Improving the Layout of Oligonucleotide Microarrays: Pivot Partitioning

Sérgio A. de Carvalho Jr.[1,2,3] and Sven Rahmann[1,3]

[1] International NRW Graduate School in Bioinformatics and Genome Research
[2] Graduiertenkolleg Bioinformatik, Bielefeld University, Germany
Sergio.Carvalho@cebitec.uni-bielefeld.de
[3] Algorithms and Statistics for Systems Biology group, Genome Informatics, Technische Fakultät, Bielefeld University, D-33594 Bielefeld, Germany
Sven.Rahmann@cebitec.uni-bielefeld.de

Abstract. The production of commercial DNA microarrays is based on a light-directed chemical synthesis driven by a set of masks or micromirror arrays. Because of the natural properties of light and the ever shrinking feature sizes, the arrangement of the probes on the chip and the order in which their nucleotides are synthesized play an important role on the quality of the final product. We propose a new model called *conflict index* for evaluating the layout of microarrays. We also present a new algorithm, called Pivot Partitioning, that improves the quality of layouts, according to existing measures, by over 6% when compared to the best known algorithms.

1 Introduction

An oligonucleotide microarray is a piece of glass or plastic on which single-stranded fragments of DNA, called *probes*, are affixed or synthesized. Affymetrix GeneChip® arrays, for instance, can contain more than one million spots as small as 11 μm, with each spot accommodating several million copies of a probe. Probes are typically 25 nucleotides long and are synthesized on the chip, in parallel, in a series of repetitive steps. Each step appends the same nucleotide to probes of selected regions of the chip. Selection occurs by exposure to light with the help of a photolithographic mask that allows or obstructs the passage of light accordingly [5].

Formally, we have a set of probes $\mathcal{P} = \{p_1, p_2, ...p_n\}$ that are produced by a series of masks $\mathcal{M} = (m_1, m_2, ...m_T)$, where each mask m_t induces the addition of a particular nucleotide $\mathcal{S}_t \in \{A, C, G, T\}$ to a subset of $\mathcal{P}$. The *nucleotide deposition sequence* $\mathcal{S} = \mathcal{S}_1\mathcal{S}_2 \ldots \mathcal{S}_T$ corresponding to the sequence of nucleotides added at each masking step is therefore a supersequence of all $p \in \mathcal{P}$ [10].

In general, a probe can be *embedded* within $\mathcal{S}$ in several ways. An embedding of p_k is a T-tuple $\varepsilon_k = (e_{k,1}, e_{k,2}, ...e_{k,T})$ in which $e_{k,t} = 1$ if probe p_k receives nucleotide $\mathcal{S}_t$ (at step t), or 0 otherwise (Fig. 1). In particular, a *left-most embedding* is an embedding in which the bases are synthesized as early as possible (see ε_3 in Fig. 1).

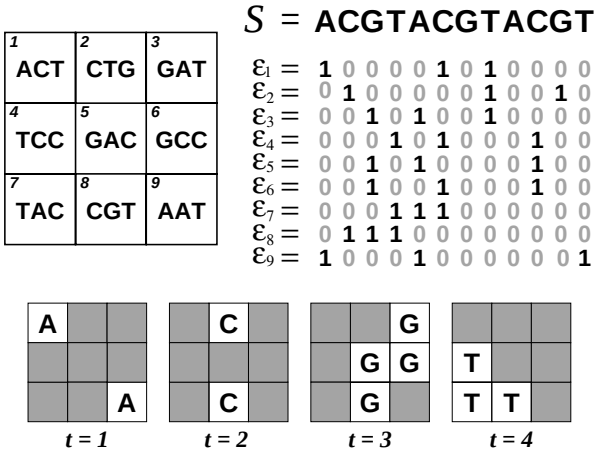

Fig. 1. Synthesis of a hypothetical 3×3 chip. Top left: chip layout and the 3 nt probe sequences. Top right: deposition sequence and probe embeddings. Bottom: first four resulting masks.

The deposition sequence is often taken as a repeated permutation of the alphabet, mainly because of its regular structure and because such sequences maximize the number of distinct subsequences [4].

