A Note on Randomized Algorithm for String Matching with Mismatches

Baba, Kensuke
Faculty of Information Science and Electrical Engineering, Kyushu University

Shinohara, Ayumi
Faculty of Information Science and Electrical Engineering, Kyushu University

Takeda, Masayuki
Faculty of Information Science and Electrical Engineering, Kyushu University

Inenaga, Shunsuke
Faculty of Information Science and Electrical Engineering, Kyushu University

他

出版情報：Nordic journal of computing. 10 (1), pp.2-12, 2003-03. Publishing Association Nordic Journal of Computing
バージョン：
権利関係：

A Note on Randomized Algorithm for String Matching with Mismatches*

Kensuke Baba† Ayumi Shinohara Masayuki Takeda Shunsuke Inenaga
Setsuo Arikawa

Abstract

Atallah *et al.* introduced a randomized algorithm for string matching with mismatches, which utilized fast Fourier transformation (FFT) to compute a convolution. It estimates the score vector of matches between a text string and a pattern string, that is, the vector obtained when the pattern is slid along the text and the number of matches is counted for each position. In this paper, we simplify the algorithm and give an exact analysis of the variance of the estimates.
Keywords: Pattern matching, mismatch, FFT, convolution, randomized algorithm.

1 Introduction

String matching problem is to find all occurrences of a pattern string in a text string. Let $T = t_1 \cdots t_n$ be a text string and $P = p_1 \cdots p_m$ be a pattern string over an alphabet Σ. *Approximate string matching problem* is to find all occurrences of small variations of the original pattern P in the text T. Substitution, insertion, and deletion operations are often allowed to introduce the variations. In this paper, we allow the substitution operation only. The derived problem is usually called *string matching with mismatches*. Refer the textbooks [5, 6, 8] to know the history and various results. The string matching with mismatches is essentially regarded as a problem to compute the *score vector* $C(T,P) = (c_1, \ldots, c_{n-m+1})$ between T and P, where each c_i counts the number of matches between the substring $t_i \cdots t_{i+m-1}$ of the text T and the pattern P. If $c_i = m$, the pattern exactly occurs at position i in the text. Fig. 1 shows an example of the score vector.

Although the naive algorithm which solves this problem needs $O(mn)$, it can be computed in $O(n \log m)$ time by utilizing the *fast Fourier transformation (FFT)*. This approach was essentially developed by Fischer and Paterson [7], and other related work is found in [1, 3, 4]. An algorithm on this approach is described simply in [8], which compares two strings over $\{0,1\}$ converted from T and P for the arithmetic operation. However, the time complexity of a straightforward application of this method depends on the cardinality of Σ. For example, the algorithm in [8] repeats the $O(n \log m)$ operation $|\Sigma|$ times to compute the score vector exactly.

Recently, Atallah *et al.* [2] introduced a randomized algorithm of Monte-Carlo type which returns an estimation of the score vector $C(T,P)$. The estimation is performed by averaging independent equally distributed estimates, which is yielded by mapping the characters into a complex plane uniformly. Let k be the number of randomly sampled estimations, then the time complexity is $O(kn \log m)$. They showed that the expected value of the estimation is equal to the score vector, and that the variance is bounded by $(m - c_i)^2/k$.

In this paper, we give a slight simplification of their algorithm. Concretely, characters are converted to $\{-1, 1\}$ instead of complex numbers of size $|\Sigma|$. Moreover, we analyze the variance of the estimator exactly.

2 Preliminaries

Let $\mathcal{N}$ be the set of non-negative integers. Let Σ be a finite *alphabet*. An element of Σ^* is called a *string*. The length of a string w is denoted by $|w|$. The empty string is denoted by ε, that is, $|\varepsilon| = 0$. We denote the cardinality of a set S by $|S|$ or $\#S$.

We define a function δ from $\Sigma \times \Sigma$ to $\{0,1\}$ by

$$\delta(a,b) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{if } a \neq b. \end{cases}$$

For a text string $T = t_1 \cdots t_n$ and a pattern string $P = p_1 \cdots p_m$, *the score vector of matches between T and P* is defined as

$$C(T,P) = (c_1, c_2, \ldots, c_{n-m+1}),$$

where

$$c_i = \sum_{j=1}^{m} \delta(t_{i+j-1}, p_j).$$

*An edited version of this report was published in: *Nordic Journal of Computing, vol.10(1), pp.2–12, Mar, 2003.*

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

i	1	2	3	4	5	6	7	8	9	10
text	a	c	b	a	b	b	a	c	c	b
pattern	a	b	b	a	c					
		a	b	b	a	c				
			a	b	b	a	c			
				a	b	b	a	c		
					a	b	b	a	c	
						a	b	b	a	c
c_i	3	1	1	5	2	0				

Figure 1: Score vector between the text `acbabbaccb` and the pattern `abbac`.

