Algorithms for Improving the Design and Production of Oligonucleotide Microarrays

Sérgio Anibal de Carvalho Junior

January 2007

PhD thesis submitted to the
Faculty of Technology of Bielefeld University, Germany,
for the degree of Dr. rer. nat.

Supervisor:
Dr. Sven Rahmann

1. Referee: Prof. Dr. Jens Stoye
2. Referee: Prof. Dr. Robert Giegerich

Contents

Abbreviations and Notation

Abbreviations in alphabetical order

A, `A`	adenine
Å	Angstrom; 1Å = 10^{-10} m = 1/10 nm
aRNA	anti-sense RNA, amplified RNA; complementary to mRNA
bp	base pair(s)
C, `C`	cytosine
°C	degrees centigrade (Celsius)
cDNA	complementary DNA; DNA synthesized from mRNA by RT
CDS	coding sequence
cRNA	complementary RNA; same as aRNA
CSMS, csms	cumulative statistics of matching statistics
Cy3	Cyanine 3-dNTP; in fluorescently labeled DNA
Cy5	Cyanine 5-dNTP; in fluorescently labeled DNA
DMD	digital micromirror device
DNA	deoxyribonucleic acid
dN	any deoxynucleotide; any of dA, dC, dG, dT
dNTP	any deoxynucleotide-triphosphate; any of dATP, dCTP, dGTP, dTTP
EST	expressed sequence tag
G, `G`	guanine
HPLC	high-pressure liquid chromatography
IVT	in-vitro transcription
K	Kelvin; physical unit of temperature
LCF	longest common factor
LCFS	longest common factor statistics
M	physical unit of molar concentration: 1 M = 1 mol/l
MCMC	Markov chain Monte Carlo
MES	2-(N-morpholino)ethanesulfonic acid
mol	physical unit of amount of substance 1 mol = as many entities as atoms in 0.012 kg of carbon 12
mRNA	messenger RNA
MS, ms	matching statistics
N, `N`	any of the bases A, C, G, T/U
$[Na^+]$	salt (sodium ion) concentration
NaCl	salt (sodium chloride)
NN	nearest neighbor
nt	nucleotide(s)

oligo-(dT)	primer of 12–20 dTs binding to the poly(A) tail of eukaryotic mRNA
P	phosphor
^{33}P	a radioactive phosphor isotope
PAGE	polyacrylamide gel electrophoresis
PCR	polymerase chain reaction
PEM	percent of expected mutations (unit of evolutionary time)
RNA	ribonucleic acid
RT	reverse transcription
RTase	reverse transcriptase; an enzyme
RT-PCR	reverse transcription-polymerase chain reaction
SAGE	serial analysis of gene expression
SAPE	streptavidin phycoerythrin
SBH	sequencing by hybridization
SCS	shortest common supersequence
SCSP	shortest common supersequence problem
SNP	single nucleotide polymorphism
SSC	saline sodium citrate buffer
SSPE	saline sodium phosphate buffer
SVD	singular value decomposition
T, `T`	thymine
T7	a bacteriophage that infects *E. coli*; these viral parasites are numbered T1 through T7
Taq	*Thermus aquaticus*; a heat-resistant bacterium
TIFF	tagged image file format
transfrag	transcript fragment
tRNA	transfer RNA
U, `U`	uracil
UTR	untranslated region; part of mRNA transcripts

Notation by topic

Notation introduced in early chapters is also used in subsequent chapters.

Chapter 1

p	probe sequence; a string
$\|p\|$	length of p
t	transcript or target sequence
T	transcriptome $T = (t_1, \ldots, t_n)$
m	number of probes
i	probe index
n	number of transcripts
j	transcript index
θ	hybridization conditions; parameters
$a(p, t; \theta)$	affinity coefficient between p and t, given θ

$p \lhd t$	p is a factor (substring) of t
ε	lower affinity limit for specific hybridization

Chapter 2

$x = (x_j)$	n-vector of transcript expression levels
$y = (y_i)$	m-vector of measured probe signals
$A = (A_{ij})$	$m \times n$-matrix of probe-transcript affinity coefficients; $y = A \cdot x$
$t(i)$	index of the unique target that probe i binds to
a, a_i	particular affinity coefficients
ρ_{ij}	position dependent factor of A_{ij}
β_{ij}	hybridization stability dependent factor of A_{ij}
γ_{ij}	sequence composition dependent factor of A_{ij}
σ_{ij}	self-complementarity dependent factor of A_{ij}
ε_i	stochastic additive noise of probe signal intensity
c_i, c	systematic additive offset of signal intensity
α	scale of signal intensity
η_{ij}	stochastic multiplicative noise of signal intensity
d	differential operator
Δ	denotes a difference
U; q; w	internal energy; heat; work
p	pressure
V	volume
T	temperature
H; H°_m; $\Delta_\mathrm{r} H^\circ$	enthalpy; standard molar enthalpy; standard reaction enthalpy
S; S°_m; $\Delta_\mathrm{r} S^\circ$	entropy; standard molar entropy; standard reaction entropy
G; G°_m	Gibbs energy; standard molar Gibbs energy
$\Delta_\mathrm{r} G^\circ$; $\Delta_\mathrm{r} G$	standard Gibbs energy of reaction; reaction Gibbs energy
ξ	extent of reaction
R	gas constant; $R = 8.3145$ J K^{-1} mol^{-1}
K	equilibrium constant
$[\mathrm{S}_2]_\mathrm{eq}$	equilibrium concentration of reactant S_2 (single stranded probes)
T_M	melting temperature

Chapter 3

$\mathrm{lcf}(p,t)$	length of the longest common factor of p and t
$\mathrm{lcf}^1(p,t)$	same, allowing one mismatch
$\mathrm{lcf}^*(p,t)$	combined measure derived from lcf and lcf^1
$\mathrm{LCF}(p \mid T)$	LCF vector of probe p against transcriptome T
$\mathrm{LCFS}(p \mid T; \Delta)$	LCF statistics of p against T of width Δ
δ	index of LCF statistics, $0 \le \delta < \Delta$ difference between full probe length and LCF length
$T(i)$	index set of intended targets for probe i
T	transcriptome; see Chapter 1
$U'(p_i \mid T)$	unspecificity of probe i in T; formal definition
τ	temperature; to avoid confusion with transcriptome T

$\beta(\delta)$	average hybridization probability as a function of δ
ζ	offset constant; $\zeta = \ln(\beta(0)/(1-\beta(0)))$
$b > 0$	average Gibbs energy per base pair in units of $(R\tau)$
$u(\delta)$	approximation to $\beta(\delta)$
$U(p_i \mid T)$	unspecificity of probe i in T; practical definition

Chapter 4

$T = (T_c)$	transcriptome of transcript collections T_c $(c = 1..C)$
C	number of collections
$T_c = \{t_{c,j}\}$	transcript collection c with transcripts $t_{c,j}$ $(j = 1..N_c)$
N_c	number of transcripts in collection c
s	target sequence
Σ	alphabet; here $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$
Σ^*; Σ^+	all strings (non-empty strings) over alphabet Σ
`$`	string end marker and separator
`X`	unspecified or wildcard character
`pos`	suffix array
$\mathrm{lcp}(s,t)$	length of the longest common prefix of s and t
`lcp`	longest common prefix array
q	bucket prefix length
`bck`	bucket array
$\gamma = \langle Q \rangle$	numerical radix-q representation of a q-gram Q
`cl`	collection number array
$\mathrm{ms}^{s\|t} = (\mathrm{ms}_i^{s\|t})$	matching statistics of s against t
$\mathrm{ms}^{s\|t;f}$	matching statistics allowing f mismatches of s against t
$R^0_{\min}$	minimum value of relevant matching statistics
$R^1_{\min}$	minimum value of relevant ms. with one mismatch
$R_{\max}$	maximum value of relevant matching statistics
$\mathtt{MS}[i][c]$	matching statistics array; $\mathtt{MS}[i][c] = \mathrm{ms}_i^{s\|T_c}$
(i, J)	jump at position i to level J
n, m	string lengths; $\|s\| = m$, $\|t\| = n$
$\mathbb{P}$	generic notation for a probability measure
$\mathbb{E}$	generic notation for an expectation
$\pi = (\pi_c)_{c\in\Sigma}$	character distribution for a random sequence model
p	probability that two random characters match
q	$q := 1 - p$
K_L	random number of occurrences of a random L-gram in a string
ρ_L	probability that a jump to level L occurs at a fixed position
E_L	expected number of jumps to level L in matching statistics
E_L^+	expected number of jumps to level $\geq L$ in matching statistics
L°	center of the distribution of the LCF of two random strings

Chapter 5

d	probe distance from the transcript's $3'$-end

$p_i = (d_i, \ell_i)$	probe i defined by end distance d_i and length ℓ_i
$f(d)$; f_i	position preference factor for distance d; $f_i := f(d_i)$
h_i	hairpin formation probability for probe i
U_i	unspecificity for probe i
B_i	badness value for probe i
$\mathcal{B}(\delta)$	additional badness for probes clustering within distance δ
B'_i; $\overline{B}'$	modified badness value for probe i; threshold $\overline{B}'$
S	selected probe set

Chapter 6

$\mathbb{1}_{\{\cdot\}}$	generic notation for an indicator function
$S = (s_c)$	target sequences for all collections $c = 1, \dots, C$
$H = (H_{ij})$	binary $m \times n$ hybridization matrix of probes vs. targets
D	design, i.e. a selection of probes; $D \subset \{1, \dots, m\}$
A^D; H^D	affinity and hybridization matrix restricted to rows from D
$\mathcal{M}$	minimum target coverage
$\mathcal{A}$	average target coverage
A^{T}	transpose of A
Σ	diagonal matrix with singular values (σ_j) of A
σ_j	j-th largest singular value
$\operatorname{diag}(\cdot)$	diagonal matrix with specified entries
A^-	pseudo-inverse of A
$\lVert \cdot \rVert_2$	Euclidean norm of a vector; spectral norm of a matrix
$\lVert \cdot \rVert_\infty$	maximum norm of a vector
$\operatorname{cond}(A)$	condition number of A
$\operatorname{cond}_{\mathrm{s}}(A)$	Skeel condition number of A
μ	maximal number of probes that may be selected
S,T	sets of target indices
$S \Delta T$	symmetric set difference of S and T
$P(S)$	set of probes that hybridize to any target in S
$T(i)$	set of targets that hybridize to probe i
γ	desired minimum target coverage
σ	desired minimum target set separation
$\delta \in \{0,1\}^m$	binary vector representation of design D
h_j	j-th column of binary hybridization matrix H
$z^{S,T} = (z_i^{S,T})$	indicates whether probe i separates sets S and T
δ^+, h_j^+, z^+	vectors extended for virtual unique probes
B	maximal size of target sets to be separated
f_0; f_1	false negative (positive) error rate
$\mathbb{P}(x)$	prior probability of binary expression vector $x \in \{0,1\}^n$
$\mathbb{P}(y \mid x)$	likelihood of probe signals $y \in \{0,1\}^m$, given x
$\pi(x) = \mathbb{P}(x \mid y)$	posterior probability of x for fixed y
$q(z \mid x)$	proposal probability for new solution z at current solution x
$\mathcal{N}(x)$	neighborhood of x

