted approximate seed $w$ of $T$ is the smallest possible integer $l_m$ such that $w$ is a restricted approximate seed of $T$ with maximum Hamming distance $l_m$.

DFA is *partial* if there may exist a pair of state $q_i$ and symbol $a$ such that $\delta(q_i, a)$ is undefined. In this paper, only partial DFA are considered. A deterministic *trie* is a DFA that may be represented as a tree, i.e. for each state $q^j$, there exists at most one $q^i$ such that for any $a \in A$, $\delta(q^i, a) = q^j$. An *extended transition function*, denoted by $\delta^*$, is for a DFA defined for $a \in A, u \in A^*$ in this way:

$$\delta^*(q, \varepsilon) = q, \ \delta^*(q, ua) = \delta(\delta^*(q, u), a)$$

An extended transition function of an NFA is defined as:

$$\delta^*(q, \varepsilon) = \{q\}, \ \delta^*(q, au) = \bigcup_{q_i \in \delta(q,a)} \delta^*(q_i, u)$$

A *left language* of a state $q$ of a DFA is set of all strings $w$ for that holds $\delta^*(q_0, w) = q$ for initial state $q_0$. Left language of a state $q$ of a trie contains one string, denoted by *factor*$(q)$.

A nondeterministic Hamming suffix automaton for string $T$ and maximum Hamming distance $k$ is denoted by $M^k_{S^N}(T)$.

A deterministic suffix automaton for string $T$ and maximum Hamming distance $k$, denoted by $M^k_{S^D}(T)$, is a DFA that accepts all strings from $\mathit{Suff}^k(T)$ (see Figure 5). A *depth* of a state $q$ of the automaton is length of the longest string $w$ such that $\delta^*(q_0, w) = q$ for initial state $q_0$. An element of a $d$–subset is denoted by $r_i$, where the subscript $i$ means an index (order) of the element $r_i$ within the $d$–subset. In figures, states of nondeterministic automata and elements of $d$–subsets of deterministic automata are denoted by their depths and levels, e.g. $3''$ means state or element with depth 3 and level 2.

**Problem formulation** (All restricted seeds with the smallest Hamming distance)**.** Given string $T$ and maximum Hamming distance $k$, find all restricted approximate seeds of $T$ with respect to $k$ and compute their smallest distances.

The algorithm for searching (exact) seeds in generalized strings presented in [2] obviously works for (non-generalized) strings as well (as string is special case of generalized string). It is based on the following idea. First, $M_{S^N}(T)$ is constructed. Equivalent deterministic automaton $M_{S^D}(T) = (Q, A_T, \delta, q_0, F)$ is computed using subset construction. One of conditions for any factor to be a seed of $T$ is its length. Seed $w$ must cover central part of $T$ (i.e. the part of $T$ between the leftmost and the rightmost position of $w$ within $T$), and it must cover the uncovered suffix of $T$ and the uncovered prefix of $T$. All sufficiently long factors are then checked whether they cover the uncovered suffix, prefix of $T$, respectively. If $M_{S^D}(T)$ accepts some prefix of factor $w$ then $w$ covers uncovered suffix of $T$. If a suffix automaton $M_{S^D}(T^R)$ for reversed string $T$ accepts some prefix of reversed factor $w$ then $w$ covers uncovered prefix of $T$. When $w$ satisfies all the conditions, $w$ is a seed of $T$.

Computation of the smallest Hamming distance of a cover (presented in [6]) is based on the following idea: when the maximum approximation of the first and the last position of cover $w$ in $T$ is $l_{\min}$, for its smallest distance $l_m$ holds $l_m \geq l_{\min}$, because cover is an approximate prefix and suffix of $T$ and thus it cannot cover $T$ without its first and last position. When cover $w$ of $T$ has positions with approximation at most $l$, for its smallest distance $l_m$ clearly holds $l_m \leq l$. When the positions of $w$ with the maximum approximation equal to $l$ are no longer considered (the first and the last position must be still considered) and $w$ is still cover of $T$, then for $l_m$ holds $l_m \leq l - 1$. $l_m$ is decremented till $w$ still covers $T$. This may be used with modifications for computation of seeds.

The algorithm for searching exact seeds from [2] uses two phases: first, deterministic suffix automaton is constructed and then $d$–subsets are analyzed and seeds are computed. This means that complete automaton or at least all the $d$–subsets to be analyzed need to be stored in memory at a time. By contrast, the algorithm for searching covers from [6] uses merge of the phases, each $d$–subset is analyzed just after its construction. A depth-first search like algorithm is used and the states that are no longer needed are removed. For approximate seeds searching, there is also no need to store all elements of $d$–subsets of the automaton at a time.

## 3 Problem solution

Some properties are common for exact and approximate seeds with Hamming distance. Hence the algorithm presented in [2] is used as a base of algorithm for the problem studied in this paper, using some (but not all) techniques for searching covers in [6] for further improvements.

