BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–8

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 a new approach to the design of 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 $t_k \in \{A, C, G, T\}$ 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 the alphabet. 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; hence, 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. All Affymetrix chips that we know of can be asynchronously synthesized in 74 masking steps[1].

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

[1] We understand that Affymetrix uses the same truncated repetition of TGCA to synthesize all of their chips, which suggests that their probe selection only chooses probes that fit into that deposition sequence.

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 (Hannenhalli *et al*., 2002), and define an extended model for evaluating microarray layouts. In Section 3, we briefly review exising placement strategies. In Section 4, we propose a new approach to the design of microarrays based on the quadratic assignment problem (QAP). The results of using a QAP heuritic algorithm, called GRASP, to design small artificial chips are presented in Section 5, where 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 Length 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*., 2003a, 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:

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

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.

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:

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

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

The distance-dependent weighthing function δ is as suggested in Kahng *et al*., 2003a. 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 a mismatched base occurs at the extremities of the formed duplex instead of at its center. The precise 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 at 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 observe a reduction of the conflict index at the expense of an increase in border length, and vice-versa.

a)

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	s	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

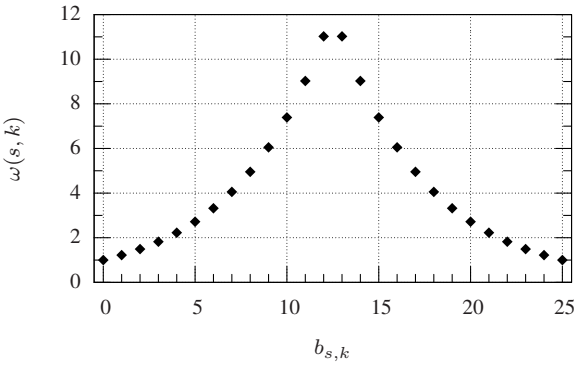

Figure 2. Ranges of values for both δ and ω on a typical Affymetrix chip where probes of length $\ell = 25$ are synthesized in $\mu = 74$ masking steps. a) Distance-dependent weighthing function $\delta(s, s', k)$ for a spot s (shown in the center) and all close neighbors s', assuming that s is masked and s' is unmasked at step k. b) Position-dependent weights $\omega(s, k)$ at each value of $b_{s,k}$, assumming that spot s is masked at step k.

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 two-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, employing 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}. For each possible assignment of p_j to s_i, it computes a cost $c(s_i, p_j) := k_{n_i} H_{ij}/n$, 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 cost. With this algorithm, they 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 their experiments, 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*., 2002).

This observation has led to the development of two new algorithms by Kahng *et al*., 2003a. 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 described by Kahng *et al*., 2003a, is a 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.

Their experimental results 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, immediately constitute a final solution. In this way, algorithms with non-linear time or space complexities can be used compute the layout of larger chips 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 small, specially if the partitioning is able to place similar probes together.

The only partitioning algorithm available in the literature is the centroid-based quadrisection (Kahng *et al*., 2003b). It is a recursive procedure that works as follows. First, it randomly selects a probe c_1 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 c_j with 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 algorithm is needed. For this purpose, Kahng *et al*., 2003b, have used the row-epitaxial algorithm.

The results presented in Kahng *et al*., 2003b, show 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 without any partitioning.

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 since they have restricted their experiments to $L \leq 3$. Moreover, there is no clear trend toward reduction or increase in border length as L varies from 0 to 3.

4 QUADRATIC ASSIGNMENT PROBLEM

We now explore a different approach to the design of microarrays 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 must be assigned to n locations. In this scenario, F is called the flow matrix as f_{ij} represents the flow of materials from facility i to facility j, which are assumed to have an associated cost proportional to the distance between the facilities. Matrix D is called the distance matrix as d_{ij} gives the distance between locations i and j. The permutation π defines 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 use the facility location example 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 gives 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 predefined 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 divide it by since, in (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, that is

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

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 (4) as

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

and our objective function (8) turns into

$$\sum_i \sum_j \mathcal{C}_j(i), \tag{11}$$

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

Note the similarities between (5) and (11). Now we 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

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

where

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

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

Equation (12) suggests how f_{ij} and d_{ij} can be set to produce $\mathcal{C}_j(i)$. The latter is trivial:

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

where "near" means that spot j is at most three cells away from i. This definition also accounts for the difference in range of j in (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, the dependence of ω and ϕ on the spots is a mere convenience since the exact location of the spots is irrelevant. The embeddings of their probes is what matters. Hence, we set

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

where

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

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

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

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. 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 (see Çela, 1998, for a survey).