We distinguish between *synchronous* and *asynchronous* embeddings. In the first case, each probe has exactly one nucleotide synthesized in every cycle of the deposition sequence; hence, 25 cycles or 100 steps are needed to synthesize probes of length 25. In the case of asynchronous embeddings, probes can have any number of nucleotides synthesized in any given cycle, allowing shorter deposition sequences. All Affymetrix chips that we know of can be asynchronously synthesized in 74 steps (18.5 cycles), which is probably due to careful probe selection.

Because of diffraction of light or internal reflection, untargeted spots can be accidentally activated in a certain masking step, producing unpredicted probes that can compromise experimental results. This problem is more likely to occur near the borders between masked and unmasked spots [5]. This observation has given rise to the term *border conflict*.

We are interested in finding an *arrangement* of the probes on the chip together with *embeddings* in such a way that the chances of unintended illumination during mask exposure steps are minimized. The problem appears to be hard because of the exponential number of possible arrangements, although we are not aware of an NP-hardness proof. In a separate work [2], we present a formulation of the above problem as a quadratic assignment problem (QAP), a classical combinatorial optimization problem that is, in general, NP-hard and particularly hard to solve in practice [3]. Optimal solutions are thus unlikely to be found even for small chips and even if we assume that all probes have a single predefined embedding.

If we consider all possible embeddings (up to several million for a typical Affymetrix probe), the problem is even harder. For this reason, the problem has been traditionally tackled in two phases. First, an initial embedding of the probes is fixed and an arrangement of these embeddings on the chip with minimum border conflicts is sought. This is usually referred to as the *placement*. Second, a *post-placement* optimization phase re-embeds the probes considering their location on the chip, in such a way that the conflicts with the neighboring spots are further reduced.

It seems intuitive that better results should be achieved if the placement and embedding phases are considered together, not separately. Because of the generally high number of embeddings of each single probe in the asynchronous setting, it is not easy to design algorithms that make efficient use of this additional freedom and achieve reasonable running times in practice. In fact, so far we know of no single publication that merges the two phases; in this article we propose such a strategy called Pivot Partitioning.

The rest of this paper is structured as follows. Section 2 details two different ways of evaluating computed layouts and embeddings; they form the objective functions that we aim to minimize. As a refinement of the "classical" border length, we introduce the *conflict index* measure. Section 3 reviews existing placement, partitioning, and post-placement strategies. In Section 4 we discuss an extension of the optimal single probe embedding (OSPE) algorithm (that first appeared in [7]) to support our new measure. Our partitioning strategy, that for the first time combines partitioning the chip with embedding the probes, is described in Section 5. Computational results follow in Section 6.

2 Evaluating Layouts and Embeddings

Border length. Hannenhalli and co-workers [6] were the first to give a formal definition of the problem of unintended illumination in the production of microarrays. They formulated the *Border Length Minimization Problem* which aims at finding an arrangement of the probes together with their embeddings in such a way that the number of border conflicts during mask exposure steps is minimal.

The *border length* $\mathcal{B}_t$ of a mask m_t is defined as the number of borders shared by masked and unmasked spots at masking step t. The total border length of a given arrangement is the sum of border lengths over all masks. For example, the initial four masks shown in Fig. 1 have $\mathcal{B}_1 = 4$, $\mathcal{B}_2 = 6$, $\mathcal{B}_3 = 6$ and $\mathcal{B}_4 = 4$. The total border length of that arrangement is 50 (masks 5 to 12 not shown).

Conflict Index. The border length of an individual mask measures the quality of that mask. We are more interested in estimating the risk of synthesizing a faulty probe at a given spot, that is, we need a per-probe measure instead of a per-mask measure. Additionally, the definition of border length does not take into account two important practical considerations [8]:

a) stray light might activate not only adjacent neighbors but also probes that lie as far as three cells away from the targeted spot;

b) imperfections produced in the middle of a probe are more harmful than in its extremities.