Every approximate restricted seed of string $T$ is necessarily an exact factor of $T$ with other possible approximate occurrences. Suffix automaton constructed for $T$

and maximum Hamming distance $k$ has extended transitions defined for all factors of $T$ with respect to $k$. When $M^k_{SD}(T)$ is constructed using subset construction from $M^k_{SN}(T)$, each element of any $d$–subset of any state of $M^k_{SD}(T)$ contains information not only about position (depth within $M^k_{SN}(T)$) but also about approximation (level within $M^k_{SN}(T)$). Therefore, it may be easily determined whether string from left language of any state of $M^k_{SD}(T)$ is an exact factor of $T$.

*Note* 1. For string $T$, string $p$ such that $p$ is not a factor of $T$, and any string $u$ being a superstring of $p$ holds:

$$\forall T, p, u \in A^*, p \in \mathit{Fact}(u) : p \notin \mathit{Fact}(T) \Rightarrow u \notin \mathit{Fact}(T)$$

**Lemma 1.** *For DFA $M^k_{SD}(T) = (Q_D, A_T, \delta_D, q^D_0, F_D)$ created using subset construction from $M^k_{SN}(T)$ and state $q_i \in Q_D$ with d–subset $d(q_i)$ such that $\forall r \in d(q_i)$ : $level(r) > 0$ holds that any successor of $q_i$ cannot contain element $r^j$ such that $level(r^j) = 0$.*

**Corollary.** *As only exact factor of $T$ may be a restricted seed of $T$, there is no need to construct any state of $M^k_{SD}(T)$ having only non-zero-level elements in its d–subset, as such state contains no exact factor of $T$ in its left language. Therefore, when such state is created during construction, it may be removed and any of its successors need not be constructed. Such deterministic suffix automaton that contains only states having at least one zero-level element in its d–subset is denoted by $\tilde{M}^k_{SD}(T)$.*

*Note* 2. Special type of deterministic suffix automaton, suffix trie, is considered in this paper. Construction of the trie and left language extraction is simpler than for general suffix automaton. As left language of any state (but the initial one) of the trie contains exactly one string, extraction of left language of any state takes linear time with respect to length of the string (e.g. using inverted transition function). See Fig. 5 for example of suffix trie.

The relation between length and positions of any seed (presented in [2]) holds also for approximate positions with Hamming distance, as the distance is defined for strings of equal lengths only.

*Note* 3. When searching for covers with Hamming distance [6], it is possible to remove all states $q$ of deterministic suffix trie that do not represent prefix, i.e. such $q$ that $|factor(q)| < depth(r_1)$, where $d(q) = r_1, \ldots, r_{|d(q)|}$. Similar property between the first position and length of a factor is used for searching seeds: $|factor(q)| \leq \frac{depth(r_1)}{2}$. Unlike in computing covers, this condition cannot be used for removing states $q$ and their successors.

**Example 1.** Let us consider suffix trie for string $T =$ *bbbbbbaaabb* and maximum Hamming distance $k = 2$. Factor *aaa* cannot be a seed of $T$ as its first approximate position within $T$ is 6. Factor *aaabb* is a seed of $T$ with respect to $k$. It is obvious that for states $q_1, q_2$ of the trie, where $factor(q_1) =$ *aaa* and $factor(q_2) =$ *aaabb*, holds: $q_2$ is a successor of a state that is a successor of $q_1$. Therefore, $q_1$ must not be removed to be able to find *aaabb*.

```
bbbbbbaaa
bbba
 bbba
  bbba
   bbba
    bbba
```

Figure 1: Possible covering of string *bbbbbbaaa* with string *bbba* and Hamming distance 2 from Example 2

```
bbbbbbaaa
bbba
 bbba
  bbba
   bbba
       b
```

Figure 2: Possible covering of a superstring of *bbbbbbaaa* with *bbba* and Hamming distance 1 from Example 2

For computation of the smallest distance $l_m$ of each seed, the idea used for searching covers ([6]) may be used for searching seeds. Unlike searching covers, any position may be removed, including the first and the last, thus the only lower bound of $l_m$ is 0. Determination whether continue to decrement $l$ is for seeds more complex than for covers, as computation of covering of central part of $T$ is not sufficient condition for seeds. For factor $w$ of $T$, not only positions and their approximation need to be considered, but also distance of the uncovered prefix, suffix, of $T$, and some suffix, prefix, of $w$, respectively. See Algorithm 3 for further information.

**Example 2.** Let us have string $T =$ *bbbbbbaaa* and maximum Hamming distance $k = 2$. One seed of $T$ with respect to $k$ is *bbba*. It may be seed of $T$ with Hamming distance 2, because its positions in $T$ are 4, 5, 6, 7, and 8 with maximum approximation 2 (see Figures 1, 5). When the position 8 with approximation 2 is removed, *bbba* is still seed of $T$ with positions 4, 5, 6, and 7, all with approximation at most 1 (see Figure 2).

