String Matching with Mismatches by Real-valued FFT

Baba, Kensuke
Kyushu University Library

https://hdl.handle.net/2324/16818

出版情報：Lecture Notes in Computer Science (Computational Science and Its Applications - ICCSA 2010, Part IV). 6019, pp.273-283, 2010-03. Springer
バージョン：
権利関係：

String Matching with Mismatches by Real-valued FFT[*]

Kensuke Baba[†]

Abstract

String matching with mismatches is a basic concept of information retrieval with some kinds of approximation. This paper proposes an FFT-based algorithm for the problem of string matching with mismatches, which computes an estimate with accuracy. The algorithm consists of FFT computations for binary vectors which can be computed faster than the computation for vectors of complex numbers. Therefore, a reduction of the computation time is obtained by the speed-up for FFT, which leads an improvement of the variance of the estimates. This paper analyzes the variance of the estimates in the algorithm and compares it with the variances in existing algorithms.
Keywords: string matching with mismatches, FFT, randomized algorithm.

1 Introduction

Similarity on strings is one of the most important concepts in applications of information retrieval which cannot be explained completely by modeling in terms of sequences and the exact matching on it, such as, mining on a huge data base and homology search in biology. The problem of *string matching* is to find the occurrences of a (short) string called a *pattern* in a (long) string called a *text*. The problem of *string matching with mismatches* is to compute the vector whose element is the number of the matches between the pattern and every substring of the text whose length is equal to the pattern. Namely, the vector for the problem of string matching with mismatches solves generally the problem of string matching which allows substitutions of a character to introduce the variations of a pattern. It is useful for many applications of information retrieval to develop an efficient algorithm for the problem of string matching with mismatches.

For the problem of string matching with mismatches with a pattern of length m and a text of length n, there exists an $O(n \log m)$ algorithm which is based on the fast Fourier transformation (FFT), while the naive comparison-based algorithm takes $O(mn)$ time. This approach was essentially developed by Fischer and Paterson [6]. In this algorithm, two strings are converted into binary strings with respect to each character in the alphabet for the numerical computation of FFT. Hence, the computation of the algorithm is the σ-times iteration of the $O(n \log m)$ computation of FFT for the alphabet size σ. Atallah *et al.* [1] introduced a randomized algorithm to reduce the iteration number by a trade-off with the accuracy of the estimates for the vector. In the algorithm, a text and a pattern are converted into two sequences of complex numbers by a function chosen randomly from the set of size σ^σ.

The aim of this paper is to improve the accuracy of the estimates in the randomized algorithm for string matching with mismatches. Schoenmeyr and Yu-Zhang [10] modified the previous algorithm such that the functions which convert characters into complex numbers are restricted to the bijective functions, and therefore the size of the set is $\sigma!$. The upper bound of the variance of the estimates decreases and it is notable for small alphabets. Nakatoh *et al.* [8, 9] reduced the size of the set of functions which covert characters into complex numbers to $2\sigma - 2$ [8] and $\sigma - 1$ [9]. Since the sizes of the sets are small compared with those in the previous two algorithms, the variance decreases greatly in the case where the sampling of the function is operated without replacement.

The main idea of our method is an improvement of the variance of the estimates by reducing the computation time of the $O(n \log m)$ computation of FFT. By converting strings to vectors of binary numbers instead of complex numbers, the practical computation time of a single operation of FFT can be reduced [11]. Therefore, the iteration number in a given time, that is, the number of the samples for an estimate increases, which implies an improvement of the variance since the variance is inverse proportion to the number of samples. Baba *et al.* [2] proposed a randomized algorithm as an improvement of the algorithm in [1]. In this algorithm, each character is converted into 1 or -1 and the number of the possible functions from Σ to $\{-1, 1\}$ is 2^σ. In this paper, we propose an algorithm in which

- input strings are converted to vectors on $\{-1, 1\}$,
- the upper bound of the variance of the estimates is explicitly lower than that in the algorithm in [2],

[*] An edited version of this report was published in: *Lecture Notes in Computer Science (Computational Science and Its Applications - ICCSA 2010, Part IV), 6019, pp.273–283, Springer, Mar, 2010.*

[†] Research and Development Division, Kyushu University Library, *baba@lib.kyushu-u.ac.jp*

- the size of the population for samples is $\sigma - 1$ if σ is a power of two.

