BIOINFORMATICS

Vol. 00 no. 00 2006
Pages 1–8

Modeling Microarray Layout as a Quadratic Assignment Problem

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

[a] *Graduiertenkolleg Bioinformatik, Bielefeld University, Germany,*
[b] *International NRW Graduate School in Bioinformatics and Genome Research, Bielefeld University, Germany,*
[c] *Algorithms and Statistics for Systems Biology group, Genome Informatics, Technische Fakultät, Bielefeld University, D-33594 Bielefeld, 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. 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.

Results: We propose a new model called *conflict index* for evaluating microarray layouts, and we show that the probe placement problem is an instance of the *quadratic assignment problem* (QAP), which opens up the way for using QAP heuristics. We use a existing heuristic 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.

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

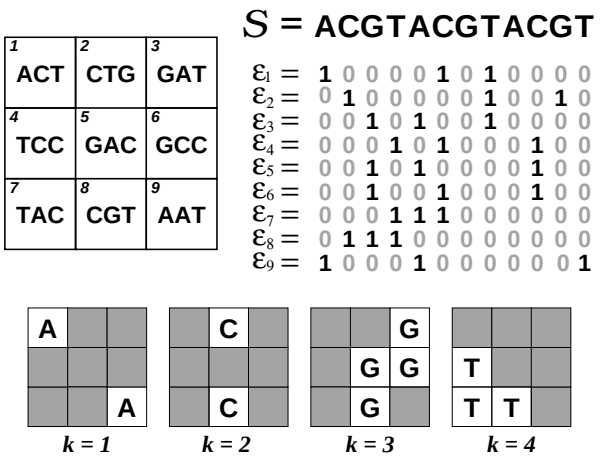

Figure 1. Synthesis of a hypothetical 3 x 3 chip. On the top left, the chip layout and its 3-base-long probe sequences. On the top right, the deposition sequence and the probe embeddings. On the bottom, the first four resulting photolithographic masks.

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 $p_i \in \mathcal{P}$ (Rahmann, 2003).

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 synthesized 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].

Because of 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. This problem appears to be hard (although the authors are not aware of an NP-hardness proof, and our QAP formulation has several special properties) because of 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 referred 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 existing 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 heuristic 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. Finally, we discuss how this approach can be used to design larger microarrays (Section 6).

2 MODELING

2.1 Border Length

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* $\mathcal{B}_k$ 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. For example, in Figure 1, the four masks shown 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.

2.2 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-spot measure instead of a per-mask measure. Additionally, 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.

This motivates the following definition of the *conflict index* $\mathcal{C}(s)$ of a spot s whose probe of length ℓ_s is synthesized in μ masking steps. 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'. This form of weighting function is the same as suggested in Kahng *et al*. (2003a). Note that δ is a "closeness" measure between s and s' only if the spot in the neighboring spot s' is not masked (and thus creates the potential of illumination at s). To restrict the number of neighbors that need to be considered, we restrict the support of $\delta(s, s', \cdot)$ to those $s' \neq s$ that are in a 7×7 grid centered around s (see Figure 2 top).

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

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

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

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

is the distance of the nucleotide synthesized in step k (if any) from the start or end of the probe: $b_{s,k}$ denotes the number of nucleotides synthesized at spot s up to and including step k

[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.

and ℓ_s is the probe length (see Figure 2 bottom). We set the constants c and θ as follows:

$$\theta = \frac{5}{\ell_s}; \qquad c = \frac{1}{\exp\theta}.$$

It is generally agreed that the chances of a successful hybridization 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 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.

We now define the conflict index of a spot s as

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

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

Finally, we note the following relation between conflict indices and border lengths: Define $\delta(s,s',k) := 1$ if s' is a direct neighbor of s and is unmasked in step k, and $:= 0$ otherwise. Also define $\omega(s,k) := 1$ if s is masked in step k, and $:= 0$ otherwise. Then $\sum_s \mathcal{C}(s) = 2\sum_{k=1}^{\mu} \mathcal{B}_k$, as each border conflict is counted twice: for s' and s.