The deterministic suffix trie is needed not only to determine positions of each factor $w$ of $T$, but also for checking whether $w$ is able to cover uncovered prefix and suffix of $T$ (see Algorithm 4). Thus, the trie must be able to accept strings of length at least $|w| - 1$. Therefore, the depth-first search with removing states from [6] cannot be used. By contrast, only elements of $d$–subset $d(q)$ may be removed after construction of all successors of $q$, transitions must be preserved. Thus, breadth-first search in the automaton is used (see Algorithm 1 and usage of queues $L, L^R$). As no state of trie $\tilde{M}^k_{SD}(T)$ is removed and the last element of each $d$–subset is preserved, it is possible to recognize all approximate suffixes of $T$ of length at least $|w| - 1$ and their distance.

Like an exact seed, an approximate one must also cover the uncovered prefix and suffix of $T$ (i.e. some prefix of seed $w$ must be an approximate suffix of $T$ and some suffix of $w$ must be an approximate prefix of $T$). Similar technique as for exact seeds ([2]) is used (Algorithm 4), but with tries $\tilde{M}^k_{SD}(T)$ and $\tilde{M}^k_{SD}(T^R)$. When some suffix of seed $w$ of $T$ (i.e. some prefix of reversed $w$) is accepted by $\tilde{M}^k_{SD}(T^R)$, i.e. $w = \mathit{factor}(q)$ for some final state $q$ of $\tilde{M}^k_{SD}(T^R)$, $w$ covers the uncovered prefix of $T$ with approximation equal to level of the last element $r_{|d(q)|}$ of $d(q)$. Similarly for a prefix of $w$, the uncovered suffix of $T$ and $\tilde{M}^k_{SD}(T)$.

For complete solution of the problem see Algorithm 1.

**Algorithm 2** Compute state of a deterministic suffix trie $\tilde{M} = Q_D, A_T, \delta_D, q_0^D, F_D$.

**Input:** NSA $(Q_N, A_T, \delta_N, q_0^N, F_N)$, state $q^t \in Q_D$, symbol $a \in A_T$, queue $L$ of states.
**Output:** Modified $\tilde{M}$ with possibly added successor $q^u$ of state $q^t$ for symbol $a$, modified queue $L$.

```
1: create new state $q^u$
2: define $depth(q^u) = depth(q^t) + 1$
3: for all $r^i \in d(q^t)$ (in order as stored in $d(q^t)$) do
4:    append all $r^j \in \delta_N(r^i, a)$ to $d(q^u)$ in ascending order by $depth(r^j)$
5: end for
6: if exists $r \in d(q^u)$ where $level(r) = 0$ then
7:    $Q_D \leftarrow Q_D \cup \{q^u\}$
8:    enqueue$(L, q^u)$
9:    if $r^u_{|d(q^u)|} \in F_N, d(q^u) = r^u_1, \dots r^u_{|d(q^u)|}$ then
10:      $F_D \leftarrow F_D \cup \{q^u\}$
11:   end if
12: end if
```

**Algorithm 3** The smallest distance of a seed of $T$.

**Input:** $d$–subset $d(q) = r_1, r_2, \dots, r_{|d(q)|}$ representing seed $w$ of $T$.
**Output:** The smallest distance $l_m$ of $w$.

```
1: $t \leftarrow d(q)$
2: $l_{\max} \leftarrow \max_{r \in t}\{level(r)\}$
3: $l \leftarrow l_{\max}$
4: repeat
5:    for all $r \in t : level(r) = l$ do
6:       remove $r$ from $t$
7:    end for
8:    $l \leftarrow l - 1$
9: until $w$ is a seed of $T$ using positions determined by $t$ with respect to $l$ (Algorithm 4)
10: $l_m \leftarrow l + 1$.
```