Therefore, the accuracy of the estimates of the proposing algorithm is better than [2], and expected to be better than the other existing algorithms if some fast algorithms are applied to the computation of FFT for binary vectors.

2 Preliminaries

Let $\mathbf{N}$ be the set of non-negative integers. Let Σ be an alphabet and Σ^n the set of the strings of length $n \in \mathbf{N}$ over Σ. The size of a set S is denoted by $|S|$. The j-th character of a string $s \in \Sigma^n$ is denoted by s_j for $1 \le j \le n$. The j-th element of an n-dimensional vector v is denoted by v_j for $1 \le j \le n$.

Let δ be the Kronecker function from $\Sigma \times \Sigma$ to $\{0, 1\}$, that is, for $a, b \in \Sigma$, $\delta(a, b)$ is 1 if $a = b$, and 0 otherwise. Then, for $t \in \Sigma^n$ and $p \in \Sigma^m$, the j-th element of the *score vector* $C(t, p)$ between t and p is

$$c_j = \sum_{k=1}^{m} \delta(t_{j+k-1}, p_k)$$

for $1 \le j \le n-m+1$. The problem of *string matching with mismatches* is to compute the score vector for two strings.

Example 1 *The score vector between* $t =$ `adcbabac` *and* $p =$ `abac` *is* $C(t,p) = (1, 0, 2, 0, 4)$.

The discrete Fourier transformation (DFT) of an n-dimensional vector v is the n-dimensional vector V of which k-th element is

$$V_k = \sum_{j=1}^{n} v_j \cdot \omega_n^{(j-1)(k-1)}$$

for $1 \le k \le n$, where $\omega_n = e^{2\pi i/n}$ and $i^2 = -1$. Let u and v be n-dimensional vectors and w the correlation of u and v, that is, for $1 \le k \le n$

$$w_k = \sum_{j=1}^{n} u_j \cdot v_{j+k},$$

where $v_{n+j} = v_j$ for $1 \le j \le n-1$. Let U, V, and W be the DFTs of u, v, and w, respectively. Then, by the basic property of DFT, for $1 \le j \le n$

$$W_j = U_j \cdot \overline{V_j},$$

where $\bar{c}$ is the conjugate complex number of c. The DFT and its inverse (IDFT) of an n-dimensional vector can be computed in $O(n \log n)$ time by FFT, respectively. Therefore, w is computed from u and v in $O(n \log n) + O(n) + O(n \log n) = O(n \log n)$ time [4].

Lemma 1 *The correlation of two* n*-dimensional vectors can be computed in* $O(n \log n)$ *time.*

3 Deterministic Algorithm

Let H_n be an n-dimensional Hadamard's matrix for $n \in \mathbf{N}$, that is, any element of H_n is -1 or 1 and

$$H_n^T H_n = nI_n,$$

where M^T is the transposed matrix of a matrix M and I_n is the n-dimensional unit matrix. It is known that H_n exists if n is a power of two.

Example 2

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Let $\sigma = |\Sigma|$ and $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$. $M(j,k)$ denotes the (j,k)-element of a matrix M. We assume that H_σ exists for σ. For $1 \le \ell \le \sigma$, ϕ_ℓ is defined to be the function from Σ to $\{-1, 1\}$ such that

$$\phi_\ell(a_j) = H_\sigma(j, \ell)$$

for $1 \le j \le \sigma$. Then, by the property of Hadamard's matrix,

$$\sum_{\ell=1}^{\sigma} \phi_\ell(a_j) \cdot \phi_\ell(a_k) = \sum_{\ell=1}^{\sigma} H_\sigma(j,\ell) \cdot H_\sigma(k,\ell) = \sigma\delta(a_j, a_k)$$

