Border Length Minimization Problem on a Square Array

VAMSI KUNDETI, SANGUTHEVAR RAJASEKARAN, and HIEU DINH

ABSTRACT

Protein/peptide microarrays are rapidly gaining momentum in the diagnosis of cancer. High-density and high-throughput peptide arrays are being extensively used to detect tumor biomarkers, examine kinase activity, identify antibodies having low serum titers, and locate antibody signatures. Improving the yield of microarray fabrication involves solving a hard combinatorial optimization problem called the *border length minimization problem* (BLMP). An important question that remained open for the past 7 years is if the BLMP is tractable or not. We settle this open problem by proving that the BLMP is $\mathcal{NP}$ -hard. We also present a hierarchical refinement algorithm that can refine any heuristic solution for the BLMP and prove that the TSP + 1-threading heuristic is an $O(N)$ -approximation.

Key words: algorithms, combinatorial optimization, DNA self-assembly, genomics, machine learning.

1. INTRODUCTION

CANCER DIAGNOSIS RESEARCH HAS TAKEN A NEW DIRECTION recently by adopting peptide microarrays for reliable detection of tumor biomarkers (Chatterjee et al., 2006; Melle et al., 2004; Welsh et al., 2003). These high-throughput arrays also find application in examining kinase activity, identifying antibody signatures against tumor antigens, etc. High-density peptide arrays are currently fabricated using technologies such as photolithography or *in situ* synthesis based on micromirror arrays. The manufacturers of these arrays are facing serious fabrication challenges because unintended illumination effects such as diffraction and scattering of light. These illumination effects can be reduced dramatically by selecting the right placement of the peptide probes before fabrication. Finding this placement can be formulated as a combinatorial optimization problem, known as the *border length minimization problem* (BLMP). Hannenhalli et al. (2002) first introduced BLMP in 2002. Although the BLMP was formulated in the context of DNA microarrays, peptide arrays share a similar fabrication technology.

The BLMP can be stated as follows. Given N^2 strings of the same length, how do we place them in a grid of size $N \times N$ such that the Hamming distance summed over all the pairs of neighbors in the grid is minimized? The BLMP has received a lot of attention from many researchers. The earliest algorithm suggested by Hannenhalli et al. (2002) reduces BLMP to TSP (traveling salesman problem) by computing a tour of the strings and then threading the tour on the grid. Kahng et al. (2003) have proposed several other heuristic algorithms that are considered the best performing algorithms in practice. De Carvalho and

Rahmann (2006) introduced a quadratic program formulation of the BLMP but unfortunately the quadratic program is an intractable problem. Later, Kundeti and Rajasekaran (2009) formulated the problem as an integer linear program, which performs better than the quadratic program in practice.

Certain generalizations of the BLMP are known to be $\mathcal{NP}$ -hard. Kundeti and Rajasekaran (2010) proved that the BLMP problem on a rectilinear $(M \times N)$ grid is $\mathcal{NP}$ -hard. Recall that if a generalization of a problem is $\mathcal{NP}$ -hard, then the special cases of the problem could still be polynomial. This is true for the k -SAT problem. Generalization of the k -SAT problem (Boolean satisfiability with k literals per clause) when $k \geq 3$ is $\mathcal{NP}$ -hard. However, for $k=2$ the problem can be solved in polynomial time. Since a square $(N \times N)$ grid is a special case of a rectangular grid the complexity of BLMP on a square remained an unsolved puzzle.

Despite many studies on the BLMP, the question of whether BLMP is tractable or not remained open for the past seven years. In this article, we show that the BLMP is $\mathcal{NP}$ -hard. We also consider a generalization of the BLMP called the *Hamming graph placement minimization problem* (HGMP). We show that some special cases of the HGMP are also $\mathcal{NP}$ -hard. On the algorithmic side, we show that a simple version of the algorithm suggested by Hannenhalli et al. is an $O(N)$ -approximation. On the practical side, we propose a refinement algorithm that takes any solution and tries to improve it. An experimental study of this refinement algorithm is also included.

Our article is organized as follows. Section 2 formally defines the BLMP and HGMP. Section 3 provides the $\mathcal{NP}$ -hardness proof of the BLMP and some special cases of the HGMP. Section 4 gives the $O(N)$ -approximation algorithm and the refinement algorithm for the BLMP. Section 5 provides an experimental evaluation of the refinement algorithm. Finally, Section 6 concludes our article and discusses some open problems.

