Sankhyā : The Indian Journal of Statistics
2017, Volume 79-B, Part 1, pp. 1-20

Theoretical Lower Bound for Border Length Minimization Problem

S Srinivasan
Defence Research and Development Organization (DRDO), Chennai, India
V Kamakoti
Department of Computer Science and Engineering, Indian Institute of Technology (IIT), Chennai, India
A Bhattacharya
Defence Research and Development Organization (DRDO), Hyderabad, India

Abstract

Biochemicalanalysis procedures, that include genomics and drug discovery, have been formalized to an extent that they can be automated. Large Microarrays housing DNA probes are used for this purpose. Manufacturing these microarrays involve depositing the respective DNA probes in each of its cells. The deposition is carried out iteratively by masking and unmasking cells in each step. A masked cell of the microarray that is adjacent (shares a border) to an unmasked one is at a high risk of being exposed in a deposition step. Thus, minimizing the number of such borders (Border length minimization) is crucial for reliable manufacturing of these microarrays. Given a microarray and a set of DNA probes, computing a lower bound on the border length is crucial to study the effectiveness of any algorithm that solves the border length minimization problem. A Numerical method for computing this lower bound has been proposed in the literature. This takes prohibitively large time. In practice, the DNA probes are random sequences of nucleotides. Based on this realistic assumption, this paper attempts to estimate the lower bound for the border length analytically using a probability theoretic approach by reducing the same to the problems of computing the probability distribution functions (PDF) for the Hamming Distance and the length of the longest common subsequence (LCS) between two random strings. To the best of our knowledge, no PDF is reported earlier for the length of the LCS between two random strings.

AMS (2000) *subject classification.* Primary 46N30; Secondary 60C05, 62P10.
Keywords and phrases. DNA, Microarrays, Longest Common Subsequence (LCS), Border Minimization Problem (BMP)

Nomenclature

p_n, $n \in \{A, C, G, T\}$	Probability of occurrence of the nucleotide n
L	Length (number of nucleotides) of input probes
l	Length of Longest Common Subsequence (LCS)
i, j	Indices for matrix/sequence.
r	Variable for distance between a pair of probes
$p_{conflict}$	Probability of an arbitrary pair of nucleotides being different.
$p_{identical}$	Probability of an arbitrary pair of nucleotides being identical.
$P_{i,j}(r)$	Probability of a pair of strings of lengths i and j having an LCS of length r characters.

1 Introduction

A study of genetic characteristics of an organism is crucial for various purposes like disease-diagnosis, drug discovery and forensics. These genetic characteristics are housed in the DNA (Deoxyribo Nucleic Acid) molecule of the organism. Any DNA molecule D is composed of two *strands* X and Y. Each strand is a sequence made of four basic nucleotides (also called *bases*), namely, *Adenine (A)*, *Guanine (G)*, *Cytosine (C)* and *Thymine (T)*. Therefore, the strands X and Y may be represented as follows:

$$X : x_1x_2x_3.....x_L$$

$$Y : y_1y_2y_3.....y_L$$

where, $x_i,\ y_i \in \{A, C, G, T\}\ \forall i,\ 1 \leq i \leq L$. The DNA molecule made of these two strands (X and Y) can be represented as:

$$D : (x_1, y_1)(x_2, y_2)(x_3, y_3).....(x_L, y_L) \tag{1.1}$$

The DNA molecule D is said to be L *base-pair*s long with each strand being L *nucleotides* long. Such a DNA molecule is said to be *stable* if and only if each (x_i, y_i) $1 \leq i \leq L$, satisfy the *Watson-Crick base pairing rule* (Watson and Crick, 1953). The Watson-Crick rule imposes a constraint on the type of nucleotides that can be paired together and is summarized in Table 1. As can be seen in Table 1:

if $(x_i = A)$ **then** y_i *has* to be T and vice versa.
if $(x_i = C)$ **then** y_i *has* to be G and vice versa.

Table 1: Complement of a Nucleotide as per Watson-Crick rule

x_i	y_i
A	T
C	G
G	C
T	A

Figure 1 shows two DNA molecules that are 5-base-pairs long. The DNA molecule in Fig. 1a is stable as each of the pairs satisfy the Watson-Crick rule. The DNA molecule in Fig. 1b is not stable as the fourth pair, namely, (CT) does not satisfy the Watson-Crick rule.

Analyzing the individual DNA strands of a given biological sample (like human blood) is important for many activities that include disease diagnosis and drug discovery. A DNA microarray is used for this purpose. Figure 2 depicts the procedure followed for such an analysis using a DNA microarray. Figure 2a shows the given stable DNA molecule that is 7 base-pairs long. Figure 2b shows the decomposition of the DNA molecule into two strands, each 7 nucleotides long. These strands are to be analyzed using the DNA microarray shown in Fig. 2c. The DNA microarray in Fig. 2c comprises 9 sites arranged as a 3×3 matrix. Each site stores a sequence of nucleotides called *probes*. Thus, there are 9 probes stored in the DNA microarray shown in Fig. 2c, namely, TAC, TAG, AAC, AGC, ATG, GCT, TGC, GAA and CGA. The analysis of the DNA strands in Fig. 2b using the microarray