for any $1 \le j, k \le \sigma$. Therefore, the score vector between $t \in \Sigma^n$ and $p \in \Sigma^m$ is

$$\begin{aligned} c_j &= \sum_{k=1}^{m} \delta(t_{j+k-1}, p_k) \\ &= \sum_{k=1}^{m} \left(\frac{1}{\sigma} \sum_{\ell=1}^{\sigma} \phi_\ell(t_{j+k-1}) \cdot \phi_\ell(p_k) \right) \\ &= \frac{1}{\sigma} \sum_{\ell=1}^{\sigma} \sum_{k=1}^{m} \phi_\ell(t_{j+k-1}) \cdot \phi_\ell(p_k) \\ & \quad (1 \le j \le n-m+1). \end{aligned}$$

Let s^ℓ be the $(n-m+1)$-dimensional vector such that

$$s_j^\ell = \sum_{k=1}^{m} \phi_\ell(t_{j+k-1}) \cdot \phi_\ell(p_k) \quad (1 \le j \le n-m+1) \quad (1)$$

for $1 \le \ell \le \sigma$. Let τ be the n-dimensional vector $(\phi_\ell(t_j))$, and π the m-dimensional vector $(\phi_\ell(p_j))$ for each ℓ. Then, s^ℓ is a part of the correlation of τ and π' which is obtained by padding $n - m$ 0's to π. Therefore, by Lemma 1, s^ℓ is computed from τ and π' in $O(n \log n)$ time.

Additionally, the following standard technique [5] is applied. We part τ into overlapping chunks each of size $(1+\alpha)m$. One chunk and the $(1+\alpha)m$-dimensional vector π' with α 0's yield $\alpha m+1$ elements of s^ℓ. Since we have $n/\alpha m$ chunks and each chunk can be computed in $O((1+\alpha)m\log((1+\alpha)m))$ time, the total time complexity is $(n/\alpha m)\cdot O((1+\alpha)m\log((1+\alpha)m)) = O(n\log m)$ by choosing $\alpha = O(m)$.

Thus, since a single correlation is computed for a single ϕ_ℓ and $1 \le \ell \le \sigma$, the score vector $C(t,p)$ is obtained by repeating the $O(n\log m)$ computation σ times. Even if we consider the assumption of the existence of the Hadamard's matrix, the iteration number is less than 2σ. The algorithm is summarized in Figure 1.

Theorem 1 *The deterministic algorithm A computes the score vector between $t \in \Sigma^n$ and $p \in \Sigma^m$ in $O(\sigma n\log m)$ time.*

In the rest of this section, we analyze the number of the $O(n\log m)$ computations in the iteration in terms of σ in the strict sense.

By the argument of the existence of the Hadamard's matrix, the iteration number is at least σ and at most $2\sigma-2$. Moreover, if we construct H_σ by Sylvester's method, that is, $H_1 = [1]$ and for $1 \le k \le \log\sigma$

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}$$

is applied recursively, then $\phi_1(a_j) = 1$ for any $1 \le j \le \sigma$. Therefore, we can skip a single $O(n\log m)$ computation for $\ell = 1$. Thus, the iteration number μ is

$$\sigma - 1 \le \mu \le 2\sigma - 3.$$

4 Randomized Algorithm

We assume that σ is a power of two. A *sample* of the score vector is the $(n-m+1)$-dimensional vector s^ℓ in Equation 1. An *estimate* of the score vector is defined to be

$$\hat{s_j} = \frac{1}{h}\sum_{\ell\in L} s_j^\ell \quad (1 \le j \le n-m+1), \qquad (2)$$

where L is a set of h integers which are chosen independently and uniformly from $\{1,2,\ldots,\sigma\}$. The algorithm is described in Figure 2.