2. PROBLEM DEFINITION

Let S be a set of strings of the same length with $S = \{s_1, \dots, s_n\}$, and let $G = (V, E)$ be a graph with $|V| = n$. A placement of S on G is a bijective map $f: S \rightarrow V$. Let $f^{-1}(u)$ be the string that is mapped to vertex u by the placement f . We denote the Hamming distance between two strings s_i and s_j as $\delta(s_i, s_j)$. The cost of placement f is $Cost(f) = \sum_{e=(u,v) \in E} \delta(f^{-1}(u), f^{-1}(v))$. The Hamming graph placement minimization problem (HGMP) is defined as follows. Given S and G , find a placement of S on G of minimum cost. We denote the optimal cost as $OPT(S, G)$, or simply as OPT if it is clear what S and G are.

Obviously, if G is a ring graph, then HGMP is the same as the well-known Hamming traveling salesman problem (HTSP). If G is a grid graph of size $N \times N$ (where $N^2 = n$), then HGMP becomes the border length minimization problem (BLMP), which is the main study of our article.

3. $\mathcal{NP}$ -HARDNESS OF THE BLMP AND HGMP

Theorem 1. *The BLMP is $\mathcal{NP}$ -hard.*

We will show that the Hamming traveling salesperson problem (HTSP) for strings (with the Hamming distance metric) polynomially reduces to the BLMP. The HTSP is already defined in Section 2.

The idea of the proof is that given $4N$ strings for the HTSP, we construct $(N+1)^2$ strings for the BLMP such that from an optimal solution to this BLMP, we can easily obtain an optimal solution for the HTSP. So we need to consider the variant of the HTSP in which the number of strings is divisible by 4. The proof will be presented in stages. The next three subsections present some preliminaries needed for the proof of the theorem. Followed by these subsections, the proof is presented.

3.1. $4N$ -strings traveling salesperson problem

Define an instance of the HTSP as a $4N$ -strings HTSP if the number of strings in the input is $4N$ (for some integer N). In this section, we show that the $4N$ -strings HTSP is $\mathcal{NP}$ -hard.

Theorem 2. *$4N$ -strings HTSP is $\mathcal{NP}$ -hard.*

Given an integer n , we can construct a set of n strings of length n each, $A_n = \{a_1, a_2, \dots, a_n\}$, such that $\delta(a_i, a_j) = 2$ for any $1 \leq i \neq j \leq n$. One way to construct A_n is to let $a_i = \overline{00 \dots 0100 \dots 0}$, where there are $(i-1)$ $\overline{0}$'s before $\overline{1}$. It is easy to check that $\delta(a_i, a_j) = 2$ for any $1 \leq i \neq j \leq n$.

3.4. Proof of the main theorem

Now we are ready to present the proof of Theorem 1. Let $S = \{s_1, s_2, \dots, s_{4N}\}$ be the input for any instance of the $4N$ -strings HTSP. Each s_i has the length l . We will generate $(N+1)^2$ strings such that an optimal solution for the BLMP on these $(N+1)^2$ strings will yield an optimal solution for the $4N$ -strings HTSP on S .

The input for the BLMP instance that we generate will be $T = \{t_1, t_2, \dots, t_{4N}, t, t, \dots, t\}$, where t occurs $N^2 - 2N + 1$ times. We set $t_i = REP_h(a_i) + REP_2(s_i)$, where a_i is the i -th string in the set A_{4N} defined in Section 3.3. We will choose h later. Also, we set $t = REP_{4Nh}(\overline{0}) + \overline{0101 \dots 01}$, where the string $\overline{01}$ is repeated l times. We can easily check that:

$$\delta(t_i, t) = h + l \text{ for any } 1 \leq i \leq 4N \quad (1)$$

$$\delta(t_i, t_j) = 2h + 2\delta(s_i, s_j) \leq 2h + 2l \quad (2)$$

$$\text{for any } 1 \leq i \neq j \leq 4N$$

We choose h so that T satisfies the condition in Lemma 1. Particularly, choose $h = 8l$. Now we will show that $OPT_{BLMP}(T) = 4(N-1)(h+l) + 8Nh + 2OPT_{HTSP}(S)$, which in turn means that an optimal solution for the BLMP on T will yield an optimal solution for the $4N$ -strings HTSP on S .

Let $A = s_{i_1}, s_{i_2}, \dots, s_{i_{4N}}$ be an optimal tour for the $4N$ -string HTSP on S . We construct a solution A' for the BLMP on T by placing t_{i_j} 's on the border of the grid in the order $t_{i_1}, t_{i_2}, \dots, t_{i_{4N}}$ and placing the copies of t on the center of the grid. By the equalities (1) and (2), the cost of A' is $Cost(A') = 4(N-1)(h+l) + 8Nh + 2Cost(A)$. Therefore, $OPT_{BLMP}(T) \leq 4(N-1)(h+l) + 8Nh + 2OPT_{HTSP}(S)$.

