BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–7

Quadratic Assignment and the Layout of Oligonucleotide Microarrays

Sérgio A. de Carvalho Jr.[a] and Sven Rahmann[b]

[a]Graduiertenkolleg Bioinformatik, Bielefeld University, Germany,
[b]Algorithms and Statistics for Systems Biology, Genome Informatics, Bielefeld University, Germany.

ABSTRACT

Motivation: The production of commercial DNA microarrays is based on a light-directed chemical synthesis driven by a set of masks or micromirror arrays. Due to 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. In this paper, we propose a new model for evaluating microarray layouts called *conflict index*. We also describe and show experimental results of a new approach to the problem of designing high-density microarrays based on the *quadratic assignment problem* (QAP).

Results: We used an existing QAP heuristic algorithm called GRASP to design the layout of small artificial chips with promising results. We compare this approach with the best known algorithm and describe how it can be combined with other existing algorithms to design the latest million-probe microarrays.

Availability: Source code is available from the authors upon request.

Contact: Sergio.Carvalho@cebitec.uni-bielefeld.de

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. The chips produced by Affymetrix, for instance, can contain more than one million spots (or *features*) as small as 11 μm, with each spot accommodating several million copies of a probe. Probes are typically 25 nucleotides long and are synthesized in parallel, on the chip, 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 (Fodor *et al.*, 1991).

Formally, we have a set of probes $\mathcal{P} = \{p_1, p_2, ...p_n\}$ that are produced by a series of masks $\mathcal{M} = (m_1, m_2, ...m_\mu)$, where each mask m_k induces the addition of a particular nucleotide $t_k \in \alpha = \{A, C, G, T\}$ to a subset of $\mathcal{P}$. The *nucleotide deposition sequence* $\mathcal{S} = t_1 t_2 \dots t_\mu$ corresponding to the sequence of nucleotides added at each masking step is therefore a supersequence of all $p_i \in \mathcal{P}$.

Figure 1. Synthesis of a hypothetical 3 x 3 chip. a) Chip layout and 3-base-long probe sequences; b) deposition sequence and embeddings of the highlighted probes; c) the first three resulting photolithographic masks.

In general, a probe can be *embedded* within $\mathcal{S}$ in several ways. An embedding of p_i is a μ-tuple $\varepsilon_i = (e_{i,1}, e_{i,2}, ...e_{i,\mu})$ in which $e_{i,k} = 1$ if probe p_i receives nucleotide t_k (at step k), or 0 otherwise (Figure 1).

Deposition sequences are usually cyclical, that is $\mathcal{S}$ is a repeated permutation of α. This is mainly because such sequences maximize the number of possible subsequences (Chase, 1976). In this context, we can distinguish between *synchronous* and *asynchronous* embeddings. In the first case, each probe has one and only one nucleotide synthesized in every cycle of the deposition sequence; thus, 100 masking steps are needed to synchronously synthesize probes of length 25. In the case of asynchronous embeddings, probes can have any number of nucleotides syntesized in any given cycle. This allows for shorter deposition sequences. Most (if not all) Affymetrix chips, for instance, can be synthesized in 74 masking steps.

Due to diffraction of light or internal reflection, untargeted spots can sometimes be accidentally activated in a certain masking step, producing unpredicted probes that can compromise the results of an experiment. This issue was described by Fodor *et al.*, 1991 who noted that the problem is more likely to occur near the borders between masked and unmasked spots. 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 their embeddings in such a way that we minimize the chances of unintended illumination during mask exposure steps. As we show in later sections, this problem is intrinsically hard due to the exponential number of possible arrangements, and optimal solutions are unlikely to be found even for very small chips and even if we consider the probes as having a single pre-defined embedding.

If we consider all valid embeddings, the problem is even harder. A typical probe of an Affymetrix chip, for instance, can have up to several million possible embeddings. 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 refered to as the *placement* problem. Second, a *post-placement* optimization phase re-embeds the probes considering its location on the chip, in such a way that the conflicts with the neighboring spots are further reduced.