**Example 3.** Let us compute set of all seeds with maximum Hamming distance $k = 2$ for string $T = \mathit{bbbbbaaa}$.

| seed | $d$–subset | $l_m$ | occurrences | $p$ | $s$ |
|---|---|---|---|---|---|
| $ba$ | $2'3'4'5'67'8'$ | 1 | 2,3,4,5,6,7,8 | $\varepsilon$ | $\varepsilon$ |
| $baa$ | $3''4''5''6'78'$ | 2 | 3,4,5,6,7,8 | $\varepsilon$ | $\varepsilon$ |
| $bba$ | $3'4'5'67'8''$ | 1 | 3,4,5,6,7 | $b$ | $\varepsilon$ |
| $bbb$ | $3456'7''$ | 2 | 3,4,5,6,7 | $b$ | $\varepsilon$ |
| $baaa$ | $6''7'8$ | 2 | 6,7,8 | $\varepsilon$ | $aa$ |
| $bbaa$ | $4''5''6'78'$ | 2 | 4,5,6,7,8 | $\varepsilon$ | $aa$ |
| $bbba$ | $4'5'67'8''$ | 1 | 4,5,6,7 | $b$ | $\varepsilon$ |
| $bbbb$ | $456'7''$ | 2 | 4,5,6,7 | $b$ | $\varepsilon$ |
| $bbaaa$ | $6''7'8$ | 2 | 6,7,8 | $\varepsilon$ | $a$ |
| $bbbaa$ | $5''6'78'$ | 1 | 6,7,8 | $\varepsilon$ | $a$ |
| $bbbba$ | $5'67'8''$ | 1 | 5,6,7 | $b$ | $\varepsilon$ |
| $bbbbb$ | $56'7''$ | 2 | 5,6,7 | $b$ | $\varepsilon$ |
| $bbbaaa$ | $6''7'8$ | 1 | 7,8 | $\varepsilon$ | $a$ |
| $bbbbaa$ | $6'78'$ | 1 | 6,7,8 | $\varepsilon$ | $\varepsilon$ |
| $bbbbba$ | $67'8''$ | 1 | 6,7 | $b$ | $\varepsilon$ |
| $bbbbaaa$ | $7'8$ | 1 | 7,8 | $\varepsilon$ | $\varepsilon$ |
| $bbbbbaa$ | $78'$ | 1 | 7,8 | $\varepsilon$ | $\varepsilon$ |
| $bbbbbaaa$ | $8$ | 0 | 8 | $\varepsilon$ | $\varepsilon$ |

Table 1: All seeds of string $T = \mathit{bbbbbaaa}$ with maximum Hamming distance $k = 2$ and their smallest distances $l_m$; $p$ is used prefix of a seed, $s$ is used suffix (both computed by Algorithm 4); see Example 3

Nondeterministic suffix automata $M^k_{SN}(T)$ (see Figure 3) and $M^k_{SN}(T^R)$ (see Figure 4) are constructed. Next, subset construction of deterministic suffix trie $M^k_{SD}(T)$ from $M^k_{SN}(T)$ starts state-by-state (see transition diagram of $M^k_{SD}(T)$ with all states, that need to be constructed, at Figure 5), the same is done with trie $M^k_{SD}(T^R)$ from $M^k_{SN}(T^R)$. Some states may have only elements with non-zero level in its $d$–subset (e.g. $7''8'$). Such states are removed and their successors are not constructed as strings from their left languages (e.g. $aaaa$) are not factors of $T$.

All other states need to be checked whether their left languages contain some seeds. For example, state with $d$–subset $6''7'8$ contains string $aaa$ in its left language. The string occurs approximately at positions 6, 7 in $T$ and exactly at position 8 in $T$, thus it cannot be seed of $T$, as its leftmost occurrence within $T$ ends at position 6 and its length is 3 (i.e. the occurrence starts at position 4), so any proper suffix of $aaa$ cannot cover the uncovered prefix (positions 1 to 3) of $T$.

Other example is state with $d$–subset $4'5'67'8''$, which contains string $bbba$ in its left language. This string covers $T$ with Hamming distance 2, and therefore it is seed of $T$ (see Figure 1). When all positions with the maximum distance (i.e. 8) are not considered, $bbba$ is still seed of $T$, as proper prefix $b$ of $bbba$ covers uncovered suffix $a$ of $T$ with Hamming distance 1 (see Figure 2). See resulting table of all seeds and their distances in Table 1

**Algorithm 1** Compute set of seeds of $T$ with the smallest Hamming distances.

```
Input: String $T$, maximum Hamming distance $k$.
Output: Set $hseeds^k(T)$ of all seeds of $T$.
1: $hseeds^k(T) \leftarrow \emptyset$
2: construct $M^k_{SN}(T) = (Q_N, A_T, \delta_N, q_0^N, F_N)$
3: construct $M^k_{SN}(T^R) = (Q^R_N, A_T, \delta^R_N, q_0^{NR}, F^R_N)$
4: create new state $q_0^D$ as the initial one of the deterministic suffix trie $M^k_{SD}(T) = (Q_D, A_T, \delta_D, q_0^D, F_D)$
5: create new state $q_0^{DR}$ as the initial one of the deterministic suffix trie $M^k_{SD}(T^R) = (Q^R_D, A_T, \delta^R_D, q_0^{DR}, F^R_D)$
6: define $factor(q_0^D) = \varepsilon$, $depth(q_0^D) = 0$, $depth(q_0^{DR}) = 0$
7: create $L, L^R$ new empty queues of states
8: enqueue($L, q_0^D$), enqueue($L^R, q_0^{DR}$)
9: while $L^R$ is not empty {construct complete $\tilde{M}^k_{SD}(T^R)$ in this loop} do
10:   $q^{tR} \leftarrow$ dequeue($L^R$)
11:   for all $a \in A_T$ do
12:     compute new state $q^{uR}$ as a successor of state $q^{tR}$ for symbol $a$ using Algorithm 2
13:     discard all elements of $d(q^{tR})$ but the last one {all successors of $d(q^{tR})$ have just been computed}
14:   end for
15: end while
16: while $L$ is not empty {construct $\tilde{M}^k_{SD}(T)$ and compute seeds in this loop} do
17:   $q^t \leftarrow$ dequeue($L$)
18:   for all $a \in A_T$ do
19:     compute new state $q^u$ as a successor of state $q^t$ for symbol $a$ using Algorithm 2
20:     if exists $r \in d(q^u)$ where $level(r) = 0$ {only state $q^u$ that is part of $\tilde{M}^k_{SD}(T)$ is further processed} then
21:       define $w = factor(q^u) = factor(q^t).a$
22:       if $w$ is a seed of $T$ using positions determined by $d(q^u)$ (Algorithm 4) then
23:         compute the smallest distance $l_m$ of $w$ (Algorithm 3)
24:         if $|w| > k$ or $l_m < |w|$ {all strings of length less or equal to $l_m$ are seeds} then
25:           $hseeds^k(T) \leftarrow hseeds^k(T) \cup \{(w, l_m)\}$
26:         end if
27:       end if
28:     end if
29:   end for
30:   discard all elements of $d(q^t)$ but the last one
31: end while
```