On the other hand, let B be an optimal solution for the BLMP on T . By Lemma 1, t_i 's lie on the border of the grid, and the copies of t lie on the center of the grid. Assume that t_i 's lie in the order $t_{i_1}, t_{i_2}, \dots, t_{i_{4N}}$. We can construct a tour B' for the $4N$ -strings HTSP on S in the order $s_{i_1}, s_{i_2}, \dots, s_{i_{4N}}$. By the equalities (1) and (2), $Cost(B) = 4(N-1)(h+l) + 8Nh + 2Cost(B')$. Hence, $OPT_{BLMP}(T) \geq 4(N-1)(h+l) + 8Nh + 2OPT_{HTSP}(S)$.

This completes the proof of Theorem 1. ■

3.5. $\mathcal{NP}$ -hardness of the HGPMP for other special cases

We can generalize the result in Theorem 1 for other special cases of the HGPMP. We say graph G is “bordered-ring” if G is undirected and G has a ring of size $\Omega(n^\alpha)$ for some constant $\alpha > 0$ such that every vertex in the ring has degree no greater than d and every vertex outside the ring has degree greater than d for some $d \geq 3$. For grid graphs, $\alpha = \frac{1}{2}$ and $d = 3$. Some variants of grid graphs like Manhattan grids are bordered-ring as well.

Theorem 3. *The HGPMP is $\mathcal{NP}$ -hard even if G is bordered-ring.*

Proof. By a similar reduction to that of the BLMP above, the theorem follows. ■

3.6. An alternate $\mathcal{NP}$ -hardness proof for the BLMP

In this section, we give an alternate $\mathcal{NP}$ -hardness proof for the BLMP by showing that another variant of the HTSP, called k -segments HTSP, polynomially reduces to the BLMP. We believe that the techniques introduced in both of our proofs will find independent applications.

3.6.1. k -Segments traveling salesperson problem. We define the k -segments HTSP and show that it is NP-hard. Consider an input of n strings: $s_1, s_2, \dots, s_n$. The problem of k -segments HTSP is to partition the n strings into k parts such that the sum of the optimal tour costs for the individual parts is minimum.

Theorem 4. *The k -segments HTSP for strings is $\mathcal{NP}$ -hard.*

Proof. We will prove this for $k=4$ (since this is the instance that will be useful for us to prove the main result), and the theorem will then be obvious.

We will show that the HTSP polynomially reduces to the 4-segments HTSP. Let $S = \{s_1, s_2, \dots, s_n\}$ be the input to any instance of the HTSP. We will generate an instance of the 4-segments HTSP that has as input $(n+3)$ strings. Let l be the length of each string in S . Note that the optimal cost for the HTSP with input S is $\leq nl$.

Consider the 4 strings: $\overline{1110}$, $\overline{1101}$, $\overline{1011}$, and $\overline{0111}$. The distance between any two of them is 2. Now replace each $\overline{1}$ in each of these 4 strings with a string of nl $\overline{1}$'s. Also, replace each $\overline{0}$ in each of these strings with a string of nl $\overline{0}$'s. Call these new strings t_1, t_2, t_3, t_4 . The distance between any two of these strings is $2nl$.

The input strings for the 4-segments HTSP are $q_1, q_2, \dots, q_{n+3}$ and are constructed as follows: q_i is nothing but s_i with t_1 appended to the left, for $1 \leq i \leq n$. Additionally, q_{n+1} is a string of length $4nl + l$ whose l LSBs are $\overline{0}$'s and whose $4nl$ MSBs equal t_2 ; q_{n+2} is a string of length $4nl + l$ whose l LSBs are $\overline{0}$ and whose $4nl$ MSBs equal t_3 . Also, q_{n+3} has all $\overline{0}$'s in its l LSBs, and its $4nl$ MSBs equal t_4 .

Clearly, in an optimal solution for the 4-segments HTSP instance, the four parts have to be $\{q_1, q_2, \dots, q_n\}$, $\{q_{n+1}\}$, $\{q_{n+2}\}$, and $\{q_{n+3}\}$. As a result, we can get an optimal solution for the HTSP instance given an optimal solution for the 4-segments HTSP instance. ■

3.6.2. A special instance of the BLMP. Consider the following n^2 strings as an input for the BLMP: $t_1, t_2, \dots, t_n, t, t, \dots, t$. Here there are $n^2 - n$ copies of t . Also, $\delta(t_i, t_j) = 16$ for any i and j less than or equal to n , and $\delta(t_i, t) = 9$ for any $i \leq n$.