In the next section, we review the Border Length Minimization Problem introduced by Hannenhalli *et al.*, 2002, and define an extended model for evaluating microarray layouts called *conflict index*. In section 3 we briefly review exising placement strategies. In section 4 we propose a new approach to the problem based on a quadratic assignment problem (QAP) formulation. We present the results of using a QAP heuritic algorithm called GRASP to design small artificial chips in section 5; we also compare its performance with the best known placement algorithm and describe how this approach can be used to design larger microarrays.

2 MODELING

Hannenhalli *et al.*, 2002 were the first to give a formal definition to the problem of unintended illumination in the production of microarrays. They formulated the Border Minimization Problem, which aims at finding an arrangement of the probes together with their embeddings in such a way the number of border conflicts during mask exposure steps is minimal.

The *border length* of a mask m_k is simply defined as the number of borders shared by masked and unmasked spots at masking step k. The total border length of a given arrangement is the sum of border lengths over all masks.

2.1 Conflict Index

Kahng *et al.*, 2003/1 noted that the definition of border length does not take into account two simple yet important practical considerations:

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.

With these observations in mind, we define the conflict index $\mathcal{C}(s)$ of a spot s whose probe of length ℓ_s is synthesized in μ masking steps as follows. First we define a distance-dependent weighting function, $\delta(s, s', k)$, that accounts for observation a) above:

Figure 2. Range of values for both δ and ω on a typical Affymetrix chip where probes of length $\ell = 25$ are synthesized in $\mu = 74$ masking steps. a) Distance-dependent weighthing function δ; b) position-dependent weights ω.

$$\delta(s, s', k) := \begin{cases} (d(s, s'))^{-2} & \text{if } s' \text{ is unmasked at step } k, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $d(s, s')$ is the Euclidian distance between spots s and s'. We also use position-dependent weights to account for observation b):

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

where

$$\lambda(s, k) := 1 + \min(b_{s,k}, \ell_s - b_{s,k}), \tag{3}$$

c and θ are constants, and $b_{s,k}$ denotes the number of nucleotides synthesized at spot s up to and including step k.

TODO: review this definition.

We now define the conflict index of a spot s as

$$\mathcal{C}(s) := \sum_{k=1}^{\mu} \left(\omega(s, k) \sum_{s'} \delta(s, s', k) \right), \tag{4}$$

where s' ranges over all spots that are at most three cells away from s, in accordance with observation a).

Our definition of conflict index aims at capturing the characteristics of the problem of unintended illumination but some decisions were rather arbitrary. We set the constants c and θ as follows:

$$\theta = \frac{5}{l_s},$$

$$c = \frac{1}{\theta}.$$

The distance-dependent weighthing function δ is as suggested in Kahng *et al.*, 2003/1. Our position-dependent weights ω, however, are different. Instead of $\sqrt{\lambda(s, k)}$, we use an exponential function. It is generally agreed that the chances of a successful hibridization between probe and target are higher if the mismatched base occurs at the extremities of the formed duplex instead of at its center. The exact effects of this position, however, is not yet fully understood and has been an active topic of research (Binder and Preibisch, 2005). The range of values for both δ and ω of a typical Affymetrix chip are illustrated in Figure 2.

The definition of border length is clearly related to our definition of conflict index. However, while the first measures the quality of a mask, the latter estimates the risk of producing faulty probes in a given spot. A good layout is one with low border length as well as low average conflict index, although it is clearly possible to reduce the conflict index at the expense of an increase in border length, and vice-versa.

3 PREVIOUS WORK

In this section we review existing algorithmic techniques for designing oligonucleotide microarrays. We make a distinction between placement algorithms and partitioning algorithms. Post-placement optimizations such as the Chessboard (Kahng *et al.*, 2002) are not covered.

3.1 Placement Algorithms

The first to formally address the border length problem were Feldman and Pevzner, 1994. They showed how an optimal placement can be constructed based on a 2-dimensional Gray code. However, their work is restricted to *uniform arrays* (arrays containing all possible probes of a given length) and synchronous embeddings.