$\alpha(x, z)$	acceptance probability for z at x
Q	rate matrix of an evolutionary Markov process
t	evolutionary time
P^t	Markov transition kernel for time t
$\mathrm{expm}(Q)$	matrix exponential; $\mathrm{expm}(Q) := \sum_{n \geq 0} Q^n/n!$

Chapter 7

$\mathrm{csms}^{s\|t}$	cumulative statistics of matching statistics of s against t $\mathrm{csms}^{s\|t}_{i,\mu}$ refers to position i in s and matches of length $\geq \mu$
$\mathtt{CSMS}[i][\mu]$	CSMS array; $\mathtt{CSMS}[i][\mu] = \mathrm{csms}^{s\|t}_{i,\mu}$ if $\mu \in [R^0_{\min}, R_{\max}]$
$\sigma^i_{k,u}$	unexpected CSMS at offset k for a probe starting at i
$L_{i,\delta}$	whole genome surrogate for $\mathrm{LCFS}(p_i)_\delta$
L	length of hypothetical transfrags
D	distance between start points of hypothetical transfrags
K	transfrags known to be present
P_1	probes expected to show a signal because of K
M	transfrags of unknown status; $M = \{1..n\} \setminus K$
N	transfrags most likely not present
J	transfrags whose status is inferred by sampling; $J = \{1..n\} \setminus (K \cup N)$
χ	fraction of transfrags hybridizing to additional probes

Chapter 8

Σ	DNA alphabet
π	permutation of the nucleotides
$\mathcal{S}$	sequence set
e	binary embedding vector of a string into a supersequence
E	embedding matrix with rows e_i, $i = 1..\|\mathcal{S}\|$
$\mathcal{U}$	upper bound on the supersequence length
$\mathcal{L}$	lower bound on the supersequence length
L_i	length of the i-th sequence in $\mathcal{S}$
$N_i(x)$	number of occurrences of $x \in \Sigma$ in the i-th sequence
$N(x)$	$\max_i N_i(x)$
C_i	completion step of the i-th sequence
$W_{i,k}$	number of unproductive steps between $(k-1)$-st and k-th productive step for sequence i
Φ	cumulative distribution function of the standard Normal distribution

Foreword

Microarrays are a ubiquitous tool in molecular biology with a wide range of applications on a whole-genome scale including high-throughput gene expression analysis, genotyping, and resequencing.

Several different microarray platforms exist, but this thesis focuses on high-density oligonucleotide arrays. The advantage of higher densities arrays is that they allow, for instance, the simultaneous measurement of the expression of several thousand genes at once.

High-density microarrays are usually produced by light-directed combinatorial chemistry that builds the probe sequences base-by-base. Because of the natural properties of light, the quality of a microarray can be improved by carefully designing the physical arrangement, or *layout*, of its probes.

In this thesis, we review models for evaluating the layout of oligonucleotide microarrays and survey algorithmic approaches that can be used in their design.

DNA chips allow to quickly obtain and compare gene expression profiles of different cell or tissue samples. As one of many applications, one hopes to better understand various types and subtypes of cancer and to improve cancer therapy by characterizing the differences between the expression profiles of healthy cells and tumor cells.

The first chapter of this thesis offers an overview of existing microarray technologies and contrasts them with other techniques for gene expression analysis. High-density oligonucleotide arrays are described in detail.

Gene expression analysis with DNA chips is a high-throughput technique that produces massive amounts of data; however, this technique is not error-free. Each step of an experiment must be performed carefully. In particular, the DNA chip must be carefully designed and manufactured. This thesis proposes and describes solutions for some of the algorithmic problems that arise in this phase. These problems are formulated in detail in the last section of Chapter x.

My research started with the general question of how to find oligonucleotide probes for highly homologous transcripts when it cannot be guaranteed that a sufficiently large set of clearly transcript-specific (or *unique*) probes can be found. An interesting future large-scale application could be the individual expression measurement of all splice variations of all genes in the human genome, for example. The basic idea was to find several *non-unique* probes such that different probes hybridize to different combinations of targets, with the hope that the measured signal can then be decoded into the individual expression levels. Two question then follow immediately: How does the decoding work, and how can the probes be designed in a way that makes the decoding as simple as possible?

In early 2001, custom probe design was in its infancy. The design of the large commercial chips, such as the Affymetrix GeneChip®, was a well guarded secret of the respective companies. Several other research groups were also beginning to work on probe selection, and their results, made the problem more popular. I soon realized that, before actually working on non-unique probes, more fundamental questions should be addressed. A high-performance large-scale probe selection system for unique probes would be very useful but did not exist. Such a system could then be used as a basis for non-unique probe design. These considerations are reflected in this thesis.

Once a set of probes is chosen for a chip design, the chip must be produced. With several technologies, such as the Affymetrix GeneChip® arrays and febit's geniom® one system, the probes are synthesized in situ on the chip with a combination of photolithography and combinatorial chemistry. We consider the problem of optimizing the nucleotide deposition sequence to lower production costs and to decrease the overall error rate during synthesis.

A concluding discussion about the results of this thesis and an outlook into the future can be found in Chapter 10.

Publications. Parts of this thesis have been published in advance.

This thesis also contains previously unpublished material, namely

- new placement algorithm that simultaneously places and re-embeds the probes

Acknowledgments. This work was carried out while I was at the... It has been a very enjoyable experience, thanks to all present and former colleagues and students in the Bioinformatics program.

I would especially like to thank Sven Rahmann for suggesting the topic, providing initial ideas, and for the opportunity to write this thesis under his guidance.

The support of the whole computer service group at the CeBiTec was crucial to the success of this work, and I cannot thank them enough for their help.

I enjoyed fruitful discussions with many people while this thesis was underway; in particular, I am grateful to ... for their valuable hints and comments.

... read early drafts of several chapters of this thesis and helped to improve it in many ways through their comments.

Sérgio A. de Carvalho Jr. Bielefeld, January 2007

Chapter 1

Introduction

1.1 DNA and Genes

Choose another title for this sub-section.

1.2 Functional Genomics

Choose another title for this sub-section.

1.3 High-density Oligonucleotide Microarrays

Oligonucleotide microarrays consist of short DNA fragments, called *probes*, affixed or synthesized at specific locations, called *features* or *spots*, of a solid surface. Microarrays are based on the principle of Watson-Crick base pairing. Each probe is a single-stranded DNA molecule of 10 to 70 nucleotides that perfectly matches with a specific part of a *target* molecule. The probes are used to verify whether (or in which quantity) the targets are present in a given biological sample.

This type of microarray was originally designed in the late 1980s as a tool for DNA sequencing, a technology that is known as DNA Sequencing by Hybridization.

The first step of a microarray experiment consists of collecting mRNAs or genomic DNA from the cells or tissue under investigation. The mixture to be analyzed is prepared with fluorescent tags and loaded on the array, allowing the targets to hybridize with the probes. Any unbound molecule is washed away, leaving on the array only those molecules that have found a complementary probe. Finally, the array is exposed to a light source that induces fluorescence, and an optical scanner reads the intensity of light emitted at each spot.

Under ideal conditions, each probe will hybridize only to its target. Thus, it is possible to infer whether a given molecule is present in the sample by checking whether there is light coming from the corresponding spot of the array. The expression level of a gene in a cell can also be inferred because each spot contains several million identical probes, and the strength of the fluorescent signal on a spot is expected to be proportional to the concentration of the target in the sample. In practice, each target is queried by several probes (its *probe set*), and complex statistical calculations are performed to infer the concentration from the observed signals.

High-density microarrays, also called microarray chips, can have more than a million spots, and are thus able to query tens of thousands of genes, covering the entire genome of an organism. The pioneering Affymetrix GeneChip® arrays, for instance, have up to 1.3 million spots on a coated quartz substrate measuring a little over 1 cm^2. The spots are as narrow as 5 μm (5 microns, or 0.005 mm), and are arranged in a regularly-spaced rectangular grid.

1.3.1 Microarray Production with Photolithographic Masks

GeneChip arrays are produced by combinatorial chemistry and techniques derived from micro-electronics and integrated circuits fabrication. Probes are typically 25 bases long and are synthesized on the chip, in parallel, in a series of repetitive steps. Each step appends the same kind of nucleotide to probes of selected regions of the chip. The selection of which probes receive the nucleotide is achieved by a process called *photolithography* (Fodor et al., 1991).

Figure 1.1 illustrates this process: The quartz wafer of a GeneChip array is initially coated with a chemical compound topped with a light-sensitive protecting group that is removed when exposed to ultraviolet light, activating the compound for chemical coupling. A lithographic mask is used to direct light and remove the protecting groups of only those positions that should receive the nucleotide of a particular synthesis step. A solution containing adenine (A), thymine (T), cytosine (C) or guanine (G) is then flushed over the chip surface, but the chemical coupling occurs only in those positions that have been previously deprotected. Each coupled nucleotide also bears another protecting group so that the process can be repeated until all probes have been fully synthesized.

1.3.2 Maskless Microarray Production

Photolithographic masks are notoriously expensive and cannot be changed once they have been manufactured. Thus, any change in the chip layout requires the production of a new set of masks. A similar method of *in situ* synthesis known as Maskless Array Synthesizer (MAS) was later developed to eliminate the need of such masks

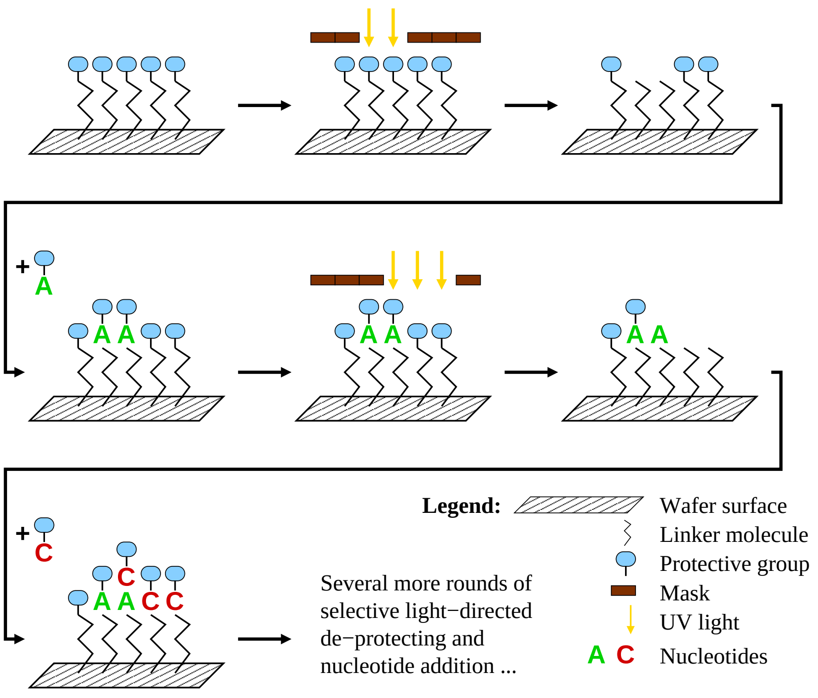

Figure 1.1: Affymetrix's probe synthesis via photolithographic masks. The chip is coated with a chemical compound and a light-sensitive protecting group; masks are used to direct light and activate selected probes for chemical coupling; nucleotides are appended to deprotected probes; the process is repeated until all probes have been fully synthesized.

