Annual Report 2006

Title: Algorithms for Optimization of Microarray Design

Name: Sérgio A. de Carvalho Jr.

Date: March 2006

Project Summary: Microarrays have become an indispensable tool for gene expression analysis with several different technologies offering a range of functionalites. The production of high-density oligonucleotide microarrays, in particular, poses a number of interesting computational problems. For instance, the shortest common supersequence (SCS), a classical computer science problem, arises during the production of *in situ* synthesized probes. Another problem is posed by the design of the photolithographic masks used in the chemical synthesis of the probes. The aim of this project is analyze some of these problems, building models and designing new algorithms with ultimate goal of improving the quality of final product.

Project Plan: Previous Year

Overview

In the last year I have worked mainly in designing, implementing and evaluating several algorithms for the placement problem buildin an extensive development framework. The Two-dimensional Gray-Code-based Partitioning algorithm has produced good results although it still needs to be fully evaluated. A new post-placement algorithm, called Priority-driven Re-embedding, was implemented but has failed to outperform existing approaches. The layout problem has also been modelled as a Quadratic Assignment Problem (QAP) and a heuristic QAP solver (GRASP) has been integrated into the framework giving promising results, although it still needs to be better evaluated. Finally, a project seminar has also been offered in the Winter semester 2005, allowing students to work on our framework and to develop algorithms for microarray design.

Steps Completed

- Analysis of the layout of Affymetrix microarrays;
- Development of a programming framework for implementing algorithms for microarray design;
- Development and evaluation of a new partitioning algorithm (Two-dimensional Gray-Code-based Partitioning);
- Preparation of a project seminar (offered in the winter semester 2005) allowing students to implement algorithms for microarray design;
- Study of the Quadratic Assignment Problem (QAP) and modelling the array layout problem as a QAP;
- Integration of a QAP heuristic solver (GRASP) into the algorithmic framework and evaluation of results;

- Development and evaluation of several existing post-placement algorithms as well as a novel optimization algorithm (Priority-driven Re-embedding).

Steps not Completed or Failed

All steps were completed, although some did not yield the results expected.

Unforseen Extra Work Done

- Development of several Perl script for handling Affymetrix layout files.

Work Summary

In the next year we plan to further investigate the potential of modelling the layout problem as a QAP and to implement a new partitioning algorith (Pivot Partitioning). We also plan to publish a few papers on the results. The last part of the year will be reserved for writing the thesis.

Project Plan: Next Year

- *Feb 27th - Mar 15th:* Further investigate the potential of modelling the layout problem as a QAP by comparing the results produced by the GRASP algorithm with existing placers and submit a paper with the results to ECCB 2006;
- *March 15th - mid Apr:* Implement a new partitioning algorithm and submit a paper comparing all exising approaches to WABI 2006;
- *Mid Apr - May 15th:* Prepare a poster abstract for the ISMB 2006 to present either the results on QAP/GRASP or the new partitioning algorithm and submit another short paper on the evaluation of Affymetrix microarrays to GCB 2006;
- *Late Jul - Aug 4th:* Prepare the poster for ISMB 2006.

List of Lectures and Seminars

SS 2004	Prof. Dr. J. Stoye	Bioinformatics Journal Club
	Prof. Dr. J. Stoye	Advanced Topics in Bioinformatics and Genome Research
	Dr. D. Evers	Software Engineering: Making Programs Fast
	Dr. D. Evers Dr. S. Becker	Algorithms for SNP-Analysis
WS 2004/2005	Dr. D. Evers Dr. S. Becker Dr. S. Rahmann	Emerging Biotechnologies: Bioinformatics of Tomorow
	Prof. Dr. J. Stoye	Bioinformatics Journal Club
SS 2005	Prof. Dr. R. Giegerich	Seminar GK Bioinformatik
	Prof. Dr. J. Stoye	Arbeitsgruppenseminar Genominformatik
	Dr. Veli Makinen	Data Compression Techniques
WS 2005/2006	Dr. S. Rahmann	Project: Optimization of Microarray Design

Complete List of Publications

S. de Carvalho Jr. and S. Rahmann, *Searching for the Shortest Common Supersequence*, Technical Report 2005-03, Technische Fakultät, Universität Bielefeld, Apr 2005.

S. de Carvalho Jr. and S. Rahmann, *Improving the Layout of Oligonucleotide Microarrays*, BREW Bioinformatics Research and Education Workshop, Berlin, Apr 2005.