The expectation of x is described by $E[x]$ and the variance by $V[x]$. For $1 \le j \le n-m+1$,

$$E[\hat{s_j}] = c_j$$

since $E\left[s_j^\ell\right] = c_j$. By the definition of ϕ_ℓ, $|\phi_\ell(a_j)\cdot\phi_\ell(a_k)| = 1$ and $(\phi_\ell(a_j))^2 = 1$ for any $1 \le j,k,\ell \le \sigma$, and hence $-(m-c_j) \le s_j^\ell - E[s_j^\ell] \le m - c_j$ for any $1 \le \ell \le \sigma$ and $1 \le j \le n-m+1$. Therefore,

$$\begin{aligned} V[\hat{s_j}] &= \frac{1}{h}V\left[s_j^\ell\right] \\ &= \frac{1}{h}\left(\frac{1}{\sigma}\sum_{\ell=1}^{\sigma}\left(s_j^\ell - E[s_j^\ell]\right)^2\right) \\ &\le \frac{(m-c_j)^2}{h}. \end{aligned}$$

This upper-bound of the variance does not depend on σ, and therefore the same result is obtained for any Σ by considering H_ν for a power of two $\nu \ge \sigma$ instead of H_σ.

Theorem 2 *The randomized algorithm B computes an estimate for the score vector between $t \in \Sigma^n$ and $p \in \Sigma^m$ in $O(hn\log m)$ time for the number h of samples. The expectation of the estimates is equal to c_j and the variance of the estimates is bounded by $(m-c_j)^2/h$ for $1 \le j \le n-m+1$.*

In the case where the Hadamard's matrix is Sylvester-type, the variance of the estimates of the score vector decreases.

Let ν be the power of two such that $\sigma \le \nu < 2\sigma$. Then,

$$\begin{aligned} \sum_{\ell=2}^{\nu}\phi_\ell(a_j)\cdot\phi_\ell(a_k) &= \sum_{\ell=1}^{\nu}H_\nu(j,\ell)\cdot H_\nu(k,\ell) - 1 \\ &= \nu\delta(a_j,a_k) - 1 \end{aligned}$$

and hence the score vector is

$$\begin{aligned} &c_j \\ &= \sum_{k=1}^{m}\left(\frac{1}{\nu}\sum_{\ell=2}^{\nu}\phi_\ell(t_{j+k-1})\cdot\phi_\ell(p_k) + \frac{1}{\nu}\right) \\ &= \frac{1}{\nu-1}\sum_{\ell=2}^{\nu}\left(\frac{\nu-1}{\nu}\sum_{k=1}^{m}\phi_\ell(t_{j+k-1})\cdot\phi_\ell(p_k) + \frac{m}{\nu}\right) \\ &\quad (1 \le j \le n-m+1). \end{aligned}$$

An estimate $(\hat{s_j})$ of the score vector is defined by Equation 2 for the following sample

$$\begin{aligned} s_j^\ell &= \frac{\nu-1}{\nu}\sum_{k=1}^{m}\phi_\ell(t_{j+k-1})\cdot\phi_\ell(p_k) + \frac{m}{\nu} \\ &\quad (1 \le j \le n-m+1) \end{aligned}$$

for $2 \le \ell \le \nu$.

For $1 \le j \le n-m+1$, clearly $E[\hat{s_j}] = c_j$ and, by the basic properties of variance,

$$V[\hat{s_j}] = \frac{1}{h}V\left[s_j^\ell\right]$$

```
A:
Input: a text $t \in \Sigma^n$ and a pattern $p \in \Sigma^m$
Output: the score vector $C(t,p) = (c_1, c_2, \ldots, c_{n-m+1})$

Let $H_\nu$ be a $\nu$-dimensional Hadamard's matrix for $\nu \geq \sigma = |\Sigma|$ and
    $\phi_\ell(a_j) = H_\nu(j,\ell)$ for $1 \leq \ell \leq \nu$ and $a_j \in \Sigma$.

1. For $1 \leq \ell \leq \nu$,
    1.1. compute $T_i^\ell = \phi_\ell(t_i)$ for $1 \leq i \leq n$ and $P_i^\ell = \phi_\ell(p_i)$ for $1 \leq i \leq m$,
    1.2. compute $s_j^\ell = \sum_{k=1}^{m} T_{j+k-1}^\ell \cdot P_k^\ell$ for $1 \leq j \leq n-m+1$ by FFT;
2. compute $c_i = \frac{1}{\nu} \sum_{\ell=1}^{\nu} s_j^\ell$ for $1 \leq j \leq n-m+1$.
```