This motivates the following definition of the *conflict index* $\mathcal{C}(p)$ of a probe of length ℓ_p that is synthesized in T masking steps. First, we define a distance-dependent weighting function, $\delta(p, p', t)$, that accounts for observation a) above:

$$\delta(p, p', t) := \begin{cases} (d(p, p'))^{-2} & \text{if } p' \text{ is unmasked at step } t, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $d(p, p')$ is the Euclidean distance between the spots of p and p'. This form of weighting function is the same as suggested in [8]. Note that δ is a "closeness" measure between p and p' only if p' is not masked (and thus creates the potential of illumination at p). To limit the number of neighbors that need to be considered, we restrict the support of $\delta(p, p', \cdot)$ to those $p' \neq p$ that are in a 7×7 grid centered around p (see Fig. 2 left).

We also define position-dependent weights to account for observation b):

$$\omega(p, t) := \begin{cases} c \cdot \exp(\theta \cdot \lambda(p, t)) & \text{if } p \text{ is masked at step } t, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $c > 0$ and $\theta > 0$ are constants, and

$$\lambda(p, t) := 1 + \min(b_{p,t}, \ell_p - b_{p,t}) \tag{3}$$

is the distance, from the start or end of the final probe sequence, of the last base synthesized before step t: $b_{p,t}$ denotes the number of nucleotides synthesized within p up to and including step t, and ℓ_p is the probe length (see Fig. 2 right).

The motivation behind an exponentially increasing weighting function is that the probability of a successful stable hybridization of a probe with its target should increase exponentially with the absolute value of its Gibbs free energy, which increases linearly with the length of the longest perfect match between probe and target. The parameter θ controls how steeply the exponential weighting function rises towards the middle of the probe. In our experiments, we set $\theta := 5/\ell_p$ and $c = 1/\exp(\theta)$.

We now define the conflict index of a probe p as

$$\mathcal{C}(p) := \sum_{t=1}^{T} \Big(\omega(p, t) \sum_{p'} \delta(p, p', t) \Big), \tag{4}$$

where p' ranges over all probes that are at most three cells away from p. $\mathcal{C}(p)$ can be interpreted as the fraction of faulty p-probes (because of unwanted illumination).

We note the following relation between conflict index and border length. Define $\delta(p, p', t) := 1$ if p' is a direct neighbor of p and is unmasked in step t, and $:= 0$ otherwise. Define $\omega(p, t) := 1$ if p is masked in step t, and $:= 0$ otherwise. Then $\sum_s \mathcal{C}(p) = 2 \sum_{t=1}^{T} \mathcal{B}_t$, as each border conflict is counted twice, once for p

0.06	0.08	0.10	0.11	0.10	0.08	0.06
0.08	0.13	0.20	0.25	0.20	0.13	0.08
0.10	0.20	0.50	1.00	0.50	0.20	0.10
0.11	0.25	1.00	p	1.00	0.25	0.11
0.10	0.20	0.50	1.00	0.50	0.20	0.10
0.08	0.13	0.20	0.25	0.20	0.13	0.08
0.06	0.08	0.10	0.11	0.10	0.08	0.06

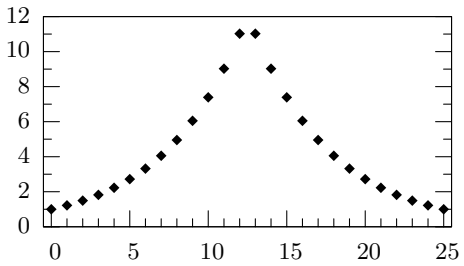

Fig. 2. Ranges of values for both δ and ω on a typical Affymetrix chip where probes of length 25 are synthesized in 74 masking steps. Left: approximate values of the distance-dependent weighting function $\delta(p, p', t)$ for a probe p (shown in the center) and close neighbors p', assuming that p' is unmasked at step t. Right: position-dependent weights $\omega(p, t)$ on the y-axis for each value of $b_{p,t}$ on the x-axis, assuming that p is masked at step t.

and once for p'. Therefore border length and total conflict are equivalent for a particular choice of δ and ω. For our choice (1) and (2), they are not equivalent, but still correlated: a good layout has both low border length and low conflict indices.

