Expected Length of Longest Common Subsequences of Two Biased Random Strings and Its Application

Hironobu AOKI[1], Ryuhei UEHARA[2] and Koichi YAMAZAKI[1]

[1] Department of Computer Science,
Gunma University
1-5-1 Tenjin-cho, Kiryu
Gumma 376-8515, Japan
{aoki,koichi}@comp.cs.gunma-u.ac.jp

[2] Natural Science Faculty,
Komazawa University
1-23-1 Komazawa, Setagaya-ku,
Tokyo 154-8525, Japan
uehara@komazawa-u.ac.jp

Abstract: The longest common subsequence (LCS) of two strings is one of the main problems in combinatorial pattern matching. A famous open problem for the LCS is its expected length of two random strings over a uniformly distributed alphabet. In this paper, we consider the expected length of the LCS for two biased random strings, and show the upper and lower bounds depending on the bias. We next consider some approximation algorithms for decomposability problem of shuffled strings. We propose two deterministic algorithms and two randomized algorithms. Using the results for the LCS of two biased random strings, we show good approximation ratios for the randomized algorithms.

Keywords: approximation algorithm, longest common subsequence, decomposability of shuffled strings.

1 Introduction

The longest common subsequence (LCS) of two strings is one of the main problems in combinatorial pattern matching. We say that s is a subsequence of u if we can obtain s by deleting zero or more letters of u. The LCS of two strings u and v is defined as the longest subsequence s common to u and v. The LCS is related to DNA or protein alignments, file comparison, speech recognition, etc. A famous open problem for the LCS is its expected length of two random strings of length n over a uniformly distributed alphabet of size k. It is known that the expected length is proportional to the length of the given sequences, but the exact value of this proportion, denoted by γ_k, is not known even for $k=2$. For $k=2$, the best known upper and lower bounds are $0.77391 \leq \gamma_2 \leq 0.83763$, and an experimental interval is $0.8120 \leq \gamma_2 \leq 0.8125$ [Dan94, DP95, PD94]. For general k, $1 \leq \gamma_k\sqrt{k} \leq e$ is known [CS75, Dek79] (see [BYGNS99] for a comprehensive reference).

From the practical point of view, the expected length of the LCS can be used for the inference of randomness. That is, if an alignment of common subsequence of two given sequence is relatively larger than γ_k, we may infer that it is more than a coincidence, and that the similarity should be studied. For example, two long random DNA sequences will have on average an alignment comprising 65% of its length, and that amount of similarity might be counterintuitive.

However, if two given sequences are biased, we cannot use the value γ_k to infer the randomness. In this paper, we first consider the expected length of two *biased* random strings of length n. More precisely, the random strings are generated over the alphabet $\{0,1,\cdots,k-1\}$ such that each alphabet i occurs with positive probability p_i with $\sum_{i=0}^{k-1} p_i = 1$. We first show that the expected length of two biased random strings takes minimum value when the alphabet is uniformly distributed. That is, γ_k is the minimum proportion to the length of the given biased random sequences. We also show the upper and lower bounds of the expected length for two biased random strings.

Iwama
stated formal treatment of asynchronous time-multiplexed communication, and discussed several decomposability of shuffled strings [Iwa83].

As an application of the expected length of two biased random strings, we consider a problem for shuffled strings. The shuffle of y and z is defined by $y \odot z = \{y_1 z_1 y_2 z_2 \cdots y_n z_n | y_1 y_2 \cdots y_n = y$ and $z_1 z_2 \cdots z_n = z\}$. A string x is said to be decomposed into strings y and z if x is in $y \odot z$. We consider the equally decomposable problem (EDP) that determines whether given string x is decomposable into two same strings, that is, x is in $y \odot y$ for some string y. Interestingly, the EDP is NP-complete even on binary alphabet [Iwa82, Iwa00, Ueh00]. In this paper, we consider the approximation algorithms for the EDP. To consider the approximation ratio, we modify the EDP that finds two strings y and z with $x \in y \odot z$. The measure is the length of the longest common substrings of y and z.

We first show two deterministic approximation algorithms. The first one works for the EDP on alphabet of size k, and achieves the approximation ratio $\frac{2}{k+1}$. The second one works for the EDP on binary alphabet, and achieves the approximation ratio 0.833. But the second one needs a large amount of memory ($\sim 2^{36}$) to store a table, and it takes several weeks to obtain the table.

We next show two online randomized approximation algorithms. Their approximation ratio is at least $\max\{\gamma_k, \frac{1}{4}\}$. Thus, the ratio is better than the deterministic algorithm for $k > 2$. Moreover, even if $k = 2$, the ratio is at least 0.77391, and 0.8120 experimentally.

2 Preliminaries

2.1 Expected Length of Longest Common Subsequences

We say that s is a *subsequence* of u if we can obtain s by deleting zero or more letters of u. The *longest common subsequence* (LCS) of two strings u and v is defined the longest subsequence s common to u and v. Let $lcs(u, v)$ denote the length of the LCS for two strings u and v. A famous open problem for the LCS is its expected length for two random strings of length n over a uniformly distributed alphabet Σ. More precisely, given alphabet $\Sigma = \{0, 1, \cdots, k-1\}$, it is open the exact value of γ_k that is defined below:

$$\gamma_k = \lim_{n \to \infty} \frac{1}{k^{2n}} \sum_{|u|=|v|=n} \frac{lcs(u,v)}{n}.$$

(Intuitively, γ_k is the expected value of $\frac{lcs(u,v)}{n}$ of which u and v are two random strings of length n over Σ.) Several results have been obtained (see [BYGNS99] for a comprehensive reference), and the following results will be used in this paper.

Fact 1 (1) For $k = 2$, the best known upper and lower bounds are $0.77391 \leq \gamma_2 \leq 0.83763$, and an experimental interval is $0.8120 \leq \gamma_2 \leq 0.8125$, given by Dančík and Paterson [Dan94, DP95, PD94].