Hannenhalli *et al.*, 2002 were the first to work with arrays of arbitrary probes. They reported that the first Affymetrix chips were designed using a heuristic algorithm for the traveling salesman prolem (TSP). The idea consisted of building a weighted graph with nodes representing probes and edges containing the Hamming distance between the probes. A TSP tour with minimum weight was then constructed, resuling in consecutive probes in the tour being likely to be similar. The TSP tour was finally *threaded* on the array in a row-by-row fashion. Hannenhalli *et al.*, 2002 enhanced this approach by suggesting a different threading of the TSP tour on the chip, called *1-threading*, to achieve up to 20% reduction in border length.

Kahng *et al.*, 2002 proposed the epitaxial placement algorithm that places a random probe in the center of the array and continues to insert probes in spots adjacent to already filled spots. It employs a greedy heuristic to select the next spot to be filled and the probe that is assigned to it.

Priority is given to spots whose all four neighbors are already filled, in which case the algorithm places the probe with minimum sum of Hamming distances to its neighbors. If no such a spot exists, the algorithm examines all non-filled spots s_i with $n_i \geq 1$ filled neighbors and finds a non-assigned probe p_j with minimum sum of Hamming distances to the neighboring probes H_{ij}. It then computes a normalized cost of each possible assignment of p_j to s_i as $c(s_i, p_j) = k_{n_i} H_{ij}/n$, where k_{n_i} are scaling coefficients ($k_1 = 1$, $k_2 = 0.8$, and $k_3 = 0.6$), and makes the assignment with minimum $c(s_i, p_j)$.

With this algorithm, Kahng *et al.*, 2002 claimed to achieve up to 10% reduction in border conflicts over the TSP-based approach of Hannenhalli *et al.*, 2002.

The major problem with the epitaxial and the TSP-based algorithm is that they have at least quadratic time complexity and thus are not scalable for the latest million-probe microarrays. According to the experiments of Kahng *et al.*, 2003/1, the TSP approach needed around 32 minutes to produce the layout of a 200 x 200 chip, whereas the epitaxial algorithm needed 74 minutes on average. For a 500 x 500 chip, the TSP took over 30 hours to complete, whereas the epitaxial algorithm did not complete "due to prohibitively large running time or memory requirements" (Kahng *et al.*, 2003/1).

This observation has led to the development of two new algorithms in Kahng *et al.*, 2003/1. The first one, called sliding-window matching (SWM), is not exactly a placement algorithm as it iteratively improves an initial placement that can be constructed by, for instance, TSP and 1-threading. Improvements are achieved by selecting an independent set of spots inside the window and optimally replacing their probes using a minimum-weight perfect matching algorithm. The term independent refers to probes that can be replaced without affecting the border length of the other selected probes.

The other algorithm presented in Kahng *et al.*, 2003/1 is a simple variant of the epitaxial algorithm, called row-epitaxial, with two main differences: spots are filled in a pre-defined order, namely row-by-row, and only probes of a limited list of candidates Q are considered when filling each spot.

The experimental results of Kahng *et al.*, 2003/1 showed that the row-epitaxial is the best placement algorithm in terms of solution quality, achieving up to 9% reduction in border length when compared to the TSP-based approach of Hannenhalli *et al.*, 2002. The SWM is the fastest algorithm in practice.

3.2 Partitioning Algorithms

The ever growing number of probes of the latest microarray chips and the properties of the placement problem naturally suggest the use of partitioning strategies to reduce the running time of the algorithms.

The placement problem can be trivially 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. These smaller sub-problems, when solved, can be combined to produce a final solution. In this way, algorithms with non-linear time or space complexities can be used solve smaller prolem instances to produce the layout of a large chip that otherwise would not be feasible. A partitioning is clearly a compromise in solution quality. However, due to the large number of probes, this compromise can be negligible if the partitioning is able to places similar probes together.

