Microarray Layout as Quadratic Assignment Problem

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

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

Abstract: The production of commercial DNA microarrays is based on a light-directed chemical synthesis driven by a set of masks or micromirror arrays. Because of the natural properties of light and the ever shrinking feature sizes, the arrangement of the probes on the chip and the order in which their nucleotides are synthesized play an important role on the quality of the final product. We propose a new model called *conflict index* for evaluating 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 an 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.

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 [3].

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

In general, a probe can be *embedded* within $\mathcal{S}$ in several ways. An embedding of p_k is a T-tuple $\varepsilon_k = (e_{k,1}, e_{k,2}, ...e_{k,T})$ in which $e_{k,t} = 1$ if probe p_k receives nucleotide $\mathcal{S}_t$ (at step t), or 0 otherwise (Figure 1). The deposition sequence is often taken as a repeated permutation of the alphabet, mainly because of its regular structure and because such sequences maximize the number of distinct subsequences.

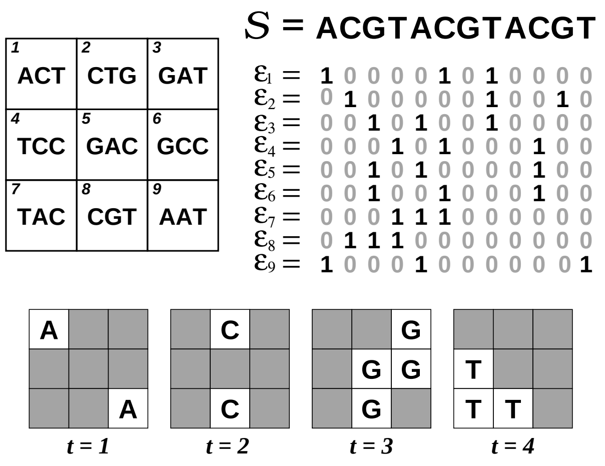
S = ACGTACGTACGT
1 ACT
2 CTG
3 GAT
4 TCC
5 GAC
6 GCC
7 TAC
8 CGT
9 AAT
ε1 = 1 0 0 0 0 1 0 1 0 0 0 0
ε2 = 0 1 0 0 0 0 0 1 0 0 1 0
ε3 = 0 0 1 0 1 0 0 1 0 0 0 0
ε4 = 0 0 0 1 0 1 0 0 0 1 0 0
ε5 = 0 0 1 0 1 0 0 0 0 1 0 0
ε6 = 0 0 1 0 0 1 0 0 0 1 0 0
ε7 = 0 0 0 1 1 1 0 0 0 0 0 0
ε8 = 0 1 1 1 0 0 0 0 0 0 0 0
ε9 = 1 0 0 0 1 0 0 0 0 0 0 1
t = 1
t = 2
t = 3
t = 4

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

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

Because of diffraction of light or internal reflection, untargeted spots can be accidentally activated in a certain masking step, producing unpredicted probes that can compromise the results of an experiment. This problem is more likely to occur near the borders between masked and unmasked spots [3]; this observation has given rise to the term *border conflict*.

We are interested in finding an *arrangement* of the probes on the chip together with *embeddings* in such a way that the chances of unintended illumination during mask exposure steps are minimized. The problem appears to be hard because of the exponential number of possible arrangements, although we are not aware of an NP-hardness proof (and our QAP formulation has several special properties). Optimal solutions are thus unlikely to be found even for small chips and even if we assume that all probes have a single predefined embedding.

If we consider all possible embeddings (up to several million for a typical Affymetrix probe), the problem is even harder. For this reason, the problem has been traditionally tackled in two phases. First, an initial embedding of the probes is fixed and an arrangement of these embeddings on the chip with minimum border conflicts is sought. This is usually referred to as the *placement*. Second, a *post-placement* optimization phase re-embeds the probes considering 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 [4], and define an extended model for evaluating microarray layouts. In Section 3, we briefly review existing placement strategies. In Section 4, we present a new formulation of the microarray placement problem 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 compare its performance with the best known placement algorithm and discuss how this approach can be used to design and improve larger microarrays.

2 Modeling

Border length. Hannenhalli and co-workers [4] were the first to give a formal definition of the problem of unintended illumination in the production of microarrays. They formulated the *Border Length Minimization Problem*, which aims at finding an arrangement of the probes together with their embeddings in such a way the number of border conflicts during mask exposure steps is minimal.