3 Previous Work

Up to now, the tasks of probe placement and probe embedding were considered separately. Placement is often handled by (recursively) partitioning the chip into smaller regions before applying a placement algorithm. We now review existing placement algorithms, partitioning algorithms and post-placement strategies.

Placement Algorithms. The border length problem on large oligonucleotide arrays of arbitrary probes was first formally addressed in [6]. The article reports that the first Affymetrix chips were designed using a heuristic for the traveling salesman problem (TSP). The idea consists of building a weighted graph with nodes representing probes, and edges containing the Hamming distance between the probe sequences. A TSP tour is approximated, resulting in consecutive probes in the tour being likely similar. The TSP tour is then *threaded* on the array in a row-by-row fashion. A different threading of the TSP tour, called *1-threading*, is suggested to achieve up to 20% reduction in border length.

A different strategy called *Epitaxial* placement [7] places a random probe in the center of the array and continues to insert probes in spots adjacent to already filled spots. Priority is given to spots with the largest numbers of filled neighbors. At each iteraction, it examines all non-filled spots and finds a non-assigned probe with minimum sum of Hamming distances to the neighboring probes, employing a greedy heuristic to select the next spot. A further 10% reduction in border conflict over TSP + 1-threading is claimed.

Both the Epitaxial algorithm and the TSP approach do not scale well to large chips. For this reason, [8] proposes a simpler variant of the Epitaxial algorithm, called *Row-epitaxial*, with two main differences: spots are filled in a pre-defined order, namely from top to bottom, left to right, and only probes of a limited list of candidates are considered when filling each spot. Experiments show that Row-epitaxial is the best large-scale placement algorithm, achieving up to 9% reduction in border length over the TSP + 1-threading.

Partitioning Algorithms. The placement problem can be partitioned by dividing the set of probes into smaller sub-sets, and assigning these sub-sets to sub-regions of the chip. Each sub-region can then be treated as an independent chip or recursively partitioned. In this way, algorithms with non-linear time or space complexities can be used to compute the layout of larger chips that otherwise would not be feasible.

The only partitioning that we know of is the Centroid-based Quadrisection [9]. It starts by randomly selecting a probe $c_1 \in \mathcal{P}$. Then, it selects another probe c_2 maximizing $h(c_1, c_2)$, the Hamming distance between their embeddings. Similarly, it selects c_3 and c_4 maximizing the sum of Hamming distance between these four probes that are called centroids. All other probes $p \in \mathcal{P}$ are then compared to the centroids and assigned to a sub-set $\mathcal{P}_k$ associated with c_k with minimum $h(p, c_k)$. The chip is divided into four quadrants, each being assigned to a sub-set $\mathcal{P}_k$. The procedure is repeated recursively on each quadrant until a given recursion depth is reached. In the end, the Row-epitaxial algorithm is used to produce the placement of the probes in each final sub-region.

Post-placement Optimization. Once the placement is done, further reduction of conflicts can be achieved by re-embedding the probes without changing their locations. The paper [7] presents a dynamic programming algorithm, that we call Optimum Single Probe Embedding (OSPE), for computing an optimum embedding of a probe with respect to the neighboring probes, whose embeddings are considered fixed. Originally, it was developed for border length minimization; in Sect. 4 we give a slightly more general form that also applies to the conflict index measure.

The OSPE algorithm is the basic operation of several post-placement optimization algorithms: Batched Greedy [7], Chessboard [7] and Sequential [9]. Their main difference lies in the order in which the re-embeddings take place. Since the OSPE never increases the amount of conflicts in a region, all optimization algorithms can be executed several times until a local optimal solution is found, or until the improvements drop below a given threshold.

The Sequential algorithm just proceeds spot by spot, from top to bottom, left to right, re-embedding all probes with the OSPE algorithm. Surprisingly, it achieves the greatest reduction of border conflicts with a running time comparable to Batched Greedy, the fastest among the three.