(2) In general cases, $1 \leq \gamma_k \sqrt{k} \leq e$ [CS75, Dek79].

2.2 Shuffled Strings

Let x and y be strings over an alphabet Σ. Then the *shuffle* of x and y, denoted by $x \odot y$, is defined by $x \odot y = \{x_1 y_1 x_2 y_2 \cdots x_n y_n | n \geq 1, x_i$ and y_i in $\Sigma^*, x_1 x_2 \cdots x_n = x$ and $y_1 y_2 \cdots y_n = y\}$. For example, $01 \odot 101 = \{01101, 01011, 10101, 10011\}$. Let x, y and z be strings over an alphabet Σ. We say z is *decomposable* into x and y if $z \in x \odot y$. We now define *equally decomposable problem* as follows:

Input: A string z of length $2n$ over an alphabet Σ.

Question: Determine whether z is decomposable into two same strings, i.e., $z \in x \odot x$ for some string x.

We denote by $EDP(k)$ the equally decomposable problem on an alphabet $\Sigma = \{0, 1, \cdots, k-1\}$. Then the following fact is known.

Fact 2 $EDP(k)$ is NP-complete [Iwa82, Theorem 3], even if $k = 2$ [Iwa00, Ueh00].

(See also [Iwa83] for other general problems and applications for decomposability of shuffled strings.)

In this paper, we will consider approximation algorithms for the $EDP(k)$. Thus we slightly modify the problem to evaluate the approximation ratio. The modified $EDP(k)$ is defined as follows:

Input: A string z of length $2n$ over an alphabet $\Sigma = \{0, 1, \cdots, k-1\}$.

Output: Two strings x and y of length n.

Measure: $lcs(x, y)$.

Remark that we only consider the output strings of length n. The following lemma guarantees that it is sufficient to consider the strings of length n.

Lemma 3 Let z be a string of length $2n$, and x and y be two strings such that $|x| < n < |y|$ and $z \in x \odot y$. Then there are two strings $\hat{x}$ and $\hat{y}$ such that $|\hat{x}| = |\hat{y}| = n$, $z \in \hat{x} \odot \hat{y}$, and $lcs(\hat{x}, \hat{y}) \geq lcs(x, y)$.

Proof. Since $|y| > |x| \geq lcs(x, y)$, y contains at least one letter, say ℓ, that does not used in the longest common subsequence of x and y. We then delete the letter ℓ from y, and insert into x. That is, when we decompose z into x and y, we output ℓ as a part of x instead of y. Then, we obtain the new strings x' and y'. It is easy to see that $lcs(x', y') \geq lcs(x, y)$, $|y'| = |y| - 1$, and $|x'| = |x| + 1$. We can repeat this process until we obtain two strings of the same length. ∎

3 Longest Common Subsequences of Two Biased Random Strings

We consider the expected length of the longest common subsequences of two *biased* random strings. That is, given alphabet $\Sigma = \{0, 1, \cdots, k-1\}$, we assume that each letter i occurs with fixed positive probability p_i with $\sum_{i=0}^{k-1} p_i = 1$. We now denote by $\gamma_k(p_0, p_1, \cdots, p_{k-1})$ the expected value of $\lim_{n\to\infty} \frac{lcs(u,v)}{n}$ of which u and v are two random strings of length n on the probability space defined by those p_is.

Theorem 4 For any k, the expected value of $\gamma_k(p_0, p_1, \cdots, p_{k-1})$ takes the minimum value when $p_0 = p_1 = \cdots = p_{k-1} = \frac{1}{k}$. That is, γ_k is the minimum value of $\gamma_k(p_0, p_1, \cdots, p_{k-1})$.

Proof. We first assume that $k = 2$. Let $x = x_0 x_1 \cdots x_{n-1}$ and $y = y_0 y_1 \cdots y_{n-1}$ be any strings of length n over $\{0, 1\}$. We let $x = x' x_{n-1}$ and $y = y' y_{n-1}$ for short.

Adding one letter to the end of one string, the longest common subsequence grows at most one. Thus we have $\mathrm{E}[lcs(x', y')] \leq \mathrm{E}[lcs(x', y)] \leq \mathrm{E}[lcs(x', y')] + 1$ for any distribution.

In [Dan94, Section 2.2], Dančík have showed that

$$lcs(x, y) = \begin{cases} lcs(x', y') + 1 & \text{if } x_{n-1} = y_{n-1} \\ lcs(x, y) = \max\{lcs(x', y), lcs(x, y')\} \\ \qquad \text{if } x_{n-1} \neq y_{n-1}. \end{cases}$$

When we choose x and y randomly, the event $x_{n-1} = y_{n-1}$ occurs with probability $p_0^2 + p_1^2$, and $\mathrm{E}[lcs(x', y)] = \mathrm{E}[lcs(x, y')]$ from their symmetrically. Thus we have $\mathrm{E}[lcs(x, y)] = (p_0^2 + p_1^2)(\mathrm{E}[lcs(x', y')] + 1) + 2p_0 p_1 \max\{\mathrm{E}[lcs(x', y)], \mathrm{E}[lcs(x, y')]\} = (p_0^2 + p_1^2)(\mathrm{E}[lcs(x', y')] + 1) + 2p_0 p_1 \mathrm{E}[lcs(x', y)]$.

Since $\mathrm{E}[lcs(x', y)] \leq \mathrm{E}[lcs(x', y')] + 1$, to minimize the formula $(p_0^2 + p_1^2)(\mathrm{E}[lcs(x', y')] + 1) + 2p_0 p_1 \mathrm{E}[lcs(x', y)]$, we shall minimize the probability $p_0^2 + p_1^2 = 2(p_0 - \frac{1}{2})^2 + \frac{1}{2}$ with $0 < p_0 < 1$. Thus $\mathrm{E}[lcs(x, y)]$ takes the minimum value when $p_0 = p_1 = \frac{1}{2}$.