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

Conflict Index. The border length of an individual mask measures the quality of that mask. We are more interested in estimating the risk of synthesizing a faulty probe at a given spot, that is, we need a per-probe measure instead of a per-mask measure. Additionally, the definition of border length does not take into account two important practical considerations [6]: a) stray light might activate not only adjacent neighbors but also probes that lie as far as three cells away from the targeted spot; b) imperfections produced in the middle of a probe are more harmful than in its extremities.

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

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

where $d(p, p')$ is the Euclidean distance between the spots of p and p'. This form of weighting function is the same as suggested in [6]. Note that δ is a "closeness" measure between p and p' only if p' is not masked (and thus creates the potential of illumination at p). To limit the number of neighbors that need to be considered, we restrict the support of $\delta(p, p', \cdot)$ to those $p' \neq p$ that are in a 7×7 grid centered around p (see Figure 2 left).

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

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

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

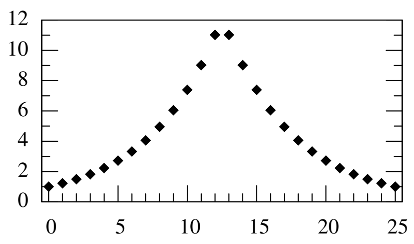

Figure 2: Ranges of values for both δ and ω on a typical Affymetrix chip where probes of length 25 are synthesized in 74 masking steps. Left: approximate values of the distance-dependent weighting function $\delta(p, p', t)$ for a spot containing probe p (shown in the center) and close neighbors p',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

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

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

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

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

We now define the conflict index of a probe p as

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

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

Note the following relation between conflict index and border length: Define $\delta(p, p', t) := 1$ if p' is a direct neighbor of p and is unmasked in step t, and $:= 0$ otherwise. Define $\omega(p, t) := 1$ if p is masked in step t, and $:= 0$ otherwise. Then $\sum_p \mathcal{C}(p) = 2 \sum_{t=1}^{T} \mathcal{B}_t$, as each border conflict is counted twice, once for p' and once for p. Therefore border length and total conflict are equivalent for this particular choice of δ and ω. For our choice (1) and (2), they are still correlated: a good layout has both low border lengths and low conflict indices.

3 Review of Placement and Partitioning Algorithms

Placement Algorithms. All methods mentioned here assume fixed (synchronous, left-most, or otherwise pre-computed) embeddings. Post-placement optimizations such as the Chessboard [5] exist but separate the embedding problem from the placement problem.

The border length problem was first formally addressed in [4]. The article reports that the first Affymetrix chips were designed using a heuristic for the traveling salesman problem (TSP). The idea consists of building a weighted graph with nodes representing probes, and edges containing the Hamming distance between the probe sequences. A TSP tour is approximated, resulting in consecutive probes in the tour being likely similar. The TSP tour is then *threaded* on the array in a row-by-row fashion. A different threading of the TSP tour, called *1-threading*, is suggested to achieve up to 20% reduction in border length [4].

A different strategy called *Epitaxial* placement [5] places a random probe in the center of the array and continues to insert probes in spots adjacent to already filled spots. Priority is given to spots with the largest numbers of filled neighbors. At each iteraction, it examines all non-filled spots and finds a non-assigned probe with minimum sum of Hamming distances to the neighboring probes, employing a greedy heuristic to select the next spot. A further 10% reduction in border conflict over TSP + 1-threading is claimed.

Both the Epitaxial algorithm and the TSP approach do not scale well to large chips. For this reason, [6] proposes a simpler variant of the Epitaxial algorithm, called *Row-epitaxial*, with two main differences: spots are filled in a pre-defined order, namely from top to bottom, left to right, and only probes of a limited list of candidates are considered when filling each spot. Experiments show that Row-epitaxial is the best large-scale placement algorithm, achieving up to 9% reduction in border length over the TSP + 1-threading.

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

The first known partitioning algorithm is called Centroid-based Quadrisection [7]. It starts by randomly selecting a probe $c_1 \in \mathcal{P}$. Then, it selects another probe c_2 maximizing $h(c_1, c_2)$, the Hamming distance between their embeddings. Similarly, it selects c_3 and c_4 maximizing the sum of Hamming distance between these four probes, which are called centroids. All other probes $p \in \mathcal{P}$ are then compared to the centroids and assigned to the sub-set $\mathcal{P}_k$ associated with c_k with minimum $h(p, c_k)$. The chip is divided into four quadrants, each being assigned to a sub-set $\mathcal{P}_k$. The procedure is repeated recursively on each quadrant until a given recursion depth is reached. In the end, the Row-epitaxial algorithm is used to produce the placement of the probes in each final sub-region.