We now briefly describe GRASP, a heuristic proposed by Feo and Resende, 1995, which was first used for solving the QAP by Li *et al.*, 1994. We also outline a GRASP variant known as GRASP with path-relinking (Oliveira *et al.*, 2004) that we have used in the design of microarray chips. In the description that follows we use the terms of the facility location problem: f_{ij} is the flow between facilities i and j, d_{kl} is the distance between locations i and j.

GRASP is an acronym for Greedy Randomized Adaptive Search Procedure. It is an iterative process comprised of two phases: a construction phase where it builds a random feasible solution, and a local search phase where it seeks a local optimum in the neighborhood of that solution.

Before the first iteration, GRASP sorts the (n^2-n) elements of the distance matrix in increasing order, keeping the first $\lfloor \beta(n^2-n) \rfloor$:

$$d_{k_1 l_1} \leq d_{k_2 l_2} \leq \cdots \leq d_{k_{\lfloor \beta(n^2-n) \rfloor} l_{\lfloor \beta(n^2-n) \rfloor}},$$

where $0 < \beta < 1$ is a restriction parameter. It also sorts the (n^2-n) elements of the flow matrix in decreasing order, again

keeping the first $\lfloor \beta(n^2 - n) \rfloor$:

$$f_{i_1 j_1} \geq f_{i_2 j_2} \geq \cdots \geq f_{i_{\lfloor \beta(n^2-n) \rfloor} j_{\lfloor \beta(n^2-n) \rfloor}}.$$

Finally, it sorts the costs of assignments:

$$d_{k_1 l_1} f_{i_1 j_1}, d_{k_2 l_2} f_{i_2 j_2}, \ldots, d_{k_{\lfloor \beta(n^2-n) \rfloor} l_{\lfloor \beta(n^2-n) \rfloor}} f_{i_{\lfloor \beta(n^2-n) \rfloor} j_{\lfloor \beta(n^2-n) \rfloor}},$$

keeping the $\lfloor \alpha\beta(n^2 - n) \rfloor$ smallest elements, where $0 < \alpha < 1$ is another restriction parameter. Note that $d_{k_1 l_1} f_{i_1 j_1}$ gives the cost of simultaneously assigning facility i_1 to location k_1 and facility j_1 to location l_1.

The construction phase of GRASP consists of two stages. In the first stage, the algorithm makes a simultaneous assignment selected at random among those with the $\lfloor \alpha\beta(n^2 - n) \rfloor$ smallest costs.

In the second stage of the construction phase, GRASP build a feasible solution by making a series of greedy assignments as follows. First, it computes the costs of all m possible assignments with respect to assignments already made. Then, it randomly selects one assignment among those with $\lfloor \alpha m \rfloor$ smallest costs.

In the local search phase, GRASP searches for a local optimum in the neighborhood of the constructed solution. Different search strategies and different definitions of the neighborhood can be used. One possible approach is to check every possible swap of assignments and make those which improve the current solution until no futher improvements can be made.

GRASP repeats the construction and local search phases for a given number of times, keeping the best solution found. Each iteration is indenpendent in the sense that every construction phase builds a new solution from scratch. The best solutions are kept, but GRASP takes no advantage of the knowledge gained in previously iterations to build or impove a new solution. This is exactly where the concept of path-relinking comes into play.

GRASP with path-relinking is an extension of the basic GRASP that uses an elite set P to store the best solutions found. It also incorporates a third phase that consists of choosing at random one elite solution $q \in P$ to be combined with the last solution p produced after the local search phase.

Solutions p and q are combined as follows. For every location $k = 1, \ldots, n$, the path-relinking algorithm attempts to exchange facility p_k assigned to location k in solution p with facility q_k assigned to location k in the elite solution. In order to keep the solution p feasible, it actually exchanges p_k with p_l, where $p_l = q_k$. This exchange is performed only if it results in a better solution. The result of the path-relinking phase is a solution r that is as good as p and q.

For more detail on GRASP with path-relinking, we refer the interested reader to Oliveira *et al.*, 2004.

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 $\mid P \mid = 10$. The 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.*, 2003a. The running times and the border length of the resulting layouts are shown on Table 1.

Our results show that GRASP-PR produces layouts with lower border lengths than the row-epitaxial algorithm for the smaller chips. For 6 x 6 chips GRASP-PR outperforms row-epitaxial by 2.5 percentage points on average when compared to the initial random layout. This is a promising result given that GRASP-PR is a general QAP heuristic. For 10 x 10 chips, however, this difference drops to 0.6 percentage point. The row-epitaxial generates better layouts for 11 x 11 or larger chips. This is probably because a larger chip means that there are more probes to choose from when filling the spots.

In terms of running time, the row-epitaxial is faster and shows little variation as the number of probes grows. In contrast, the time required to compute a layout with GRASP-PR increases at a fast rate.