When $k > 2$, using the same argument, we have $\mathrm{E}[lcs(x', y)] \leq \mathrm{E}[lcs(x', y')] + 1$, and $\mathrm{E}[lcs(x, y)] = (p_0^2 + p_1^2 + \cdots + p_{k-1}^2)(\mathrm{E}[lcs(x', y')] + 1) + 2\sum_{i \neq j} p_i p_j \mathrm{E}[lcs(x', y)]$. Since $p_0^2 + p_1^2 + \cdots + p_{k-1}^2$ takes the minimum value when $p_0 = p_1 = \cdots = p_{k-1} = \frac{1}{k}$, we have the theorem. ∎

We next turn to obtain the values of $\gamma_2(p, 1-p)$. Figure 1 depicts the values of $\gamma_2(p, 1-p)$. The solid line depicts the lower bound, and the plus marks are the upper bounds, which are the results in the rest of this section. The x marks are the experimental values obtainded as follows: The experimental value of $\gamma_2(0.5, 0.5)$ is obtained by Dančík. Clearly, $\gamma_2(0, 1) = 1$, and $\gamma_2(p, 1-p) = \gamma_2(1-p, p)$. The experimental values for $p = 0.1, 0.2, 0.3, 0.4$ are obtained from the average values of 1000 random strings of length 100000. For example, $\gamma_2(0.3, 0.7)$ is at least 0.7988 and at most 0.8909 from the theoretical results, and is approximately equal to 0.8397 from the experimental results.

3.1 Upper bound

We here show upper bound of $\gamma_2(p, 1-p)$.

Theorem 5 Let $H(x)$ be the binary entropy function $-x \log x - (1-x)\log(1-x)$, and $w(x) =$

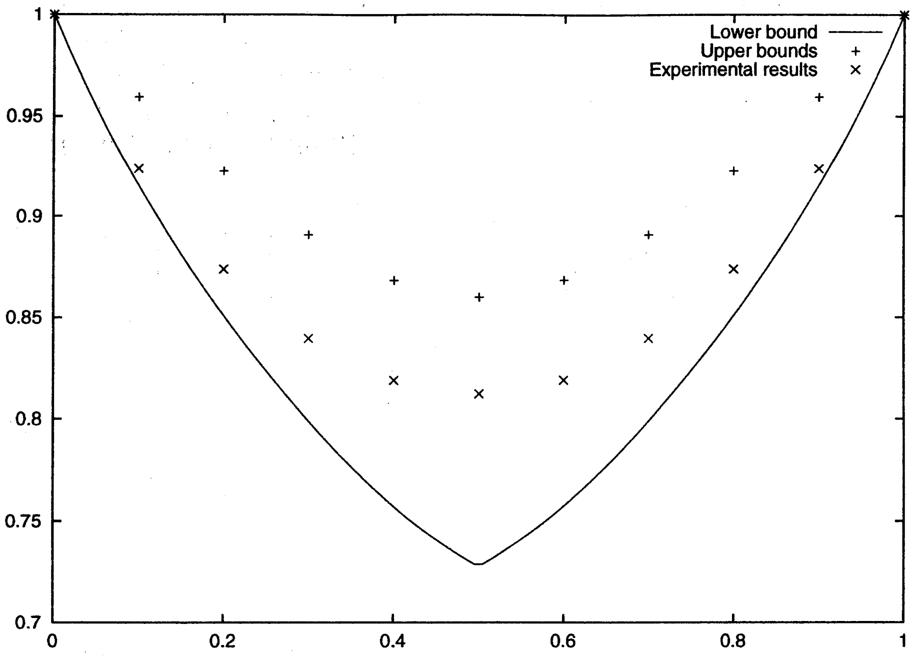

Figure 1: Bounds of $\gamma_2(p, 1-p)$

$\frac{1}{2}(2 - x - \sqrt{5x^2 - 8x + 4})$. Then $\gamma_2(p, 1-p)$ satisfies $\gamma_2(p, 1-p)H\left(\frac{w(\gamma_2(p,1-p))}{\gamma_2(p,1-p)}\right) + (1 - \gamma_2(p, 1-p))H\left(\frac{w(\gamma_2(p,1-p))}{1-\gamma_2(p,1-p)}\right) + (1 - w(\gamma_2(p, 1-p)))H\left(\frac{\gamma_2(p,1-p)-w(\gamma_2(p,1-p))}{1-\gamma_2(p,1-p)}\right) \geq (2 - \gamma_2(p, 1-p))H(p)$.

We note that the values of the plus marks in Figure 1 are obtained from numerical analysis of the inequation in the theorem using Mathematica system.

Proof. We prove the theorem expanding the results in [BYGNS99]. Using the standard Kolmogorov complexity arguments, they first obtain the following results for general k: $(\log I_{n,\gamma_k})/n + 2(1-\gamma_k)\log(k-1) \geq (2-\gamma_k)\log k$, where I_{n,γ_k} is the number of position sets that correspond to LCSs of two strings. (Remind that γ_k is the abbreviation of $\gamma_k(\frac{1}{k}, \frac{1}{k}, \cdots, \frac{1}{k})$.)

In their analysis, they use uniformity of the distribution to obtain the terms $\log k$ and $\log(k-1)$. The first term $\log k$ means that $\log k$ bits are required to represent each letter in an alphabet of size k on the uniform distribution. Thus we need replace the term $\log k$ by $-\sum_{i=0}^{k-1} p_i \log p_i$ in our case by the basic fact of information theory. We next turn to the term $\log(k-1)$. The term $\log(k-1)$ represents the information of the sequence of letters of two given strings that do not belong to the LCS of the strings. Thus, as noted in [BYGNS99], this information is empty for $k = 2$. Thus, for $k = 2$, we have $(\log I_{n,\gamma_2(p,1-p)})/n \geq -(2-\gamma_2(p,1-p))(p\log p + (1-p)\log(1-p)) = (2-\gamma_2(p,1-p))H(p)$.