(Singh-Gasson et al., 1999). Probes are still built by repeating cycles of deprotection and chemical coupling of nucleotides. The illumination, however, relies on an array of miniature mirrors that can be independently controlled to direct or deflect the incidence of light on the chip.

NimbleGen Systems, Inc. uses its own Digital Micromirror Device (DMD) that can control up to 786 000 individual mirrors to produce microarrays with spots as small as 16 μm $\times$ 16 μm. The Geniom® system of febit biotech GmbH, a platform for customized microarray production, also uses a micromirror array to direct the synthesis process.

1.3.3 The Problem of Unintended Illumination

Regardless of which method is used to direct light (masks or micromirror arrays), it is possible that some probes are accidentally activated for chemical coupling because of light diffraction, scattering or internal reflection on the chip surface. This unwanted illumination of regions introduces unexpected nucleotides that change probe sequences, significantly reducing their chances of successful hybridization with their targets. Moreover, these faulty probes may also introduce cross-hybridizations, which can interfere in the experiments performed with the chip.

This problem is more likely to occur near the borders between a masked and an unmasked spot (in the case of maskless synthesis, between a spot that is receiving light and a spot that is not). This observation has given rise to the term *border conflict*.

It turns out that by carefully designing the *arrangement* of the probes on the chip and their *embeddings* (the sequences of masked and unmasked steps used to synthesize each probe), it is possible to reduce the risk of unintended illumination. This issue becomes even more important as there is a need to accommodate more probes on a single chip, which requires the production of spots at higher densities and, consequently, with reduced distances between probes.

In this thesis, we address the problem of designing the layout of a microarray with the goal of reducing the chances of unintended illumination, which we call Microarray Layout Problem (MLP). We use the term *layout* to refer to where and how the probes are synthesized on the chip (their arrangement and their embeddings).

Chapter 2

The Microarray Layout Problem

In this chapter we give a more precise definition of the Microarray Layout Problem (MLP) and define criteria for evaluating a given layout. The description that follows assumes that probes are synthesized with photolithographic masks, but the concepts apply to the maskless production as well (with micromirror arrays). Two evaluation criteria are presented: *border length* and *conflict index*. As shown later, the conflict index model can be seen as a generalization of the border length model.

Formally, we have a set of probes $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, where each $p_k \in \{\text{A,C,G,T}\}^*$ is produced by a series of T synthesis steps (frequently, but not necessarily, all probes have the same length ℓ). Each synthesis step t uses a mask M_t to induce the addition of a particular nucleotide $N_t \in \{\text{A,C,G,T}\}$ to a subset of $\mathcal{P}$ (Fig. 2.1). The *nucleotide deposition sequence* $N = N_1 N_2 \ldots N_T$ corresponding to the sequence of nucleotides added at each synthesis step is a supersequence of all $p \in \mathcal{P}$.

A microarray chip consists of a set of spots, or sites, $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$, where each spot s is specified by its coordinates on the chip surface and accommodates a unique probe $p_k \in \mathcal{P}$. Note that we usually refer to s as containing a single probe p_k although, in practice, it contains several million copies of it. Each probe is synthesized at a unique spot, hence there is a one-to-one assignment between probes and spots (if we assume that there are as many spots as probes, i.e., $m = n$). Real microarrays may have complex physical structures but, in this thesis, we shall assume that the spots are arranged in a rectangular grid with n_r rows and n_c columns. We also assume that probes can be assigned to any spot.

In general, a probe can be *embedded* within N in several ways. An embedding of p_k is a T-tuple $\varepsilon_k = (\varepsilon_{k,1}, \varepsilon_{k,2}, \ldots, \varepsilon_{k,T})$ in which $\varepsilon_{k,t} = 1$ if probe p_k receives nucleotide N_t (at step t), and 0 otherwise. In particular, a *left-most embedding* is an embedding in which the bases are added as early as possible (as in ε_1 in Fig. 2.1). Similarly, a *right-most embedding* is an embedding in which the bases are added as late as possible (as in ε_8 in Fig. 2.1).

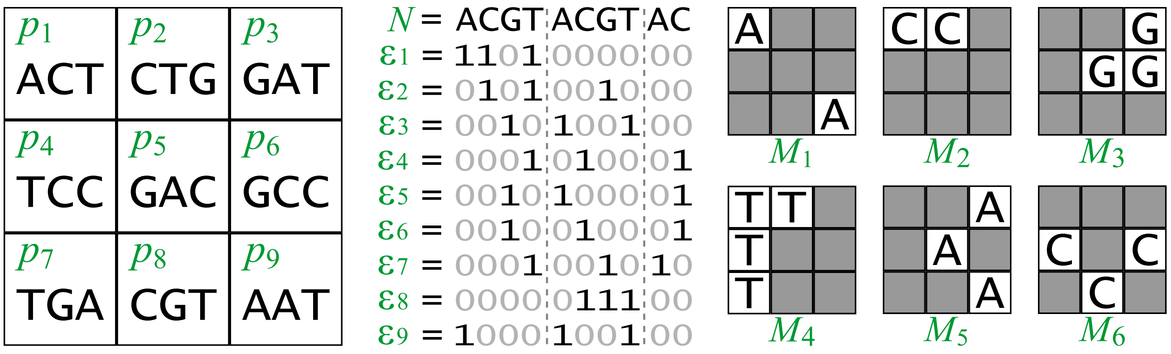

Figure 2.1: Synthesis of a hypothetical 3×3 chip with photolithographic masks. Left: chip layout and the 3-mer probe sequences. Center: deposition sequence with 2.5 cycles and probe embeddings (asynchronous). Right: first six masks.

We say that an embedding ε_k is *productive* (unmasked) at step t if $\varepsilon_{k,t} = 1$, or *unproductive* (masked) otherwise. The terms productive and unproductive can also be used to denote unmasked and masked spots, respectively.

The deposition sequence is often a repeated permutation of the alphabet, mainly because of its regular structure and because such sequences maximize the number of distinct subsequences (Chase, 1976). The deposition sequence shown in Fig. 2.1 is a 2.5-time repetition of ACGT, and we thus say that it has two and a half *cycles*.

For cyclic deposition sequences, it is possible to distinguish between two types of embeddings: *synchronous* and *asynchronous*. In the former, 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. With asynchronous embeddings, probes can have any number of nucleotides synthesized in any given cycle, allowing shorter deposition sequences. For this reason, asynchronous embeddings are usually the choice for commercial microarrays. For instance, all GeneChip arrays that we know of can be asynchronously synthesized in 74 steps with 18.5 cycles of TGCA — we will refer to this sequence as the *standard Affymetrix deposition sequence*.

Ideally, the deposition sequence should be as short as possible in order to reduce manufacturing time, cost and probability of errors (Rahmann, 2003). Finding the shortest deposition sequence to synthesize a set of probes is an instance of a classical computer science problem known as the Shortest Common Supersequence problem, which will be the focus of Chapter 9. For the MLP, however, we assume that N is a fixed sequence given as input.

2.1 Problem statement

Given a set of probes $\mathcal{P}$, a geometry of spots $\mathcal{S}$, and a deposition sequence N as specified above, the MLP asks to specify a chip layout (k, ε) that consists of

1. a bijective assignment $k : \mathcal{S} \to \{1, \ldots, n\}$ that specifies a probe index $k(s)$ for each spot s (meaning that $p_{k(s)}$ will be synthesized at s),
2. an assignment $\varepsilon : \{1, \ldots, n\} \to \{0,1\}^T$ specifying an embedding $\varepsilon_k = (\varepsilon_{k,1}, \ldots, \varepsilon_{k,T})$ for each probe index k, such that $N[\varepsilon_k] :\equiv (N_t)_{t:\varepsilon_{k,t}=1} = p_k$,

such that a given penalty function is minimized. We introduce two such penalty functions: total border length and total conflict index.

2.2 Border length

The first formal definition to the problem of unintended illumination was given by Hannenhalli et al. (2002), who defined the *border length* $\mathcal{B}_t$ of a mask M_t as the number of borders separating masked and unmasked spots at synthesis step t, that is, the number of border conflicts in M_t. Formally,

$$\mathcal{B}_t := \frac{1}{2} \cdot \sum_{s,s' \in \mathcal{S}} \mathbb{1}_{\{s \text{ and } s' \text{ are adjacent}\}} \cdot \mathbb{1}_{\{\varepsilon_{k(s),t} \neq \varepsilon_{k(s'),t}\}}. \tag{2.1}$$

where $\mathbb{1}_{\{cond\}}$ is the indicator function that equals 1 if condition *cond* is true, and 0 otherwise. The total border length of a given layout (k, ε) is the sum of border lengths over all masks, that is

$$\mathcal{B}(k, \varepsilon) := \sum_{t=1}^{T} \mathcal{B}_t. \tag{2.2}$$

The *Border Length Minimization Problem* was then defined as the problem of finding a layout minimizing the total border length (Hannenhalli et al., 2002). As an example, the six masks shown in Fig. 2.1 have $\mathcal{B}_1 = 4$, $\mathcal{B}_2 = 3$, $\mathcal{B}_3 = 5$, $\mathcal{B}_4 = 4$, $\mathcal{B}_5 = 8$ and $\mathcal{B}_6 = 9$. The total border length of that layout is 52 (masks 7 to 10 not shown).

Hamming distance. In the rest of this thesis, we will refer to the *Hamming distance* $H(k, k')$ between the embeddings ε_k and $\varepsilon_{k'}$ as the number of synthesis steps in which they differ. Formally,

$$H(k, k') := \sum_{t=1}^{T} \mathbb{1}_{\{\varepsilon_{k,t} \neq \varepsilon_{k',t}\}}. \tag{2.3}$$

Note that $H(k, k')$ gives the number of border conflicts generated when probes with embeddings ε_k and $\varepsilon_{k'}$ are placed in adjacent spots.

2.2.1 Lower Bounds

Lower bounds for the BLMP with synchronous and asynchronous embeddings were given by Kahng et al. (2002), based on a simple graph formulation. Unfortunately, both lower bounds are not tight, and their computation is time-consuming, especially for large chips.