That is, c_i is the number of matches between the text and the pattern when the first character of the pattern in positioned at the ith character of the string.

3 Deterministic Algorithm

In this section, we introduce a deterministic algorithm to compute the score vector for given text T and pattern P. Although it might not be practical for large alphabet, it will be a base for the randomized algorithm explored in the next section.

3.1 Binary Alphabet Case

We first consider a binary alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. We define a function $\psi : \Sigma \to \{-1, 1\}$ by $\psi(\mathtt{a}) = 1$ and $\psi(\mathtt{b}) = -1$. By using ψ, we convert the strings T and P into the sequences of integers as follows.

$$\begin{aligned} \psi(T) &= \psi(t_1)\psi(t_2)\cdots\cdots\cdots\psi(t_n), \\ \psi(P) &= \psi(p_1)\psi(p_2)\cdots\psi(p_m). \end{aligned}$$

Let $A^\psi(T,P) = (a_1^\psi, a_2^\psi, \ldots, a_{n-m+1}^\psi)$ where $a_i^\psi = \sum_{j=1}^{m} \psi(t_{i+j-1}) \cdot \psi(p_j)$.

Lemma 1 *For any* $1 \le i \le n-m+1$, $c_i = (a_i^\psi + m)/2$.

Proof. Since $c_i = \#\{j \mid t_{i+j-1} = p_j,\ 1 \le j \le m\}$, we have $a_i^\psi = \#\{j \mid t_{i+j-1} = p_j,\ 1 \le j \le m\} - \#\{j \mid t_{i+j-1} \ne p_j,\ 1 \le j \le m\} = c_i - (m - c_i) = 2c_i - m$. Thus $c_i = (a_i^\psi + m)/2$. □

The above lemma implies that we have only to compute $A^\psi(T,P)$ to get the score vector $C(T,P)$. Since the sequence $A^\psi(T,P)$ is the convolution of $\psi(T)$ with the reverse of $\psi(P)$, we can calculate all the a_i's simultaneously by the use of FFT in $O(n \log m)$ time as follows. As is stated in [2], we additionally apply the standard technique [5] of partitioning the text into overlapping chunks of size $(1+\alpha)m$ each, and then processing each chunk separately. Processing one chunk gives us αm components of C. Since we have $n/(\alpha m)$ chunks and each chunk can be computed in $O((1+\alpha)m \log((1+\alpha)m))$ by FFT, the total time complexity is $(n/\alpha m) \cdot O((1+\alpha)m \log((1+\alpha)m)) = O\left(\frac{(1+\alpha)}{\alpha} n \log((1+\alpha)m)\right) = O(n \log m)$ by choosing $\alpha = O(m)$.

Theorem 1 *For a binary alphabet, the score vector C can be exactly computed in $O(n \log m)$ time.*

3.2 General Case

We now consider general case $|\Sigma| > 2$. Let Ψ_Σ be the set of all mappings from Σ to $\{-1, 1\}$. Remark that $|\Psi_\Sigma| = 2^{|\Sigma|}$. We abbreviate Ψ_Σ with Ψ when Σ is clear from the context. The next lemma is obvious.

Lemma 2 *For any $\psi \in \Psi_\Sigma$ and any $a, b \in \Sigma$,*

$$\psi(a) \cdot \psi(b) = \begin{cases} 1 & \text{if } \psi(a) = \psi(b), \\ -1 & \text{if } \psi(a) \ne \psi(b). \end{cases}$$

Lemma 3 *For any $a, b \in \Sigma$,*

$$\frac{1}{|\Psi|} \sum_{\psi \in \Psi} \psi(a) \cdot \psi(b) = \delta(a,b).$$

Proof. In the case where $a = b$, then $\psi(a) = \psi(b)$ for any $\psi \in \Psi$. Therefore $\psi(a) \cdot \psi(b) = 1$ for any ψ by Lemma 2, and the sum $\sum_{\psi \in \Psi} \psi(a) \cdot \psi(b)$ equals to the cardinality of Ψ. Thus, the left side of the equation is unity.