The only partitioning algorithm available in the literature is a simple recursive procedure called centroid-based quadrisection (Kahng *et al.*, 2003/1). It starts by selecting a probe c_1 randomly from the probe set $\mathcal{P}$. Then, it examines all other probes of $\mathcal{P}$ and selects c_2 with maximum $h(c_1, c_2)$, where $h(c_1, c_2)$ is the Hamming distance between the embeddings of c_1 and c_2. Similarly it finds c_3 with maximum $h(c_1, c_3) + h(c_2, c_3)$ and c_4 with maximum $h(c_1, c_4) + h(c_2, c_4) + h(c_3, c_4)$. Probes c_1, c_2, c_3 and c_4 are called centroids. All other probes $p_i \in \mathcal{P}$ are then compared to the centroids and assigned to the sub-set $\mathcal{P}_j$ associated with centroid c_j that has minimum $h(p_i, c_j)$. Each sub-set $\mathcal{P}_j$ is assigned to a sub-region of the chip. The procedure is repeated recursively on each sub-region until a given maximum recursion depth L is reached.

The result of this algorithm is a partitioning of the chip into several sub-regions and an assignment of sub-sets of $\mathcal{P}$ to each sub-region. For the actual placement of the probes in each sub-region, another placement algorithm is needed. For this purpose, Kahng *et al.*, 2003/1 have used the row-epitaxial algorithm.

The results presented in Kahng *et al.*, 2003/1 shows that the running time of the row-epitaxial algorithm drops significantly with increasing L. The time required to place the probes of a 500 x 500 chip, for instance, droped by 69% with $L = 3$ when compared with the time required by the row-epitaxial algorithm alone. It is not clear from their experiments, however, how the choice of L impaired the performance of the row-epitaxial algorithm in terms of solution quality.

4 QUADRATIC ASSIGNMENT PROBLEM

We now explore a different approach to the placement problem based on the quadratic assignment problem (QAP), a classical combinatorial optimization problem introduced by Koopmans and Beckmann, 1957.

The QAP can be formally stated as follows. Given $n \times n$ matrices $F = (f_{ij})$ and $D = (d_{ij})$, find a permutation π of the natural numbers $1, 2, \ldots n$ minimizing

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)} \tag{5}$$

The QAP has been used to model a variety of real-life problems. One of its major applications is to model the facility location problem where n facilities need to be assigned to n locations. In this context, F is called the *flow* matrix where f_{ij} represents the flow of materials from facility i to facility j, which are assumed to have an associated cost. Matrix D is called the *distance* matrix where d_{ij} represents the distance between locations i and j. The permutation π then gives a one-to-one assignment of facilities to locations with minimum cost.

4.1 Quadratic Assignment Formulations

The microarray placement problem discussed in previous sections can be seen as an instance of a QAP. We can draw a parallel with the facility location by viewing the probes as facilities and the spots as locations. The flow matrix then contains the number of conflicts between probe embeddings whereas the distance matrix contains the distance between the spots. The exact contents of F and D depends whether the goal is to minimize border length or conflict index.

In the following formulations, we consider the probes as having a single pre-defined embedding in order to force a one-to-one relationship. A more elaborate formulation would consider all possible embeddings of a probe but, then, it would be necessary to ensure that only one embedding of a probe is assigned to a spot.

4.1.1 Border Length Minimization The QAP formulation for the case of border length minimization is trivial. We set

$$f_{ij} = \frac{h(p_i, p_j)}{2} \tag{6}$$

where $h(p_i, p_j)$ is the Hamming distance between the embeddings of probes p_i and p_j. We need to divide it by two because in equation 5, the conflicts between p_i and p_j appears twice (in f_{ij} and f_{ji}). For the distance matrix, we set

$$d_{ij} = \begin{cases} 1 & \text{if spots } i \text{ and } j \text{ are adjacent,} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

since only conflicts between adjacent spots are relevant for the border length. It is easy to verify that this formulation reflects the definition of border length.