Here, clearly, $I_{n,\gamma_k(p_0,p_1,\cdots,p_{k-1})} = I_{n,\gamma_k}$ for any $0 < p_0, p_1, \cdots, p_{k-1} < 1$. Thus, we can use the same upper bounds of I_{n,γ_k} given in [BYGNS99], that is, $(\log I_{n,\gamma_2(p,1-p)})/n \leq \gamma_2(p, 1-p)H\left(\frac{w(\gamma_2(p,1-p))}{\gamma_2(p,1-p)}\right) + (1 - \gamma_2(p, 1-p))H\left(\frac{w(\gamma_2(p,1-p))}{1-\gamma_2(p,1-p)}\right) + (1 - w(\gamma_2(p, 1-p)))H\left(\frac{\gamma_2(p,1-p)-w(\gamma_2(p,1-p))}{1-w(\gamma_2(p,1-p))}\right)$, which completes the proof. ∎

3.2 Lower bound

We next show a lower bound of $\gamma_2(p, 1-p)$.

Theorem 6 For any p with $0 < p < 1$, $\gamma_2(p, 1-p)$ is at least $\max\{f(p), f(1-p)\}$, where $f(p) = \frac{2p^3-3p^2+p+1}{p^3-3p^2+2p+1}$.

Proof. We present a finite automaton that models an algorithm which finds a common subsequence

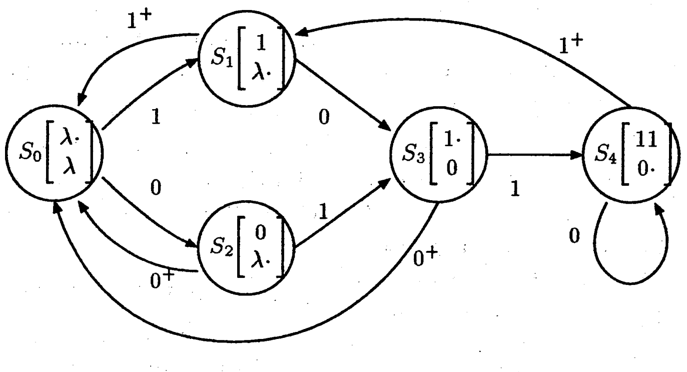

Figure 2: Finite automaton A finding a common subsequence of two biased strings

on two finite (biased) strings. By analyzing the associated Markov chain, a lower bound on the expected length of the extended LCS is obtained. The idea is based on the work by Deken [Dek79] and Dančík and Paterson [Dan94, PD94]. Since their strings are not biased, their automaton (e.g., [Dan94, Figure 3.2]) are simplified using the fact that $p = 0.5$. Hence we extend the automaton and obtain new one, say A, in Figure 2. The number of states is 5; initial state is S_0, and each state stores the previous inputs (the symbol λ describes empty). Each dot represents the string which is read at the next step. Each letter associated with an edge describes the next letter of the input string. For example, A at state S_0 moves to S_1 if the next input letter is 1, and it moves to S_2 if the next input letter is 0. Consequently, A at state S_0 moves to S_1 with probability $1-p$ and it moves to S_2 with probability p. The plus mark means that, through the transition, A produces the letter as a common substring (and the letters stored in the state are thrown away). For example, when A stays at the state S_3, A reads the letter 0 (with probability p), and then it moves to S_0 with producing 0 as a common substring (and throws away all letters stored at the state S_3).

We define the *transition matrix* of A by a matrix $\mathbf{T}_A = \{t_{i,j}\}$ such that $t_{i,j}$ is the probability that the state i changes the state j in next step. That is, we have

$$\mathbf{T}_A = \begin{pmatrix} 0 & 1-p & p & 0 & 0 \\ 1-p & 0 & 0 & p & 0 \\ p & 0 & 0 & 1-p & 0 \\ p & 0 & 0 & 0 & 1-p \\ 0 & 1-p & 0 & 0 & p \end{pmatrix}.$$

When $0 < p < 1$, every element in $\mathbf{T}_A^4$ is positive. Thus $\mathbf{T}_A$ is regular. That is, there exists a unique vector $\mathbf{d} = (d_0, d_1, d_2, d_3, d_4)$ such that $\mathbf{d}\mathbf{T}_A = \mathbf{d}$ and $\sum_{i=0}^4 d_i = 1$. Solving this system, we get $d_0 = \frac{p^2-p+1}{2(p^3-3p^2+2p+1)}$, $d_1 = \frac{p^3-2p^2+1}{2(p^3-3p^2+2p+1)}$, $d_2 = \frac{p(p^2-p+1)}{2(p^3-3p^2+2p+1)}$, $d_3 = \frac{p(1-p)}{p^3-3p^2+2p+1}$, and $d_4 = \frac{p(1-p)}{p^3-3p^2+2p+1}$. We say that a state S_i *produces* a common subsequence if it outputs a common letter when the state changes. Modifying [Dan94, Theorem 3.2] to fit our notations, we immediately have the following fact: For sufficiently large n, the expected length of the common subsequence produced by A is given by $2\sum_{i=0}^4 d_i \Pr[S_i$ produces a common subsequence$]$. Thus, we have the expected length equals to $2((1-p)d_1+pd_2+pd_3+(1-p)d_4) = \frac{2p^3-3p^2+p+1}{p^3-3p^2+2p+1} (= f(p))$. That is a lower bound of $\gamma_2(p, 1-p)$. Using the fact that $\gamma_2(p, 1-p) = \gamma_2(1-p, p)$, we also have another lower bound $f(1-p)$. Hence $\max\{f(p), f(1-p)\}$ is the lower bound of $\gamma_2(p, 1-p)$. ∎

Remark that $f(p) \neq f(1-p)$ since the automaton A is not symmetric for the letters 0 and 1.

4 Approximation algorithms

In this section, we consider the approximation algorithms for the equally decomposable problem. Throughout this section, we assume that $\Sigma = \{0, 1, \cdots, k-1\}$. We also let $s = s_0 s_1 \cdots s_{2n-1}$ be the input string, and $x = x_0 x_1 \cdots x_{n-1}$ and $y = y_0 y_1 \cdots y_{n-1}$ be the two output strings produced by the algorithm. We consider two deterministic approximation algorithms and two randomized online approximation algorithms.