Lemma 2 *In an optimal solution to the above BLMP instance, $t_1, t_2, \dots, t_n$ lie on the boundary of the $n \times n$ grid and moreover these strings are found in four segments of successive nodes.*

Proof. Let T be the collection of strings $t_1, t_2, \dots, t_n$. By Lemma 1, we conclude that all the strings of T lie on the boundary of the grid in an optimal solution.

Let S_1 and S_2 be two segments such that S_1 and S_2 consist of strings from T , strings in S_1 are in successive nodes, strings in S_2 are in successive nodes, and these two segments are not successive. Consider the case in which none of these strings are in a corner of the grid. Let $S_1 = \{a_1, a_2, \dots, a_{n_1}\}$ and $S_2 = \{b_1, b_2, \dots, b_{n_2}\}$. Let $C(S_1) = \sum_{i=1}^{n_1-1} \delta(a_i, a_{i+1})$ and $C(S_2) = \sum_{i=1}^{n_2-1} \delta(b_i, b_{i+1})$. The total cost for these two segments is $C(S_1) + C(S_2) + 9(n_1 + n_2) + 36$. If we join these two segments into one, the new cost will be $C(S_1) + C(S_2) + 9(n_1 + n_2) + 34$.

Thus it follows that all the strings of T will be on the boundary, and they will be found in successive nodes in any optimal solution. Also it helps to utilize the corners of the grid since each use of a corner will reduce the total cost by nine. Therefore, in an optimal solution there will be four segments such that all the segments are in the boundary of the grid, each segment has strings from T in successive nodes, and one string of each segment occupies a corner of the grid. In other words, an optimal solution for the BLMP instance contains an optimal solution for the 4-segments TSP corresponding to T . The optimal cost for this BLMP instance is $25n - 28$. ■

3.6.3. Construction of strings for the above BLMP instance. We can construct n^2 strings that have the same properties as the ones in the above BLMP instance. To begin with, we construct $(n+1)$ binary strings of length n each. The string t_i has all $\overline{1}$'s except in position i , for $1 \leq i \leq n$. The position of the LSB of any string is assumed to be $\overline{1}$. String t_{n+1} has all $\overline{1}$'s. Clearly, $\delta(t_i, t_j) = 2$ for any i and j less than or equal to n . Also, $\delta(t_i, t_{n+1}) = 1$ for any $1 \leq i \leq n$.

Now, in each t_i (for $1 \leq i \leq (n+1)$), replace every $\overline{1}$ with a string of eight $\overline{1}$'s and replace each $\overline{0}$ with a string of eight $\overline{1}$'s. After this change, $\delta(t_i, t_j) = 16$ for any $1 \leq i, j \leq n$ and $\delta(t_i, t_{n+1}) = 8$ for any $1 \leq i \leq n$.

Finally, append a $\overline{0}$ to the left of each t_i (for $1 \leq i \leq n$) as the MSB. Also, append a $\overline{1}$ to the left of t_{n+1} . In this case, $\delta(t_i, t_j) = 16$ for any $1 \leq i, j \leq n$ and $\delta(t_i, t_{n+1}) = 9$ for any $1 \leq i \leq n$.

3.6.4. The alternate proof of the main theorem. Let $S = \{s_1, s_2, \dots, s_n\}$ be the input for any instance of the HTSP. We will generate n^2 strings such that an optimal solution for the BLMP on these n^2 strings will yield an optimal solution for the 4-segments HTSP on S .

We will use as the basis the $(n+1)$ strings generated in the above section. Recall that these strings $t_1, t_2, \dots, t_{n+1}$ are of length $(8n+1)$ each. Also, $\delta(t_i, t_j) = 16$ for any $1 \leq i, j \leq n$ and $\delta(t_i, t_{n+1}) = 9$ for any $1 \leq i \leq n$.

Replace each $\bar{0}$ in each of the above strings with nl $\bar{0}$'s and replace each $\bar{1}$ in each of these strings with nl $\bar{1}$'s. Now, $\delta(t_i, t_j) = 16nl$ for any $1 \leq i, j \leq n$, and $\delta(t_i, t_{n+1}) = 9nl$ for any $1 \leq i \leq n$. Each of these strings is of length $(8n+1)nl$.

Replace each $\bar{0}$ in each s_i with two $\bar{0}$'s (for $1 \leq i \leq n$), and replace each $\bar{1}$ in each s_i with two $\bar{1}$'s and let s'_i be the resultant string. Note that an optimal solution for the 4-segments HTSP on the revised S will also be an optimal solution for the 4-segments HTSP on the old S . If l is the length of each string in the old S , then $2l$ will be the length of each revised input string.