4 Optimum Single Probe Embedding

The Optimum Single Probe Embedding (OSPE) algorithm finds an optimal embedding of a single probe on a given spot, assuming that all neighboring embeddings are fixed. It can be seen as a special case of a global alignment between the probe sequence p of length ℓ and the deposition sequence $\mathcal{S}$ of length T. We use an $(\ell + 1) \times (T + 1)$ array D, where $D[i, j]$ is defined as the minimum cost of an embedding of $p[1..i]$ into $\mathcal{S}[1..j]$. The cost is the sum of conflicts induced by the embedding of p on its neighbors plus the conflicts suffered by p because of the embeddings of its neighbors.

At every step j of the deposition sequence, the probe p can be either masked or unmasked. Thus, entry $D[i, j]$ is computed as the minimum between the costs resulting from each possible state:

$$D[i, j] = \min(D[i, j-1] + M_{ij}, D[i-1, j-1] + U_j).$$

The costs M_{ij} and U_j depend on probe p and neighboring probes p'. M_{ij} denotes the cost of masking probe p at step j given that base i of p has been synthesized previously. Any unmasked neighbor p' generates a conflict on p with cost $\omega(p, i) \cdot \delta(p, p', j)$; therefore the total cost is

$$M_{ij} = \sum_{p'} \omega(p, i) \cdot \delta(p, p', j).$$

U_j denotes the cost of unmasking probe p at step j, which generates a conflict on each masked neighbor p' with cost $\omega(p', j) \cdot \delta(p', p, j)$; therefore

$$U_j = \sum_{p'} \omega(p', j) \cdot \delta(p', p, j).$$

The first column of D is initialized as follows: $D[0, 0] = 0$ and $D[i, 0] = \infty$ for $0 < i \leq \ell$. The first row is $D[0, j] = D[0, j-1] + M_{0j}$ for $0 < j \leq T$. The time complexity of the OSPE algorithm is obviously $O(\ell \cdot T)$.

5 Pivot Partitioning

Traditionally, the microarray layout problem has been tackled in two phases: placement, during which an initial embedding of the probes is fixed, and post-placement optimization, when probes are re-embedded using the OSPE algorithm. We believe that better layouts can be produced if the placement phase also considers the various embeddings that a probe can have. In this section we propose a new partitioning algorithm called Pivot Partitioning (PP).

Our algorithm has some similarities with the Centroid-based Quadrisection (CQ) described in Sect. 3. Its main differences are motivated by the following observation. As mentioned earlier, some probes can have up to several millions different embeddings, while others may have only a few or even only one possible embedding. Probes with more embeddings can better "adapt" to the other

Algorithm 1 PivotPartitioning

Input: chip dimension,
set of probes $\mathcal{P} = \{p_1, p_2, ...p_n\}$,
maximum partitioning depth t_{max}
Output: placement of the probes $p \in \mathcal{P}$ on the chip

1. Select probes p with minimum number of embeddings, $E(p)$, as pivot candidates:
   (a) Let $\mathcal{Q} = \{p \in \mathcal{P} | E(p)$ is minimal$\}$
   (b) Set $\mathcal{P} \leftarrow \mathcal{P} \setminus Q$
2. Let the region R consist of all rows and columns.
3. Call the Recursive Partitioning with the initial partitioning depth 1:
   return RecursivePartitioning (1, t_{max}, R, $\mathcal{Q}$, $\mathcal{P}$)

probes, that is, when placed on a particular spot, they are more likely to have an embedding with fewer conflicts than a probe that has only a limited number of embeddings.

We use the probes with fewer embeddings, which we call "pivots", to drive the partitioning of the probe set and to re-embed the probes just before their placement (as a partitioning algorithm, PP also works in combination with another placement algorithm). Also, we designed our algorithm to work for border length as well as conflict index minimization.

5.1 Pivot Candidates

The first step of the Pivot Partitioning (Algorithm 1), is to select the pivot candidates $\mathcal{Q}$, a set of probes that can later be chosen as pivots. Our pivots are the equivalent of the centroids of the CQ algorithm: they are used to partition the probe set. They are restricted, however, to the probes having fewer embeddings.