This shows that total border length and total conflict are correlated: 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.

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

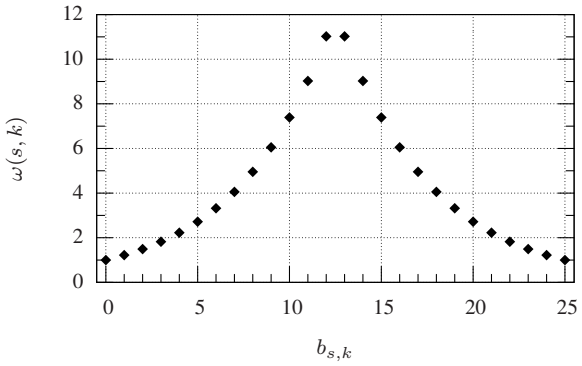

Figure 2. Ranges of values for both δ and ω on a typical Affymetrix chip where probes of length $\ell = 25$ are synthesized in $\mu = 74$ masking steps. Top: Distance-dependent weighting function $\delta(s,s',k)$ for a spot s (shown in the center) and all close neighbors s', assuming that s' is unmasked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

3 PREVIOUS WORK

We briefly review existing layout techniques, distinguishing between placement algorithms and partitioning algorithms. Post-placement optimizations such as the Chessboard (Kahng *et al*., 2002) are not covered.

3.1 Placement Algorithms

Hannenhalli *et al*. (2002) were the first to consider the border length problem on large oligonucleotide arrays of arbitrary probes. They reported 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 probes. A TSP tour is approximated, resulting in consecutive probes in the tour being likely to be similar. The TSP tour is finally *threaded* on the array in a row-by-row fashion. Hannenhalli *et al*. (2002) suggested 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) propose 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 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).

Both the TSP and the epitaxial approach do not scale well to large chips. Another 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 large-scale placement algorithm, achieving up to 9% reduction in border length when compared to the TSP-based approach of Hannenhalli *et al*. (2002).

3.2 Partitioning Algorithms

The ever growing number of probes on the latest microarrays 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 partitioned by dividing the set of probes into smaller subsets, 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 recursion depth 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.

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 stated as follows. Given $n \times n$ real-valued matrices $F = (f_{ij}) \geq 0$ and $D = (d_{ij}) \geq 0$, find a permutation π of $\{1, 2, \ldots n\}$ such that

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

The attribute *quadratic* stems from the fact that the target function can be written with n^2 binary indicator variables $x_{ik} \in \{0, 1\}$, where $x_{ik} := 1$ if and only if $k = \pi(i)$. The objective (5) then becomes

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot \sum_{k=1}^{n} \sum_{l=1}^{n} d_{kl} \cdot x_{ik} \cdot x_{jl} \rightarrow \min \tag{6}$$

such that $\sum_k x_{ik} = 1$ for all i, $\sum_i x_{ik} = 1$ for all k and $x_{ik} \in \{0, 1\}$ for all (i, k). The objective function is a quadratic form in x.

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. One unit of flow is assumed to have an associated cost proportional to the distance between the facilities. Matrix D is called the distance matrix as d_{kl} gives the distance between locations k and l. The optimal permutation π defines a one-to-one assignment of facilities to locations with minimum cost.

4.1 QAP Formulation of Probe Placement

The probe placement problem can be seen as an instance of the QAP. To formulate it, we use the facility location example by viewing the probes as locations and the spots as facilities, i.e., the spots are assigned to the probes. In this case the flow matrix F contains the "closeness" values between spots, and the distance matrix D contains the (weighted) Hamming distances between probe embeddings. We first give the general formulation for conflict index; the case of border length minimization is obtained by using the particular weight functions given at the end of Section 2.