4.1.2 Conflict Index Minimization In case of conflict index minimization, the formulation is slightly more elaborate. Our goal is to design a microarray minimizing the sum of conflict indices over all spots i

$$\sum_{i} \mathcal{C}(i). \tag{8}$$

From the point of view of a spot i, there is a conflict at step k only when i is masked and a close neighbor j is unmasked, in which case we say that there is an *induced conflict* of j onto i, $\mathcal{C}_j(i)$, that can be derived from equation 4 as

$$\mathcal{C}_j(i) := \sum_{k=1}^{\mu} \omega(i,k) \cdot \delta(i,j,k). \tag{9}$$

We can then rewrite equation 4 as

$$\mathcal{C}(i) := \sum_{j} \mathcal{C}_j(i), \tag{10}$$

and our problem stated in equation 8 turns into minimizing

$$\sum_{i} \sum_{j} \mathcal{C}_j(i), \tag{11}$$

where j ranges over all spots that are at most three cells away from i.

Equations 5 and 11 are conveniently similar. Now we just need to set f_{ij} and d_{ij} in such a way that their multiplication results in $\mathcal{C}_j(i)$. The dependence of δ on k is due to the fact that $\delta(i,j,k) = 0$ if spot j is masked at step k. It is thus possible to rewrite equation 9 as

$$\mathcal{C}_j(i) := \left(\sum_{k=1}^{\mu} \omega(i,k) \cdot \phi(j,k) \right) \cdot (d(i,j))^{-2}, \qquad (12)$$

where

$$\phi(j,k) := \begin{cases} 0 & \text{if spot } j \text{ is masked at step } k, \\ 1 & \text{otherwise,} \end{cases} \qquad (13)$$

and $d(i,j)$ is the Euclidean distance between spots i and j as used in the definition of δ (equation 1).

Looking at equation 12, we can already see how f_{ij} and d_{ij} can be set to produce $\mathcal{C}_j(i)$. The latter is easy:

$$d_{ij} = \begin{cases} (d(i,j))^{-2} & \text{if spot } j \text{ is "near" spot } i, \\ 0 & \text{otherwise.} \end{cases} \qquad (14)$$

where "near" means that spot j is at most three cells away from i (this accounts for the differences in range of j in equations 5 and 11).

The only remaining problem is that $\mathcal{C}_j(i)$ is defined in terms of spots i and j, whereas f_{ij} must be defined in terms of probes i and j, independently of which spots they are assigned to. However, this is not really a problem since the dependence of ω and ϕ on the spots is a mere convenience. The exact location of the spots is irrelevant; the embeddings of their probes is what matters. Having said that, we set

$$f_{ij} = \sum_{k=1}^{\mu} \omega'(i,k) \cdot \phi'(j,k), \qquad (15)$$

where

$$\omega'(i,k) := \begin{cases} c \cdot \exp\left(\theta \cdot \lambda'(i,k)\right) & \text{if embedding of } i \\ & \text{is masked at step } k, \\ 0 & \text{otherwise,} \end{cases} \qquad (16)$$

$$\lambda'(i,k) := 1 + \min(b'_{i,k}, \ell'_i - b'_{i,k}), \qquad (17)$$

$$\phi'(j,k) := \begin{cases} 0 & \text{if embedding of } j \text{ is masked at step } k \\ 1 & \text{otherwise,} \end{cases} \qquad (18)$$

c and θ are constants, ℓ'_i is the length of probe i, and $b'_{i,k}$ denotes the number of nucleotides of probe i synthesized up to and including step k.

4.2 QAP Heuristics

In the previous sub-section we showed how the microarray placement problem can be modeled as a quadratic assignment problem. This is interesting because we can now use existing QAP algorithms to design the layout of microarrays minimizing either the sum of border lengths or conflict indices.

The QAP is known to be NP-hard and NP-hard to approximate, and instances of size larger than $n = 20$ are generally considered to be impossible to solve (to optimallity). Fortunately, several heuristics are available including approaches based on tabu search, simulated annealing and genetic algorithms (Çela, 1998).

Another interesting heuristic approach is GRASP (greedy randomized adaptive search procedure), proposed by Feo and Resende, 1995 and used by Li *et al.*, 1994 for solving QAP instances.

We also outline a GRASP variant known as GRASP with path-relinking (Oliveira *et al.*, 2004) that we have used to design the layout of microarray chips.

GRASP ...

GRASP with path-relinking (GRASP-PR)...