Synchronous embeddings. Let L be a complete directed graph over the set of probes $\mathcal{P}$ with arcs weighted with the Hamming distance between the (unique) embeddings of the corresponding probes.

Since a probe can have at most four neighbors on the chip, we delete all but the four arcs with the least weights of every node. Furthermore, assuming that the chip is a rectangular grid with n_r rows and n_c columns, we delete the heaviest $2 \cdot (n_r + n_c)$ remaining arcs, because the spots on the borders of the chip have less than four neighbors. It is not difficult to see that the cost of any placement must be greather than the total weight of L, and we obtain the following theorem.

Theorem 2.1. *The total arc weight of L is a lower bound on the total border length of the optimum layout with synchronous embeddings.*

Asynchronous embeddings. With asynchronous embeddings, we can construct a similar complete directed graph L'. For the arc weights, however, it is necessary to estimate the minimum number of border conflicts between the two probes (among all of their possible embeddings).

Kahng et al. (2002) observed that the number of bases of a probe p_k that can be "aligned" with bases of $p_{k'}$ cannot exceed the length of $LCS(p_k, p_{k'})$, where $LCS(p_k, p_{k'})$ is the *longest common subsequence* of p_k and $p_{k'}$. Therefore, an arc of L' between probes p_k and $p_{k'}$ can be weighted with $\ell - |LCS(p_k, p_{k'})|$, where ℓ is the length of both probe sequences (assuming probes have the same length).

We can then delete all but the four arcs with the least weights of each probe and, subsequently, the heaviest $2 \cdot (n_r + n_c)$ remaining arcs of L', to obtain the following theorem.

Theorem 2.2. *The total arc weight of L' is a lower bound on the total border length of the optimum layout with asynchronous embeddings.*

2.3 Conflict index

The border length measures the quality of an individual mask or set of masks. With this model, however, it is not possible to know how the border conflicts are distributed among the probes. Ideally, all probes should have roughly the same risk of being damaged by unintended illumination, so that all signals are affected in approximately the same way.

The *conflict index* is a quality measure defined with the aim of estimating the risk of damaging probes at a particular spot (de Carvalho Jr. and Rahmann, 2006b) – it is thus a per-spot or per-probe measure instead of a per-mask measure. Additionally, it takes into account two practical considerations observed by Kahng et al. (2003a):

a) stray light might activate not only adjacent neighbors but also spots 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.

For a proposed layout (k, ε), the conflict index $\mathcal{C}(s)$ of a spot s whose probe $p_{k(s)}$ is synthesized in T masking steps according to its embedding vector $\varepsilon_{k(s)}$ is

$$\mathcal{C}(s) := \sum_{t=1}^{T} \Big(\mathbb{1}_{\{\varepsilon_{k(s),t}=0\}} \cdot \omega(\varepsilon_{k(s)}, t) \cdot \sum_{\substack{s'\text{: neighbor}\\ \text{of } s}} \mathbb{1}_{\{\varepsilon_{k(s'),t}=1\}} \cdot \gamma(s, s') \Big). \tag{2.4}$$

The indicator functions ensure the following conflict condition: During step t, there is a conflict at spot s if and only if s is masked ($\varepsilon_{k(s),t} = 0$) and a close neighbor s' is unmasked ($\varepsilon_{k(s'),t} = 1$), since light directed at s' may somehow reach s. When s is productive, it does not matter if it accidentally receives light targeted at a neighbor, and when s' is unproductive, there is no risk that it damages probes of s.

Function $\gamma(s, s')$ is a “closeness” measure between s and s' (to account for observation a). It is defined as

$$\gamma(s, s') := (d(s, s'))^{-2}, \tag{2.5}$$

where $d(s, s')$ is the Euclidean distance between the spots s and s'. Note that in (2.4), s' ranges over all neighboring spots that are at most three cells away from s (see Fig. 2.2 left), which is in accordance with observation a.

In general, we use the terms *close neighbor* or simply *neighbor* of a spot s to refer to a spot s' that is at most three cells away (vertically and horizontally) from s. In other words, s' is inside a 7×7 region centered around s. This is in contrast to the terms *direct* or *immediate neighbor* of s, used to denote a spot s' that is adjacent to s (in other words, s' shares a common border with s on the chip). Obviously, an immediate neighbor s' is also a close neighbor of s.

The position-dependent weighting function $\omega(\varepsilon, t)$ accounts for the significance of the location inside the probe where the undesired nucleotide is introduced in case of accidental illumination (observation b). It is defined as:

$$\omega(\varepsilon, t) := c \cdot \exp\left(\theta \cdot \lambda(\varepsilon, t)\right) \tag{2.6}$$

where $c > 0$ and $\theta > 0$ are constants, and for $1 \leq t \leq T$,

$$\lambda(\varepsilon, t) := 1 + \min(b_{\varepsilon,t}, \ell_\varepsilon - b_{\varepsilon,t}), \tag{2.7}$$

$$b_{\varepsilon,t} := \sum_{t'=1}^{t} \varepsilon_{t'}, \qquad \ell_\varepsilon := \sum_{t=1}^{T} \varepsilon_t = b_{\varepsilon,T}. \tag{2.8}$$

In other words, ℓ_ε is the length of the final probe specified by ε (equal to the number of ones in the embedding), and $b_{\varepsilon,t}$ denotes the number of nucleotides synthesized up to and including step t.

Note that $\omega(\varepsilon, t)$ grows exponentially from the extremities of the probe to its center (see Fig. 2.2 right). The motivation behind this definition 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 Fig. 2.2 and our experiments, we use probes of length $\ell = 25$, and parameters $\theta = 5/\ell$ and $c = 1/\exp(\theta)$.

Although 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 (Hubbell et al., 1999). The precise effects of this position is not yet fully understood and has been an active topic of research (Binder and Preibisch, 2005).

Conflict Index distance. Many of the algorithms that will be discussed in later chapters were initially developed for border length minimization, and they usually rely on the Hamming distance defined earlier (2.3). We have adapted some of these algorithms to work with conflict index minimization by using the *conflict index distance*, which extends the Hamming distance by taking into account the position inside the probe where the conflict occurs (observation b). We define the conflict index distance $C(k, k')$ between the embeddings ε_k and $\varepsilon_{k'}$ as:

$$C(k, k') := \sum_{t=1}^{T} \Big(\mathbb{1}_{\{\varepsilon_{k,t}=0 \text{ and } \varepsilon_{k',t}=1\}} \cdot \omega(\varepsilon_k, t) + \mathbb{1}_{\{\varepsilon_{k',t}=0 \text{ and } \varepsilon_{k,t}=1\}} \cdot \omega(\varepsilon_{k'}, t) \Big). \tag{2.9}$$

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

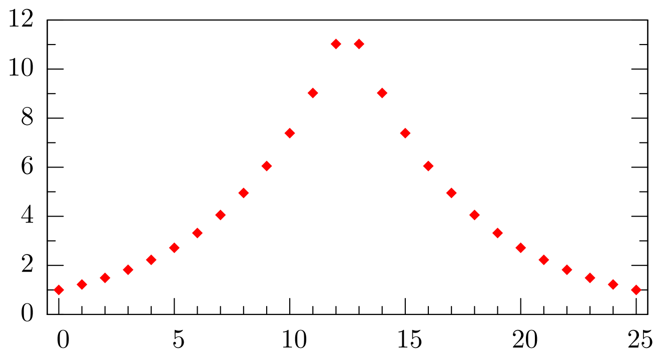

Figure 2.2: Ranges of values for both γ and ω on a typical Affymetrix chip where probes of length 25 are synthesized in 74 masking steps. Left: approximate values of the distance-dependent weighting function $\gamma(s, s')$ for a spot s in the center and close neighbors s'. Right: position-dependent weights $\omega(\varepsilon, t)$ on the y-axis for each value of $b_{\varepsilon,t} \in \{0, \ldots, 25\}$ on the x-axis, assuming $\ell_\varepsilon = 25$.

The conflict index distance $C(k, k')$ can be interpreted as the sum of the conflict indices resulting from placing probes p_k and $p_{k'}$ at hypothetical neighboring spots, ignoring the distance between these spots (note that there is no dependency on γ) and the conflicts generated by other neighbors.

2.3.1 The Meaning of Conflict Index

The conflict index $\mathcal{C}(s)$ attempts to estimate the risk of damaging the probes of a spot s due to the unwanted illumination problem. The definitions of γ and ω given here are an arbitrary choice in an attempt to capture the characteristics of the problem.

The most appropriate choice of γ and ω depend on several attributes of the specific technology used to produce the chips such as the physical properties of the light (intensity, frequency, etc.), the size of the spots, the density of the probes on the chip, the distance between the light source and the mask and the distance between the mask and the chip surface.

Finding the best choice of γ and ω for a particular technology is beyond the scope of this thesis. We note, however, that all algorithms discussed in the next chapters were developed to work independently of the values given by these functions.

2.4 Chip Quality Measures

Most of the algorithms discussed in the next chapters can work with border length as well as conflict index minimization. The relation between these two measures becomes clear if $\gamma(s, s')$ and $\omega(\varepsilon, t)$ are re-defined as follows: Set $\gamma(s, s') := 1$ if s' is a direct neighbor of s, and $:= 0$ otherwise. Also, set $\omega(\varepsilon, t) := 1/2$, so that conflicts always have the same weight, independently of where they occur. Now $\sum_s \mathcal{C}(s) = \sum_{t=1}^{T} \mathcal{B}_t$; that is, total border length is equivalent to the sum of conflict indices for a particular choice of γ and ω. For the choices (2.5) and (2.6), they are not equivalent but still correlated, since a good layout has low border lengths as well as low conflict indices.

To better compare border lengths for chips of different sizes, we divide by the number of internal borders of the chip n_b, which equals $n_b = n_r(n_c - 1) + n_c(n_r - 1)$, assuming that the chip is a rectangular grid with n_r rows and n_c columns. We thus call $1/n_b \cdot \sum_{t=1}^{T} \mathcal{B}_t$ the *normalized border length*. This can be further divided by the number of synthesis steps to give the *normalized border length per mask* $1/(n_b \cdot T) \cdot \sum_{t=1}^{T} \mathcal{B}_t$.

Similarly, it is useful to divide the total conflict index by the number of probes, and we define the *average conflict index* as $1/|\mathcal{P}| \cdot \sum_s \mathcal{C}(s)$.

2.5 How hard is the Microarray Layout Problem?

The MLP appears to be hard because of the super-exponential number of possible arrangements, although no NP-hardness proof is yet known. A formulation of the MLP as a Quadratic Assignment Problem (QAP) was given by de Carvalho Jr. and Rahmann (2006b). The QAP is a classical combinatorial optimization problem that is, in general, NP-hard, and particularly hard to solve in practice (Çela, 1997). 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 MLP 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 conflicts is sought. This is usually referred to as the *placement* phase. Second, a post-placement optimization phase *re-embeds* the probes considering their location on the chip, in such a way that the conflicts with neighboring spots are further reduced. Often, the chip is *partitioned* into smaller sub-regions before the placement phase in order to reduce running times, especially on larger chips.