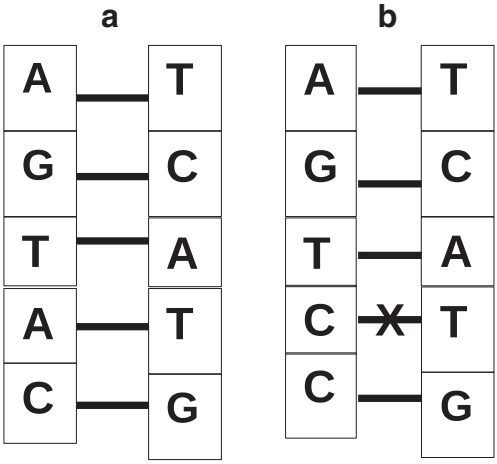

Figure 1: Watson-Crick rule in formation of a stable DNA molecule

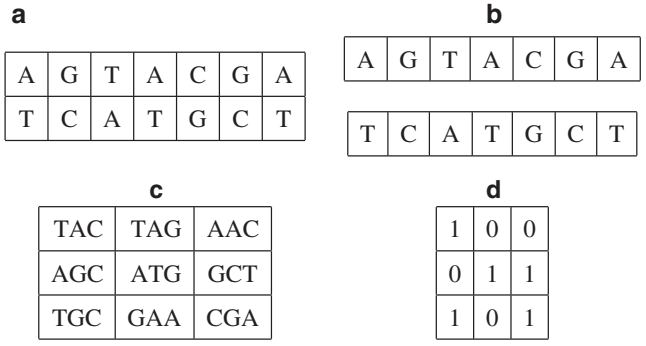

Figure 2: Example for Usage of Microarray. **a** Given DNA Sample (Both Strands Bonded). **b** Given DNA Sample decomposed into individual strands. **c** The Microarray used for analyzing the DNA sample decomposed into individual strands as in (**b**). **d** The output of microarray says whether the given sample contains the corresponding probe as its substring (1) or not (0)

in Fig. 2c involves passing both the strands into each of the nine probes. Once the strands are passed, each site in the microarray produces a binary response (0 or 1) as defined below:

if the probe stored in the site is a *substring* of any one of the strands, **then**

output 1 **else** output 0.

Figure 2d shows the response of the 3×3 microarray on passing both the DNA strands in Fig. 2b. Note that, the site storing the probe TAC outputs a 1 as TAC is a substring of the strand $AG\underline{\mathbf{TAC}}GA$, while the site storing probe TAG outputs 0 as TAG is not a substring of any of the two strands in Fig. 2b. This binary response of the microarray is used extensively for many biological and biochemical activities like disease-diagnosis and drug-discovery.

Manufacturing of a microarray is similar to the *Photo-lithography* process used for manufacturing integrated circuits and involves a sequence of *Deposition* steps. In each deposition step, every site in the microarray is deposited with at most one nucleotide. For each site the nucleotides should be deposited in the same order as they are listed in the probe. For example, the nucleotide T, should be deposited first followed by A and then G in the site housing the probe TAG in Fig. 2c. Figure 3 shows the deposition steps involved in manufacturing of microarray shown in Fig. 2c with the deposition sequence ($TAGCATCAGA$). It means during the first deposition step, the nucleotide T is deposited. Since the probes TAC, TAG and

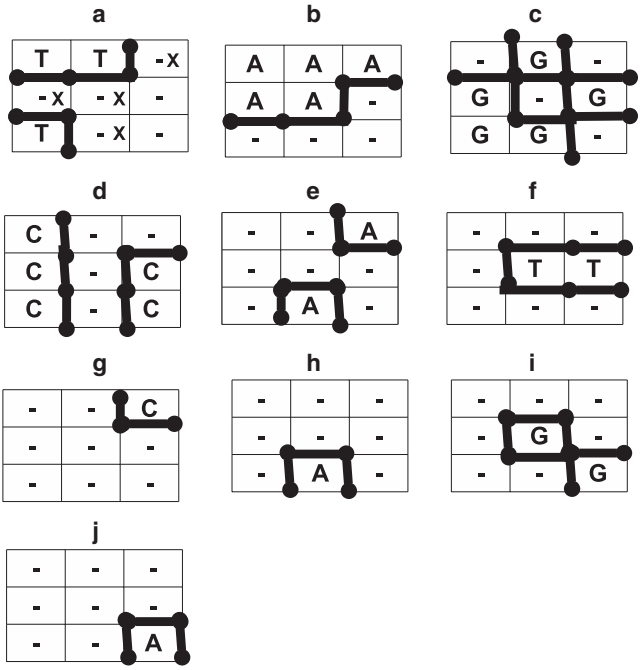

Figure 3: Sequence of masks for manufacturing microarray in Fig. 2

TGC are the only probes that require the deposition of nucleotide T as the first one, corresponding sites are marked with T in Fig. 3a. Deposition does not take place in those sites that are marked with a '$-$'. These sites are masked while depositing T. Out of 6 masked cells 4 of them marked with **x** in Fig. 3a are adjacent to unmasked cells sharing a border. The border is marked with block lines in Fig. 3. The border length for this deposition step is 5. Similarly, the border length for the second deposition (Fig. 3b, where A is deposited) is 4. The *total border length* for the entire deposition sequence (Fig. 3a–j) is 48. As mentioned earlier, the masked cells that share a border with unmasked cells on which a deposition happens, have a risk of getting exposed during the deposition due to optical effects like diffraction (Hannenhalli et al., 2002; Kahng et al., 2003). It is straightforward to note that *minimizing the total border length across all the depositions shall minimize the above mentioned risk.*

This problem is called the *Border Minimization Problem (BMP)* and is well studied in literature (Hannenhalli et al., 2002; Kahng et al., 2002, 2003).

An estimate of lower bound on border length (a value of border length, which is lower than or equal to whatever could be achieved by any deposition

sequence) for a given arrangement of probes on a microarray is crucial to evaluate the effectiveness of any algorithm that solves the BMP. A numerical technique for evaluation of such a lower bound is reported in literature (Kahng et al., 2003, 2002). This technique (described in Section 2) involves extensive computation on the input set of probes. Therefore, it is suitable only for small microarray sizes as it takes prohibitively large CPU time for current commercial microarray sizes.