To prove the lemma in the case where $a \ne b$, we show a more general proposition: *for* $n \ge 0$

$$\sum_{\psi \in \Psi} \psi(d_1) \cdot \cdots \cdot \psi(d_n) \cdot \psi(b) = 0 \text{ if } d_1 \ne b, \ldots, d_n \ne b.$$

By the assumption that b is distinct from $d_1, \cdots, d_n$,

$$\sum_{\psi \in \Psi} \psi(d_1) \cdot \cdots \cdot \psi(d_n) \cdot \psi(b)$$

$$
\begin{aligned}
&= \sum_{\psi(b)=1,\psi\in\Psi} \psi(d_1)\cdot\cdots\cdot\psi(d_n)\cdot 1 \\
&\quad + \sum_{\psi(b)=-1,\psi\in\Psi} \psi(d_1)\cdot\cdots\cdot\psi(d_n)\cdot(-1) \\
&= 0.
\end{aligned}
$$

Thus, by the proposition for $n = 1$, the left side of the equation is zero. □

Theorem 2 *For any* $1 \le i \le m - n + 1$,

$$c_i = \frac{1}{|\Psi|}\sum_{\psi\in\Psi} a_i^{\psi}. \quad (1)$$

Proof. By the definition of a_i^{ψ} and Lemma 3, the right side of the equation can be written as follows.

$$
\begin{aligned}
\frac{1}{|\Psi|}\sum_{\psi\in\Psi} a_i^{\psi} &= \frac{1}{|\Psi|}\sum_{\psi\in\Psi}\sum_{j=1}^{m}\psi(t_{i+j-1})\cdot\psi(p_j) \\
&= \sum_{j=1}^{m}\frac{1}{|\Psi|}\sum_{\psi\in\Psi}\psi(t_{i+j-1})\cdot\psi(p_j) \\
&= \sum_{j=1}^{m}\delta(t_{i+j-1},p_j).
\end{aligned}
$$

Since the last formula is the definition of c_i, the theorem is proved. □

Theorem 3 $C(T,P)$ *can be exactly computed in* $O(2^{|\Sigma|}n\log m)$ *time.*

Proof. By Theorem 2, each c_i is the mean of a_i^{ψ} for every $\psi \in \Psi_\Sigma$, therefore $C(T,P)$ is obtained by computing all $A^{\psi}(T,P)$. Since each $A^{\psi}(T,P)$ can be computed in $O(n\log m)$ time, we can calculate $C(T,P)$ in $O(2^{|\Sigma|}n\log m)$ time. □

We note that if the alphabet Σ is infinite, by splitting the text in chunks of length $O(m)$ to be dealt with independently ensures it will work with an alphabet size $O(m)$, $C(T,P)$ can be exactly computed in $O(2^{O(m)}n\log m)$.

4 Randomized Algorithm

A shortcoming of the deterministic algorithm in the last section is that the running time is exponential with respect to the size of alphabet. It is not practical for large alphabet. In this section, we propose a randomized algorithm which was inspired by Atallah *et al.* [2].

Let us noticed that Theorem 2 can be interpreted as follows. Each c_i is the mean of random variable $X_i = \sum_{j=1}^{m}\psi(t_{i+j-1})\cdot\psi(p_j)$, assuming that ψ is drawn uniformly randomly from Ψ. The observation leads us to the following randomized algorithm. Instead of computing all vectors $A_\psi(T,P) = (a_1^{\psi}, a_2^{\psi}, \ldots, a_{n-m+1}^{\psi})$ where $a_i^{\psi} = \sum_{j=1}^{m}\psi(t_{i+j-1})\cdot\psi(p_j)$ to average them, we compute only k samples of them for randomly chosen $\psi_1,\ldots,\psi_k \in \Psi$. Since the expected value of X_i equals to c_i, it will give a good estimation for large enough k. We will give a formal proof of it, and exactly analyze the variance of X_i in the sequel. Fig. 2 illustrates the core part of the algorithm for the basic case $n = (1+\alpha)m$.