The most important placement algorithms are surveyed in Chapter 4. The QAP formulation of the MLP is then presented in Chapter 5. Re-embedding algorithms are discussed in Chapter 6, whereas partitioning algorithms are the focus of Chapter 7. Finally, we present recent developments that simultaneously place and re-embed probes in Chapter 8.

Chapter 3

Analysis of Affymetrix Microarrays

General physical structure of GeneChip arrays. Control and special probes, checkerboard patterns on the borders, empty spots.

Probe pairs: perfect match (PM) and mismatch (MM) probes. Rows of PM and MM probes on the chip.

As we mentioned in Chaper 3, all GeneChip arrays that we know of can be asynchronously synthesized in 74 steps with the standard Affymetrix deposition sequence (18.5 cycles of TGCA).

This suggests that only sub-sequences of this sequence can be used as probes on Affymetrix chips. Rahmann (2006) shows that this covers about 98.45% of all 25-mers, however, it seems that Affymetrix uses an even more restrictive probe selection criterion. This is because GeneChip probes always appear in pairs, with the perfect match (PM) and the mismatch (MM) probes being located next to each other (in alternating rows of PM and MM probes), and there is evidence that the embeddings of these probes are aligned in such a way that only the middle bases are not aligned.

Chapter 4

Placement Algorithms

The input for a placement algorithm consists of the deposition sequence N, a set of probes $\mathcal{P}$ (each probe is assumed to have at least one embedding in N) and a geometry of spots $\mathcal{S}$.

The output of a placement algorithm is a one-to-one assignment of probes to spots. If there are more spots than probes to place, we can add enough "empty" probes that do not introduce any conflicts with the other probes (since light is never directed to such spots).

All algorithms discussed in this section assume that an initial embedding of the probes is given, which can be a left-most, right-most, synchronous or otherwise pre-computed embedding — a placement algorithm typically does not change the given embeddings.

4.1 Early approaches

Feldman and Pevzner (1994) were the first to formally address the unintended illumination problem. They showed how an optimal placement can be constructed based on a two-dimensional Gray code. Their work, however, is restricted to *uniform arrays*

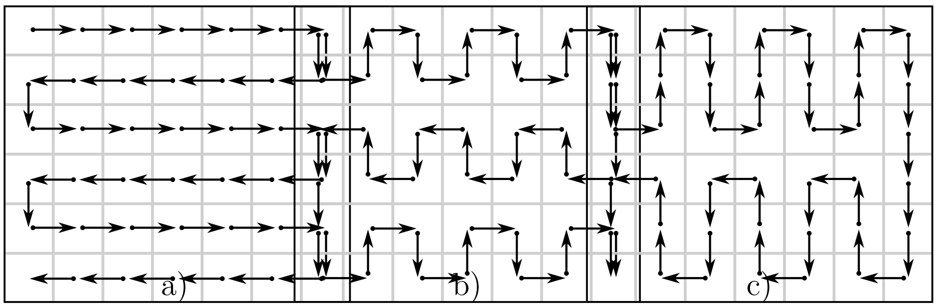

Figure 4.1: Different ways of *threading* probes on a chip. a) Standard row-by-row (0-threading); b) 1-threading; c) 2-threading.

(arrays containing all 4^ℓ probes of a given length ℓ) and synchronous embeddings, being thus of limited practical importance for current microarrays.

The border length problem on arrays of arbitrary probes was first discussed by Hannenhalli et al. (2002). The article reports that the first Affymetrix chips were designed using a heuristic for the traveling salesman problem (TSP). The idea is to build a weighted graph with nodes representing probes, and edges containing the Hamming distances between their embeddings, that is, the number of times their embeddings differ at a particular synthesis step. A TSP tour on this graph is heuristically constructed, 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 (Fig. 4.1a).

Hannenhalli and co-workers studied several threading alternatives, which they collectively called *k-threading* (Fig. 4.1b,c). A k-threading is a variation of the standard row-by-row threading, in which the right-to-left and left-to-right paths are interspaced with alternating upward and downward movements over k sites. (The row-by-row threading can be seen as a k-threading with $k = 0$.) Hannenhalli and co-workers experimentally observed that 1-threading may reduce border length in up to 20% for large chips when compared to row-by-row threading.

A different strategy, called Epitaxial placement, was proposed by Kahng et al. (2002). It was originally designed for chips with synchronous embeddings, but it can be trivially implemented for asynchronous embeddings as well. The algorithm starts by placing 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 whose all four neighbors are full, in which case a probe with the minimum number of border conflicts with the neighbors is placed. Otherwise, all spots s with $i \geq 1$ filled neighbors are examined. For each spot, the algorithm finds a non-assigned probe p whose number of conflicts with the filled neighbors, $c(s, p)$, is minimal, and assigns a cost $C(s, p) = k_i \cdot c(s, p)/i$ for this assignment, where $0 < k_i \leq 1$ are scaling coefficients (the authors propose $k_1 = 1$, $k_2 = 0.8$, and $k_3 = 0.6$). The assignment with minimum $C(s, p)$ is made and the procedure is repeated until all probes have been placed. With this algorithm, Kahng and co-workers claim a further 10% reduction in border conflicts over TSP + 1-threading.

Both the Epitaxial algorithm and the TSP approach have at least quadratic time complexity and hence do not scale well to large chips. This observation motivated the design of two new placement algorithms: Sliding-Window Matching (SWM) and Row-Epitaxial (Kahng et al., 2003a).

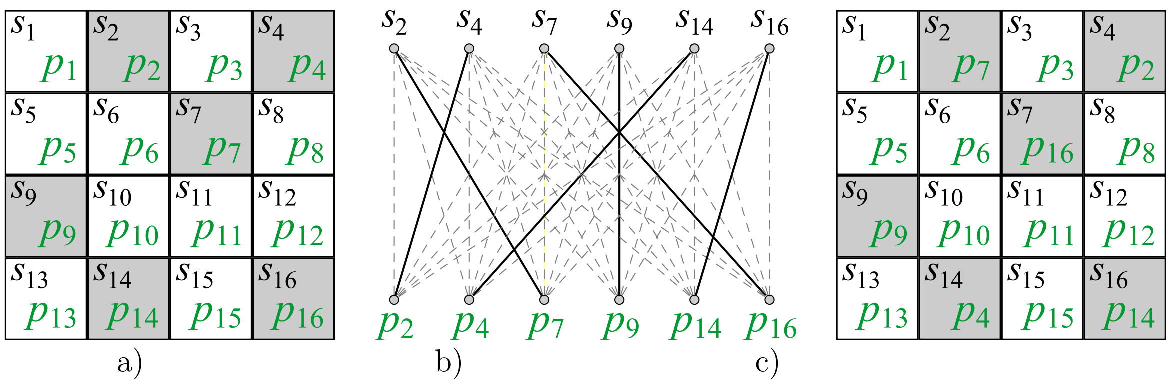

Figure 4.2: Sliding-Window Matching algorithm. a) Initial arrangement of probes $p_1 \dots p_{16}$ inside a 4×4 window and the selected independent set of spots (shaded). b) Bipartite graph and a minimum weight perfect matching (dark edges). c) New arrangement inside the window.

4.2 Sliding-Window Matching

The SWM is not exactly a placement algorithm as it iteratively improves an existing placement that can be constructed, for instance, by TSP + 1-threading, or much simpler, by lexicographically sorting the binary embedding vectors with a linear-time radix sort. The sorting is several times faster but it is also likely to produce a worse initial placement than the TSP, with consecutive embeddings being similar only in their first synthesis steps. This, however, should be of little importance given that this placement is only used as a starting point for the SWM algorithm.

The SWM works inside a window that starts at the top left of the chip and slides from left to right, top to bottom, while maintaining a certain amount of overlap between each iteration. When the window reaches the right-end of the chip, it is re-started at the left-end of the next set of rows, also retaining an overlap with the preceding rows.

At each iteration, the algorithm attempts to reduce the total border length inside the window by relocating some probes (Fig. 4.2a). First, a random maximal independent set of spots is selected, and the probes assigned to these spots are removed. The term independent refers to the fact that selected spots can be re-assigned to probes without affecting the border length of other selected spots. The algorithm creates a bipartite graph with nodes representing the removed probes and the now vacant spots (Fig. 4.2b). The edges of this graph are weighted with the number of border conflicts that are generated by the corresponding assignment. Finally, a minimum weight perfect matching on this graph is computed, and the indicated assignments are made (Fig. 4.2c).

Selecting an independent set of spots ensures that the cost of each new assignment can be computed independently of the other assignments. The SWM was designed for border length minimization and it takes advantage of the fact that, in this model, an independent set of spots can be constructed by selecting sites that are not immediate neighbors (spots that do not share a common border). SWM can be adapted for conflict index minimization (to our knowledge, this has not been implemented) by using larger windows containing relatively sparse independent sets. Therefore several random independent sets should be constructed before moving the window.

4.3 Row-Epitaxial

The Row-Epitaxial algorithm is a variant of the Epitaxial algorithm with two main differences introduced to improve scalability: i) spots are filled in a pre-defined order, namely, from top to bottom, left to right, and ii) only a limited number Q of probes are considered for filling each spot.

Like SWM, Row-Epitaxial improves an initial placement that is constructed by TSP + 1-threading or Radix-sort + 1-threading. For each spot s of the chip, it looks at the next Q probes that lie in close proximity (to the right or below s), and swaps the current probe of s with the probe that generates the minimum number of border conflicts with the top and left neighbors of s. Row-Epitaxial can be adapted to conflict index minimization by restricting the computation of the conflict index of s to those neighboring probes that are to the left or above s (those which have already found their final positions).

Fig. 4.3 shows computational results for normalized border length and average conflict index for various chip dimensions and different values of Q. The running time of Row-Epitaxial is $O(Qn)$, i.e., linear in the chip size, where Q is a user-defined constant. In this way, solution quality can be traded for running time: More candidates yield better layouts but also demand more time. For border length minimization, increasing Q above 10 000 has little positive effect.

According to experiments conducted by Kahng et al. (2003a), Row-Epitaxial is the best known large-scale placement algorithm, achieving up to 9% reduction in border length over the TSP + 1-threading, whereas SWM achieves slightly worse results but requires significantly less time.