We recently developed an approach that, for the first time, combines the partitioning of the chip with the embedding of the probes [1]. Our algorithm, called Pivot Partitioning, achieves up to 6% reduction in conflicts when compared to the best known algorithms.

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 that can be stated as follows. Given $n \times n$ real-valued matrices $F = (f_{ij}) \geq 0$ and $D = (d_{kl}) \geq 0$, 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 $\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot \sum_{k=1}^{n} \sum_{l=1}^{n} d_{kl} \cdot x_{ik} \cdot x_{jl} \rightarrow \min$, 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.

QAP Formulation of Probe Placement. The probe placement problem can be seen as an instance of the QAP, where we want to find a one-to-one correspondence between spots and probes. In a realistic setting, we may have more spots available than probes to place. Below we show that this does not cause problems as we can 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.

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

The "flow" f_{ij} 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

$$f_{ij} := \begin{cases} (d(i,j))^{-2} & \text{if spot } j \text{ is "near" spot } i, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

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.

The "distance" d_{kl} 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

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

where d_{klt} is the potential contribution of probe l's embedding to the failure risk of probe p_k in the t-th synthesis step. According to the conflict index model,

$$d_{klt} = \begin{cases} c \cdot \exp(\theta \cdot \lambda(p_k, t)) & \text{if } p_k \text{ is masked and } p_l \text{ unmasked in step } t, \\ 0 & \text{otherwise.} \end{cases}$$

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 p_k and p_l.

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

$$\begin{aligned} \sum_i \sum_j f_{ij} \cdot d_{\pi(i)\pi(j)} &= \sum_i \sum_j \Big(\sum_{t=1}^{T} \delta(p_{\pi(i)}, p_{\pi(j)}, t) \cdot \omega(p_{\pi(i)}, t) \Big) \\ = \quad \sum_i \sum_{t=1}^{T} \Big(\omega(p_{\pi(i)}, t) \cdot \sum_j \delta(p_{\pi(i)}, p_{\pi(j)}, t) \Big) &= \sum_i \mathcal{C}(\pi(i)) = \sum_k \mathcal{C}(k), \end{aligned}$$

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

QAP Heuristics. We have shown how the microarray placement problem can be modeled as a quadratic assignment problem. The QAP is known to be NP-hard and particularly hard to solve in practice. Instances of size larger than $n = 20$ are generally considered to be impossible to solve to optimality. Nevertheless, our formulation is of interest because we can now use existing QAP heuristics (see [2] for a survey) to design the layout of microarrays minimizing either the sum of border lengths or conflict indices.

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 random layout.

		Random	Row-epitaxial			GRASP with path-relinking		
Chip dimension	Number of probes	Border length	Border length	Reduction (%)	Time (sec.)	Border length	Reduction (%)	Time (sec.)
6×6	36	1 989.20	1 714.60	13.80	0.01	1 672.20	15.94	2.73
7×7	49	2 783.20	2 354.60	15.40	0.02	2 332.60	16.19	6.43
8×8	64	3 721.20	3 123.80	16.05	0.03	3 099.13	16.72	12.49
9×9	81	4 762.00	3 974.80	16.53	0.05	3 967.20	16.69	25.96
10×10	100	5 985.20	4 895.60	18.20	0.06	4 911.40	17.94	47.57
11×11	121	7 288.40	5 954.40	18.30	0.10	5 990.73	17.80	87.48
12×12	144	8 714.00	7 086.20	18.68	0.11	7 159.80	17.84	152.42

As an example, we use a QAP heuristics called GRASP [8] (Greedy Randomized Adaptive Search Procedure), and an improved version called GRASP with path-relinking [9]. GRASP is comprised of two phases: a construction phase where a random feasible solution is built, and a local search phase where a local optimum in the neighborhood of that solution is sought.

Initially, the elements of the distance and flow matrices are sorted in increasing and decreasing order, respectively. The first β elements of each are kept (where $0 < \beta < 1$) and their products are computed. A simultaneous assignment of a pair of facilities to a pair of locations is selected at random among those with the α smallest costs, where $0 < \alpha < 1$. A feasible solution is then built by making a series of greedy assignments.