4.1 Deterministic Approximation Algorithms

Our first deterministic algorithm first reads $k+1$ input letters. Then, by the pigeon hole principle, at least two of them are the same. Thus the algorithm outputs the same letters into x and y, respectively. The remainder letters are output appropriately. The algorithm repeats the process until the input string ends up. (Remark that it is easy to make the algorithm to output x and y with $|x| = |y| = n$.) We call this algorithm *Pigeon Hole Algorithm.*

Theorem 7 For *EDP*(k), Pigeon Hole Algorithm achieves the approximation ratio $\frac{2}{k+1}$.

Proof. At least two letters of $k+1$ letters are used as a part of common subsequence. Moreover, the longest common subsequence has at most length n. Thus the approximation ratio is at least $\frac{1}{2}2n\frac{2}{k+1}\frac{1}{n} = \frac{2}{k+1}$. ∎

We now focus on the *EDP*(2). In the case, the Pigeon Hole Algorithm reads only 3 letters on each process. We modify the algorithm to read more letters. To evaluate its approximation ratio, we define min - max $LCS(k,n)$ as follows: We can decompose *any* string s of length $2n$ into two strings x and y with $|x| = |y| = n$ such that $lcs(x,y)$ is at least min - max $LCS(k,n)$. More precisely, min - max $LCS(k,n)$ is equal to

$$\min_{s\in\Sigma^{2n}} \max_{\substack{s\in u\odot v,\\ |u|=|v|=n}} lcs(u,v).$$

We can compute the exact values of min - max $LCS(k,n)$ for small k and n. Letting $k=2$, we obtain the data in Table 1. (It takes several weeks to obtain the data by our PCs.) Using Table 1, we construct the extended Pigeon Hole Algorithm working for *EDP*(2); it first reads 36 letters, and computes all possible decompositions to maximize the length of the longest common subsequence. We call this algorithm *Brute Force Algorithm.* By Table 1, we immediately have the following theorem.

Theorem 8 For *EDP*(2), Brute Force Algorithm achieves the approximation ratio $15/18 = 0.833$ for sufficiently large n.

4.2 Randomized Approximation Algorithms

In this section, we first construct an online randomized algorithm. Let l_x and l_y be the length of the strings that are already produced by the algorithm as x and y, respectively. The algorithm, say R_1, is easy to describe; R_1 outputs given input s_i as x_{l_x} with probability $\frac{1}{2}$, and as y_{l_y} with probability $\frac{1}{2}$. (We remark that R_1 does not output two strings of the same length in general. We will show another randomized algorithm R_2 that outputs two strings of the same length, and achieves the same approximation ratio as R_1.) The R_1 has the following property.

Lemma 9 (1) Each possible decomposition occurs with the same probability.

(2) $\Pr[x_i = j] = \Pr[y_i = j]$ for every $0 \le i \le 2n-1$ and $0 \le j \le k-1$.

(3) For each $0 \le i \le 2n-1$ and $0 \le j \le 2n-i$, R_1 outputs s_{i+j} as x_i with probability $\frac{1}{2^{i+j}}\binom{i+j-1}{i-1}$.

Proof. The number of possible decomposition is equal to 2^{2n}, and each decomposition occurs with probability $\frac{1}{2^{2n}}$. Thus (1) holds. Symmetry of the algorithm immediately implies (2). It is not difficult to see that s_{i+j} can become x_i when $0 \le j \le 2n-i$. Here s_{i+j} becomes x_i if and only if j letters of $s_0, s_1, \cdots, s_{i+j-1}$ become a part of y, $i-1$ letters of $s_0, s_1, \cdots, s_{i+j-1}$ become a part of x, and s_{i+j} becomes x_i. Thus, we have $\Pr[s_{i+j}$ becomes $x_i] = \binom{i+j-1}{i-1}\left(\frac{1}{2}\right)^{i+j-1}\frac{1}{2} = \binom{i+j-1}{i-1}\left(\frac{1}{2}\right)^{i+j}$. ∎

Next we analyze and state two approximation ratios of R_1.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	17	18
$\min\text{-}\max LCS(2,n)$	0	1	2	2	3	4	5	6	7	8	8	9	10	11	14	15
$\min\text{-}\max LCS(2,n)/n$	0	0.5	0.667	0.5	0.6	0.667	0.714	0.75	0.778	0.8	0.727	0.75	0.769	0.786	0.824	0.833

Table 1: $\min\text{-}\max LCS(2,n)$

Theorem 10 (1) R_1 achieves the approximation ratio $\frac{1}{4}$ [Che00].

(2) R_1 achieves the approximation ratio γ_k.

Thus, using Fact 1, we immediately have the following corollary.

Corollary 11 For sufficiently large n, the approximation ratio of R_1 is

(1) for $EDP(2)$, at least 0.77391, and 0.8120 experimentally,

(2) for $EDP(k)$ with $2 < k \leq 16$, at least $\frac{1}{\sqrt{k}}$, and

(3) for $EDP(k)$ with $16 < k$, at least $\frac{1}{4}$.

We first show the proof of Theorem 10(1).

Proof. Given input s, let $x^* = x_0^* x_1^* \cdots x_{n-1}^*$ and $y^* = y_0^* y_1^* \cdots y_{n-1}^*$ be an optimal solution of the $EDP(k)$. We also let $c^* = c_0 c_1 \cdots c_{n'-1}$ be one of the longest common substrings of x^* and y^*. Each c_i corresponds to some $x^*_{\phi(i)}$ and $y^*_{\psi(i)}$, where ϕ, ψ are some functions with $0 \leq \phi(0) < \phi(1) < \cdots < \phi(n'-1) \leq n-1$ and $0 \leq \psi(0) < \psi(1) < \cdots < \psi(n'-1) \leq n-1$. Let x and y be the output of R_1. Then, one of the following four cases occurs for each i with $0 \leq i \leq n'-1$:

(1) Both $x^*_{\phi(i)}$ and $y^*_{\psi(i)}$ are output in x.