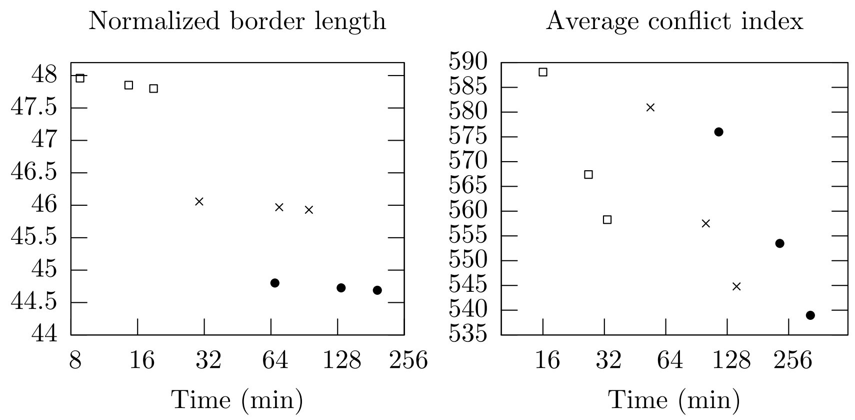

Figure 4.3: Trade-off between solution quality and running time with the Row-Epitaxial algorithm, on random chips of dimensions 200 × 200 (□), 300 × 300 (×) and 500 × 500 (•). The number Q of candidates per spot are 10 000, 20 000, and 30 000 from left to right. Layouts are measured by normalized border length (left) and average conflict index (right).

Chapter 5

MLP and the Quadratic Assignment Problem

Chapter 6

Re-embedding Algorithms

Once the probes have been placed, conflicts can be further reduced by re-embedding the probes without changing their locations. All re-embedding algorithms presented in this section are based on the Optimum Single Probe Embedding (OSPE) introduced by Kahng et al. (2002). OSPE is a dynamic programming for computing an optimum embedding of a single probe with respect to its neighbors, whose embeddings are considered as fixed. The algorithm was originally developed for border length minimization but here we present a more general form designed for the conflict index model (de Carvalho Jr. and Rahmann, 2006a).

6.1 Optimum Single Probe Embedding

The OSPE algorithm can be seen as a special case of a global alignment between a probe sequence p of length ℓ and the deposition sequence N of length T, disallowing mismatches and gaps in N. We assume that p is placed at spot s, and that we know the embeddings of all probes placed at spots near s.

The optimal embedding of p into N is built by determining the minimum cost of embedding a prefix of p into a prefix of N: We use an $(\ell + 1) \times (T + 1)$ matrix D, where $D[i, t]$ is defined as the minimum cost of an embedding of $p[1..i]$ into $N[1..t]$. The cost is the sum of conflicts induced by the embedding of $p[1..i]$ on its neighbors, plus the conflicts suffered by $p[1..i]$ because of the embeddings of its neighbors.

We can compute the value for $D[i, t]$ by looking at two previous entries in the matrix: $D[i, t-1]$ and $D[i-1, t-1]$. The reason is that $D[i, t]$ is the minimum cost of embedding $p[1..i]$ up to the t-th synthesis step, which can only be obtained from the previous step $(t-1)$ by either masking or unmasking spot s at step t.

If s is productive at step t, base N_t is appended to $p[1..i-1]$; this is only possible if $p[i] = N[t]$. In this case a cost U_t is added for the risk of damaging probes at neighboring spots s'. We know that $p[1..i-1]$ can be embedded in $N[1..t-1]$ with optimal cost $D[i-1, t-1]$. Hence, the minimum cost at step t, if s is productive, is $D[i-1, t-1] + U_t$. According to the conflict index model,

$$U_t := \sum_{\substack{s'\text{: neighbor}\\ \text{of } s}} \mathbb{1}_{\{\varepsilon_{k(s'),t}=0\}} \cdot \omega(\varepsilon_{k(s')}, t) \cdot \gamma(s', s).$$

If s is masked at step t, no base is appended to $p[1..i]$, but a cost $M_{i,t}$ must be added for the risk of damaging p (by light directed at neighboring spots s'). Since $D[i, t-1]$ is the minimum cost of embedding $p[1..i]$ in $N[1..t-1]$, the minimum cost up to step t, if s is unmasked, is $D[i, t-1] + M_{i,t}$.

Note that $M_{i,t}$ depends on the number of bases p already contains (that is, on i): Each unmasked neighboring spot s' generates a conflict on p with cost $\gamma(s, s') \cdot c \cdot \exp[\theta \cdot (1 + \min\{i, \ell - i\})]$, in accordance with (2.6)–(2.8). Thus

$$M_{i,t} := c \cdot \exp[\theta \cdot (1 + \min\{i, \ell - i\})] \cdot \sum_{\substack{s'\text{: neighbor}\\ \text{of } s}} \mathbb{1}_{\{\varepsilon_{k(s'),t}=1\}} \cdot \gamma(s, s').$$

Finally, $D[i, t]$ is computed as the minimum cost of the possible actions,

$$D[i,t] := \begin{cases} \min\{\, D[i, t-1] + M_{i,t},\ D[i-1, t-1] + U_t \,\} & \text{if } p[i] = N[t], \\ D[i, t-1] + M_{i,t} & \text{if } p[i] \neq N[t]. \end{cases}$$

The first column of D is initialized as follows: $D[0, 0] = 0$ and $D[i, 0] = \infty$ for $0 < i \leq \ell$, since no probe of length $\ell > 0$ can be embedded into an empty deposition sequence. The first row is initialized by setting $D[0, t] = D[0, t-1] + M_{0,t}$ for $0 < t \leq T$.

If we assume that costs U_t and $M_{i,t}$ can be computed in constant time, the time complexity of the OSPE algorithm is $O(\ell T)$ since there are $O(\ell T)$ entries in D to compute. The algorithm can be rather time-consuming in the general form presented here, since we have to look at the embeddings of up to 48 neighbors around s. Naturally, it runs much faster for border length minimization, since there are only four neighbors, and there are neither position-dependent (ω) nor distance-dependent (γ) weights to compute. In any case, a few optimizations significantly reduce the running time. For instance, in each row, only the columns between the left-most and the right-most embedding of p in N need to be computed.

Once D is computed, the minimum cost is $D[\ell, T]$, and an optimal embedding of p into N can be constructed by tracing a path from $D[\ell, T]$ back to $D[0, 0]$, similarly to the procedure used to build an optimal global alignment. This takes $O(T)$ time.

6.2 Re-embedding algorithms

The OSPE algorithm is the basic operation of several post-placement optimization algorithms: Greedy, Batched Greedy and Chessboard (Kahng et al., 2002), and Sequential (Kahng et al., 2003b). Their main difference lies in the order in which the probes are re-embedded.

Since OSPE never increases the amount of conflicts in the region around the re-embedded probe, optimization algorithms can execute several re-embedding operations without risk of worsening the current solution. Moreover, probes can be re-embedded several times since new improvements may be possible once neighbors are changed. In fact, the following algorithms work in repeating cycles of optimization until no more improvements are possible (when a local optimal solution is found), or until improvements drop below a given threshold.

The Greedy algorithm uses OSPE to compute, for each spot of the chip, the maximum reduction of border conflicts achievable by optimally re-embedding its probe. It then selects a spot s with the highest gain (reduction of conflicts) and re-embeds its probe optimally, updating the gains of affected neighboring spots.

A faster version of this algorithm, called Batched Greedy, pre-selects several spots for re-embedding and thus sacrifices its greedy nature by postponing the update of gains.

The Chessboard optimization is based on the fact that a chip can be bi-colored like a chessboard, in such a way that the embeddings of probes located on white spots are independent of those placed on black spots (with respect to border length), and vice-versa. The Chessboard uses this coloring to alternate the optimal re-embedding of probes located on black and white spots.

The Sequential optimization is the simplest algorithm among the four. It proceeds spot by spot, from top to bottom, from left to right, re-embedding each probe optimally. Once the end of the array is reached, it restarts at the top left for the next iteration.

Surprisingly, the Sequential algorithm achieves the greatest reduction of border conflicts with a running time comparable to Batched Greedy, the fastest among the four. All re-embedding algorithms mentioned here were initially developed for border length minimization, but they can all be applied to the conflict index model as well. For the Chessboard optimization, $4 \times 4 = 16$ colors must be used instead of 2.

Chapter 7

Partitioning Algorithms

We mentioned earlier that the MLP is usually approached in two phases: placement and re-embedding. The placement, however, is sometimes preceded by a *partitioning* phase, which breaks the problem into smaller sub-problems that are easier to manage. This is especially helpful for placement algorithms with non-linear time or space complexities that are otherwise unable to handle very large chips.

A partitioning algorithm divides the set of probes $\mathcal{P}$ into smaller subsets, and assigns them to defined regions of the chip. Each region can then be treated as an independent chip (and processed by a placement algorithm) or recursively partitioned. Linear-time placement algorithms may also benefit from a partitioning since probes with similar embeddings are typically assigned to the same region (Row-Epitaxial, for instance, is more likely to find good candidates for filling a spot).

We describe four partitioning algorithms: 1-Dimensional Partitioning, 2-Dimensional Partitioning, Centroid-based Quadrisection (CQ), and Pivot Partitioning (PP). Like placement algorithms, they assume that an initial (left-most, right-most, synchronous or otherwise pre-computed) embedding of the probes is given. Pivot Partitioning is the only algorithm that modifies these embeddings. As we shall see, 1-D and 2-D Partitioning generate a few masks with extremely few conflicts, leaving the remaining masks with high levels of conflicts that are difficult to handle. CQ and PP offer a more uniform optimization over all masks. Results of de Carvalho Jr. and Rahmann (2006a) indicate that PP produces better layouts than CQ on large chips.

Partitioning is a compromise in solution quality since it restricts the space of solutions and may lead to conflicts at partition borders. However, it can improve solution quality in practice when the placement algorithm cannot handle large regions well. It is not advisable to perform too many levels of partitionings because smaller sub-regions mean less freedom for optimization during placement. The right balance depends on both the placement algorithm and the partitioning algorithm.

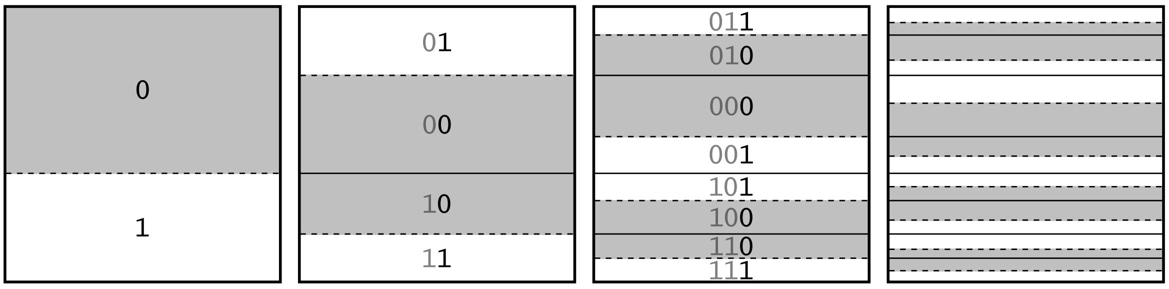

Figure 7.1: First four levels of 1-Dimensional Partitioning. Dashed lines show the divisions performed in each step; solid lines indicate regions delimited in previous steps (there are no border conflicts between spots separated by solid lines). Masked (shaded) regions are labeled "0", unmasked (white) regions are labeled "1". This labeling forms a Gray code (shown in the first three steps only).