5 RESULTS AND DISCUSSION

We now present experimental results of using GRASP with path-relinking (GRASP-PR) for designing the layout of small artificial chips.

Due to the large number of probes on industrial microarrays, it is not feasible to use GRASP-PR (or any other QAP method) to design the layout of an entire microarray chip. However, it is certainly possible to use it on small sub-regions of a chip. This is interesting because we can combine GRASP-PR with a partitioning strategy such as the centroid-based quadrisection described in section 3.2.

We have run GRASP-PR as well as the best know placement algorithm, row-epitaxial (see section 3.1), on several small random chips. For GRASP-PR, we used a C implementation provided by Oliveira *et al.*, 2004 with default parameters: 32 iterations, $\alpha = 0.1$, $\beta = 0.5$, and elite set with size $\mid P \mid= 10$. Their main routine takes three arguments: matrices F and D and the dimension of the problem n (in our case, the number of spots or probes). We generated the matrices using the formulations presented in section 4.1. For the row-epitaxial algorithm we used a C implementation provided by Kahng *et al.*, 2003/1. The running times and the border length of the resulting layouts are shown on table 1. Experiments were conducted on a Sun Fire V1280 server with 12 UltraSparc III+ processors with 900Mhz and 96 Gb of RAM under similar load balances.

Our results show that GRASP-PR consistently produces layouts with lower border lengths than those produced by the row-epitaxial algorithm, although its running time increases at a much higher rate with larger chip sizes. It is also clear that the rate of improvement of GRASP-PR over row-epitaxial is higher on smaller chips. On a 6 x 6 chip GRASP-PR outperforms row-epitaxial by 2.5 percentage points when compared to the initial random layout. On a 10 x 10 chip, this difference drops to 0.6 percentage point.

It is interesting to extrapolate the times shown on table 1 to predict the total time that would be required to design the layout of larger chips, if we were to combine GRASP-PR with a partitioning algorithm. If the partitioning produced

Table 1. Border length of layouts produced by row-epitaxial and GRASP-PR on chips with random probes of length 25 synchronously embedded. Reduction in border conflicts are reported in percentage compared to the initial layout.

		random	row-epitaxial			GRASP-PR		
chip dimension	number of probes	border length	border length	reduction (%)	time (sec.)	border length	reduction (%)	time (sec.)
6 x 6	36	2 242.4	1 952.4	12.93	3.2	1 896.4	15.43	3.5
7 x 7	49							
8 x 8	64							
9 x 9	81							
10 x 10	100	6 684.8	5 562.8	16.78	8.7	5 522.8	17.38	33.5
11 x 11	121							
12 x 12	144							

Border length in each case are averages over a set of five chips; running times are averages over three runs for each chip of a set.

Figure 3. Predicted time to compute the layout of large microarrays using a partitioning algorithm and GRASP-PR on the resulting sub-regions. Each curve represents a chip with the dimension of an existing GeneChip array. The maximum dimension of sub-regions produced by the partitioning is shown on the horizontal axis. The time required for the partitioning itself is ignored.

10 x 10 regions, 13 689 sub-regions would be created from the 1164 x 1164 Affymetrix Human Genome U133 Plus 2.0 GeneChip®. Since each sub-region takes around 30 seconds to compute with GRASP-PR, the total time required for designing such a chip would be a little less than two hours (ignoring the time for the partitioning itself).

If the partitioning produced 12 x 12 regions, 9 409 sub-regions would be created and, since GRASP-PR takes around 3 minutes to compute each of them, the total time would be more than 19 days. This is probably prohibitive, although it is certainly possible to reduce the time of each GRASP-PR execution by running it on faster machines or using a parallel implementation (GRASP is known to be easily parallelized; see Li *et al.*, 1994). Figure 3 shows similar predictions based on our results with varying chip dimensions and partitioning sizes.

We believe that solution quality is more important than the running time of a placement algorithm. Even if an algorithm takes a couple of days to complete, it is time well spent given that commercial microarrays are likely to be produced in large scale. This is specially true when we consider the time required for the whole design process of a microarray chip. Even customer designed chips that usually have a limited number of produced units are likely to benefit from a few extra hours of computing time.