The construction and local search phases are repeated for a given number of times. Each iteration is independent in the sense that a new solution is always built from scratch. GRASP with path-relinking is an extension of the basic algorithm that uses an "elite set" to store the best solutions found. It incorporates a third phase that chooses, at random, one elite solution that is used to improve the solution produced at the end of the local search phase.

5 Results and Discussion

We present experimental results of using GRASP with path-relinking (GRASP-PR) for designing the layout of small artificial chips, and compare them with the layouts produced by Row-epitaxial. We used a C implementation of GRASP-PR provided by [9] with default parameters (32 iterations, $\alpha = 0.1$, $\beta = 0.5$, and elite set of size 10) and our own implementation of Row-epitaxial. The chips have 25-nt probes uniformly generated and asynchronously embedded in a deposition sequence of length 74. The running times and the border lengths of the resulting layouts are shown in Table 1 (all results are averages over a set of ten chips).

Our results show that GRASP-PR produces layouts with lower border lengths than Row-epitaxial on the smaller chips. On 6 x 6 chips, GRASP-PR outperforms Row-epitaxial by 2.14 percentage points on average when compared to the initial random layout. On 9 x 9 chips, however, this difference drops to 0.16 percentage point, while Row-epitaxial

Table 2: Average conflict indices of random chips compared with the layouts produced by Row-epitaxial and GRASP with path-relinking.

		Random	Row-epitaxial			GRASP with path-relinking		
Chip dimension	Number of probes	Avg. C. Index	Avg. C. Index	Reduction (%)	Time (sec.)	Avg. C. Index	Reduction (%)	Time (sec.)
6×6	36	524.28	495.15	5.56	0.05	467.08	10.91	3.68
7×7	49	558.25	521.90	6.51	0.07	489.32	12.35	8.84
8×8	64	590.51	551.84	6.55	0.09	515.69	12.67	19.48
9×9	81	613.25	568.62	7.28	0.11	533.79	12.96	38.83
10×10	100	628.50	576.49	8.28	0.11	539.69	14.13	73.09
11×11	121	642.72	588.91	8.37	0.12	551.41	14.21	145.67
12×12	144	656.86	598.21	8.93	0.12	561.21	14.56	249.19

generates better layouts on 11 x 11 or larger chips. In terms of running time, 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.

Table 2 shows improved results in terms of conflict indices. For these experiments, we used the same implementation of GRASP-PR and a version of Row-epitaxial implemented for conflict index minimization, which fills every spot with a probe minimizing the resulting conflict index on that spot. GRASP-PR consistently produces better layouts on all chip dimensions, achieving up to 6.38% less conflicts on 10 x 10 chips, for example, when compared to Row-epitaxial. In terms of running times, however, GRASP-PR is even slower for the case of conflict index minimization. This is because it takes more time to compute matrices F and D in the conflict index model as they are clearly more elaborate. However, GRASP-PR also seems to be faster when the distance matrices contain more zero entries.

The gains in terms of conflict index of both approaches are clearly less than the gains in terms of border length. This may be because the embeddings are fixed and the reduction of conflicts is restricted to the relocation of the probes, which only accounts for one part of the conflict index model. The fact that the distance matrix contains less zero entries with the conflict index formulation, however, might explain why GRASP-PR performs better in terms of conflict index minimization when compared to Row-epitaxial.

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 an entire microarray chip. However, we showed that it is certainly possible to use it on small sub-regions of a chip, which opens up the way for two interesting alternatives. First, our QAP approach could be used combined with a partitioning strategy such as the Centroid-based Quadrisection or our new Pivot Partitioning, to the design the smaller regions that result from the partitioning.

Second, an existing layout could be improved, iteratively, by relocating probes inside a defined region of the chip, in a sliding-window fashion. Each iteration produces an instance of a QAP whose size equals the size of the window. The QAP heuristics can be used to check whether a different arrangement of the probes inside the window can reduce the conflicts. For this approach to work, we also need to take into account the conflicts due to the spots around the window. Otherwise, a new layout with less internal conflicts could be achieved at the expense of an increase of conflicts on the borders of the window.

A simple way of preventing this situation 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. Then, 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.

Summary. 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 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.

References

[1] de Carvalho Jr., S., Rahmann, S.: Improving the Layout of Oligonucleotide Microarrays: Pivot Partitioning. Submitted (2006).

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

[3] 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.

[4] 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.

[5] 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*.

[6] 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*, 148–156.

[7] 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.

[8] 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.

[9] 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.

[10] 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.