7.1 1-Dimensional Partitioning

1-Dimensional Partitioning divides the set of probes based on the state of their embeddings at a particular synthesis step. It starts by creating two subsets of $\mathcal{P}$:

$$\mathcal{P}_0 = \{p_k \in \mathcal{P} | \varepsilon_{k,1} = 0\}, \qquad \mathcal{P}_1 = \{p_k \in \mathcal{P} | \varepsilon_{k,1} = 1\}.$$

In other words, $\mathcal{P}_0$ contains all probes whose embeddings are unproductive during the first synthesis step, whereas $\mathcal{P}_1$ contains the probes with productive embeddings. The chip is then divided into two horizontal bands, proportionally to the number of probes in $\mathcal{P}_0$ and $\mathcal{P}_1$, so each band accommodates one subset of $\mathcal{P}$.

This procedure is recursively applied to each band, using the the next synthesis steps to further divide each subset of probes. For instance, the following subsets of $\mathcal{P}_0$ and $\mathcal{P}_1$ are created during step $t = 2$:

$$\mathcal{P}_{00} = \{p_k \in \mathcal{P}_0 | \varepsilon_{k,2} = 0\}, \qquad \mathcal{P}_{01} = \{p_k \in \mathcal{P}_0 | \varepsilon_{k,2} = 1\},$$

$$\mathcal{P}_{10} = \{p_k \in \mathcal{P}_1 | \varepsilon_{k,2} = 0\}, \qquad \mathcal{P}_{11} = \{p_k \in \mathcal{P}_1 | \varepsilon_{k,2} = 1\}.$$

The next assignments of subsets to the upper or lower band of their regions are made in such a way that regions with the same "state" — productive (unmasked) or unproductive (masked) — are joined as far as possible, resulting in masks that consist of alternating layers of masked and unmasked spots. This process is illustrated in Fig. 7.1, where at each step t, a band is labeled "0" when its embeddings are unproductive, and "1" when its embeddings are productive. The resulting binary numbers from top to bottom form a Gray code, i.e., two successive numbers differ in only one bit.

The Gray code highlights an interesting property of 1-D Partitioning. After d levels of partitionings (based on steps 1 to d), the embeddings of any two immediate neighbors differ among the first d steps in at most one step. As a result, masks $M_1 \dots M_d$ exhibit a layered structure that effectively reduces border conflicts.

Unfortunately, the Gray code is disrupted as soon as a region cannot be divided (because all probes of that region are, for instance, masked at a particular step). This will certainly happen as several binary numbers are unlikely to be substrings of embeddings (think of, for example, a long run of zeros).

Moreover, 1-D Partitioning can optimize only a limited number of masks because the sub-regions soon become too narrow to be further divided. The maximum *partitioning depth* d_{max} is primarily limited by the number of rows in the chip. In practice, since regions are likely to be unevenly divided, d_{max} varies between regions. The algorithm can also be configured to stop partitioning a region once its dimensions drop below a given threshold.

1-D Partitioning is easier to implement if the partitionings always produce rectangular regions (i.e., splitting a row between two regions is not allowed). In order to force an exact division of a region, however, it might be necessary to move a few probes from one subset of probes to the other one.

For example, imagine that a chip with $|\mathcal{P}| = 900$ probes, $n_r = 30$ rows and $n_c = 30$ columns is to be partitioned based on the state of the embeddings at the first synthesis step, resulting in sub-sets $\mathcal{P}_0$ and $\mathcal{P}_1$ with, say, 638 and 262 probes, respectively. The chip must thus be divided into two sub-regions, the upper one containing $[30 \cdot 638/900] = 21$ rows and the lower one with $[30 \cdot 262/900] = 9$ rows ($[x]$ is x rounded to the nearest integer). The problem is that the upper region then contains $21 \cdot 30 = 630$ spots but it has to accommodate 638 probes, whereas the lower region contains $9 \cdot 30 = 270$ spots but only 262 probes. The solution is to (arbitrarily) move 8 probes from $\mathcal{P}_0$ to $\mathcal{P}_1$, which, results in some imperfections in the layers of the corresponding mask (a few masked spots in a region of unmasked spots, for instance).

7.2 2-Dimensional Partitioning

The 2-Dimensional Partitioning algorithm extends the idea of 1-D Partitioning to two dimensions, with the potential of optimizing twice as many masks. The algorithm is similar: $\mathcal{P}$ is divided into subsets based on the state of the embeddings at a particular synthesis step. The difference is that 2-D Partitioning alternates horizontal and vertical divisions of regions, and that the assignments of probes to regions obey a two-dimensional Gray code (Fig. 7.2).

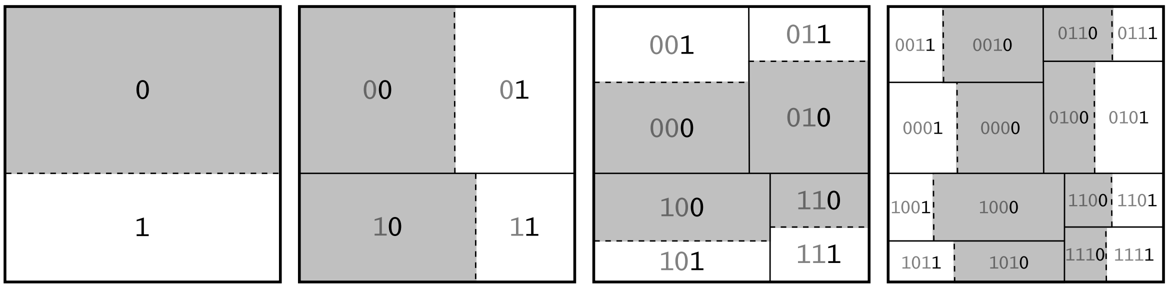

Figure 7.2: First four levels of 2-Dimensional Partitioning. Dashed lines show the divisions performed in each step; solid lines indicate regions delimited in previous steps. Masked regions are labeled with "0", unmasked regions with "1"; this labeling forms an approximation to a two-dimensional Gray code.

In a 2-D Gray code, two neighboring numbers differ in at most one bit. Thus, regions whose embeddings are at the same state (productive or unproductive) are joined as far as possible.

If regions were always equally divided, 2-D Partitioning would have the same property as 1-D Partitioning: After d levels of partitionings (based on steps 1 to d), the embeddings of any two immediate neighbors would differ among the first d steps in at most one step. However, this is not always the case since 2-D Partitioning is likely to create regions with different dimensions, forcing some regions to share a border with more than its four natural neighbors (for example, region "1100" in Fig. 7.2 borders with "0101" and "1111").

So far we have described both 1-D and 2-D Partitionings using the state of the first d synthesis steps to divide the set of probes. The result of this approach is that, while the first masks are optimized, the remaining masks are left with high levels of border conflicts; we call this a *left-most mask optimization.*

However, a defect in the middle of the probe is more harmful than in its extremities, so it is more important to optimize the central masks, which synthesize the middle bases. Thus we partition the chip based on the following sequence of synthesis steps, assuming that T is even and d is odd: $T/2, (T/2) \pm 1, (T/2) \pm 2, \ldots, (T/2) \pm \lfloor d/2 \rfloor$; we call this a *centered mask optimization.*

For left-most optimization, it makes sense to embed the probes in a left-most fashion in order to reduce conflicts in the last masks (which are not optimized by the partitioning); the left-most embeddings reduce the number of unmasked spots in the last steps, resulting in masks that largely consist of masked spots. Similarly, centered mask optimization produces better results with *centered embeddings.* A centered embedding is constructed by shifting a left-most embedding to the right so that the number of masked steps to the left of the first productive step approximately equals the number of masked steps to the right of the last productive step.

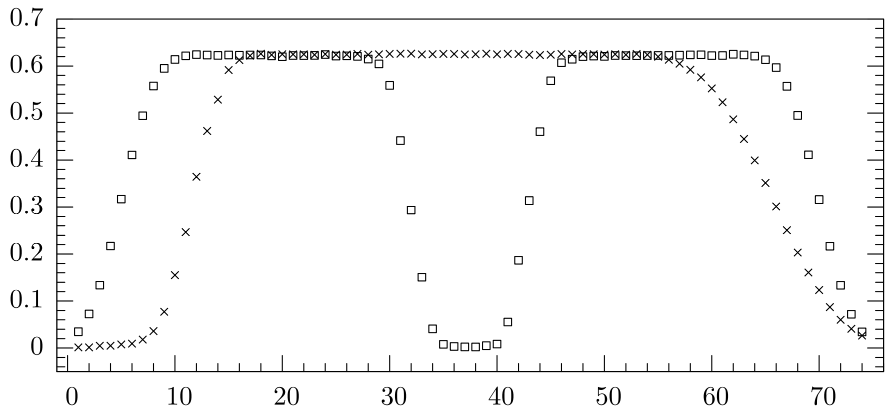

Figure 7.3: Normalized border length (on the y-axis) per masking step (on the x-axis) of a layout produced by 2-Dimensional Partitioning for a $1\,000 \times 1\,000$ chip with random probe sequences (embedded in the standard 74-step Affymetrix deposition sequence). Partitioning stops when a region becomes smaller than 64×64; Row-Epitaxial is used for the placement. (×) Left-most mask optimization with left-most embeddings; (□) centered mask optimization with centered embeddings.

Fig. 7.3 shows the results of 2-D Partitioning on a $1\,000 \times 1\,000$ chip with both optimizations. For left-most mask optimization, we obtain a normalized border length of 33.89 (up to approximately 0.6 per step). For centered mask optimization, the normalized border length improves slightly to 33.59. The average conflict index (not shown in the figure) for left-most mask optimization is 571.8; for centered mask optimization, it improves considerably to 383.5 because of the higher weight of the middle bases.

7.3 Centroid-based Quadrisection

Centroid-based Quadrisection or CQ (Kahng et al., 2003b) employs a different criterion for dividing the set of probes and a different approach for partitioning. At each iteration, a region R is quadrisectioned into R_1, R_2, R_3, and R_4. Each sub-region R_i is associated with a selected probe $p_{c_i} \in \mathcal{P}$, called *centroid*, that is used to guide the assignment of the remaining probes to the sub-regions.

A centroid is a representative of its region; it should symbolize the "average embedding" in that region. The remaining probes $p_k \in \mathcal{P} \setminus \{p_{c_1}, p_{c_2}, p_{c_3}, p_{c_4}\}$ are compared to each centroid and assigned to the sub-region R_i whose centroid's embedding ε_{c_i} has minimum $H(k, c_i)$, where $H(k, k')$ is the Hamming distance between the embeddings ε_k of p_k and $\varepsilon_{k'}$ of $p_{k'}$ (i.e., the number of steps in which ε_k and $\varepsilon_{k'}$ differ).