The input for the BLMP instance that we generate will be $q_1, q_2, \dots, q_n, t, t, \dots, t$, where t occurs $n^2 - n$ times. Each of these strings will be of length $(8n+1)nl + 2l$. The string q_i will have s'_i in its $2l$ LSBs, and it will have t_i in its $(8n+1)nl$ MSBs, for $1 \leq i \leq n$. The string t will have t_{n+1} in its $(8n+1)nl$ MSBs. Its $2l$ LSBs will be $\bar{0}101\dots\bar{0}1$, that is, the string $\bar{0}1$ is repeated l times. Note that $\delta(q_i, q_j) = 16nl + \delta(s'_i, s'_j)$ for any $1 \leq i, j \leq n$. Also, $\delta(q_i, t) = 9nl + l$ for any $1 \leq i \leq n$.

Note that strings of this BLMP instance are comparable to the strings we had for Lemma 2. This is because the interstring distances are very nearly in the same ratios for the two cases. As a result, using a proof similar to that of Lemma 2, we can show that the strings $t_1, t_2, \dots, t_n$ will all lie in the boundary of the grid in an optimal solution to the above BLMP. Let $T = \{t_1, t_2, \dots, t_n\}$. Also, the strings of T will be found in four segments such that one string of each segment occupies one of the corner nodes of the grid. Let S_1, S_2, S_3 , and S_4 stand for the strings in these four segments, respectively. Let C_1, C_2, C_3 , and C_4 be the optimal tour costs for S_1, S_2, S_3 , and S_4 , respectively.

Let $|S_i| = n_i$ for $1 \leq i \leq 4$. The total cost (i.e., the border length) for the above BLMP solution can be computed as follows. Consider S_1 alone. The cost because of this segment is $C_1 + 2(9nl + l) + (n_1 - 1)(9nl + l)$. The cost $2(9nl + l)$ is due to the two end points of the segment S_1 . The cost $(n_1 - 1)(9nl + l)$ is because each string of S_1 (except for the one in a corner of the grid) is a neighbor of a t . Upon simplification, the cost for S_1 is $C_1 + (n_1 + 1)(9nl + l)$. Summing over all the four segments, the total cost for the BLMP solution is $C_1 + C_2 + C_3 + C_4 + (n + 4)(9nl + l)$. The minimum value of this is obtained when S_1, S_2, S_3 , and S_4 form a solution to the 4-segments HTSP on T .

Clearly, an optimal solution for the 4-segments HTSP on T will also yield an optimal solution for the 4-segments HTSP on S . This can be seen as follows. Consider the strings in S_i and let $Q_i = a_1^i, a_2^i, \dots, a_{n_i}^i$ be the corresponding input strings (of S), for $1 \leq i \leq 4$. Note that C_i is nothing but $(n_i - 1)(16nl)$ plus twice the optimal tour cost for Q_i , for $1 \leq i \leq 4$. Thus, $C_1 + C_2 + C_3 + C_4$ is equal to $(n - 4)16nl + 2(C_1' + C_2' + C_3' + C_4')$ where C_i' is the optimal tour cost for Q_i , for $1 \leq i \leq 4$.

This completes the proof of Theorem 1. ■

4. ALGORITHMS FOR THE BLMP

4.1. An $O(N)$ -approximation algorithm

In this section, we will show that a simple version of the algorithm suggested by Hannenhalli et al. (2002) is actually an $O(N)$ -approximation algorithm. This algorithm can be described as follows. Assume that the input is the set of strings $S = \{s_1, s_2, \dots, s_{N^2}\}$. The algorithm first computes a tour T on strings in S . Then it threads the tour T into the grid in row-major order (Fig. 2). The first step can be done by calling the $\frac{3}{2}$ -approximation algorithm for the HTSP as suggested by Christofides (1976).

Lemma 3 $OPT_{HTSP}(S) \leq 2OPT_{BLMP}(S)$.