Figure 1: The deterministic algorithm A for the problem of string matching with mismatches.

```
B:
Input: a text $t \in \Sigma^n$, a pattern $p \in \Sigma^m$, and the number $h$ of the samples
Output: an estimate $(\hat{s_1}, \hat{s_2}, \ldots, \widehat{s_{n-m+1}})$ of the score vector $C(t,p)$

Let $H_\nu$ be a $\nu$-dimensional Hadamard's matrix for $\nu \geq \sigma = |\Sigma|$ and
    $\phi_\ell(a_j) = H_\nu(j,\ell)$ for $1 \leq \ell \leq \nu$ and $a_j \in \Sigma$.

1. Make a set $L$ of $h$ integers randomly chosen from $\{1, 2, \ldots, \nu\}$;
2. For $\ell \in L$,
    2.1. compute $T_i^\ell = \phi_\ell(t_i)$ for $1 \leq i \leq n$ and $P_i^\ell = \phi_\ell(p_i)$ for $1 \leq i \leq m$,
    2.2. compute $s_j^\ell = \sum_{k=1}^{m} T_{j+k-1}^\ell \cdot P_k^\ell$ for $1 \leq j \leq n-m+1$ by FFT;
3. compute $\hat{s_i} = \frac{1}{h} \sum_{\ell \in L} s_j^\ell$ for $1 \leq j \leq n-m+1$.
```

Figure 2: The randomized algorithm B for the problem of string matching with mismatches.

$$\begin{aligned} &= \frac{(\nu-1)^2}{\nu^2 h} V\left[\sum_{j=1}^{m} \phi_\ell(t_{j+k-1}) \cdot \phi_\ell(p_j)\right] \\ &\leq \frac{(\nu-1)^2}{\nu^2 h} \left(\sum_{j=1}^{m} \sqrt{V\left[\phi_\ell(t_{j+k-1}) \cdot \phi_\ell(p_j)\right]}\right)^2 . \end{aligned}$$

By the definition of ϕ_ℓ, $V[\phi_\ell(a) \cdot \phi_\ell(a)] = 0$ for any $a \in \Sigma$. In the case where $a \neq b$, since $(\phi_\ell(a) \cdot \phi_\ell(b))^2 = 1$,

$$\begin{aligned} &V[\phi_\ell(a) \cdot \phi_\ell(b)] \\ &\quad = E\left[(\phi_\ell(a) \cdot \phi_\ell(b))^2\right] - E[\phi_\ell(a) \cdot \phi_\ell(b)]^2 \\ &\quad = \frac{1}{\nu-1} \sum_{\ell=2}^{\nu} 1 - \left(-\frac{1}{\nu-1}\right)^2 \\ &\quad = \frac{\nu(\nu-2)}{(\nu-1)^2} \end{aligned}$$

for any $a, b \in \Sigma$. Therefore, since $\nu \leq 2\sigma - 2$, the variance of the estimates is bounded by

$$\begin{aligned} V[\hat{s_j}] &\leq \frac{(\nu-1)^2}{\nu^2 h} \left((m - c_j) \cdot \sqrt{\frac{\nu(\nu-2)}{(\nu-1)^2}} + c_j \cdot 0\right)^2 \\ &= \frac{(\nu-2)(m-c_j)^2}{\nu h} \\ &\leq \frac{(\sigma-2)(m-c_j)^2}{(\sigma-1)h} \end{aligned}$$

for $1 \leq j \leq n-m+1$.