In practice, the DNA probes are random sequences of nucleotides. Based on this realistic assumption, this paper attempts to estimate the lower bound for the border length analytically using a probability theoretic approach by reducing the same to the problems of computing the probability distribution functions (PDF) for the Hamming Distance and the length of the longest common subsequence (LCS) between two random strings. To the best of our knowledge, no PDF is reported earlier for the length of the LCS between two random strings.

In the following section, we present the numerical technique for estimation of lower bound on border length as reported in Kahng et al. (2002, 2003) and discuss its limitations. In subsequent sections, we present the proposed probability theoretic technique and compare it with the existing numerical technique Kahng et al. (2003, 2002). We show that the lower bound computed using the proposed technique matches with the one computed with the earlier technique for small microarray sizes, whereas, for larger microarrays, the lower bound could be computed only with the proposed technique owing to prohibitively large CPU time required by the existing technique.

2 Numerical Method for Lower Bound on Border Length

In this section, we present the best reported method for estimation of lower bound (Kahng et al., 2002, 2003) and discuss why it is not scalable. This technique can be more comprehensively described with notions of *Conflict-Distance* and *Lower Bound Graph*. Figure 3 shows the sequence of masks for manufacturing the microarray in Fig. 2 with the deposition sequence as $TAGCATCAGA$. A Conflict is said to exist between a pair of probes in a deposition step if and only if exactly one probe among the pair is exposed for deposition while the other is masked. Thus, the *conflict-distance* is the border length contributed by this pair of probes, if they are placed adjacent to each other in the microarray. The *conflict-distance* between a pair of probes for a given deposition sequence equals the *total* number of conflicts between them. For example, conflict-distance between p_1 (TAG) and p_2 (AGC) for the above deposition sequence is 2. As in the first step p_2 is masked while p_1 is deposited with T. In the second and third steps, both

	T	A	G	C	A	T	C	A	G	A
TAC	1	1	0	1	0	0	0	0	0	0
AGC	0	1	1	1	0	0	0	0	0	0
Conflict	*		*							

'*' indicates existence of *Conflict*

Figure 4: Definition of distance between a pair of probes

p_1 and p_2 are both deposited with A and G respectively. In the fourth step p_1 is masked while p_2 is deposited with C (Fig. 4 illustrates the concept of conflict).

Two variations are reported in the literature for synthesizing the microarrays, namely, *Synchronous Array Design Problem (SADP)* and *Asynchronous Array Design Problem (AADP)*. In SADP, the choice of deposition sequence (S) and embedding of each probe into S are as per the Alphabet algorithm (Ning and Leong, 2006). According to this, each probe (or site) is deposited exactly once during each period ($ACGT$) of the deposition sequence $(ACGT)^L$, wherein, L is the length of the probes to be placed on the microarray. The Hamming distance between two probes p_1 and p_2 is the number of positions at which their respective nucleotides differ. For example, the Hamming distance between probes TAG and AGC is 3. It is straightforward to see that in the case of SADP, the contribution of a pair of probes to the border length, if they are placed adjacent in the microarray is twice the Hamming distance between them. This is implied by the fact that every conflicting nucleotide position in the pair necessitates that while one of them is deposited, the other needs to be masked. The same is illustrated in Fig. 5 for the probes ACT and GCA. It is seen from Fig. 5 that the contribution by this pair to the border length is 4 for SADP which is twice of Hamming distance between them.

In the case of AADP, the deposition sequence can be a random one and do not follow any pattern as in the case of SADP. Thus, given a pair of probes of length L each, let the longest common subsequence between them be of length k. It is straightforward to infer that if these probes are placed adjacent to each other in a microarray, then, they contribute *at least* $2(L-k)$ to the border length irrespective of the deposition sequence. The value $(L-k)$ is called *edit-distance* between the given probes. This is illustrated in Fig. 6, wherein, the minimum conflict-distance between two probes $ACGGT$ and $CGATA$ is shown. The $|LCS|$ between the probes (k)

Dep. Seq.	A	C	G	T	A	C	G	T	A	C	G	T
ACT	A	-	-	-	-	C	-	-	-	-	-	T
GCA	-	-	G	-	-	C	-	-	A	-	-	-
Conflict	*	-	*	-	-	-	-	-	*	-	-	*

Deposition Sequence = $(ACGT)^3$

Hamming Distance between ACT and GCA = 2

Distance = 2× Hamming Distance = 4

(4 instances indicated by '*')

Figure 5: Distance between a pair of probes in SADP problem (every probe is deposited only once during each period of $ACGT$)

is 3 while each probe is five nucleotides long ($L = 5$). Thus, the edit distance $|L - k|$ is 2 and therefore, the minimum possible conflict distance between the probes $2|L - k| = 4$. To sum up, the *minimum possible* conflict-distance between a pair of probes each of length L is twice the Hamming distance between them in the case of SADP and twice the edit-distance between them in the case of AADP.