Proof. Let A be an optimal solution for the BLMP on S . Consider the path P' drawn as the thick dark line in Figure 2. Obviously, $Cost(P') \leq Cost(A) = OPT_{BLMP}(S)$. Let s_{i_1} and $s_{i_{N^2}}$ be the two endpoints of P' . Since the Hamming distance satisfies the triangular inequality, $\delta(s_{i_1}, s_{i_{N^2}}) \leq Cost(P')$. Consider the tour that starts at s_{i_1} , traverses along the path P' to $s_{i_{N^2}}$, and comes back to s_{i_1} . Obviously, the cost of the tour is $Cost(P') + \delta(s_{i_1}, s_{i_{N^2}}) \leq 2Cost(P') \leq 2Cost(A)$. Hence, $OPT_{HTSP}(S) \leq 2OPT_{BLMP}(S)$. ■

Theorem 5. *The above algorithm yields an $O(N)$ -approximate solution.*

Proof. First, we see that $Cost(T) \leq \frac{3}{2}OPT_{HTSP}(S) \leq 3OPT_{BLMP}(S)$. The first inequality is due to the $\frac{3}{2}$ -approximation for the HTSP. The second inequality is due to Lemma 3. Now let us analyze the cost of the solution F produced by the algorithm. Consider the path P drawn as the thick dark line in Figure 2.

FIG. 2. The thick dark line corresponds to an optimal tour on the input strings.

Obviously, $\text{Cost}(P) \leq \text{Cost}(T)$. Also, the total cost of the N rows in F is no more than $\text{Cost}(P)$. By the triangle inequality, it is not hard to see that the cost of each column in F is no more than $\text{Cost}(P)$. Therefore, $\text{Cost}(F) \leq (N+1)\text{Cost}(P) \leq (N+1)\text{Cost}(T) \leq 3(N+1)\text{OPT}_{BLMP}(S) = O(N)\text{OPT}_{BLMP}(S)$. ■

4.2. A hierarchical refinement algorithm

Several heuristics such as the epitaxial growth have been proposed to solve the BLMP problem earlier. However, most of these heuristics do not improve the cost monotonically. Local search-based algorithms are often employed to solve hard combinatorial problems. We now introduce a hierarchical refinement algorithm (HRA). This refinement technique can be applied to any heuristic placement to refine the cost and obtain a better placement. Let N be the number of probes in the placement, d a positive integer such that $d^x = N$, and $x \geq 1$ is called the degree of refinement. The refinement algorithm starts with a given placement, then it divides the placement into $s_1^0, s_2^0, \dots, s_{N/d^2}^0$ subproblems with d^2 probes per subproblem. Each of these subproblems is solved optimally—an optimal permutation among the probes is found. After this every d^2 subproblems are combined into a new subproblem $s_i^1 = \cup_{j=1}^{d^2} s_{id^2+j}^0$, $1 \leq i \leq N/d^3$. To solve s_i^1 optimally, we identify an optimal permutation among $s_{id^2+j}^0 \in s_i^1$, $1 \leq j \leq d^2$. This process continues until we are left with no subproblems to solve (Fig. 3).

We should remark that while solving a subproblem optimally, we also consider the cost contributed from the neighboring subproblems. This ensures the monotonic improvement in the placement cost. The refinement algorithm asymptotically runs in $\Theta(d^2!N)$ time. If $d = O(1)$, the refinement algorithm runs in linear time. For small values of d , the algorithm performs well in practice. HRA is a deterministic refinement algorithm. We further extend this by introducing randomness. The randomized hierarchical refinement algorithm (RHRA) is similar to the HRA algorithm. RHRA randomly selects a subsquare within the given placement and applies the HRA technique to the selected subsquare. Similar to local search algorithms, repeating RHRA algorithm several times improves the placement cost monotonically. We study the performance of both these algorithms in Section 5.

4.3. Quad epitaxial algorithm

The *epitaxial* (EPX) placement suggested in Kahng et al. (2003) places a randomly selected probe at the center of the array, it continues placing the probes greedily around the locations adjacent to the placed probes to minimize the cost (i.e., the algorithm almost spends $O(N^2)$ time to place each probe). The epitaxial algorithm gives good results for small arrays, but for larger arrays the epitaxial algorithm is impractical and extremely slow. We propose the quad epitaxial (QEPX) algorithm as a simple extension to the epitaxial algorithm. QEPX yields good performance and is very fast compared to the EPX algorithm. The basic idea behind the QEPX algorithm is to divide the array into four parts, apply EPX algorithm for each of the four parts, and finally find an optimal arrangement among the four parts. In Section 5, we compare the QEPX algorithm with EPX algorithm.

FIG. 3. Illustration of the hierarchical refinement algorithm with degree of refinement 3. This shows the possible optimal solutions (i.e., permutation among subproblems) at the top-most and penultimate levels.