**Algorithm 4** Determine whether string $w$ is a seed of $T$ with maximum Hamming distance $l$.

```
Input: Already constructed parts of deterministic suffix automata $M^k_{SD}(T) = (Q, A_T, \delta, q_0, F)$ and $M^k_{SD}(T^R) = (Q^R, A_T, \delta_R, q_0^R, F^R)$, $d$–subset $t = r_1, r_2, \ldots, r_{|t|}$ for $q \in Q$ and $w \in factor(q)$, maximum Hamming distance $l$.
Output: Resolution whether $w$ is a seed of $T$ with respect to $l$ and $t$.
1: if for all $i = 2, 3, \ldots, |t| : r_i - r_{i-1} \leq |w|$
   and $\exists p \in Pref^0(w), |p| \geq |T| - r_{|t|} : \delta^*(q_0, p) = q^1, q^1 \in F \wedge level(r^{q^1}_{|d(q^1)|}) \leq l$
   and $\exists s \in Suff^0(w), |s| \geq r_1 - |w| : \delta^*_R(q_0^R, s) = q^2, q^2 \in F^R \wedge level(r^{q^2}_{|d(q^2)|}) \leq l$ then
2:   return true
3: else
4:   return false
5: end if
```

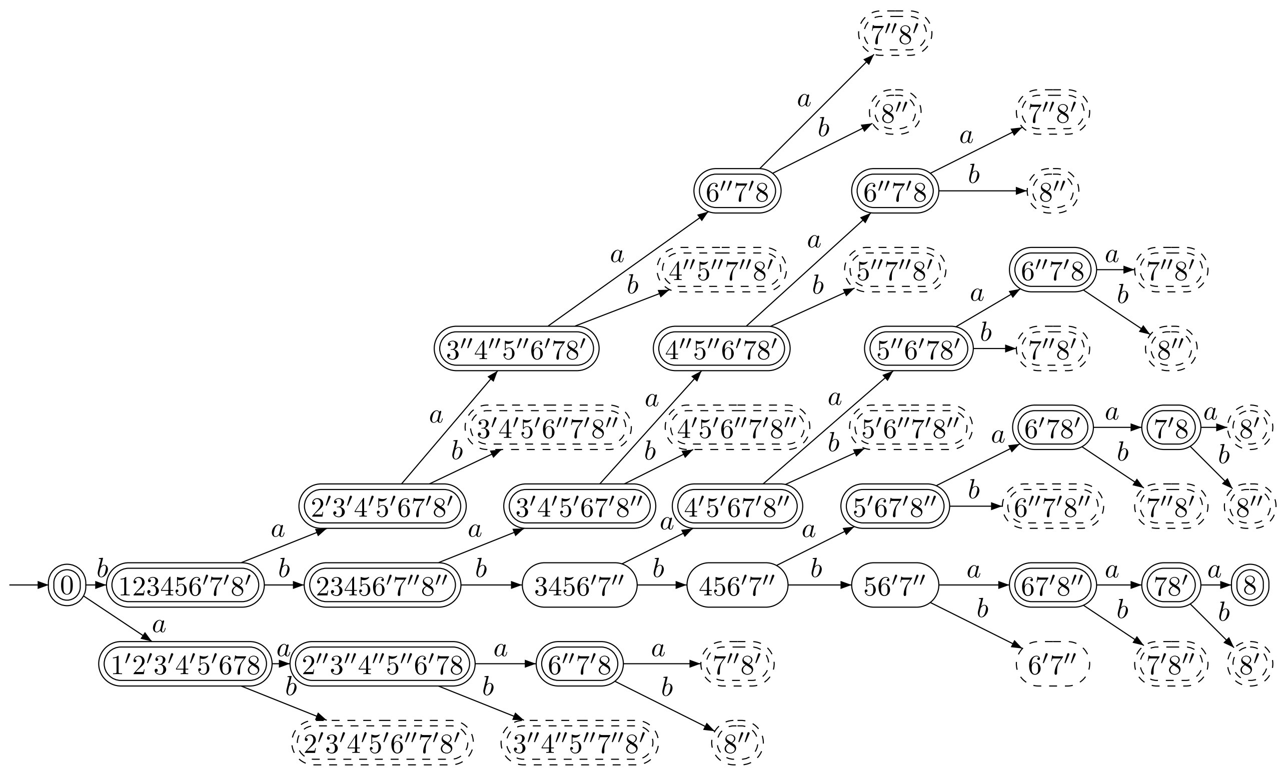

Figure 5: Transition diagram of the constructed part of the suffix trie $M^k_{SD}(T)$ for string $T = bbbbbaaa$ and maximum Hamming distance $k = 2$ from Example 3; dashed states are removed as their left language do not contain exact factor of $T$ and thus they are not states of $\tilde{M}^k_{SD}(T)$

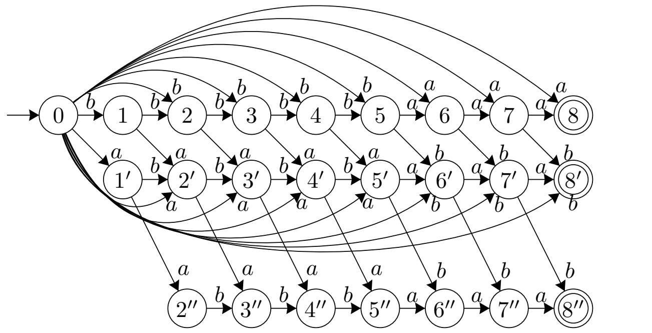

Figure 3: Transition diagram of the nondeterministic approximate suffix automaton $M^k_{SN}(T)$ for string $T = bbbbbaaa$ and maximum Hamming distance $k = 2$ from Example 3

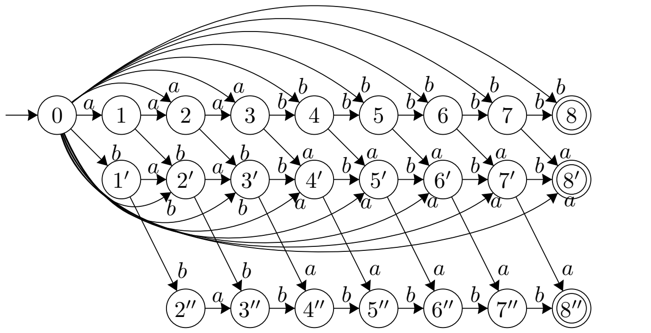

Figure 4: Transition diagram of the nondeterministic approximate suffix automaton $M^k_{SN}(T^R)$ for reversion of string $T$, i.e. $T^R = aaabbbbb$, and maximum Hamming distance $k = 2$ from Example 3