(2) Both $x^*_{\phi(i)}$ and $y^*_{\psi(i)}$ are output in y.

(3) $x^*_{\phi(i)}$ is output in x, and $y^*_{\psi(i)}$ is output in y.

(4) $x^*_{\phi(i)}$ is output in y, and $y^*_{\psi(i)}$ is output in x.

Each case occurs with the same probability $\frac{1}{4}$. Thus the expected size of the set of the letters in each case is $\frac{n'}{4}$. The letters in the case (3) produce a common substring. Thus the expected length of $lcs(x,y)$ is at least $\frac{1}{4}lcs(x^*,y^*)$. (Remark that the letters in the case (4) also produce a common substring, but we cannot use the letters in (3) and (4) at the same time.) ∎

To prove Theorem 10(2), we need a technical lemma.

Lemma 12 We fix some letter in Σ, say k'. Then for any positive real c with $0 < c < 1$ and positive real ϵ, $|\Pr[x_{cn} = k'] - \Pr[x_{cn+1} = k']| < \epsilon$ for sufficient large n.

Proof. For each i and j with $0 \leq i \leq 2n-1$ and $0 \leq j \leq k-1$, we define a function σ_j on input letter s_i such that

$$\sigma_j(s_i) = \begin{cases} 1 & \text{if } s_i = j \\ 0 & \text{if } s_i \neq j. \end{cases}$$

Then, by Lemma 9(3), we have $\Pr[x_i = k'] = \sum_{j=0}^{2n-i} \frac{1}{2^{i+j}} \binom{i+j-1}{i-1} \sigma_{k'}(s_{i+j})$. Thus, letting $\binom{i-1}{i} = 0$ for any positive integer i, we have

$$\begin{aligned} & |\Pr[x_i = k'] - \Pr[x_{i+1} = k']| \\ = & \left| \sum_{j=0}^{2n-i} \frac{1}{2^{i+j}} \binom{i+j-1}{i-1} \sigma_{k'}(s_{i+j}) \right. \\ & \left. - \sum_{j=0}^{2n-i-1} \frac{1}{2^{i+j+1}} \binom{i+j}{i} \sigma_{k'}(s_{i+j+1}) \right| \\ = & \left| \sum_{j=0}^{2n-i} \frac{1}{2^{i+j}} \binom{i+j-1}{i-1} \sigma_{k'}(s_{i+j}) \right. \\ & \left. - \sum_{j=1}^{2n-i} \frac{1}{2^{i+j}} \binom{i+j-1}{i} \sigma_{k'}(s_{i+j}) \right| \\ = & \left| \sum_{j=0}^{2n-i} \frac{1}{2^i} \sigma_{k'}(s_{i+j}) \left(\frac{1}{2^j} \binom{i-1+j}{i-1} \right. \right. \\ & \left. \left. - \frac{1}{2} \frac{1}{2^{j-1}} \binom{i+j-1}{i} \right) \right|. \end{aligned}$$

Using the equation $\sum_{k=0}^{\infty}\binom{n+k}{n}z^k = \frac{1}{(1-z)^{n+1}}$ (see [GKP89, (5.56)]), we have

$$\lim_{n'\to\infty}\sum_{j=0}^{n'}\frac{1}{2^j}\binom{i-1+j}{i-1} = 2^i, \text{ and}$$

$$\lim_{n'\to\infty}\sum_{j=0}^{n'}\frac{1}{2^{j-1}}\binom{i+j-1}{i} = 2^{i+1}.$$

Hence, for any positive real ϵ', we get $|\Pr[x_i = k'] - \Pr[x_{i+1} = k']| < \left|\sum_{j=0}^{2n-i}\frac{1}{2^i}\sigma_{k'}(s_{i+j})\epsilon'\right|$ for sufficiently large n and $i = cn$. Since $0 \leq \sigma_{k'}(s_{i+j}) \leq 1$, we finally have $|\Pr[x_i = k'] - \Pr[x_{i+1} = k']| < \epsilon$ for sufficiently large n and $i = cn$. ∎

We here show the proof of Theorem 10(2).

Proof. Repeating Lemma 12, we can show that for any positive integer c and i, each letter in Σ occurs with some fixed probability in any substrings $x_i x_{i+1}\cdots x_{i+c}$ and $y_i y_{i+1}\cdots y_{i+c}$. More precisely, for any positive integer c and positive real d with $0 < d < 1$ and positive real ϵ, $\max\{\Pr[x_{dn} = k'], \Pr[x_{dn+1} = k'], \cdots, \Pr[x_{dn+c} = k']\} - \min\{\Pr[x_{dn} = k'], \Pr[x_{dn+1} = k'], \cdots, \Pr[x_{dn+c} = k']\} < \epsilon$ for each k' in Σ and sufficiently large n, and so is $y_i y_{i+1}\cdots y_{i+c}$. Thus we can regard that, on the substrings, each letter i in Σ occurs with fixed probability $p(i)$ with $0 \leq i \leq k-1$. Thus, using Lemma 9(2), we have the approximation ratio $\gamma_k(p(0), p(1), \cdots, p(k-1))$ on the substrings. By Theorem 4, it takes the minimum value γ_k when $p(0) = p(1) = \cdots = p(k-1) = \frac{1}{k}$. Thus R_1 achieves the approximation ratio at least γ_k. ∎

Remark. When the input string is biased, we can obtain better approximation ratio. We let $k = 2$. Then, for each $0 \leq i \leq n-1$, $\Pr[x_i = 1](= \Pr[y_i = 1]) = \sum_{j=0}^{2n-i}\frac{1}{2^{i+j}}\binom{i+j-1}{i-1}\sigma_1(s_{i+j})$ by Lemma 9(3). For example, if $\max\{\Pr[x_i = 1]\} < 0.2$ for $0 \leq i \leq n-1$, R_1 achieves the approximation ratio at least $\gamma_2(0.2, 0.8) \geq 0.850932$ by Theorem 6.