For the sake of completeness, the numerical method for computing the lower bound on border length, reported in Kahng et al. (2002, 2003) is summarized as follows: Given a set of N probes $(p_1, p_2, ..., p_N)$, construct a complete weighted directed graph $G = (V, E)$, such that $V = \{v_1, v_2, ..., v_N\}$, and p_i corresponds to v_i, $1 \leq i \leq N$. The weight of the edge $< v_i, v_j >$ is half of the conflict distance between p_i and p_j. Thus, between every pair of vertices (p_i, p_j), there are two equal-weighted edges, one from p_i to p_j and other from p_j to p_i. For every vertex v_i, remove all the heaviest $N - 5$ vertices. This results in the graph G' having $4N$ edges. From G', remove the heaviest $4\sqrt{N}$ edges resulting in G'' having $4N - 4\sqrt{N}$ edges. Given any

Probe-1	A	**C**	**G**	G	**T**
Probe-2	**C**	**G**	A	**T**	A

$|LCS|$ between probe-1 and probe-2 l = 3

Length of each probe, L = 5

Edit Distance = $L - l$ = 2

Minimum Possible Distance between the probes = $2 \times (L - l) = 4$

Figure 6: Lower bound on distance between a pair of probes

placement of the probes $(p_1, p_2, ..., p_n)$ on the microarray, construct a directed graph $K = (V', E')$ such that $V' = \{v'_1, v'_2, ..., v'_N\}$ and p_i corresponds to v'_i, $1 \leq i \leq N$. If two probes p_i and p_j are placed on adjacent sites on the microarray, then there exists two weighted directed edges (from p_i to p_j and from p_j to p_i) with weight equal to half of the conflict distance between p_i and p_j. There shall be $4N - 4\sqrt{N}$ edges on K. The total weight of edges in K is a lower bound on the border length incurred during the microarray deposition process. It is straightforward to see that the total edge weight of all such graphs K considered over all possible placements of the probes and over all possible deposition sequences shall be greater than or equal to that of G''.

Given a $\sqrt{N} \times \sqrt{N}$ microarray hosting N probes, each of length L, the total time to construct the graph G is $O({}^{N}C_2 \times$ time to compute the conflict-distance between probes of length L). For the SADP recall that the conflict-distance is the Hamming distance while it is edit distance in the case of AADP. This takes prohibitively large time for microarrays greater than 200×200 thereby necessitating simpler techniques for lower bound estimation. It is realistic to assume that every probe is a random sequence of nucleotides A, G, C and T appearing with a probability of p_A, p_G, p_C and p_T respectively (Louie et al., 2003; Yamagishi and Shimabukuro, 2008). Then the problem of computing the lower bound for border length is reduced to the problem of finding the expected Hamming distance and $|LCS|$ between two such random strings of nucleotides for the SADP and AADP respectively. Once we know the probability distribution function (PDF) for conflict-distance, the lower bound can be computed as follows:

1) In an $\sqrt{N} \times \sqrt{N}$ microarray, there are $2\sqrt{N}(\sqrt{N} - 1)$ adjacent sites, while we have ${}^{N}C_2$ possible pairs.

2) Let $PD^{N,L}_{conflict}(k)$ be the probability distribution function denoting the number of pairs with conflict distance k among N random probes, each of length L.

3) Find an t such that $\sum_{i=1}^{t} PD^{N,L}_{conflict}(i) \geq 4N - 4\sqrt{N}$ and $\sum_{i=1}^{t-1} PD^{N,L}_{conflict}(i) < 4N - 4\sqrt{N}$. The lower bound is:

$$LB^{N,L} = 2 \sum_{i=1}^{t} i \times PD^{N,L}_{conflict}(i)$$

In other words, $LB^{N,L}$ represents the weight of the conflict graph built with $4N - 4\sqrt{N}$ edges in *increasing* (considering the best possible conflict distances, thereby implying the lower bound) order of their weights as specified by the probability distribution function $PD^{N,L}_{conflict}(k)$. It is straightforward to see that no other conflict-graph with lesser total edge weight can be constructed using the weights as specified by $PD^{N,L}_{conflict}(i)$, $1 \leq i \leq t$, over all possible placements and all possible deposition sequences.

The next section describes how to compute the PDF for the Hamming distance and length of LCS between two random strings.

3 Probability Distribution Function for Hamming Distance and Length of Longest Common Subsequence Between Two Random Strings

In this section, we present the proposed probability theoretic technique for estimation of lower bound on border length. It is recalled that overall border length equals the sum total of conflict-distances between all pairs of adjacent sites. For SADP problem (Fig. 5), distance between a pair of adjacent sites is twice the Hamming Distance (Number of nucleotide positions where the pair of probes differ with each other). For AADP problem (Fig. 6), the distance cannot be lower than $2(L - l)$, where L is the length of each probe and l is the length of LCS between the pair of probes (irrespective of deposition sequence and embedding). Therefore, for arriving at the lower bound, we need the probability distributions of Hamming Distance and length of LCS. This section presents these probability distributions followed by the proposed technique for estimation of the lower bound for border length.

3.1. Probability Distribution of Hamming Distance.

3.1.1. Probability of Occurrence of Each Nucleotide. There are four nucleotides A, C, G and T. In the ongoing analysis, the frequencies (probabilities) of occurence of these nucleotides are denoted by p_A, p_C, p_G and p_T respectively. These frequencies are constrained by:

$$p_A + p_C + p_G + p_T = 1$$

3.1.2. Hamming Distance Between an Arbitrary Pair of Probes. Let us consider two probes of length L. We have to find the probability that they *differ* (have different nucleotides) in exactly r nucleotide positions and *agree*

(have the same nucleotides) in the remaining $L-r$ positions. Before that let us find the probability for an arbitrary pair of nucleotides being identical:

$$p_{identical} = p_A.p_A + p_C.p_C + p_G.p_G + p_T.p_T$$

Then, the probability of conflict (p), i.e., the probability that an arbitrary pair of nucleotides differ from each other is:

$$p = 1 - p_{identical} \tag{3.1}$$

Next, we have to find the probability that the pair of probes differ in the given r locations. Since the probability for a pair of nucleotides to be distinct is p, the probability for r pairs to have distinct nucleotides is:

$$p^r$$

This equals the probability for an arbitrary pair of probes to differ in the given r nucleotide positions. For the probes to differ *only* in the chosen r positions, the nucleotides in remaining $L - r$ positions should be same in both the probes. The probability for this is:

$$(1-p)^{(L-r)}$$

Therefore, the probability that the probes differ in the chosen r locations and agree in the remaining $L - r$ locations is:

$$p^r(1-p)^{(L-r)}$$

Since there are ${}^LC_r$ ways of choosing these r locations, the probability that the Hamming distance between the pair of probes being exactly r is:

$${}^LC_r p^r(1-p)^{(L-r)} \tag{3.2}$$

It can be easily observed that (3.2) is same as that of a *Binomial Distribution.*