Annex: Scientific Essay

1 Introduction

A DNA 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 are considered the industry standard. They 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 [1].

Formally, we have a set of probe sequences $\mathcal{P} = \{p_1, p_2, ...p_n\}$ that are produced by a series of masks $\mathcal{M} = (m_1, m_2, ...m_\mu)$, where each mask m_i induces the addition of a particular nucleotide $\nu_i \in \{A, C, G, T\}$ to a subset of $\mathcal{P}$. The *nucleotide deposition sequence* $\mathcal{S} = \nu_1\nu_2 \ldots \nu_\mu$ corresponding to the sequence of nucleotides added at each masking step is therefore a supersequence of all sequences of $\mathcal{P}$.

In general, a probe can be *embedded* within $\mathcal{S}$ in several ways. We can think of an embedding of p_j as a μ-tuple $(e_{j,1}, e_{j,2}, ...e_{j,\mu})$ in which $e_{j,i}$ equals 1 if probe p_j receives nucleotide ν_i (at step i), or 0 otherwise. We say that the embedding of p_j is *productive* at step i when $e_{j,i} = 1$, or *unproductive* when $e_{j,i} = 0$. Equivalently, given an arrangement of the probes on the chip, we can say that a spot s is productive at step i if mask μ_i allows the passage of light to s (probes of s are activated to chemical coupling with nucleotide ν_i), or unproductive otherwise.

2 Layout Evaluation

Due to diffraction of light or internal reflection, untargeted spots can sometimes be accidentally activated in a certain step, producing unpredicted probes that can compromise the results of an experiment. This issue was described by Fodor et al. [1] who noted that the problem is more likely to occur near the borders between masked and unmasked spots. Feldman and Pevzner [2] were the first to formally address the problem, but they only considered *uniform arrays* (arrays containing all possible probes of a given length).

2.1 Border Length

Hannenhalli et al. [3] were the first to work with *assay* arrays with arbitrary probes. They formulated the Border Length Minimization Problem (BLMP), which aims at finding an arrangement of the probes together with their embeddings that minimizes the number of border conflicts during mask exposure steps.

Given a mask μ_i we compute its border length as the number of borders shared by masked (unproductive) and unmasked (productive) spots. The total border length of a given arrangement is the sum of border lengths over all masks.

2.2 Conflict Index

In [4], Kahng et al. noted that the definition of border length did not take into account two simple yet important practical considerations: a) stray light might activate not only immediate neighbors but also probes that lie as far as three cells away from the targeted spot; and b) imperfections produced in the middle of a probe are more harmful than in its extremities.

With the aforementioned observations in mind, we define the conflict index $\kappa(s)$ of a spot s whose probes of length ℓ_s are synthesized in μ masking steps as follows. First we define a distance-dependent weighting function, $\delta(s, s', i)$, that accounts for observation a) above:

$$\delta(s, s', i) := \begin{cases} 0 & \text{if spot } s' \text{ is masked at step } i, \\ \frac{1}{(d(s,s'))^2} & \text{otherwise,} \end{cases} \quad (1)$$

where $d(s, s')$ is the Euclidian distance between spots s and s'. We also use position-dependent weights as suggested in [4] to account for observation b):

$$\omega(s, i) := \begin{cases} 0 & \text{if spot } s \text{ is unmasked at step } i, \\ 1 + \log \min(b_{s,i} + 1, \ell_s - b_{s,i} + 1) & \text{otherwise,} \end{cases} \quad (2)$$

where $b_{s,i}$ denotes the number of nucleotides synthesized at spot s up to and including step i. We now define the conflict index of a spot s as

$$\kappa(s) := \sum_{i=1}^{\mu} \left(\omega(s, i) \sum_{s'} \delta(s, s', i) \right), \quad (3)$$

where s' ranges over all spots near s (in practice, only those inside a 7x7 grid centered in s).

It should be clear that our definition of conflict index is intimately connected to that of border length although the first estimates the risk of producing faulty probes in a given spot while the latter measures the quality of a mask.

3 Layout Optimization

We can distinguish between two main approaches to the problem of microarray layout design: placement algorithms and post-placement optimizations. The first concerns algorithms that, given a set of probes, a deposition sequence and layout constraints (chip dimensions), finds a placement of the probes minimizing the sum of conflicts (either border length or conflict indices).

The second approach takes an initial solution (a placement together with probe embeddings) and tries to reduce the conflicts by re-embedding the probes without changing their placement. In the following sections we will address some of the existing alternatives as well as some ideas for future research.