5.1 Future Work

As mentioned in section 3.2, partitioning is a compromise in solution quality in favour of running time. However, it is not clear yet how the choice of the maximum recursion depth in the centroid-based quadrisection, for instance, can undermine the effectiveness of the placement algorithms. At the moment, we are investigating alternative partitioning strategies and evaluating their effects on the quality of the solutions.

We conclude by noting that QAP heuristics such as GRASP-PR could also be used to improve existing layouts if small changes were made introduced. The idea is to run such algorithms iteratively in a sliding-window fashion. Each iteration produces an instance of a QAP whose size equals the size of the window and the QAP heuristic can be executed to check whether conflicts can be reduced with a new layout.

The only problem with this approach is that the QAP heuristic needs to take into account not only the costs due to the iteractions inside the window, but also those due to the spots around the window. Otherwise, it is likely that a reduction in conflicts inside the window is achieved at the expense of an increase on the borders of the window.

A possible change to the QAP heuristics would be to solve a larger QAP instance consisting of the spots inside the window as well as those around it. The difference is that the contents of the spots outside the window would have to be fixed, that is, the elements of the permutation π corresponding to these spots would be always the same.

REFERENCES

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

Çela,E. (1998) *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer, Massachessets, USA.

Chase,P.J. (1976) Subsequence numbers and logarithmic concavity, *Discrete Mathematics*, **16**, 123–140.

Table 2. Average conflict index of layouts produced by GRASP-PR on chips with random probes of length 25 synchronously embedded. Reduction in border conflicts are reported in percentage compared to the initial layout.

		random	GRASP-PR		
chip dimension	number of probes	avg. conf.idx.	avg. conf. idx.	reduction (%)	time (sec.)
6 x 6	36	2 242.4	1 896.4	15.43	3.5
7 x 7	49				
8 x 8	64				
9 x 9	81				
10 x 10	100	6 684.8	5 522.8	17.38	33.5
11 x 11	121				
12 x 12	144				

Average conflict index in each case are averages over a set of five chips; running times are averages over three runs for each chip of a set.

Feldman,W. and Pevzner,P. (1994) Gray code masks for sequencing by hibridization, *Genomics*, **23**, 233–235.

Feo,T.A. and Resende,M.G.C. (1995) Greedy randomized adaptive search procedures, *Journal of Global Optimization*, **6**, 109–133.

Fodor,S., Read,J., Pirrung,M., Stryer,L., Lu,A. and Solas,D. (1991) Light-directed, spatially addressable parallel chemical synthesis, *Science*, **251**, 767–73.

Hannenhalli,S., Hubell,E., Lipshutz,R. and Pevzner,P. (2002) Combinatorial algorithms for design of DNA arrays, *Advances in Biochemical Engineering / Biotechnology*, **77**, 1–19.

Kahng,A.B., Mandoiu,I.I., Pevzner,P.A., Reda,S. and Zelikovsky,A.Z. (2002) Border length minimization in DNA array design, *Proceedings of the Second Workshop on Algorithms in Bioinformatics*.

Kahng,A.B., Mandoiu,I., Pevzner,P., Reda,S. and Zelikovsky,A. (2003-1) Engineering a scalable placement heuristic for DNA probe arrays, *Proceedings of the Seventh Annual International Conference on Computational Molecular Biology*, 148–83.

Kahng, A.B., Mandoiu,I., Reda,S., Xu,X. and Zelikovsky,A. (2003-2), Evaluation of placement techniques for DNA probe array layout, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 262–269.

Koopmans,T.C. and Beckmann,M.J. (1957) Assignment problems and the location of economic activities, *Econometrica*, **25**, 53–76.

Li,Y., Pardalos,P.M. and Resende,M.G.C. (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, Pardalos,P. and Wolkowicz,H. (editors), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **16**, 237–261.

Oliveira,C.A.S., Pardalos,P.M. and Resende,M.G.C. (2004) GRASP with path-relinking for the quadratic assignment problem, AT&T Labs Technical Report.