Theorem 3 *In the randomized algorithm B with a Sylvester-type Hadamard's matrix, the variance of the estimates of the score vector is bounded by $(\sigma-2)(m-c_j)^2/(\sigma-1)h$ for $1 \leq j \leq n-m+1$.*

Additionally, in the case where the sampling of ℓ is operated without replacement, by the basic property of the variance,

$$V[\hat{s_j}] \leq \frac{\nu-h-1}{\nu-2} \cdot \frac{(\sigma-2)(m-c_j)^2}{(\sigma-1)h}$$

$$= \frac{(2\sigma - h - 3)(m - c_j)^2}{2(\sigma - 1)h}$$

for $1 \le j \le n - m + 1$.

5 Related Work

5.1 The Standard Algorithm

The main idea of FFT-based algorithms, which computes the score vector as a correlation (or a convolution) of two numerical vectors in $O(n \log m)$ time for strings of lengths m and n, was essentially developed by Fischer and Paterson [6]. A generalized algorithm on this idea is described simply in [7].

In the standard algorithm, two strings are converted into binary strings with respect to each character in the alphabet for the numerical computation of FFT, and the score vector is the sum of all results of the correlations. Namely, for the functions $\phi_x : \Sigma \to \{0, 1\}$ for $x \in \Sigma$ such that

$$\phi_x(a) = \delta(x, a)$$

for any $a \in \Sigma$, it is clear that

$$\sum_{x \in \Sigma} \phi_x(a) \cdot \phi_x(b) = \delta(a, b)$$

for any $a, b \in \Sigma$. Therefore, the score vector between $t \in \Sigma^n$ and $p \in \Sigma^m$ is

$$c_j = \sum_{x \in \Sigma} \sum_{k=1}^{m} \phi_x(t_{j+k-1}) \cdot \phi_x(p_k) \quad (1 \le j \le n - m + 1).$$

Since $\sum_{k=1}^{m} \phi_x(t_{j+k-1}) \cdot \phi_x(p_k)$ for $1 \le j \le n - m + 1$ with respect to an $x \in \Sigma$ is computed as a part of a correlation of two vectors, an $O(\sigma n \log m)$ algorithm is obtained in the same way as the algorithm A in Section 3.

5.2 Randomized Algorithms

The computation time of the standard algorithm is not practical for strings over a large alphabet. As a solution of this problem, Atallah *et al.* [1] introduced a Monte Carlo-type algorithm in which the computation time is reduced by a trade-off with the accuracy of the estimates for the score vector. In this algorithm, an estimate is the arithmetic mean of some samples, and a sample is computed with respect to a function which is chosen independently and uniformly from the set of functions from Σ to the set of complex numbers.

Let Φ be the set of the functions from Σ to $\{0, 1, \ldots, \sigma - 1\}$ and $\omega_\sigma = e^{2\pi i/\sigma}$. Then, since $\sum_{j=0}^{\sigma-1} \omega_\sigma^j = 0$,

$$\frac{1}{|\Phi|} \sum_{\phi \in \Phi} \omega_\sigma^{\phi(a)} \cdot \overline{\omega_\sigma^{\phi(b)}} = \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \omega_\sigma^{\phi(a)-\phi(b)} = \delta(a, b)$$

for any $a, b \in \Sigma$. Therefore, the score vector between $t \in \Sigma^n$ and $p \in \Sigma^m$ is

$$c_j = \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \sum_{k=1}^{m} \omega_\sigma^{\phi(t_{j+k-1})} \cdot \overline{\omega_\sigma^{\phi(p_k)}} \quad (1 \le j \le n - m + 1).$$

Thus, a randomized algorithm is obtained in the same way as the algorithm B in Section 4. The expectation of the estimate is equal to the score vector and the variance of the estimates is bounded by $(m - c_j)^2/h$ for the number h of samples. (Strictly, if we regard the real part of the output as the estimate, then the upper bound is $(m - c_j)^2/2h$ [10].)