3.1 Placement Algorithms

Several placing algorithms have been suggested but only a few have been found to be useful for the latest chip dimensions (up to 1164x1164 spots and over a million probes). In fact, the problem size and its inherent properties naturally suggest the use of partitioning strategies to minimize running time, especially for those computing-intensive algorithms.

A common feature of partitioning strategies is that the chip is recursively divided into smaller regions. At the same time, the set of probes is divided into smaller sub-sets which are assigned to one of the final regions. In the end, these algorithms they rely on another placement algorithms to actually place the sub-set of probes on their assigned regions.

In the following sub-sections we describe the only partitioning strategy published so far and two novel partitioning strategies.

3.1.1 Centroid-based Quadrisection

The Centroid-based Quadrisection algorithm is a simple recursive procedure proposed by Kahng et al [5]. It starts by randomly selecting a probe, p_1, from the probe set. Then it examines all other probes and finds probe p_2 with maximum distance to p_1. Similarly it finds p_3 maximizing the sum of distances to p_1 and p_2, and p_4 maximizing the sum of distances to p_1, p_2 and p_3. These are called centroids.

Then, all other remaining probes are compared to the centroids and assigned to the one that minimizes the Hamming distance between them (which can be computed in linear time). Then the chip is partitioned into four quadrants, each being assigned to a set of probes. Each sub-region can then be recursively partitioned until a maximum partitioning depth is reached.

3.1.2 Two-dimensional Gray-Code-based Partitioning

The inspiration for this approach comes from the work of Feldman and Pevzner [2] on uniform arrays – arrays with the complete set of probes of a given length. They proved that a placement of such an array based on a two-dimensional Gray code has an optimal border length. However, uniform arrays are of restricted interest. Moreover, their work was based on synchronous embeddings – i.e. the deposition sequence is cyclical and a probe must synthesize one (and only one) base per cycle.

Our approach, on the other hand, does not need any of the above assumptions to work. Given a set of probes, we examine their given embeddings into the deposition sequence (if no embedding is available we can quickly compute, for instance, a left-most embedding) and count how many of them are masked and how many are unmasked at the first masking step. We can then partition the chip into two horizontal bands in such a way that they can accommodate each sub-set of probes, assigning, for instance, masked probes to the top partition and unmasked probes to the bottom partition. This procedure can be repeated recursively by partitioning each sub-set of embeddings based on the next masking steps and partitioning the chip likewise, alternating vertical and horizontal partitions. Assignments of sub-sets of probes to sub-regions obey a two-dimensional gray code in such a way that masked sub-regions are placed next to other masked sub-regions in an attempt to minimize conflicts.

This strategy has proved to produce layouts that have extremely low conflicts. However, it cannot optimize the complete set of masks because at some point the regions become too small to be sub-divided, leaving the last masks with high levels of conflicts. In our experience, this algorithm needs, more than any other partitioning approach, to be combined with another placement algorithm that is able to optimize those masks that have not been optimized.

However, we believe that it has a great advantage when the goal is to minimize the conflict index of the spots. The reason is that, instead of examining the first masks, we can drive the partitioning of the chip based on the state of their embeddings at the middle masking steps, going towards both ends of the deposition sequence. This will result in a layout with extremely low conflicts in the middle masks where it has greater impact for the conflict index.

3.1.3 Pivotal-driven Partitioning

The Pivotal-driven partitioning is similar to the Centroid-based Quadrisection proposed by Kahng et al. [5]. The main difference is that it is based on the following observation. While some probes have an astonishing number of possible embeddings (up to several millions on a typical Affymetrix chip), others may have only a few or even only one valid embedding into the deposition sequence. The idea is that we can use these probes with less degree of freedom as pivots to drive the placement of the remaining probes.

Several implementations are possible. One that sounds viable can be described as follows. First we examine the set of probes and identify all pivots (those with a minimum number of embeddings). Computing the number of embeddings of a probe takes quadratic time but can be done rather quickly with a few simple optimizations (even a million probes can be examined in a matter of seconds).

Then, similarly to the Centroid-based Quadrisection, we choose two pivots, p_1 and p_2, with maximum Hamming distance between them (note that the Hamming distance reflects the minimum number of conflicts between these probes since we assume that they cannot be re-embedded in a different way). Computing the Hamming requires linear time. Then we can examine the other pivots and compute their Hamming distances to p_1 and p_2, diving them into two sub-sets. The same procedure can be repeated recursively until we build a binary tree of pivots. Note that the pivots are assigned to their corresponding sub-sets in a way that all pivots can be found in the leaves of the tree.