In general, the randomized algorithm R_1 does not produce two strings of the same length. This point can be less useful than Pigeon Hole Algorithm and Brute Force Algorithm. Hereafter, we state another online randomized algorithm R_2. It always outputs two strings of the same length, and converges to the same behavior of R_1 asymptotically. Thus, in practical, we can use R_2 instead of R_1.

Let l_x and l_y be the length of the strings that are already produced by R_2 as x and y, respectively. The algorithm R_2, step by step, outputs given input s_i as x_{l_x} with probability $\frac{n-l_x}{2n-l_x-l_y}$, and as y_{l_y} with probability $\frac{n-l_y}{2n-l_x-l_y}$. The algorithm R_2 has the following property.

Lemma 13 (1) $|x| = |y| = n$.

(2) Each possible decomposition occurs with the same probability.

(3) $\Pr[x_i = j] = \Pr[y_i = j]$ for every $0 \leq i \leq n-1$ and $0 \leq j \leq k-1$.

(4) For each $0 \leq i \leq n-1$ and $0 \leq j \leq n$, R_2 outputs s_{i+j} as x_i with probability $\binom{2n-(i+j)}{n-i}\binom{i+j-1}{i-1}/\binom{2n}{n}$.

Proof. (1) When the length of x becomes n, the probability $\frac{n-l_x}{2n-l_x-l_y}$ becomes zero. This immediately shows (1).

(2) We assume that the algorithm already produced

$x_0\cdots x_{l_x-1}$ and $y_0\cdots y_{l_y-1}$. Then the algorithm will decompose $s_{l_x+l_y}\cdots s_{2n-1}$ into $x_{l_x}\cdots x_n$ and $y_{l_y}\cdots y_n$. The number of the decomposition is $\binom{2n-(l_x+l_y)}{n-l_x}$. Among the decomposition, the number of the decomposition with $s_{l_x+l_y} = x_{l_x}$ is $\binom{2n-(l_x+l_y)-1}{n-l_x-1}$, and the number of the decomposition with $s_{l_x+l_y} = y_{l_y}$ is $\binom{2n-(l_x+l_y)-1}{n-l_x}$. Now, $\binom{2n-(l_x+l_y)-1}{n-l_x-1}/\binom{2n-(l_x+l_y)}{n-l_x} = \frac{n-l_x}{2n-l_x-l_y}$, and $\binom{2n-(l_x+l_y)-1}{n-l_x}/\binom{2n-(l_x+l_y)}{n-l_x} = \frac{n-l_y}{2n-l_x-l_y}$, that concludes the proof of (2).

(3) Symmetry of the algorithm implies.

(4) It is clear that s_i is the first letter being able to become x_i. Reversing the strings, we also have that $s_{2n-1-(n-i-1)} = s_{n+i}$ is the last letter being able to become x_i. Thus, the value of x_i depends on s_{i+j} with $0 \leq j \leq n$. Thus it is sufficient to show the following claim; The probability that $s_{i+j} = x_i$ is $\binom{2n-(i+j)}{n-i}\binom{i+j-1}{j}/\binom{2n}{n}$ with $0 \leq j \leq n$.

To become the input s_{i+j} the output x_i, among $s_0, \cdots, s_{i+j-1}$, $\binom{i+j-1}{j}$ letters of them become a part of the string y, and $\binom{i+j-1}{i-1}$ letters of them become a part of the string x. The number of such decomposition is $\binom{i+j-1}{j}$. Moreover, by (2), each possible decomposition occurs with the same probability equal to

$\frac{n(n-1)\cdots(n-i+1)\cdot n(n-1)\cdots(n-j+1)}{2n(2n-1)\cdots(2n-i+1)\cdot(2n-i)\cdots(2n-(i+j-1))}$. Thus we have the probability equal to $\binom{i+j-1}{i-1}\frac{n(n-1)\cdots(n-i+1)\cdot n(n-1)\cdots(n-j+1)}{2n(2n-1)\cdots(2n-i+1)\cdot(2n-i)\cdots(2n-(i+j-1))} = \binom{i+j-1}{i-1}\frac{(2n-(i+j)!)n!n!}{(2n)!(n-i)!(n-j)!} = \frac{\binom{i+j-1}{i-1}\binom{2n-(i+j)}{n-i}}{\binom{2n}{n}}$. ∎

Intuitively, two algorithms R_1 and R_2 converge the same behavior asymptotically because the expected probability that R_2 outputs s_i as a part of x is $\frac{1}{2}$. However, the probability depends on the previous output string. Thus we need to show more precise result.

Lemma 14 Fix any letter, say k', in Σ. For any positive real ϵ and c with $0 < c < 1$, we have $|$Prob$[R_1$ outputs k' as $x_{cn}]-$ Prob$[R_2$ outputs k' as $x_{cn}]| < \epsilon$ for sufficiently large n.