3.2. Probability Distribution of LCS Length. The *Longest Common Subsequence* problem has several applications in the areas of image and video processing (Bezerra, 2004; Hu et al., 1999), Biomedical Engineering (Parvinnia et al. 2008, Rizvi and Agarwal, 2007), Electronic Design Automation (Ranganathan and Motamarri, 1997, Tang and Wong, 2002), data comparison (Ramanathan et al., 2006), Network Intrusion Detection (Zhong et al., 2004) etc. An algorithm for the LCS problem suitable for hardware implementation is provided in Hu et al. (2008). From the Bioinformatics point of view, a more complicated problem called *Exemplar LCS* is formulated in

Bonizzoni et al. (2007). Several parallel algorithms for computation of LCS are presented in Korkin et al. (2008), Lin et al. (1991), Liu (2006), Lu and Lin (1994), & Xiao et al. (2008). A fast algorithm for run length encoded strings is provided in Apostolico et al. (1997). Several new algorithms for the LCS problem were developed by researchers for computation of LCS (Bergroth et al., 1998; Rizvi and Agarwal, 2007; Shuai, 1992; Xiang et al., 2007; Yuxiang et al., 2007). Probability theoretic studies on length of LCS is already reported in literature (Chvátal and Sankoff, 1975; Steele, 1982). The focus of the study presented in Chvátal and Sankoff (1975) & Steele (1982) was to obtain mean length of LCS or its variance. In this paper, we arrive at the *Probability Distribution Function* for length of LCS.

That is, given two strings (probes comprising the nucleotides A, C, G and T) of lengths n_1 and n_2 and $n_1 \geq n_2$ and an integer i, what is the probability that an arbitrary pair of strings of given lengths have an LCS of length i.

3.2.1. Probability of Conflict between alphabets. Consider an alphabet set Σ of cardinality M. In our case, $\Sigma = \{A, C, G, T\}$ and hence $M = |\Sigma| = 4$. Let us denote that probability of occurence of the i^{th} alphabet by $P_{oc}(i)$. The probability that a pair of alphabets chosen at random being identical is:

$$p_{identical} = \sum_{i=1}^{M} P_{oc}(i)^2, \ \forall 1 \leq i \leq M$$

If each of these M alphabets are equiprobable, then

$$p_{identical} = M \times \frac{1}{M}\frac{1}{M} = \frac{1}{M}$$

The probability that a pair contains distinct alphabets is just the complement of $p_{identical}$:

$$1 - p_{identical} = p_{conflict} \tag{3.3}$$

The basic building blocks of the algorithm for computing LCS (Cormen et al., 1990) are used in this paper for deriving the PDF of $|LCS|$.

3.2.2. Algorithm for LCS (Cormen et al., 1990). This algorithm is based on the dynamic programming approach. Given two strings A and B, let L_{ij} represent the LCS between the substrings containing first i characters of A and first j characters of B. The recurrence relation for length of LCS is therefore,

$$\begin{aligned} L_{i,j} &= L_{i-1,j-1} + 1 \text{ if A[i]==B[j]} \\ L_{i,j} &= max(L_{i,j-1}, L_{i-1,j}) \text{ , otherwise} \end{aligned} \tag{3.4}$$

The (3.4) is recursively used to find the LCS between *given pair of* strings. Our objective here is to find the probability distribution of LCS. In this paper, the PDF for LCS is computed using the same recurrence relation as stated in (3.4). To solve this recurrence, we need the *Basis* data analogous to *Boundary Conditions* which are computed in the next section.

3.2.3. LCS Between Single Character String and Another Arbitrary Length String. We are given two strings, one consisting of a single character and other of an arbitrary length l. Our alphabet consists of M elements and the single character string contains *exactly* one of them. Since, one of the strings has only one character, $|LCS| \leq 1$. Let $P_{i,j}(k)$ denote the probability that the LCS between the substrings containing first i characters of A and first j characters of B is k.

The LCS would be zero if and only if the string of length l does not have that character forming the single character string. That is, each of the l characters in the other string should be different from that character. The probability of this happening is:

$$P_{l,1}(0) = p_{conflict}^{l} \tag{3.5}$$

Since the LCS between two such (one of length L and other 1) strings could be (and have to be) either 0 or 1, probability of LCS=1 is just the probability of LCS $\neq 0$. Therefore,

$$P_{l,1}(1) = 1 - P_{l,1}(0) = 1 - p_{conflict}^{l} \tag{3.6}$$

3.2.4. Probability Distribution Recurrence. Based on the detailed derivation presented in Appendix A, the probability distribution function for length of LCS is given by (3.7).

$$\begin{aligned} P_{i,j}(r) &= p_{conflict}[P_{i-1,j}(r)\sum_{k=0}^{r} P_{i,j-1}(r) + P_{i,j-1}(r)\sum_{k=0}^{r} P_{i-1,j}(r) \\ &\quad - P_{i,j-1}(r)P_{i-1,j}(r)] \\ &\quad + p_{identical}\left[P_{i,j}(r-1)\right] \ \forall\, r > 0 \\ P_{i,j}(0) &= p_{conflict} \times P_{i,j-1}(0)P_{i-1,j}(0) \end{aligned} \tag{3.7}$$

Therefore, from this probability distribution represented by (3.7), we can compute the lower bound on border length following the procedure presented in Section 2.