We now analyze the mean and the variance of the estimator $\hat{c}_i$. Since all the random variable $\hat{c}_i$ are defined in a similar way, we generically consider the random variable

$$\hat{s} = \frac{1}{k}\sum_{\ell=1}^{k}\sum_{j=1}^{m}\psi(t_j)\cdot\psi(p_j)$$

where the t_j's and the p_j's are fixed and mapping ψ's are independently and uniformly selected from Ψ_Σ. The definition implies that $\hat{s}$ is the mean of k random variables which are drawn from independent and identical distribution. The random variable can be defined by

$$s = \sum_{j=1}^{m}\psi(t_j)\cdot\psi(p_j),$$

and the mean $E(\hat{s})$ and variance $V(\hat{s})$ are

$$E(\hat{s}) = E(s) \quad \text{and} \quad V(\hat{s}) = \frac{V(s)}{k}.$$

The number c of matches between $T = t_1\cdots t_m$ and $P = p_1\cdots p_m$ is

$$c = \sum_{j=1}^{m}\delta(t_j,p_j).$$

Lemma 4 *The mean of* $\hat{s}$ *is equal to* c.

Proof. By Lemma 3,

$$
\begin{aligned}
E(\hat{s}) = E(s) &= \frac{1}{|\Psi|}\sum_{\psi\in\Psi} s \\
&= \frac{1}{|\Psi|}\sum_{\psi\in\Psi}\sum_{j=1}^{m}\psi(t_j)\cdot\psi(p_j) \\
&= \sum_{j=1}^{m}\frac{1}{|\Psi|}\sum_{\psi\in\Psi}\psi(t_j)\cdot\psi(p_j) \\
&= \sum_{j=1}^{m}\delta(t_j,p_j).
\end{aligned}
$$

Thus, the mean of $\hat{s}$ is c. □

In order to analyze the variance of s accurately, we introduce the following function $\rho_{T,P} : \Sigma\times\Sigma \to \mathcal{N}$

Procedure EstimateScore
Input: a text $T = t_1 \cdots t_{(1+\alpha)m}$ and a pattern $P = p_1 \cdots p_m$ in Σ^*;
Output: an estimate for the score vector $C(T, P)$.
for $\ell := 1$ **to** k **do begin**
randomly and uniformly select a ψ_ℓ from Ψ_Σ;
let $T_\ell = \psi_\ell(T)$;
{Note that T_ℓ is a sequence over $\{-1, 1\}$ of length $(1 + \alpha)m$.}
let P_ℓ be the concatenation of $\psi_\ell(P)$ with trailing αm zeros;
compute the vector C_ℓ as the convolution of T_ℓ with the reverse of P_ℓ by FFT;
end
compute the vector $\hat{C} = \frac{1}{k}\sum_{\ell=1}^{k} C_\ell$ and output it as an estimate of $C(T, P)$.

Figure 2: Randomized Algorithm

depending on text $T = t_1 \cdots t_m$ and pattern $P = p_1 \cdots p_m$, which give a statistics of T and P.

$$\rho_{T,P}(a, b) = \#\{j \mid t_j = a \text{ and } p_j = b,\ 1 \le j \le m\}$$

For example, if $T = \mathtt{aabac}$ and $P = \mathtt{abbba}$, then $\rho_{T,P}(\mathtt{a}, \mathtt{b}) = 2$, $\rho_{T,P}(\mathtt{a}, \mathtt{a}) = \rho_{T,P}(\mathtt{b}, \mathtt{b}) = \rho_{T,P}(\mathtt{c}, \mathtt{a}) = 1$, and the others are zero. We omit the subscription T, P of $\rho_{T,P}$ in the sequel. In addition, we use the following expression.

$$\tau(a, b) = \rho(a, b) + \rho(b, a).$$

The next lemma is obvious from the definition.

Lemma 5 $\displaystyle\sum_{(a,b)\in\Sigma\times\Sigma} \rho(a, b) = \frac{1}{2}\sum_{(a,b)\in\Sigma\times\Sigma} \tau(a, b) = m.$

The next lemma gives the exact variance of $\hat{s}$, in terms of ρ.