The reasons are two-fold. First, less time is spent choosing the pivots since fewer candidates need to be considered. Second, probes with fewer embeddings are better representatives to drive the partitioning. The problem is that some embeddings may have their unmasked steps concentrated in one region of the deposition sequence. This is specially true if the probes are embedded in a left-most or right-most fashion. Some Affymetrix probes, for instance, can be synthesized in the first 37 masking steps, thus using only half of the total 74 steps. Such probes are clearly not good choices for pivots. Probes with fewer embeddings, on the other hand, are guaranteed to cover most (if not all) cycles of the deposition sequence.

In order to guarantee a good partitioning, we limit the size of $\mathcal{Q}$ to a minimum of 1% of the total number of probes[4]. This is achieved by selecting probes with the next minimum number of embeddings. Computing the number of embeddings of a probe takes $O(\ell T)$ time, where ℓ is the length of the probe and T is the

[4] Usually, around 1-2% of the probes of an Affymetrix array have only one possible embedding; or two, if we consider that they appear in PM/MM pairs and must be "aligned" in all but the steps that synthesize their middle bases

Algorithm 2 Recursive Partitioning

Input: current depth t,
maximum depth t_{max}
rectangular region R of the chip,
set of pivot candidates $\mathcal{Q}$,
set of probes $\mathcal{P}$,
Output: placement of the probes $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ on the region R of the chip

1. If $t = t_{max}$ then
   (a) Re-embed $p \in \mathcal{P}$ optimally with respect to all $q \in \mathcal{Q}$
   (b) Return RowEpitaxial (R, $\mathcal{P} \cup \mathcal{Q}$)
2. Select q' and $q'' \in \mathcal{Q}$ such that $h(q', q'')$ is maximal
3. Partition the set of pivot candidates:
   (a) $\mathcal{Q}' = \{q \in \mathcal{Q} \mid h(q, q') < h(q, q'')\}$
   (b) $\mathcal{Q}'' = \{q \in \mathcal{Q} \mid h(q, q') > h(q, q'')\}$
   (when $h(q, q') = h(q, q'')$, assignments are made in an attempt to achieve balanced partitionings)
4. Partition the set of probes:
   (a) $\mathcal{P}' = \{p \in \mathcal{P} \mid w(p, q') < w(p, q'')\}$
   (b) $\mathcal{P}'' = \{p \in \mathcal{P} \mid w(p, q') > w(p, q'')\}$
   (when $h(p, q') = h(p, q'')$, assignments are made in an attempt to achieve balanced partitionings)
5. Partition R into two sub-regions R' and R'' proportionally to the number of probes in $\mathcal{P}' \cup \mathcal{Q}'$ and $\mathcal{P}'' \cup \mathcal{Q}''$
6. return RecursivePartitioning ($t + 1$, t_{max}, R', $\mathcal{Q}'$, $\mathcal{P}$')
   $\cup$ RecursivePartitioning ($t + 1$, t_{max}, R'', $\mathcal{Q}''$, $\mathcal{P}$")

length of the deposition sequence. With a few optimizations, however, even a million probes can be examined in a few minutes.

5.2 Recursive Partitioning

The essence of Pivot Partitioning is its recursive procedure (Algorithm 2) that is executed until a given recursion depth t_{max} is reached.

If the maximum recursion depth has not been reached yet, we choose a pair of pivots q' and $q'' \in \mathcal{Q}$ with maximum Hamming distance between their embeddings, $h(q', q'')$. All other $q \in \mathcal{Q}$ are assigned to a sub-set of $\mathcal{Q}$ associated with the pivot whose Hamming distance to q is minimum (step 3).

The next step similarly partitions $\mathcal{P}$ into two sub-sets. A probe $p \in \mathcal{P}$ is assigned to the sub-set associated with the pivot q with minimum weighted distance $w(p, q)$. The weighted distance is computed with the OSPE algorithm, ignoring the location of the probes since they have not been placed yet. In this way, we make the assignments considering all possible embeddings of p.

Step 5 partitions R into two sub-regions, proportionally to the number of probes in $\mathcal{Q}' \cup \mathcal{P}'$ and $\mathcal{Q}'' \cup \mathcal{P}''$, alternating horizontal and vertical divisions. Since we only deal with rectangular regions, sometimes it is necessary to move

a few probes from one partition to the other in order to ensure that the probes fit in the sub-regions.