Schoenmeyr and Yu-Zhang [10] modified the previous algorithm such that Φ is restricted to the set of the bijective functions, and therefore the size of the set is $\sigma!$. The authors claim that the upper bound of the variance of the estimates in their algorithm is $\sigma(\sigma - 3)(m - c_i)^2/2(\sigma - 1)^2 h$, that is, the variance of the estimates decreases and it is notable for small alphabets.

Nakatoh *et al.* [8, 9] reduced the size of the set of functions which covert characters into complex numbers to $2\sigma - 2$ [8] and $\sigma - 1$ [9]. The upper bounds of the variance in the previous two algorithms are lower than that in [1]. Moreover, since the sizes of the sets are small compared with those in the previous two algorithms, σ^σ and $\sigma!$, the algorithms by Nakatoh *et al.* can be utilized as deterministic algorithms in the case where σ is small, and the variance decreases greatly in the case where the sampling of the function is operated without replacement.

5.3 Algorithms by Binary Vectors

In the randomized algorithms in the previous subsection, the $O(n \log m)$ computation consists of two DFTs and a single IDFT, and the DFTs are for vectors of complex numbers which are converted from input strings. If the vectors are expressed by vectors of binary numbers such as vectors over $\{0, 1\}$ or $\{-1, 1\}$, some speed-up methods can be applied to the $O(n \log m)$ computation of FFT. Generally, as to FFT for vectors of real numbers, there exist efficient algorithms [11].

First of all, the size of each vector is practically half since a complex number is treated as $a + ib$ for real numbers a and b. Therefore, the computation time is half in some standard FFT algorithms which treat a vector of complex numbers as two vectors of real numbers. Next, the product of a complex number $a + ib$ and a real number d can be computed in a short time compared with the product of $a + ib$ and a complex number $a' + ib'$, that is, the numbers of products of real numbers are 2 and 4, respectively. (Note that even if we consider vectors of real numbers, it remains computations with complex numbers in FFT.) Additionally, in the case where d is 1, -1, or 0, the products in the computations of DFTs can be ignored, and hence the number of products decreases greatly in a practical sense.

The standard algorithm, in which input strings are converted into binary vectors, can be randomized in the same way as the algorithms in the previous subsection, however the limit of the variance of the estimates as σ tends to infinity is infinity. Namely, the accuracy of the estimates in a randomized version of the standard algorithm is not practical.

Baba *et al.* [2] proposed a randomized algorithm as an improvement of the algorithm in [1]. In this algorithm, each character is converted into 1 or -1 and the number of the possible functions from Σ to $\{-1, 1\}$ is 2^σ. The upper bound of the variance of the estimates is $(m - c_i)^2/h$.

5.4 Comparison

We compare the algorithm B in Section 4 with the randomized algorithms referred in this section: the four randomized algorithms [1, 10, 8, 9] in Subsection 5.2, the randomized version of the standard algorithm, and the other algorithm [2] in Subsection 5.3.

We focus on the computation time and the variance of the estimates of the score vector. The result of the comparison is summarized in Table 1.

In Table 1, (a) is the range of the functions which convert characters into numbers for FFT, that is, the domain of the elements of numerical vectors. As mentioned in Subsection 5.3, the practical computation time of FFT for real-valued vectors, especially, for vectors of 1, -1, or 0 is short compared with vectors of complex numbers. This leads an improvement of the accuracy of an estimate which is computed in a given time, since the variance is inverse proportion to the number of samples. (c) is the limit of the upper bound of the variance as σ tends to infinity. If the computation time is reduced to be half by using vectors over $\{-1, 1\}$ instead of vectors over $\mathbf{C}$, it compensates for the double variance.

(b) is the upper bound of the variance of the estimates. By Theorem 3, the upper bound of the variance in the algorithm B is explicitly lower than that in the algorithm in [2].