4. Results And Discussions

The experimental results reported in Kahng et al. (2003,2002) assumed that the nucleotides, namely, {A,C,G,T} appear on the input probes with

Table 2: Comparison of lower bound on border length between the proposed technique and those reported earlier (Kahng et al. 2006, 2003, 2002)

Size	SADP		AADP	
	Earlier	Proposed	Earlier	Proposed
100×100	410019	422293	220497	219625
200×200	1512014	1550685	798708	790515
300×300	3233861	3378844	*	1766232
500×500	8459958	8714346	*	4775745
1000×1000		31681531	*	16551913
10000×10000		2.3×10^9	*	1.2×10^9

While the values are nearly same, the proposed technique consumes a negligibly small CPU time compared to prohibitively large CPU time consumed by earlier technique. Hence, the proposed method scales up for current microarray sizes and next generation ones while earlier method is suitable only for very small chip sizes

(The symbol '*' in this table indicates that these values are not available because of prohibitively large CPU time (Kahng et al. 2002, 2003))

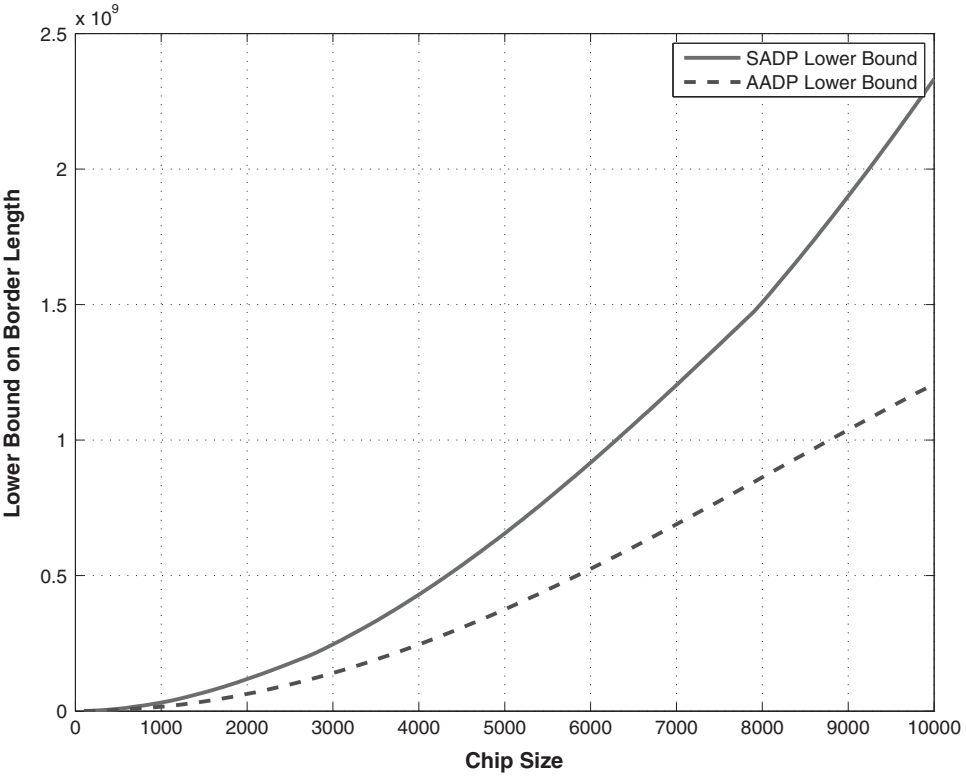

Figure 7: Theoretical Lower Bound for SADP and AADP problems

equal probability. Similar assumption is made in this paper to facilitate a common ground for comparison. Table 2 shows the results obtained using the technique reported in Kahng et al. (2003, 2002) and that reported in this paper. It is clearly seen that the proposed technique computes a lower bound which is within 4 % and 1 % (for SADP and AADP respectively) of that computed using the technique reported in Kahng et al. (2003, 2002). Results provided by Kahng et al. (2003, 2002) show that for chips larger than 300×300, it is *computationally impractical* (Kahng et al., 2003, 2002) to compute the lower-bound on border length. Therefore, corresponding entries were missing in the tables provided in Kahng et al. (2002, 2003).

But, using the proposed technique, the lower bounds are computable *within fraction of a second* with negligibly small amount of memory even for large chip sizes like 10000×10000 and beyond.

The lower bound values computed using the proposed technique, as a function of chip size, is shown in Fig. 7 for both SADP and AADP problems.

5. Conclusion

The importance of evaluating the lower bound on border length for synthesis of DNA probes onto the microarray is already well known. Researchers have previously developed technique for estimating the lower bound on border length based on numerical simulation with randomly generated set of probes. But such numerical simulation based methods were suitable only for microarrays much smaller in size than the current day microarrays. In this paper, we present a probability theoretic technique which facilitates computation of lower bound on border length for large microarrays of current day's size and even for the next generation ones. We have presented the lower bound values for microarrays containing hundred million probes, while previously, the lower bound on border length was available only for microarrays containing fewer than 90000 probes. Further, this computation consumes a very small amount of CPU time and requires a low amount of memory. We have also shown that for smaller microarrays where it is feasible to compute the lower bound through numerical simulation, the theoretical values (computed through the proposed technique) match with the earlier values (based on numerical simulation). To arrive at the lower bound on border length, it was required to know the probability distribution for length of *Longest Common Subsequence* (LCS). In this paper, we have also derived the probability distribution for $|LCS|$, which to the best of our knowledge is the first reported in literature.