Each sub-region is then processed recursively. Once the maximum partitioning depth t_{max} is reached, the Row-epitaxial [8] algorithm is used to place the probes of $\mathcal{P} \cup \mathcal{Q}$ in the region R. Before that, however, all probes $p \in \mathcal{P}$ are re-embedded optimally with respect to the pivots (again using OSPE ignoring probe locations), which improves the "alignment" of all embeddings in that region.

6 Results and Discussion

We now present the results of running our Pivot Partitioning (PP) algorithm on random chips. Table 1 shows the normalized border length (total border length divided by the number of probes) using our own implementations of Row-epitaxial (for the placement) as well as the Sequential post-placement optimization.

Our results show that, in the first level of partitioning, PP allows for a reduction in border length by as much as 16% when compared to running the Row-epitaxial alone (from 41.27 to 34.69 on 500×500 chips). The total border length for $t_{max} = 2$ on 500×500 is 8 673 722. This represents a reduction of as much as 6.8% over the layout produced by the Centroid-based Quadrisection (CQ) similarly combined with Row-epitaxial and followed by the Sequential optimization, which produced a layout with a border length of 9 307 510 as reported in [9]. In the next levels of partitioning, we observe a small increase in border length but, on the other hand, we also report a significant reduction in running times.

Table 2 shows similar results with the average conflict index. For these experiments, we use a version of Row-epitaxial implemented for conflict index minimization, which fills every spot with a probe p minimizing $\mathcal{C}(p)$. For the post-placement optimization, we use the Sequential algorithm with OSPE for conflict index minimization as described in Sect. 4. Computing the conflict index of a spot for every probe candidate is not as straight forward as computing the Hamming distance between a probe and its neighbors; thus both versions of Row-epitaxial and Sequential for conflict index minimization are significantly slower. For this reason, we set the limit on the number of probes considered by the Row-epitaxial to $Q = 2\,000$.

We also compare the performance of Pivot Partitioning with the Centroid-based Quadrisection (CQ). Table 3 shows the total border length of layouts produced by CQ as reported in [9]. We run PP on similar input and report the results with equivalent partitioning depths (two levels of PP are equivalent to one level of CQ). The results are shown as a percentage of reduction in border length compared to CQ. For instance, on 500×500 chips, PP produces layouts with 8.95% less conflicts than CQ, on average.

Our results show that PP produces layouts with less conflicts than CQ except for higher partitioning depths on the smaller chips. We suspect that this disad-

Table 1. Normalized border length of layouts produced by Pivot Partitioning on random chips with dimensions ranging from 100×100 to 500×500, with probes synchronously embedded in a deposition sequence of length 100. Partitioning depths ranges from $t_{max} = 0$ (no partitioning) to $t_{max} = 6$. Row-epitaxial is used for the placement (with $Q = 20\,000$), followed by the Sequential post-placement optimization. Running times are reported in seconds, and do not include the post-placement optimization.

	$t_{max} = 0$		$t_{max} = 2$		$t_{max} = 4$		$t_{max} = 6$	
Dim	Cost	Time	Cost	Time	Cost	Time	Cost	Time
100	42.77	34	39.19	13	40.72	10	42.11	11
200	41.63	429	37.30	155	38.53	62	40.00	85
300	41.38	1 174	36.12	766	37.22	264	38.53	139
500	41.27	3 524	34.69	3 472	35.50	1 996	36.58	713

Table 2. Average conflict index of layouts produced by Pivot Partitioning on random chips of synchronous embeddings. We use versions of the Row-epitaxial (with $Q = 2\,000$) and the Sequential algorithms for conflict index minimization.