Lemma 6 *The variance of $\hat{s}$ is*

$$V(\hat{s}) = \frac{1}{k}\sum_{a\neq b} \left(\rho(a, b)^2 + \rho(a, b) \cdot \rho(b, a)\right).$$

Proof. Since the mean of s equals to c by Lemma 4,

$$V(\hat{s}) = \frac{1}{k}V(s) = \frac{1}{k}\frac{1}{|\Psi|}\sum_{\psi\in\Psi}(s - c)^2.$$

By the definition of ρ,

$$\begin{aligned} s &= \sum_{(a,b)\in\Sigma\times\Sigma} \psi(a) \cdot \psi(b) \cdot \rho(a, b) \\ &= \sum_{a=b} \rho(a, b) + \sum_{a\neq b} \psi(a) \cdot \psi(b) \cdot \rho(a, b), \text{ and} \\ c &= \sum_{a=b} \rho(a, b). \end{aligned}$$

Therefore,

$$\begin{aligned} &\frac{1}{|\Psi|}\sum_{\psi\in\Psi}(s - c)^2 \\ &= \frac{1}{|\Psi|}\sum_{\psi\in\Psi}\left(\sum_{a\neq b} \psi(a) \cdot \psi(b) \cdot \rho(a, b)\right)^2 \\ &= \sum_{a\neq b}\left(\rho(a, b) \cdot \sum_{a'\neq b'} \rho(a', b') \cdot \alpha(a, b, a', b')\right), \end{aligned}$$

where

$$\alpha(a, b, a', b') = \frac{1}{|\Psi|}\sum_{\psi\in\Psi} \psi(a) \cdot \psi(b) \cdot \psi(a') \cdot \psi(b').$$

Then, we show that

$$\alpha(a, b, a', b') = \begin{cases} 1 & \text{if either } a = a' \text{ and } b = b', \\ & \quad \text{or } a = b' \text{ and } a' = b, \\ 0 & \text{otherwise,} \end{cases}$$

by the case analysis whether there exists a distinct character from the others in a, b, a', b'. If there exists such a character, then $\alpha(a, b, a', b') = 0$ by the proof of Lemma 3. If there does not exist such a character, then we have either $a = a'$ and $b = b'$, or $a = b'$ and $b = a'$ by the assumption that both $a \neq b$ and $a' \neq b'$. Then, by Lemma 3 and the fact that $\psi(a)^2 = 1$ for any $\psi \in \Psi$ and any $a \in \Sigma$ since $\psi(a) \in \{-1, 1\}$,

$$\alpha(a, b, a', b') = \frac{1}{|\Psi|}\sum_{\psi\in\Psi} \psi(a)^2 \cdot \psi(b)^2 = 1.$$

Thus,

$$V(\hat{s}) = \frac{1}{k}\sum_{a\neq b} \rho(a, b)\left(\rho(a, b) + \rho(b, a)\right)$$

$$= \frac{1}{k}\sum_{a\neq b}\left(\rho(a,b)^2 + \rho(a,b)\cdot\rho(b,a)\right).$$

□

Moreover, by the definition of τ, we have

$$\begin{aligned}
&\sum_{a\neq b}\left(\rho(a,b)^2 + \rho(a,b)\cdot\rho(b,a)\right)\\
&= \frac{1}{2}\sum_{a\neq b}\left(\rho(a,b)^2 + 2\rho(a,b)\cdot\rho(b,a) + \rho(b,a)^2\right)\\
&= \frac{1}{2}\sum_{a\neq b}\left(\rho(a,b) + \rho(b,a)\right)^2\\
&= \frac{1}{2}\sum_{a\neq b}\tau(a,b)^2\\
&= \sum_{a<b}\tau(a,b)^2.
\end{aligned}$$

Therefore, the variance can be exactly restated in term of τ as follows, which might be more intuitive.

Theorem 4 *The variance of $\hat{s}$ is*

$$V(\hat{s}) = \frac{1}{k}\sum_{a<b}\tau(a,b)^2.$$

Remind that $\tau(a,b)$ represented the number of positions $j = 1,\ldots,m$ in T and P, such that (t_j, p_j) is either (a,b) or (b,a). If T exactly matches P, then $V(\hat{s}) = 0$, which implies that the estimation is always m, without any error. On the other hand, since $\sum_{a<b}\tau(a,b) = m - c$, the variance $V(\hat{s})$ is maximized for inputs which have no match and are constructed by only two characters, for example, $T =$ `aaaaaa`, $P =$ `bbbbbb`, and $T =$ `aaabba`, $P =$ `bbbaab`.

We now state the bound of the variance of $\hat{s}$ in terms of m and c, that exactly fits to the one proved by Atallah *et al.* [2].