5. EXPERIMENTAL STUDY

5.1. Performance of the QEPX algorithm

In this section, we compare the performance of QEPX algorithm introduced earlier. We use randomly generated probe arrays of size 32^2 , 64^2 , 128^2 , and 256^2 . In all of our experimental studies we compute a *lower bound* on the solution by picking the smallest $2N(N-1)$ edges from the complete Hamming distance graph. Column 4 (Init Cost) in Table 1 indicates the placement cost obtained by placing the probes in the row-major order as given by the input. Column 5 indicates the final placement cost obtained by the epitaxial (quad) algorithm. As we can see from columns 7 and 10, the refinement obtained by the QEPX algorithm is very close to the EPX algorithm. On the other hand, QEPX runs $3.6\times$ faster than the EPX algorithm. As we can see from Table 1, as the chip size increases, EPX algorithm becomes very slow. We ran both EPX and QEPX algorithms on a chip size of 243×243 with a time limit of 60 minutes. The QEPX algorithm took around 12 minutes to complete and improved the input placement cost by 36%. On the other hand, the EPX algorithm did not complete the placement. From our experiments we conclude that the QEPX can provide a good placement, which we can use as an input for refinement/local search algorithms such as RHRA. In the next subsection we provide our experimental study of HRA and RHRA algorithms on various placement heuristics.

5.2. Performance of refinement algorithms

We have applied our HRA and RHRA refining algorithms on the following placement heuristics.

TABLE 1. COMPARISON BETWEEN *EPITAXIAL* AND *QUAD EPITAXIAL*

Test Case	Probes	Lower Bound	Init Cost	EPX	Time (sec)	Refined Percent	QEPX	Time (sec)	Refined Percent
t-0	1024	23480	37192	27591	0.60	25.81%	28060	0.42	24.55%
t-1	1024	23427	37029	27472	0.62	25.81%	28151	0.43	23.98%
t-0	4096	86818	151116	106471	10.70	29.54%	107805	3.05	28.66%
t-1	4096	86897	151176	106430	10.37	29.60%	107634	3.23	28.80%
t-0	16384	322129	609085	410301	180.00	32.64%	411746	43.93	32.40%
t-1	16384	—	608928	409625	185.88	32.73%	410902	44.70	32.52%
t-0	65536	—	2447885	2447885	—	0.00%	1563369	765.79	36.13%
t-1	65536	—	2427143	2427143	—	0.00%	1562630	774.33	35.62%

- (RAND) Random placement: In this placement, we use the order in which the probes are provided to our algorithm.
- (SORT) Sort placement: In this placement, the input probes are sorted lexicographically.
- (SWM) Sliding window matching placement is obtained by running the SWM (Kahng et al., 2003) algorithm with parameters (6, 3).
- (REPX) Row epitaxial placement is obtained by running the row-epitaxial algorithm with three look-ahead rows.
- (EPX) Epitaxial placement is obtained by running the EPX algorithm.
- (QEPX) Quad epitaxial placement is obtained by our quad-epitaxial algorithm.

The cost of the placement obtained by running the *HRA* algorithm exactly once is given in column 5 (*HRA*). Column 6 (*RHRA*) indicates the placement cost obtained by running our randomized refinement algorithm *RHRA* for 350 iterations. From Table 2 we can see that as initial placement moves closer and closer toward the lower bound, the refinement percentage decreases, which is logical. For test cases with 729, 6561 (1024, 4096) probes we use a refinement degree $d=3$ ($d=2$). Choosing a bigger refinement degree gives better refinements, however, takes more time. Finally we conclude that our refinement algorithms would be very useful when applied in conjunction with fast initial placement heuristics.

6. CONCLUSIONS

In this article, we have studied the border length minimization problem (BLMP), which has numerous applications in biology and medicine. We solved a seven-year-old open problem in this area by showing that the BLMP is $\mathcal{NP}$ -hard. Two different proofs have been given, and we believe that the techniques in these proofs will find independent applications. We have also shown that certain generalizations of the BLMP are $\mathcal{NP}$ -hard as well. In addition, we have presented a hierarchical refinement algorithm (*HRA*) for the BLMP. Deterministic and randomized versions of this algorithm can be used to refine the solutions

TABLE 2. COST REFINEMENT FOR VARIOUS PLACEMENT HEURISTICS BY APPLYING *HRA* (HIERARCHICAL REFINEMENT ALGORITHM) AND *RHRA* (RANDOMIZED HIERARCHICAL REFINEMENT ALGORITHM) WITH 350 ITERATIONS