Proof. By Lemmas 9 and 13, we have $|$Prob$[R_1$ outputs k' as $x_{cn}]-$ Prob$[R_2$ outputs k' as $x_{cn}]$ $| = \sum_{j=0}^{2n-i}\binom{i+j-1}{i-1}\sigma_{k'}(s_{i+j})\left|\frac{1}{2^{i+j}} - \frac{\binom{2n-(i+j)}{n-i}}{\binom{2n}{n}}\right|$. We now estimate the value of $\binom{2n-(i+j)}{n-i}/\binom{2n}{n}$. Hereafter, we denote by $f(n) \sim g(n)$ if $\lim_{n\to\infty}\frac{f(n)}{g(n)} = 1$. Now we let $i = cn$ and $j = dn$ with $0 < c < 1$ and $0 < d < 1$. Then, using Stirling's approximation formula $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ and an asymptotic expansion $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$ (see [GKP89, Exercise 9.60]), we have $\binom{2n-(i+j)}{n-i}/\binom{2n}{n} \sim \frac{((2-c-d)n)!}{((1-c)n)!((1-d)n)!}\frac{\sqrt{\pi n}}{4^n} \sim \sqrt{\frac{2-c-d}{2(1-c)(1-d)}}\left(\frac{(2-c-d)^{2-c-d}}{(1-c)^{1-c}(1-d)^{1-d}}\right)^n$. It is not difficult to check that for any $0 < c,d < 1$, $0 \le \frac{1}{2^{c+d}} - \frac{(2-c-d)^{2-c-d}}{(1-c)^{1-c}(1-d)^{1-d}} \le \frac{1}{4}$. On the other hand, $\sqrt{\frac{2-c-d}{2(1-c)(1-d)}}$ is a positive constant. Thus, since $\frac{1}{2^{i+j}} = \left(\frac{1}{2^{c+d}}\right)^n$, we have $0 \le \frac{1}{2^{i+j}} - \frac{\binom{2n-(i+j)}{n-i}}{\binom{2n}{n}} \le \frac{1}{2^{i+j}}$ for sufficiently large n. Thus we finally get $|$Pr$[R_1$ outputs k' as $x_{cn}]$ $-$ Pr$[R_2$ outputs k' as $x_{cn}]| <$ ϵ for sufficiently large n and $i = cn$ using the equation $\sum_{k=0}^{\infty}\binom{n+k}{n}z^k = \frac{1}{(1-z)^{n+1}}$ again. ∎

Using Theorem 10 and Lemma 14, we have the following theorem.

Theorem 15 The randomized online algorithm R_2 achieves the approximation ratio $\max\{\frac{1}{4},\gamma_k\}$ for sufficiently large n.

It is worth remarking that the idea of the R_2 can be applied to the following general case; We want to decompose a input string s of length n into several strings $x^0, x^1, \cdots, x^{k-1}$ of length $n_0, n_1, \cdots, n_{k-1}$ with $\sum_{i=0}^{k-1} n_i = n$. Let $l_0, l_1, \cdots, l_{k-1}$ be the length of the strings that are already produced by the extended R_2 as $x^0, x^1, \cdots, x^{k-1}$, respectively. Then the extended R_2 outputs given input s_i as x^j with probability $\frac{n_j - l_j}{n-(l_1+l_2+\cdots+l_k)}$ for each j with $0 \le j \le k-1$. Using the same strategy of the proof of Lemma 13 with multinomial coefficient, we can show that the extended R_2 has the following property; $|x^j| = n_j$, and each possible decomposition occurs with the same probability equal to $1/\binom{n}{n_0,n_1,\cdots,n_{k-1}}$ (see, e.g., [GKP89] for the notion of multinomial coefficient).

5 Concluding Remarks

There are two interesting further works related with each other. First one is evaluating the exact value of min-max$\,LCS(k,n)/n$. It is not difficult to see that $\gamma_k \le$ min-max$\,LCS(k,n)/n \le 1$ for sufficiently large n. It seems to increase for n, but we do not know if it has an upper bound less than 1. Second one is determining if $EDP(k)$ has a polynomial time approximation scheme (see [Hoc95] for the notion of the polynomial time approximation scheme). Two problems have the following relation: If min-max$\,LCS(k,n)/n$ converges to 1, $EDP(k)$ has a polynomial time approximation scheme. More precisely, for any given $\epsilon > 0$, there exists positive constant n_0 with min-max$\,LCS(k,n_0)/n_0 \ge 1-\epsilon$ if min-max$\,LCS(k,n)/n$ converges to 1. Thus we can construct the algorithm based on the same idea of Brute Force Algorithm, which first reads n_0 letters and computes all possible decompositions. Clearly this algorithm achieves the approximation ratio $1-\epsilon$.

Acknowledgments

The authors thank to Professor Zhi-Zhong Chen who pointed that the algorithm R_1 achieves the approximation ratio $\frac{1}{4}$ stated in Theorem 10(1). The authors also wish to thank Professor Kazuo Iwama for his helpful advice for this work.

References

[BYGNS99] R.A. Baeza-Yates, R. Gavaldà, G. Navarro, and R. Scheihing. Bounding the Expected Length of Longest Common Subsequences and Forests. *Theory of Computing Systems*, 32(4):435–452, 1999.

[Che00] Z.-Z. Chen. Personal communication. 2000.

[CS75] V. Chvátal and D. Sankoff. Longest Common Subsequences of Two Random Sequences. *J. Appl. Prob.*, 12:306–315, 1975.

[Dan94] V. Dančík. *Expected Length of Longest Common Subsequences.* PhD thesis, University of Warwick, 1994. available at `http://www.dcs.warwick.ac.uk/pub/reports/theses/dan94.html`.

[Dek79] J.G. Deken. Some Limit Results fo Longest Common Subsequences. *Discrete Mathematics*, 26:17–31, 1979.

[DP95] V. Dančík and M. Paterson. Upper Bounds for the Expected Length of a Longest Common Subsequence of Two Binary Sequences. *Random Structures and Algorithms*, 6(4):449–458, 1995.

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing Company, 1989.

[Hoc95] D. Hochbaum (eds.). *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1995.

[Iwa82] K. Iwama. On Equations Including String Variables. In *Proc. 23th Symp. on Foundations of Computer Science*, pages 226–235. IEEE, 1982.

[Iwa83] K. Iwama. Unique Decomposability of Shuffled Strings: A Formal Treatment of Asynchronous Time-Multiplexed Communication. In *Proc. 15th ACM Symp. on the Theory of Computing*, pages 374–381. ACM, 1983.

[Iwa00] K. Iwama. Personal communication. 2000.

[PD94] M. Paterson and V. Dančík. Longest Common Subsequences. In *Proc. 19th MFCS*, pages 127–142. Lecture Notes in Computer Science Vol. 841, Springer-Verlag, 1994.

[Ueh00] R. Uehara. NP-completeness of Equally Decomposable Problem. manuscript, 2000.