## 4 Time and space complexities

*Note* 4. As parts of $M^k_{SD}(T)$ and $M^k_{SD}(T^R)$ are constructed the same way in Algorithm 1, the time and space complexities of their construction are the same.

**Lemma 2.** *Left languages of states of deterministic suffix trie* $\tilde{M}^k_{SD}(T) = (Q_D, A_T, \delta_D, q^D_0, F_D)$ *are distinct, i.e.*

$$q_1, q_2 \in Q_D; q_1 \neq q_2 \Rightarrow \mathit{factor}(q_1) \neq \mathit{factor}(q_2)$$

**Definition 1.** Let us consider string $T$ and maximum Hamming distance $k$. When a factor $w$ approximately occurs $e$-times in $T$ with respect to $k$, we say that *number of repetitions of $w$ in $T$ with respect to $k$*, denoted by $R^k_w(T)$, is $e-1$. Then *number of repetitions of all factors of $T$ with respect to $k$*, denoted by $R^k(T)$, is defined as

$$R^k(T) = \sum_{w \in Fact(T)} R^k_w(T)$$

**Lemma 3.** *Number of states of* $\tilde{M}^k_{SD}(T)$ *is*

$$\frac{1}{2} \cdot (|T|^2 + |T|) - R^k(T) + 1$$

*Note* 5. As restricted approximate seeds of string $T$ are exact factors of $T$, it is meaningful to consider effective alphabet $A_T$ only and $|A_T| \leq |T|$ always holds (recall that effective alphabet $A_T$ consists only of symbols that occur in $T$). It is also meaningless to consider high $k$, because every factor of $T$ having length less or equal to $k$ is always approximate seed of $T$. Thus $k \leq |T|$ always holds. Usually $k$ and $|A_T|$ are independent of $|T|$.