Appendix A
Probability Distribution for Length of LCS - Derivation

We derive a recurrence equation for $P_{i,j}(r)$ similar to what is done for $L_{i,j}$ as in (3.4). The Eqn.(3.4) has two components, one dealing with the case that the current characters considered in both the strings are identical and another with the case when they are different. Similar steps are taken here for PDF computation as described below.

Effect of Appending Identical Characters to a Pair of Strings. If an identical character is appended to two strings, the LCS increases by one in accordance with (3.4). That is, if two strings of lengths $i-1$ and $j-1$ have an LCS of length $r-1$ and an identical character is added to both of them, the new pair of strings would have an LCS of length r.

$$P_{i,j}(r/(LCS(i-1,j-1)=r-1)) = p_{identical} \tag{A.1}$$

Equaiton (A.1) says that if two strings of lengths $i-1$ and $j-1$ have an LCS of length $r-1$ and if new pair of strings are obtained by adding an *arbitrary* character to each of them, the probability that new pair of strings have an LCS of length r between them is $p_{identical}$.

Effect of Appending Distinct Characters to a Pair of Strings. In this case, the resulting string would have an $|LCS|$ equal to the larger one among LCS(i-1,j) and LCS(i,j-1) in accordance with (3.4). For the resulting pair of strings of lengths i and j to have an LCS of length r, either of the following conditions need to be satisfied:

- $\mathbf{C}_1$: LCS(i-1,j) $= r$ AND LCS(i,j-1) $\leq r$. If the latter condition is not satisfied, the LCS(i,j-1) would have an LCS larger than r, therefore LCS(i,j) which is obtained by adding characters to both the strings cannot be equal to r.
- $\mathbf{C}_2$: LCS(i,j-1) $= r$ AND LCS(i-1,j) $\leq r$.

Condition $\mathbf{C}_1$ alone implies the following:

$$P_{i,j}(r/(LCS(i-1,j)=r)) = P(LCS(i,j-1)\leq r) = \sum_{k=0}^{k=r} P_{i,j-1}(k) \tag{A.2}$$

Equation (A.2) gives the probability that the resulting pair of strings of lengths i and j have an LCS of length *exactly* r given that LCS(i-1,j) is r. This equation means that once we know that LCS(i-1,j) is of length r, the LCS(i,j-1) should be utmost of length r. That is, probability of the $|LCS(i,j)|$ (the LCS between resulting pair of strings) being r, given that

LCS(i-1,j) is r, is just the probability that LCS(i,j-1) is utmost r as represented by (A.2).

Condition $\mathbf{C}_2$ above implies the following:

$$P_{i,j}(r/(LCS(i,j-1)=r)) = P(LCS(i-1,j) \leq r) = \sum_{k=0}^{k=r} P_{i-1,j}(k) \quad \text{(A.3)}$$

Equation (A.3) signifies, given that LCS(i,j-1) equals r, the probability that LCS(i,j) is of length r is same as the probability that LCS(i-1,j) is no more than r characters long.

The probability of $|LCS(i,j)|$ being r, given that distinct characters are appended to both the strings can be obtained by consolidating $\mathbf{C}_1$ and $\mathbf{C}_2$ as follows:

$$p_{i,j}(r) = p(\mathbf{C}_1) + p(\mathbf{C}_2) - p(\mathbf{C}_1 \cap \mathbf{C}_2) \quad \text{(A.4)}$$

Conditions for LCS(i,j) to be of Given Length r. The only ways in which LCS(i,j) can be exactly r characters long are:

- LCS(i-1,j-1) is exactly of length $r-1$ *and*, identical characters are appended to both of them. This is represented by (A.1)
- Between LCS(i,j-1) and LCS(i-1,j), at least one of them is exactly r characters long and the other is no more than r characters long and distinct characters are appended to the strings. This is represented by (A.4)

Therefore,

$$p_{i,j}(r) = p_{identical} \times \text{Eqn. } 9 + p_{conflict} \times \text{Eqn. } 12 \quad \text{(A.5)}$$

Recurrence for $P_{i,j}(r)$.

$$\begin{aligned} P_{i,j}(r) &= p_{conflict}[P_{i-1,j}(r)\sum_{k=0}^{r} P_{i,j-1}(r) + P_{i,j-1}(r)\sum_{k=0}^{r} P_{i-1,j}(r) \\ &\quad - P_{i,j-1}(r)P_{i-1,j}(r)] \\ &\quad + p_{identical}\,[P_{i,j}(r-1)] \end{aligned} \quad \text{(A.6)}$$

Equation (A.6) is the recurrence for probability distribution of LCS between an *arbitrary* pair of strings of lengths i and j.

Limiting Case. The random variable r denoting the length of the LCS ranges from 0 to maximum length of the strings. If r is zero, (A.6) requires the value of $P_{i,j}(0)$ which can be computed as follows:

$$P_{i,j}(0) = p_{conflict} \times P_{i,j-1}(0)P_{i-1,j}(0) \quad \text{(A.7)}$$

That is, the probability of LCS(i,j) being zero is the joint probability of LCS(i,j-1) and LCS(i-1,j) being zero and the probability of appending distinct characters to the strings of lengths $i-1$ and $j-1$.

References

APOSTOLICO, A., LANDAU, G. and SKIENA, S. (1997) Matching for run-length encoded strings. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 348–356.

BERGROTH, L., HAKONEN, H. and ARAITA, T. (1998) New approximation algorithms for longest common subsequences. In *String Processing and Information Retrieval: A South American Symposium, 1998. Proceedings*, pages 32–40.

BEZERRA, F (2004) A longest common subsequence approach to detect cut and wipe video transitions. In *Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on*, pages 154–160.