	$t_{max} = 0$		$t_{max} = 2$		$t_{max} = 4$		$t_{max} = 6$	
Dim	Cost	Time	Cost	Time	Cost	Time	Cost	Time
100	514.49	45	453.67	37	467.78	19	475.44	15
200	517.07	192	466.22	215	452.41	166	462.55	99
300	518.51	438	475.84	524	452.00	466	448.17	336
500	517.50	1 471	481.36	1 530	462.33	1 472	445.43	1 295

vantage is due to the "borrowing heuristic" used by CQ that permits, during the placement, borrowing probes from neighboring partitions in order to maintain a high number of probes that can be considered for filling the last spots of a quadrant. We are planning to implement a similar strategy on Pivot Partitioning that could improve the quality of our solutions.

Summary. We have presented a new partitioning strategy that for the first time combines the partitioning the chip with embedding of the probes. The main advantages of our approach over previous methods are: faster and better selection of pivots used to drive the assignment of probes to sub-regions; and improved assignment of probes to regions by considering all valid embeddings of a probe.

Acknowledgments. We thank Ion Mandoiu, Xu Xu and Sherief Reda for providing an implementation of their algorithms.

References

1. Binder, H., Preibisch, S.: Specific and nonspecific hybridization of oligonucleotide probes on microarrays. *Biophysical Journal* (2005) **89** 337–352.

Table 3. Comparison between Pivot Partitioning (PP) and Centroid-based Quadrisection (CQ) on random chips with dimensions ranging from 100×100 to 500×500, whose probes are synchronously embedded in a deposition sequence of length 100. The partitioning depths varies from $L = 1$ to $L = 3$ for the CQ algorithm and, equivalently, from $t_{max} = 2$ to $t_{max} = 6$ for PP. Both partitionings use Row-epitaxial for the placement (with $Q = 20\,000$) and are followed by the Sequential post-placement optimization. The data shows the total border length of chips produced by CQ (extracted from [9]), and the results of using PP on similar input, as percentage of the reduction in border length compared to CQ. For instance, PP generates on average 8.95% less border length on 500×500 chips with $t_{max} = 2$.

	CQ	PP	CQ	PP	CQ	PP
Dim	$L = 1$	$t_{max} = 2$	$L = 2$	$t_{max} = 4$	$L = 3$	$t_{max} = 6$
100	393 218	0.18%	399 312	-1.89%	410 608	-2.48%
200	1 524 803	2.27%	1 545 825	0.48%	1 573 096	-1.34%
300	3 493 552	7.12%	3 413 316	2.05%	3 434 964	-0.61%
500	9 546 351	8.95%	9 355 231	4.67%	9 307 510	1.03%

2. de Carvalho Jr., S., Rahmann, S.: Microarray Layout as a Quadratic Assignment Problem. Submitted (2006).
3. Çela,E. (1998) *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer, Massachessets, USA.
4. Chase, P.: Subsequence numbers and logarithmic concavity. *Discrete Mathematics* (1976) **16** 123–140.
5. Fodor, S., Read, J., Pirrung, M., Stryer, L., Lu, A., Solas, D.: Light-directed, spatially addressable parallel chemical synthesis. *Science* (1991) **251** 767–73.
6. Hannenhalli, S., Hubell, E., Lipshutz, R., Pevzner, P.: Combinatorial algorithms for design of DNA arrays. *Advances in Biochemical Engineering / Biotechnology* (2002) **77** 1–19.
7. Kahng, A., Mandoiu, I., Pevzner, P., Reda, S., Zelikovsky, A.: Border length minimization in DNA array design. In *Proceedings of the Second Workshop on Algorithms in Bioinformatics* (WABI 2002).
8. Kahng, A., Mandoiu, I., Pevzner, P., Reda, S., Zelikovsky, A.: Engineering a scalable placement heuristic for DNA probe arrays. In *Proceedings of the Seventh Annual International Conference on Computational Molecular Biology* (2003) 148–156.
9. Kahng, A., Mandoiu, I., Reda, S., Xu, X., Zelikovsky, A.: Evaluation of placement techniques for DNA probe array layout. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design* (2003) 262–269.
10. Rahmann, S.: The shortest common supersequence problem in a microarray production setting. In *Proceedings of the 2nd European Conference in Computational Biology* (ECCB 2003), volume 19 Suppl. 2 of *Bioinformatics*, pages ii156–ii161.