(d) is the number of the functions which convert characters into numbers, that is, the size of the population for samples in each randomized algorithms. In the case where the sampling is operated without replacement, the improvement of the variance is obtained notably when the size is small. If we consider the straightforward product for FFT, the lower bound of the size is $\sigma - 1$ [3]. In the algorithm B, the size is $\sigma - 1$ if σ is a power of two.

6 Conclusion

In this paper, we proposed a randomized algorithm by FFT for the problem of string matching with mismatches. The algorithm consists of FFT computations for binary vectors which can be computed faster than the computation for complex numbers. Therefore, an improvement of the variance of the estimates is obtained by speed-up for the computation of FFT. We analyzed the variance of the estimates in the proposed algorithm and compared it with the variances in the existing algorithms. Our future work is some experiments for practical data with specific speed-up algorithms for FFT of binary vectors.

References

[1] M. J. Atallah, F. Chyzak, and P. Dumas. A randomized algorithm for approximate string matching. *Algorithmica*, 29(3):468–486, 2001.

[2] K. Baba, A. Shinohara, M. Takeda, S. Inenaga, and S. Arikawa. A note on randomized algorithm for string matching with mismatches. *Nordic Journal of Computing*, 10(1):2–12, 2003.

[3] K. Baba, Y. Tanaka, T. Nakatoh, and A. Shinohara. A generalization of FFT algorithms for string matching. In *Proc. International Symposium on Information Science and Electrical Engineering 2003 (ISEE 2003)*, pages 191–194. Kyushu University, 2003.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms, Second Edition.* MIT Press, 2001.

[5] M. Crochemore and W. Rytter. *Text Algorithms.* Oxford University Press, 1994.

[6] M. J. Fischer and M. S. Paterson. String-matching and other products. In *Complexity of Computation (SIAM-AMS Proceedings)*, pages 113–125, 1974.

[7] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.

[8] T. Nakatoh, K. Baba, D. Ikeda, Y. Yamada, and S. Hirokawa. An efficient mapping for scores of string matching. *Journal of Automata, Languages and Combinatorics*, 10(5/6):697–704, 2005.

[9] T. Nakatoh, K. Baba, M. Mori, and S. Hirokawa. An optimal mapping for score of string matching with FFT (in Japanese). *DBSJ Letters*, 6(3):25–28, 2007.

Table 1: A comparison of randomized algorithms by FFT for the problem of string matching with mismatches. (a) is the domain of the elements of numerical vectors for FFT, (b) is the upper bound of the variance of the estimates, (c) is the limit of (b) as σ tends to infinity, and (d) is the size of the population for samples. **C** is the set of complex numbers and $\alpha = (m - c_i)^2/h$.

	(a)	(b)	(c)	(d)
ACD01 [1]	$\mathbf{C}$	$\alpha/2$	$\alpha/2$	σ^σ
SY05 [10]	$\mathbf{C}$	$\alpha \cdot \sigma(\sigma - 3)/2(\sigma - 1)^2$	$\alpha/2$	$\sigma!$
NBIYH05 [8]	$\mathbf{C}$	$\alpha \cdot (\sigma - 2)/(2\sigma - 1)$	$\alpha/2$	$\sigma - 1 \sim 2\sigma - 2$
NBMH07 [9]	$\mathbf{C}$	$\alpha \cdot (\sigma - 3)/2\sigma$	$\alpha/2$	$\sigma - 1$
(Standard)	$\{0, 1\}$	$(\sigma c_i^2 - 1)$	(∞)	σ
BSTIA03 [2]	$\{-1, 1\}$	α	α	2^σ
Proposed	$\{-1, 1\}$	$\alpha \cdot (\sigma - 2)/(\sigma - 1)$	α	$\sigma - 1 \sim 2\sigma - 3$

[10] T. Schoenmeyr and D. Yu-Zhang. FFT-based algorithms for the string matching with mismatches problem. *Journal of Algorithms*, 57:130–139, 2005.

[11] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus. Real-valued fast Fourier transform algorithms. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-35(6):849–863, 1987.