The next phase consists of examining every non-pivot probe p and computing the minimum distance of any valid embedding of p to every pivot, and assigning it to the leaf of the tree corresponding to that pivot. Note that p can be immediately re-embedded optimally in regards to

its pivot. Actually, it is possible to re-embed p optimally in regards to all probes already assigned to that leaf.

Finally, we traverse the binary tree from the root, partitioning the chip into regions proportionally to the number of probes in each sub-tree. Like in the Two-dimensional partitioning, we alternate horizontal and vertical partitions, although there is no clear connection between sub-trees that could be driven by a Gray code.

For real chips like those produced by Affymetrix, we expect to find about 1% of the probes with a single embeddings (pivots). If this is not the case, or if we need more or less pivots, some non-pivots can be arbitrarily promoted to pivots and vice-versa.

Computing the minimum distance of p to a fixed embedding takes quadratic time with a dynamic programming approach. Re-embedding a probe the computed minimum distance takes linear time once the dynamic programming matrix has been computed. Considering all other probes of a leaf node when optimally re-embedding p (instead of only considering the pivot) does not increase the time complexity and takes little extra time. We expect that computing the minimum distance and optimally re-embedding a probe will have the greatest impact on the total running time of our method. However, since we work with a limited number of pivots, we expect the algorithm to have a manageable running time.

The main advantage of this partitioning is that, as in the Centroid-based Quadrisection, the probes are placed on sub-regions based on an analysis of their full embedding string (which is the main disadvantage of the Two-dimensional partitioning). However the main weakness of the Centroid-based Quadrisection and other placement algorithms is that they do not consider all valid embeddings of a probe. This is usually only considered in later optimizations, usually when it is too late to change the placement.

3.2 Post-placement Optimization

We now turn to the post-placement optimization problem, which aims at minimizing conflicts by re-embedding probes without changing their location on the chip. A common feature of such strategies is the use of an Optimum Single-Probe Embedding (OSPE) algorithm proposed by Kahng et al [4]. This algorithm is able to compute an optimum embedding of a probe in regards to its neighbors, whose embeddings are considered fixed. The algorithm is based on a simple dynamic programming recursion that takes quadratic time.

Note that the OSPE algorithm can never increase the amount of conflicts and thus, all optimization algorithms can be repeated several times until a local optimal solution is found (or until the improvements drop below a given threshold).

In the next sub-sections we describe three post-placement algorithms described by Kahng et al. [4] [5] and propose a new heuristic.

3.2.1 Greedy and Batched Greedy Optimization

The Greedy algorithm computes, for each spot, the maximum reduction of conflicts that could be achieved by re-embedding its probe with the OSPE algorithm. Then it greedily selects the spot with higher gain and re-embeds its probe optimally in regards to its neighbors, updating the gains of affected spots. The Batched Greedy version of the algorithm sacrifices its greedy nature by postponing the update of gains and re-embedding all unaffected probes of an iteration.

3.2.2 Chessboard Optimization

The Chessboard optimization is based on a simple heuristic. From the point of view of border length, a chip can be bi-colored just like a chessboard, in such a way that the embeddings of probes located on white spots are independent of those placed on black spots, and vice-versa. The idea is to use this coloring to alternate the optimal re-embedding of probes located on black and white spots.

3.2.3 Sequential Optimization

The Sequential optimization is a strikingly simple yet effective post-placement optimization. This algorithm also re-embeds probes with the OSPE but instead of trying to find a particular order, it just proceeds spot by spot from top to bottom, left to right. Surprisingly, it outperforms all other know post-placement algorithms.

3.2.4 Priority-driven Optimization

In this section we present a new heuristic post-placement optimization algorithm, called Priority-driven Re-embedding. Our approach is based on the same observation stated for the PivotPartitioning, namely, the fact that while some probes have an astonishing number of possible embeddings (up to several millions on a typical Affymetrix chip), others may have only a few or even only one valid embedding into the deposition sequence.

Our assumption is that those with a greater number of embeddings can better adapt to their neighbors, while those with only a limited number of embeddings are less flexible and should be re-embedded first. The idea is that we should use these probes with less degree of freedom to drive the re-embedding of their neighboring spots.