**Lemma 4.** *Number of states of $M^k_{SD}(T)$ constructed using Alg. 1 is at most $|A_T| \cdot (\frac{1}{2} \cdot (|T|^2 + |T|) - R^k(T)) + 1$.*

**Lemma 5.** *For every d–subset of $\tilde{M}^k_{SD}(T)$ constructed by Algorithm 1 holds that there are no two elements having the same depth.*

**Lemma 6.** *Number of elements of all d–subsets of $\tilde{M}^k_{SD}(T)$ is not greater than $\frac{1}{2}\cdot(|T|^3+|T|^2)-|T|\cdot R^k(T)+1$.*

**Lemma 7.** *Number of elements of all d–subsets of $M^k_{SD}(T)$ constructed using Alg. 1 is not greater than*

$$|A_T| \cdot (\frac{1}{2} \cdot (|T|^3 + |T|^2) - |T| \cdot R^k(T)) + 1$$

**Lemma 8.** *Time complexity of the check whether d–subset $d(q)$ of $\tilde{M}^k_{SD}(T)$ represents a seed $w = factor(q)$ of $T$ (Alg. 4) is at most $2 \cdot |d(q)| + 2 \cdot |w| - 2$, that is $\mathcal{O}(|T|)$.*

**Lemma 9.** *Time complexity of the computation of the smallest distance of seed $w = factor(q)$ of $T$ (Alg. 3) is at most $|d(q)| + k \cdot (3 \cdot |d(q)| + 2 \cdot |w| - 2)$ that is $\mathcal{O}(k \cdot |T|)$.*

*Note* 6. Number of all seeds is $\mathcal{O}(|T|^2)$ (like number of factors). Thus, the sum of their lengths is $\mathcal{O}(|T|^3)$, denoted by $|hseeds^k(T)|$.

**Theorem 1.** *Time complexity of computation of all seeds with their smallest distance for string $T$ with maximum Hamming distance $k$ (Algorithm 1) is $\mathcal{O}(k \cdot |A_T| \cdot |T|^3)$.*

*Proof.* Construction of nondeterministic suffix automaton $M^k_{SN}(T) = (Q_N, A_T, \delta_N, q^N_0, F_N)$ for $T$ and $k$ takes $\mathcal{O}(k \cdot |A_T| \cdot |T|)$. For each state of $\tilde{M}^k_{SD}(T)$ (Lemma 3) and for each symbol of $A_T$, new $d$–subset is constructed. As each element of any $d$–subset may be constructed in constant time (just using already known $\delta_N$) and the elements are naturally ordered (no need to sort – proven in [6]), all $d$–subsets are constructed in at most

$$|A_T| \cdot (\frac{1}{2} \cdot (|T|^3 + |T|^2) - |T| \cdot R^k(T)) + 1$$

time. Each $d$–subset is checked whether it contains element with zero level in linear time. The left language extraction of state takes linear time and by Lemma 8 and 3 the theorem holds. □

**Lemma 10.** *During construction of $\tilde{M}^k_{SD}(T)$ (Algorithm 1), there are at most $\mathcal{O}(|T|^2)$ elements of d–subsets stored in memory at a time.*

**Theorem 2.** *Space complexity of computation of all seeds is $\mathcal{O}(|T|^2 + |hseeds^k(T)|)$.*

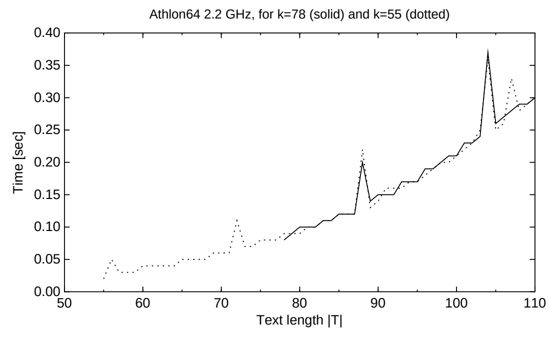

Figure 6: Time consumption of the experimental run on the Athlon64 with respect to the text size (see Section 5)

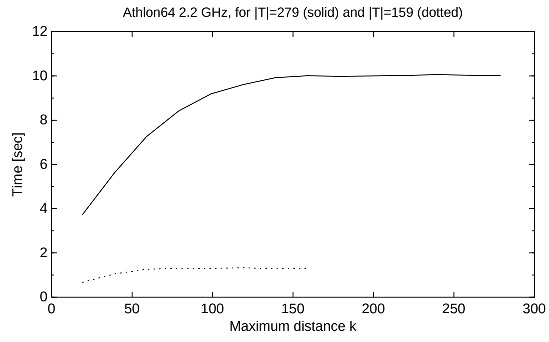

Figure 7: Time consumption of the experimental run on the Athlon64 with respect to the maximum distance (see Section 5)

*Proof.* Space complexity of construction of $M^k_{SN}(T)$ is $\mathcal{O}(k\cdot|A|\cdot|T|)$ (proven in [6]). By Lemma 10, number of elements stored in memory at a time is $\mathcal{O}(|T|^2)$, as no more elements of $d$–subsets than those in $L$ plus $\mathcal{O}(|T|)$ new are in memory at a time. By Lemma 3, number of states of $\tilde{M}^k_{SD}(T)$ is $\mathcal{O}(|T|^2)$ (they all are stored in memory with one element each). As the constructed automaton is trie, number of transitions is also $\mathcal{O}(|T|^2)$. The space complexity also depends on size of result, $|hseeds^k(T)|$. □

## 5 Experimental results

The algorithm was implemented in C++ using STL and compiled using GNU C++ 3.4.6 with O3 optimizations level. The dataset used to test the algorithm is the nucleotide sequence of Saccharomyces cerevisiae chromosome IV[1]. The string $T$ consists of the first $|T|$ characters of the chromosome.