Lemma 7 *The variance of $\hat{s}$ is bounded as follows.*

$$V(\hat{s}) \leq \frac{(m-c)^2}{k}.$$

Proof. By Lemma 5,

$$\begin{aligned}
m - c &= \sum_{(a,b)\in\Sigma\times\Sigma}\rho(a,b) - \sum_{a=b}\rho(a,b)\\
&= \sum_{a\neq b}\rho(a,b)\\
&= \frac{1}{2}\sum_{a\neq b}\tau(a,b)\\
&= \sum_{a<b}\tau(a,b).
\end{aligned}$$

Therefore, by Theorem 4,

$$\begin{aligned}
&\frac{(m-c)^2}{k} - V(\hat{s})\\
&= \frac{1}{k}\left(\sum_{a<b}\tau(a,b)\right)^2 - \frac{1}{k}\sum_{a<b}\tau(a,b)^2\\
&= \frac{1}{k}\sum_{a<b}\left(\tau(a,b)\cdot\sum_{a'<b'}\tau(a',b') - \tau(a,b)^2\right)\\
&= \frac{1}{k}\sum_{a<b}\left(\tau(a,b)\cdot{\sum_{a'<b'}}^{*}\tau(a',b')\right),
\end{aligned}$$

where ${\sum_{a'<b'}}^{*}\tau(a',b')$ expresses the sum of $\tau(a',b')$ except for the two cases $a' = a, b' = b$ and $a' = b, b' = a$. Since $\tau(a,b) \geq 0$ for any a and b, the last formula is not less than zero. □

We now have the main theorem.

Theorem 5 *Algorithm* EstimateScore *runs in $O(kn\log m)$ time. The mean of the estimation equals to the score vector C, and the variance of each entry is bounded by $(m - c_i)^2/k$.*

5 Conclusion

We gave a randomized algorithm for string matching with mismatches, which can be regarded as a slight simplification of the one due to Atallah *et al.* [2]. For comparison, we give a brief description of their algorithm. It treats the set Ψ' of all mappings from Σ to $\{0,1,\ldots,|\Sigma|-1\}$, and the basic equation is

$$c_i = \frac{1}{|\Psi'|}\sum_{\psi\in\Psi'}\sum_{j=1}^{m}\omega^{\psi(t_{i+j-1})-\psi(p_j)}, \qquad (2)$$

where ω is a primitive $|\Sigma|$th root of unity. When $|\Sigma| = 2$, we know $\omega = -1$, and that Eq. (2) directly corresponds to Eq. (1) in ours. The difference is how to treat general alphabet $|\Sigma| > 2$. In our algorithm, the converted sequence $\psi(T)$ is simply over $\{-1,1\}$, while in their algorithm $\psi(T)$ is over $\{1,\omega,\omega^2,\ldots,\omega^{|\Sigma|-1}\}$ that are complex numbers. When computing the convolution by FFT, the computation of the former will be much simpler (and possibly faster) than the latter. From the view point of the precision of the numerical calculations, the former might be preferable to the latter, although we have not yet studied explicitly. Moreover, this simplification enabled us to reach the exact estimation of the variance (Theorem 4), by fairly primitive discussion. An interesting point is that the variance is still independent from the size of alphabet, although we map Σ into $\{-1,1\}$, instead of $\{0,1,\ldots,|\Sigma|-1\}$.

In the situation that the cardinality of Σ is not large, the deterministic algorithm in [8] mentioned in Section 1 can compute the score vector exactly in practical time. By modifying the definition of ψ as

$$\psi_x(a) = \begin{cases} 1 & \text{if } a = x, \\ 0 & \text{if } a \neq x, \end{cases}$$

then we have

$$c_i = \sum_{x \in \Sigma} \sum_{j=1}^{m} \psi_x(t_{i+j-1}) \cdot \psi_x(p_j)$$

for any $1 \leq i \leq n - m + 1$. This implies that the score vector $C(T, P)$ can be exactly computed in $O(|\Sigma| n \log m)$ time. Since an estimation for ψ_x counts matches with respect to a certain character x, the expected value is not equal to the score vector. Hence the randomized algorithm cannot be applied to this algorithm.

References

[1] K. Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16(6):1039–1051, 1987.

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

[3] R. A. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Commun. ACM*, 35:74–82, 1992.

[4] R. A. Baeza-Yates and C. H. Perleberg. Fast and practical approximate string matching. *Inf. Process. Lett.*, 59(1):21–27, 1996.

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

[6] M. Crochemore and W. Rytter. *Jewels of Stringology*. World Scientific, 2003.

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

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