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 commercial microarrays, if we were to combine GRASP-PR with a partitioning algorithm. If the partitioning produced 6 x 6 regions, 37 636 sub-regions would be created from the 1164 x 1164 Affymetrix Human Genome U133 Plus 2.0 GeneChip®, one of the largest Affymetrix chips. Since each sub-region takes around 3 seconds to compute with GRASP-PR, the total time required for designing such a chip would be a little over 31 hours (ignoring the time for the partitioning itself).

If the partitioning produced 12 x 12 regions, 9 409 sub-regions would be created and, at 2.4 minutes each, the total time would be more than 16 days. This is probably prohibitive, although it is certainly possible to reduce the time of each

Table 1. Border length of random chips compared with the layouts produced by row-epitaxial and GRASP with path-relinking. Reductions in border length are reported in percentages compared to the initial layout. Chips contain 25-base long probes uniformly generated and synchronously embedded. Border length and running times are averages over a set of five chips.

		Random	Row-epitaxial			GRASP with path-relinking		
Chip dimension	Number of probes	Border length	Border length	Reduction (%)	Time (sec.)	Border length	Reduction (%)	Time (sec.)
6 x 6	36	2 239.20	1 942.40	13.25	0.010	1 882.40	15.93	2.991
7 x 7	49	3 115.20	2 675.60	14.11	0.020	2 621.60	15.84	7.074
8 x 8	64	4 202.40	3 514.00	16.38	0.024	3 481.20	17.16	13.568
9 x 9	81	5 420.00	4 471.20	17.51	0.028	4 460.40	17.70	28.076
10 x 10	100	6 740.40	5 556.20	17.57	0.034	5 536.00	17.87	55.430
11 x 11	121	8 212.00	6 726.80	18.09	0.040	6 734.80	17.99	84.659
12 x 12	144	9 872.00	7 975.20	19.21	0.044	8 038.00	18.58	148.196

Experiments were conducted on a Sun Fire V1280 server with 900Mhz UltraSparc III+ processors and 96 Gb of RAM under similar load balances.

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.

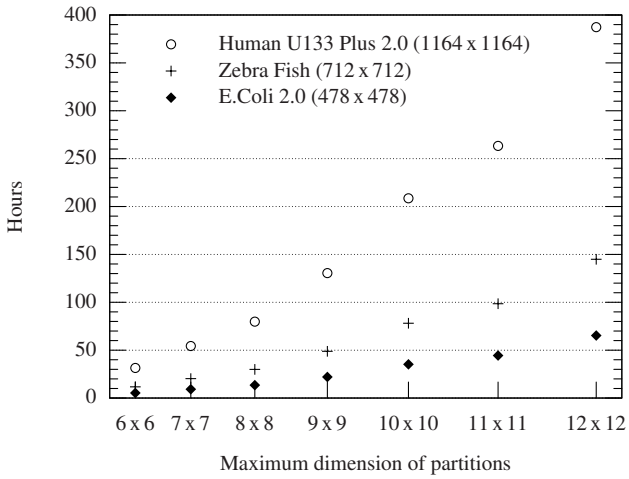

Figure 3. Predicted running times required to design selected Affymetrix GeneChip arrays using GRASP-PR and a partitioning algorithm with varying degrees of partitioning (based on data shown in Table 1). The dimensions of the chips are shown in parentheses. The time required for the partitioning itself is ignored.

5.1 Future Work

As mentioned earlier, 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 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 introduced. The idea is that we can run such algorithms iteratively in a sliding-window fashion, where each iteration produces an instance of a QAP whose size equals the size of the window. The QAP heuristic can then be executed to check whether a different arrangement of the probes inside the window can reduce the conflicts.

The only problem with this approach is that the QAP heuristic also needs to take into account the conflicts due to the spots around the window. Otherwise, the new layout may increase the conflicts on the borders of the window.

A possible solution to this problem is to solve a larger QAP instance consisting of the spots inside the window as well as those around it. The spots outside the window obviously must remain unchanged, and that can be done by fixing the corresponding elements of the permutation π. Note also that there is no need to compute d_{ij} if spots i and j are both outside the window, nor f_{ij} if probes i and j are assigned to spots outside the window.

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.

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. In *Proceedings of the Second Workshop on Algorithms in Bioinformatics*.

Kahng,A.B., Mandoiu,I., Pevzner,P., Reda,S. and Zelikovsky,A. (2003a) Engineering a scalable placement heuristic for DNA probe arrays. In *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. (2003b), Evaluation of placement techniques for DNA probe array layout. In *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 Pardalos,P. and Wolkowicz,H. (eds.), *Quadratic Assignment and Related Problems*, 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. In Ribeiro,C.C. and Martins,S.L. (eds.), *Efficient and Experimental Algorithms*, Lecture Notes in Computer Science, **3059**, 356–368, Springer-Verlag.