The first set of tests was run on an AMD Athlon 64 3200+ (2200 MHz) system, with 2.5 GB of RAM, under Gentoo Linux operating system (see Figures 6 and 7).

[1]The Saccharomyces cerevisiae chromosome IV dataset could be downloaded from http://www.genome.jp/.

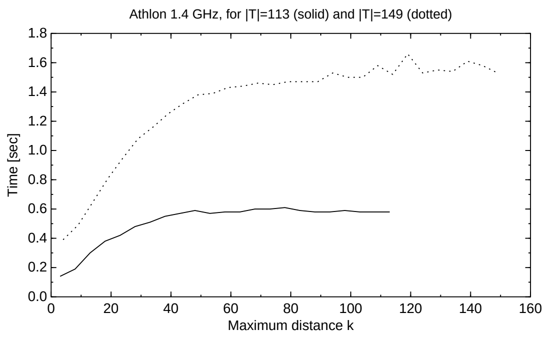

Figure 8: Time consumption of the experimental run on the Athlon with respect to the maximum distance (see Section 5)

The second set of tests was run on an AMD Athlon (1400 MHz) system, with 1.2 GB of RAM, under Gentoo Linux operating system (see Figure 8).

*Note* 7. In comparison to experimental results presented in [3], the algorithm presented in this paper runs a bit faster for the same data, even on a slightly slower computer (1.3 seconds in [3] for text length 100 vs. maximum 0.7 second for text length 113 – see Figure 8).

## 6 Conclusion

In this paper, we have shown that an algorithm design based on determinization of a suffix automaton is appropriate for computation of all restricted seeds with the smallest Hamming distance. The presented algorithm is straightforward, easy to understand and to implement and its theoretical and experimental time requirements are comparable to the existing approach ([4]).

For the future work, we would like to extend the algorithm for searching seeds to other distances and to utilize similar approach for searching other types of regularities.

## Acknowledgements

This research was supported by the Czech Technical University in Prague as grant No. CTU0803113, by the Ministry of Education, Youth and Sports of the Czech Republic under research program MSM 6840770014, and by the Czech Science Foundation as project No. 201/06/1039 and as project No. 201/09/0807.

## References

[1] Iliopoulos, C. S., Moore, D., and Park, K. S., “Covering a String,” *CPM ’93: Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, Springer-Verlag, London, UK, 1993, pp. 54–62.

[2] Voráček, M. and Melichar, B., “Computing Seeds in Generalized Strings,” *Proceedings of Workshop 2006*, Czech Technical University in Prague, 2006, pp. 138–139.

[3] Christodoulakis, M., Iliopoulos, C. S., Park, K. S., and Sim, J. S., “Implementing Approximate Regularities,” *Mathematical and Computer Modelling*, Vol. 42, October 2005, pp. 855–866.

[4] Christodoulakis, M., Iliopoulos, C. S., Park, K. S., and Sim, J. S., “Approximate Seeds of Strings,” *Journal of Automata, Languages and Combinatorics*, Vol. 10, No. 5/6, 2005, pp. 609–626.

[5] Guth, O., “Searching Approximate Covers of Strings Using Finite Automata,” *POSTER 2008*, Czech Technical University in Prague, Faculty of Electrical Engineering, Praha, 2008.

[6] Guth, O., Melichar, B., and Balík, M., “Searching All Approximate Covers and Their Distance using Finite Automata,” *Information Technologies – Applications and Theory*, Univerzita P. J. Šafárika, Košice, 2008, pp. 21–26.

[7] Melichar, B., Holub, J., and Polcar, T., “Text Searching Algorithms, Volume I,” November 2005.

[8] Melichar, B., “Text Searching Algorithms, Volume II,” March 2006.

[9] Lahoda, J., Melichar, B., and Žďárek, J., “Pattern Matching in DCA Coded Text,” *Proceedings of the 13th International Conference on Implementation and Application of Automata*, Springer, Heidelberg, 2008, pp. 151–160.

[10] Melichar, B. and Polcar, T., “The Longest Common Subsequence Problem – A Finite Automata Approach,” *Implementation and Application of Automata*, Springer, New York, 2003, pp. 294–296.

[11] Žďárek, J., “Automata and 2D Pattern Matching,” *Advances on Two-dimensional Language Theory*, University of Salerno, Salerno, 2006, p. 15.

[12] Voráček, M., “Computing Covers in Generalized Strings,” *POSTER 2005*, Czech Technical University in Prague, Faculty of Electrical Engineering, 2005.