Obviously, the QAP requires a one-to-one correspondence between spots and probes. In a realistic setting, we may have more spots available than probes to place. This does not cause problems: Below we show that we simply need to add enough "empty" probes and define their weight functions appropriately.

Perhaps more severely, we assume that all probes have 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 becomes necessary to ensure that only one embedding of a probe is assigned to a spot. This still leads to a quadratic integer programming problem, albeit with slightly different side conditions than for (6).

Our goal is to design a microarray minimizing the sum of conflict indices over all spots i, i.e., $\sum_i \mathcal{C}(i) \rightarrow \min$.

Defining f_{ij}. The "flow" between spots i and j depends on their Euclidean distance $d(i, j)$ on the array; in accordance with the conflict index model, we set

$$f_{ij} := \begin{cases} (d(i,j))^{-2} & \text{if spot } j \text{ is "near" spot } i, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where "near" means that spot j is at most three cells away from i. Note that most of the flow values on large arrays are zero. For Border Length Minimization, the case is even simpler: We set $f_{ij} := 1$ if spots i and j are direct neighbors, and $f_{ij} := 0$ otherwise.

Defining d_{kl}. The "distance" between probes k and l depends on the (weighted) Hamming distance of their embeddings. To account for possible "empty" probes to fill up surplus spots, we set $d_{kl} := 0$ if k or l or both refer to an empty probe (i.e., empty probes never contribute to the target function, we do not mind if nucleotides are erroneously synthesized on spots assigned to empty probes).

For real probes, we set

$$d_{kl} := \sum_{t=1}^{\mu} d_{klt},$$

where we now use t as the synthesis step index (the k used in Section 2 is now a probe index). The number d_{klt} is the potential contribution of probe l's embedding to the failure risk of probe k in the t-th synthesis step. According to the conflict index model,

$$d_{klt} = \begin{cases} c \cdot \exp(\theta \cdot \lambda'(k,t)) & \text{if } k \text{ masked, } l \text{ unmasked} \\ & \text{in step } t, \\ 0 & \text{otherwise.} \end{cases}$$

Because of the change in notation and reference to probes instead of spots, we repeat that $\lambda'(k,t) = 1 + \min(b'_{k,t}, \ell'_k - b'_{k,t})$, where ℓ'_k is the length of p_k and $b'_{k,t}$ is the number of nucleotides of p_k synthesized up to and including step t.

In the special case of the Border Length Minimization Problem, where $\theta = 0$ and $c = 1/2$, we obtain that $d_{kl} + d_{lk} = H_{kl} = H_{lk}$, where H_{kl} denotes the Hamming distance between the embeddings of probes k and l.

Proof of correctness. It now follows that for a given assignment (permutation) π, we have $f_{ij} d_{\pi(i)\pi(j)} = \sum_{t=1}^{\mu} \delta(i,j,t) \cdot \omega(i,t)$ in the notation of Section 2. Therefore the objective function (5) becomes

$$\begin{aligned} & \sum_i \sum_j f_{ij} \cdot d_{\pi(i)\pi(j)} \\ = \; & \sum_i \sum_j \left(\sum_{t=1}^{\mu} \delta(i,j,t) \cdot \omega(i,t) \right) \\ = \; & \sum_i \sum_{t=1}^{\mu} \left(\omega(i,t) \cdot \sum_j \delta(i,j,t) \right) \\ = \; & \sum_i \mathcal{C}_i, \end{aligned}$$

and indeed equals the total conflict index with our definitions of (f_{ij}) and (d_{kl}).

Remark. Note that it is technically possible to switch the definitions of $F = (f_{ij})$ and $D = (d_{kl})$, i.e., to assign probes to spots, without modifying the problem formulation, but this would lead to high distance value for neighboring spots many zero distance values for independent spots, a somewhat counterintuitive model. Also, QAP heuristics tend to find pairs objects with large flow values and place them close to each other initially. Therefore, the way of modeling F and D is significant.