Our algorithm can be described as follows. First, we find a set of spots whose probes have only one or a limited number of embeddings. We mark these spots as finished and add their non-empty immediate neighbors to a priority queue of pending spots whose order is determined by the number of embeddings their corresponding probes have. Then, we retrieve (and remove) the first spot from the queue (the one whose probe has the least number of embeddings) and use the OSPE algorithm to re-embed its probe optimally. Finally, we mark the spot as finished and add its immediate neighbors to the queue. The algorithm repeats until the queue is fully emptied. Whenever an empty spot is reached, it is immediately marked as finished and, of course, not added to the queue.

Sometimes the layout of a chip contains "islands" of probes surrounded by empty spots. In such cases, it is necessary to do a full scan on the chip looking for non-finished spots. If any is found, it is added to the queue and processed as described above.

When computing the optimum embedding with the OSPE algorithm, we can consider all neighbors of a spot or only those which have already been marked as finished. Our experiments showed that the latter approach is more efficient. The reason may be that, by using the OSPE algorithm we always decrease the number of conflicts towards a local optimum and, a re-embedding that does not consider the neighboring probes may be a bad decision that the algorithm will never recover from. Moreover, by considering all neighbors, the algorithm can be executed several times in succession, with further reduction of conflicts.

We have also explored two variants in the ordering of the priority queue. The first one retrieves probes with the greatest number of finished spots (instead of the least number of embeddings). The intuition is that these spots also have a lesser degree of freedom since they have to adapt to a greater number of neighboring probes and, thus, should be re-embedded first. In particular, a spot with 4 finished immediate neighbors should be immediately re-embedded. Another possibility is to order the queue based on a combination of number of embeddings and number of finished neighbors.

Experiments have shown that the three variants produce on average the same reduction in conflicts. Sometimes, when the goal is to minimize border length, a priority based on the number of neighbors gives a small upper hand. If the goal is to reduce the conflict indices, it appears that the combined priority has a small advantage.

Our experiments also showed that, surprisingly, the reduction in conflicts produced with the Priority-driven optimization is only a bit better than the simpler Sequential optimization, with the disadvantage of much higher running times.

The conclusion is that the order in which the probes are re-embedded play a fairly insignificant role in the final result, and the local optimum reached in the end is only slightly higher or lower than those found with a different ordering.

4 Microarray Layout and the Quadratic Assignment Problem

We now turn to a different approach to the array layout problem, namely, to model the placement problem as a quadratic assignment (QAP). The quadratic assignment is combinatorial problem that seeks to assign a set of objects to a set of places with a minimum cost. The cost is defined by the product of two functions: the distance between the places and the flow among the objects.

If we think of the spots as places and the probe as objects to be assigned to these places, we can see the layout problem as an instance of the QAP. The goal function is then defined as the distance between the spots multiplied by distance between the probes (the flow in the QAP formulation). This formulation conveniently resembles the definition of conflict index.

The only problem is that QAP is notoriously hard. Problems of size greater than 15 remain nearly intractable. However, heuristic algorithms can find sub-optimal solutions in a reasonable time. In particular an algorithm called GRASP due to Li et al. [6] is able to compute good solutions for problem of sizes of up to a few hundred.

We have experimentally used GRASP to improve the layout of small regions of the chip with excellent results. We believe that GRASP can be better used to find a good placement of a small region produced, for example, by a partitioning algorithm.

However, we also believe that GRASP can be used as an optimization algorithm if some changes to the implementation are introduced. The idea is that we can examine an already placed region of the chip and, in most of the times, produce a better placement of that region with GRASP. However, as it is used so far, we also observe an increase in conflicts in the border between the optimized region and the probes that lie immediately close to that region. Clearly this results from the fact that, currently, we cannot pass to GRASP the part of the cost of placing a probe in a spots that is due to the probes outside the considered region.

If a modification can be done in the main GRASP routine so that it also takes into account the cost associated with probes outside the region under consideration, we believe that we can substantially reduce conflicts in a microarray chip. We also think that such an approach could be used in a sliding-window manner to optimize the full set of probes of a chip. Moreover, with an overlapping sliding-window, we expect to move probes around and further increase the chances of reducing conflicts.

References

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

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

[3] Hannenhalli, S., Hubell, E., Lipshutz, R., Pevzner, P.: Combinatorial algorithms for design of DNA arrays. Adv. Biochem. Eng. Biotechnol. **77** (2002) 1–19

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

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

[6] Y. Li, P.M. Pardalos, and M.G.C. Resende: A greedy randomized adaptive search procedure for the quadratic assignment problem DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 237–261, 1994