BONIZZONI, P., VEDOVA, G., DONDI, R., FERTIN, G., RIZZI, R. and VIALETTE, S. (2007) Exemplar longest common subsequence. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 4(4):535–543.

CHVÁTAL, V. and SANKOFF, D. (1975) Longest common subsequences of two random sequences. *J. Appl. Probab*, 12:306–315.

CORMEN, T., LEISERSON, C. and RIVEST, R. (1990) *Introduction to Algorithms*. The MIT Press, Cambridge, MA.

HANNENHALLI, S., HUBBELL, E., LIPSHUTZ, R. and PEVZNER, P. A. (2002) Combinatorial algorithms for design of dna arrays. In *Advances in Biochemical Engineering/Biotechnology*. Springer-Verlag.

HU, S.-H., WANG, C.-W. and CHEN, H.-L. (2008) An efficient and hardware-implementable systolic algorithm for the longest common subsequence problem. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 6, pages 3150–3155.

HU, W.-C., SCHMALZ, M. and RITTER, G. (1999) Image retrieval using the longest approximate common subsequences. In *Multimedia Computing and Systems, 1999. IEEE International Conference on*, volume 2, pages 730–734.

KAHNG, A., MANDOIU, I., REDA, S., XU, X. and ZELIKOVSKY, A. (2006) Computer-aided optimization of dna array design and manufacturing. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(2):305–320.

KAHNG, A. B., MANDOIU, I., REDA, S., XU, X. and ZELIKOVSKY, A. Z. (2003) Evaluation of placement techniques for dna probe array layout. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 262, Washington, DC, USA. IEEE Computer Society.

KAHNG, A. B., MANDOIU, I. I., PEVZNER, P. A., REDA, S. and ZELIKOVSKY, A. (2002) Border length minimization in dna array design. In *WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, pages 435–448, London, UK. Springer-Verlag.

KORKIN, D., WANG, Q. and SHANG, Y. (2008) An efficient parallel algorithm for the multiple longest common subsequence (mlcs) problem. In *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pages 354–363.

LIN, H., LU, M. and FANG, J. (1991) An optimal algorithm for the longest common subsequence problem. In *Parallel and Distributed Processing, 1991. Proceedings of the Third IEEE Symposium on*, pages 630–639.

LIU, W. (2006) A fast parallel longest common subsequence algorithm based on pruning rules. In *Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on*, volume 1, pages 27–34.

LOUIE, E., OTT, J. and MAJEWSKI, J. (2003) Nucleotide frequency variation across human genes. *Genome research*, 13(12):2594–2601.

LU, M. and LIN, H. (1994) Parallel algorithms for the longest common subsequence problem. *Parallel and Distributed Systems, IEEE Transactions on*, 5(8):835–848.

NING, K. and LEONG, H. W. (2006) Towards a better solution to the shortest common supersequence problem: the deposition and reduction algorithm. *BMC Bioinformatics*.

PARVINNIA, E., TAHERI, M. and ZIARATI, K. (2008) An improved longest common subsequence algorithm for reducing memory complexity in global alignment of dna sequences. In *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, volume 1, pages 57–61.

RAMANATHAN, M., GRAMA, A. and JAGANNATHAN, S. (2006) Sieve: A tool for automatically detecting variations across program versions. In *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on*, pages 241–252.

RANGANATHAN, N. and MOTAMARRI, R. (1997) A vlsi architecture for computing the optimal correspondence of string subsequences. In *Computer Architecture for Machine Perception, 1997. CAMP '97. Proceedings Fourth IEEE International Workshop on*, pages 290–294.

RIZVI, S. and AGARWAL, P. (2007) A time efficient algorithm for finding longest common subsequence from two molecular sequences. In *Bioengineering Conference, 2007. NEBC '07. IEEE 33rd Annual Northeast*, pages 302–306.

SHUAI, D.-X. (1992) An approach to solving the longest common subsequences based on string-coding functional and neural network. In *Systems Engineering, 1992., IEEE International Conference on*, pages 564–567.

STEELE, J. M. (1982) Long common subsequences and the proximity of two random strings. *SIAM J. Appl. Math.*, 14(4):731–737.

TANG, X. and WONG, D. (2002) Floorplanning with alignment and performance constraints. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 848–853.

WATSON, J. D. and CRICK, F. (1953) Molecular structure of nucleic acids: A structure for deoxyribo nucleic acid. *nature*, 171:737–738.

XIANG, X., ZHANG, D. and QIN, J. (2007) A new algorithm for the longest common subsequence problem. In *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*, pages 112–115.

XIAO, L., SONG, H., ZHU, M. and KUANG, Y. (2008) A pc cluster based parallel algorithm for longest common subsequence problems. In *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 820–823.

YAMAGISHI, M. and SHIMABUKURO, A. (2008) Nucleotide frequencies in human genome and fibonacci numbers. *Bulletin of mathematical biology*, 70(3):643–653.

YUXIANG, X., ZHANG, D. and QIN, J. (2007) An improve algorithm for the longest common subsequence problem. In *Convergence Information Technology, 2007. International Conference on*, pages 637–639.

ZHONG, C., CHEN, G.-L. and HE, J.-H. (2004) Parallel computing for the longest common subsequences in network intrusion detection system. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 4, pages 2553–2557.

S Srinivasan
Defence Research and Development Organization (DRDO), Chennai, India
E-mail: srinivasan.research@gmail.com

V Kamakoti
Department of Computer Science and Engineering, Indian Institute of Technology (IIT), Chennai, India
E-mail: veezhi@gmail.com; kama@cse.iitm.ac.in

A Bhattacharya
Defence Research and Development Organization (DRDO), Hyderabad, India
E-mail: abhijit689@yahoo.com

Paper received: 8 September 2014.