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 optimality). 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 k and l.

GRASP is an acronym for Greedy Randomized Adaptive Search Procedure. It is a 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. GRASP performs a number of independent iterations of these two phases and outputs the best solution found.

Before the first iteration, GRASP sorts the $(n^2 - n)$ elements of the distance matrix in increasing order, keeping the first $N := \lfloor \beta(n^2 - n) \rfloor$, where $0 < \beta < 1$ is a restriction parameter.

$$d_{k_1 l_1} \leq d_{k_2 l_2} \leq \cdots \leq d_{k_N l_N},$$

It also sorts the $(n^2 - n)$ elements of the flow matrix in decreasing order, again keeping the first N:

$$f_{i_1 j_1} \geq f_{i_2 j_2} \geq \cdots \geq f_{i_N j_N}.$$

Finally, it sorts the costs of assigning initial pairs of facilities to pairs of locations: The cost of initially assigning facility i_q to location k_q and facility j_q to location l_q for some $q \in \{1, \ldots, N\}$ is $d_{k_q l_q} f_{i_q j_q}$. GRASP sorts the vector

$$(d_{k_1 l_1} f_{i_1 j_1},\ d_{k_2 l_2} f_{i_2 j_2},\ \ldots,\ d_{k_N, l_N} f_{i_N j_N}),$$

keeping the $\lfloor \alpha N \rfloor$ smallest elements, where $0 < \alpha < 1$ is another restriction parameter.

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 builds a feasible solution by making a series of greedy assignments as follows. First, it computes the costs of all m remaining 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 further 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 independent 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 improve 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 details on GRASP with path-relinking, we refer to (Oliveira *et al*., 2004).

5 RESULTS

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

Because of 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 known 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 $|P| = 10$. 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 in 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.

6 DISCUSSION

We have identified the probe placement or microarray layout problem with general distance-dependent and position-dependent weights as a (specially structured) quadratic assignment problem. QAPs are notoriously hard to solve, and currently known exact methods start to take prohibitively long already for slightly more than 20 objects, i.e., we barely could solve the problem for 5×5 arrays. However, the literature on QAP heuristics is quite rich, as many problems in operations research can be modeled as QAPs. Here we used one such heuristic to identify the potential of the probe-placement-QAP-relation.

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,

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.

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 quantities. 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.

Future Work. As mentioned earlier, partitioning is a compromise in solution quality in favor 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.

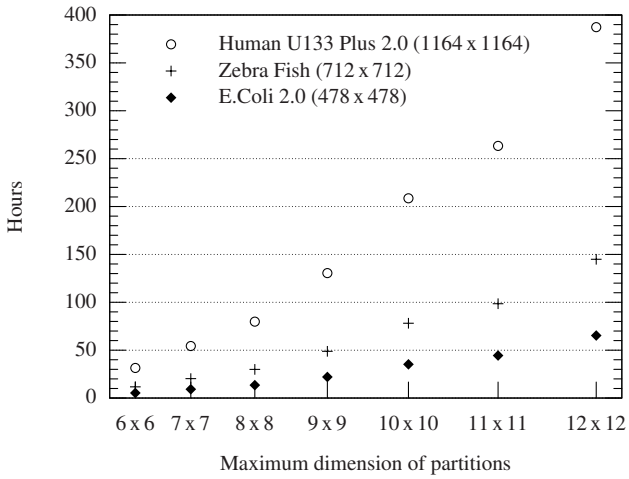

Figure 3. Predicted running times required to design selected Affymetrix GeneChip arrays using GRASP-PR and a partitioning algorithm with varying degrees of partitioning (based on data shown in Table 1). The dimensions of the chips are shown in parentheses. The time required for the partitioning itself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

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 f_{ij} if spots i and j are both outside the window, nor d_{kl} if probes k and l 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.

Rahmann,S. (2003) 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.