<i>Probes</i>	<i>Algo</i>	<i>Lower Bound</i>	<i>Init Cost</i>	<i>HRA</i>	<i>RHRA</i>	<i>Refined Percent</i>	<i>Time</i>
729	RAND	17087	26401	23970	22631	14.280%	2.83 (min)
729	SORT	17087	24082	22415	21649	10.103%	2.81 (min)
729	SWM	17087	22267	22195	22069	0.889%	2.81 (min)
729	REPTX	17087	21115	21107	21101	0.066%	2.81 (min)
729	EPTX	17087	19733	19726	19726	0.035%	2.81 (min)
6561	RAND	136820	243125	221090	209514	13.825%	17.55 (min)
6561	SORT	136820	210326	198972	191915	8.754%	17.02 (min)
6561	SWM	136820	204955	204525	203412	0.753%	17.20 (min)
6561	REPTX	136820	185386	185362	185341	0.024%	17.16 (min)
6561	EPTX	136820	168676	168623	168544	0.078%	17.15 (min)
1024	RAND	23480	37192	35236	33046	11.148%	0.28 (sec)
1024	SORT	23480	33784	32326	31026	8.164%	0.26 (sec)
1024	SWM	23480	31424	31383	31323	0.321%	0.13 (sec)
1024	QEPX	23480	28060	28035	28028	0.114%	0.47 (sec)
1024	REPTX	23480	29574	29557	29546	0.095%	0.11 (sec)
1024	EPTX	23480	27591	27567	27565	0.094%	0.11 (sec)
4096	RAND	86818	151116	143246	134485	11.005%	6.93 (sec)
4096	SORT	86818	131291	127033	121742	7.273%	4.46 (sec)
4096	SWM	86818	127516	127357	127092	0.333%	1.27 (sec)
4096	QEPX	86818	107805	107766	107702	0.096%	5.04 (sec)
4096	REPTX	86818	116406	116395	116376	0.026%	1.02 (sec)
4096	EPTX	86818	106471	106462	106448	0.022%	1.04 (sec)

obtained from any algorithm for solving the BLMP. Our experimental results indicate that indeed HRA can be useful in practice.

One of the best-performing algorithms for the BLMP is the epitaxial algorithm (EPX). This algorithm takes too much time, especially when the number of probes is large. In this article we present a variant called the quad-epitaxial algorithm (QEPX), which is much faster than EPX while yielding a solution that is very close to that of EPX in quality. QEPX partitions the input into four parts, works on each part separately, and finally combines these solutions. This idea can be extended further to partition the input into more parts and hence this algorithm is ideal for parallelism.

Some of the additional goals are: 1) We have used a simple lower bound on the quality of solution for the BLMP, and it would be nice to develop tighter lower bounds; 2) to develop more efficient algorithms than EPX; and 3) to design parallel algorithms for the BLMP.

ACKNOWLEDGMENTS

This work has been supported in part by the following grants: NSF 0326155, NSF 0829916, NIH 1R01GM079689-01A1, and NIH R01-LM010101.

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

- Chatterjee, M., Mohapatra, S., Ionan, A., et al. 2006. Diagnostic markers of ovarian cancer by high-throughput antigen cloning and detection on arrays. *Cancer Res.* 66, 1181–1190.
- Christofides, N. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. *Graduate School of Industrial Administration* [Report 388].
- de Carvalho Jr., S., and Rahmann, S. 2006. Microarray layout as a quadratic assignment problem. In: *Proc. German Conference on Bioinformatics*. Lecture Notes in Informatics P-83, 11–20.
- Hannenhalli, S., Hubell, E., Lipshutz, R., and Pevzner, P.A. 2002. Combinatorial algorithms for design of dna arrays. *Advances in Biochemical Engineering/Biotechnology* 77, 1–19.
- Kahng, A., Mandoiu, I., Pevzner, P., et al. 2003. Engineering a scalable placement heuristic for DNA probe arrays. *Intl. Conf. on Research in Computational Molecular Biology*, 148–156.
- Kundeti, V., and Rajasekaran, S. 2009. On the hardness of the border length minimization problem. *IEEE International Conference on Bioinformatics and Bio-engineering*, 248–253.
- Kundeti, V., and Rajasekaran, S. 2010. On the hardness of the border length minimization problem on a rectangular array. *International Journal of Foundations of Computer Science* 21, 1089–1100.
- Melle, C., Ernst, G., Schimmel, B., et al. 2004. A technical triade for proteomic identification and characterization of cancer biomarkers. *Cancer Res.* 64, 4099–4104.
- Welsh, J.B., Sapinoso, L.M., Kern, S.G., et al. 2003. Large-scale delineation of secreted protein biomarkers over-expressed in cancer tissue and serum. *Proceedings of the National Academy of Sciences of the United States of America* 100, 3410–3415.

Address correspondence to:
 Dr. Sanguthevar Rajasekaran
 University of Connecticut
 257 ITE Building
 371 Fairfield Way U-2155
 Storrs, CT 06269-4155

E-mail: rajasek@engr.uconn.edu