In order to improve the clustering of similar probes, the four centroids should be very different from each other. The following heuristic is used: First, a probe index c_1 is randomly selected from $\{1, \ldots, |\mathcal{P}|\}$. Then, a probe index $c_2 \neq c_1$ maximizing

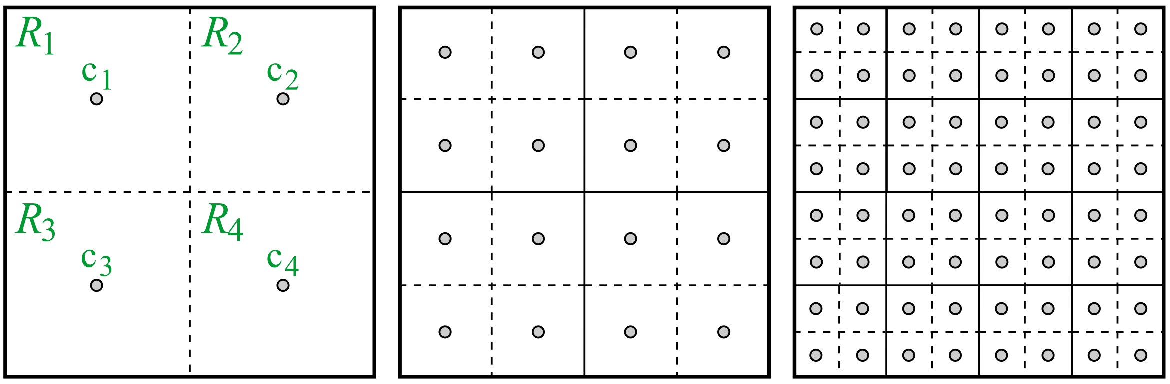

Figure 7.4: First three levels of Centroid-base Quadrisection Partitioning. Dashed lines show the divisions performed in each step; solid lines indicate regions delimited in previous steps. The centroids of each partition $R_1 \dots R_4$ are represented by small circles (labeled with $q_1 \dots q_4$ in the first step).

$H(c_2, c_1)$ is selected. Similarly, c_3 maximizing $H(c_3, c_1) + H(c_3, c_2)$ and c_4 maximizing $H(c_4, c_1) + H(c_4, c_2) + H(c_4, c_3)$ are selected. The assignment of centroids to the quadrisections of the chip is arbitrary.

In order to recover from a possibly bad choice of centroids, one can use a "multi-start heuristic", running the centroid selection procedure several times (using different "seeds" for c_1), and keeping those that lead to the best partitioning (partitioning quality is measured by the sum of Hamming distances of probe embeddings to their corresponding centroid embeddings).

The partitioning continues recursively until a pre-defined depth has been reached.

CQ was developed for border length minimization, but can be adapted for conflict index minimization by using the *conflict index distance* $C(k, k')$ instead of the Hamming distance $H(k, k')$ between the embeddings ε_k and $\varepsilon_{k'}$.

7.4 Pivot Partitioning: Merging partitioning and re-embedding

Pivot Partitioning or PP (de Carvalho Jr. and Rahmann, 2006a) is, to a certain extent, similar to CQ: Sub-regions are recursively associated with special probes p_{c_i}, here called *pivots* instead of centroids, that are used to guide the assignment of the other probes to the sub-regions. The main differences between PP and CQ are as follows.

Instead of quadrisectioning the chip, PP creates sub-regions by alternating horizontal and vertical divisions (like 2-D Partitioning). The advantage is that regions are divided proportionally to the size of each subset of probes, so they are not required to have the same size. Furthermore, for each partitioning level, only two pivots need to be selected.

Another distinction is motivated by the fact that different probes have different numbers of embeddings, ranging from a single one to several millions. Probes with more embeddings can more easily adapt to the other probes, that is, they are more likely to have an embedding with fewer conflicts to fill a particular spot than a probe that has only a limited number of embeddings. For this reason, PP uses probes with a single embedding (or few embeddings) as pivots, and chooses the other probes' embeddings and region assignments accordingly.

Indeed, the most important feature of PP is the simultaneous embedding and assignment of probes to sub-regions. Let $M(k, c_i)$ denote the minimum conflict index distance $C(k, c_i)$, as defined in (2.9), over all embeddings of p_k; we call it the *minimum conflict index distance* between probes p_k and p_{c_i}. It can be efficiently computed with a variant of the OSPE algorithm that ignores the location of the probes and the distance-dependent weights γ. Now, a non-pivot probe p_k is assigned to the region R_i whose pivot p_{c_i} has minimum $M(k, q_i)$ over $i = 1, 2$. Pivot Partitioning continues recursively up to a pre-defined depth. Finally, each probe is embedded to minimize conflicts with its assigned pivot.

Chapter 8

Merging Placement and Re-embedding

The problem with the traditional “place and re-embed” approach is that the arrangement of probes on the chip is decided based on embeddings that are likely to change during the re-embedding phase. Intuitively, better results should be obtained when the placement and embedding phases are considered simultaneously instead of separately. However, because of the generally high number of embeddings of each single probe, it is not easy to design algorithms that efficiently use the additional freedom and run reasonably fast in practice.

We describe Greedy+, the first placement algorithm that simultaneously places and re-embeds the probes, and compare it with Row-Epitaxial, the best known large-scale placement algorithm.

8.1 Greedy+

The goal is to design an algorithm that is similar to Row-Epitaxial, so that we can make a better assessment of the gains resulting from merging the placement and re-embedding phases.

Greedy+ fills the spots row-by-row, from left to right, in a greedy fashion, similarly to Row-Epitaxial. Also, for each spot s, it looks at Q probe candidates and chooses the one that can be placed at s with minimum cost. The difference is that we now consider all possible embeddings of a candidate p instead of only p's initial embedding. This is done by temporarily placing p at s and computing its optimal embedding with respect to the already-filled neighbors of s (using OSPE from Sec. 6.1).

Compared to Row-Epitaxial, Greedy+ spends more time evaluating each probe candidate p for a spot s. While Row-Epitaxial takes $O(T)$ time to compute the conflict index or the border length resulting from placing p at s, Greedy+ requires $O(\ell T)$ time since it uses OSPE (recall that ℓ is the probe length and T is the length of the deposition sequence). To achieve a running time comparable to Row-Epitaxial, we must therefore consider lower candidate numbers Q.

There are a few optimizations that reduce the time spent with OSPE computations when several probe candidates are examined in succession for the same spot. First, we note that if two probe candidates p and p' share a common prefix of length l, the first $l + 1$ rows of the OSPE dynamic programming matrix D will be identical. In other words, if we have calculated the minimum cost of p, we can speed up the calculation of the minimum cost of p' by skipping the first $l + 1$ rows of D.

In order to fully exploit this fact, we examine the probes in lexicographical order so that we maximize the length of the common prefix between two consecutive candidates. We keep a doubly-linked list of probes and remove a probe p from the list when it is placed. For the next spot to be filled, we look at Q probes in the list around p's former position, e.g., at $Q/2$ probes to the left and to the right of p.

Second, the U_t costs of OSPE need to be computed only once for a given spot s since U_t does not depend on the probe placed at s. Thus, in order to examine another candidate, we only need to recompute the $M_{i,t}$ costs.

Finally, once we know that a probe candidate p can be placed at s with minimum cost C, we can stop the OSPE computation for another candidate p' as soon as all values in a row of D are greater than or equal to C.

8.2 Results

We compare the results of Greedy+ with Row-Epitaxial. To be fair, since Row-Epitaxial is a traditional placement algorithm that does not change the embeddings, we need to compare the layouts obtained by both algorithms after a re-embedding phase. For this task we use the Sequential algorithm (Sec. 6.2) with thresholds of $W = 0.1\%$ for border length minimization and $W = 0.5\%$ for conflict index minimization, so that the algorithm stops as soon as the improvement in one iteration drops below W.

Table 8.1 shows the total border length and the average conflict index of layouts produced by both algorithms on two chips with dimensions 335×335 and 515×515, filled with probes randomly selected from existing GeneChip arrays (E.Coli Genome 2.0 and Maize Genome, respectively). Probes are initially left-most embedded into the standard 74-step Affymetrix deposition sequence $\{\text{TGCA}\}^{18}\text{TG}$. The parameter Q is chosen differently for both algorithms so that the running time is approximately comparable (e.g., for border length minimization, $Q = 350$ for Greedy+ corresponds to $Q = 10\,000$ for Row-Epitaxial). We make the following observations.

First, increasing Q linearly increases placement time, while only marginally improving chip quality for border length minimization.

Table 8.1: Normalized border length (NBL) and average conflict index (ACI) of layouts produced by Row-Epitaxial and Greedy+ placement (Pl.), followed by Sequential re-embedding (Re-emb.) with thresholds $\boldsymbol{W = 0.1\%}$ for border length minimization and $\boldsymbol{W = 0.5\%}$ for conflict index minimization. $\boldsymbol{Q}$ is the number of probe candidates considered for each spot during placement. Running times are given in seconds.

Border length min.	335×335 (E.Coli)		515×515 (Maize)	
Row-Epitaxial. Q	10 K	20 K	10 K	20 K
Time (Pl. + Re-emb.)	629 + 9	1211 + 9	3333 + 38	6806 + 38
NBL (Pl.)	34.11	33.81	33.11	32.87
NBL (Pl. + Re-emb.)	33.93	33.66	32.95	32.73
Greedy+. Q	350	700	350	700
Time (Pl. + Re-emb.)	596 + 12	1158 + 12	2633 + 53	4974 + 53
NBL (Pl.)	33.79	33.22	33.07	32.38
NBL (Pl. + Re-emb.)	33.53	32.98	32.82	32.16

Conflict index min.	335×335 (E.Coli)		515×515 (Maize)	
Row-Epitaxial. Q	5 K	10 K	5 K	10 K
Time (Pl. + Re-emb.)	930 + 1169	1732 + 1167	4082 + 4424	7856 + 4415
ACI (Pl.)	584.92	544.93	604.04	574.68
ACI (Pl. + Re-emb.)	544.23	514.10	554.87	532.74
Greedy+. Q	200	300	200	300
Time (Pl. + Re-emb.)	522 + 788	685 + 788	2131 + 2926	2757 + 2930
ACI (Pl.)	462.52	450.15	459.38	446.76
ACI (Pl. + Re-emb.)	458.02	445.98	454.84	442.55

Second, re-embedding runs very quickly for border length minimization, even on the larger chip. For conflict index minimization, the time for the re-embedding phase exceeds the time for the placement phase for both algorithms.

Finally, Greedy+ always produces better layouts in the same amount of time (or less) while looking at fewer probe candidates. In particular, for conflict index minimization on the 515×515 chip with $Q = 5\,000$ resp. 200, Greedy+ and Sequential improve the average conflict index by 18% (from 554.87 to 454.84) and need only 60% of the time, compared to Row-Epitaxial and Sequential.

Chapter 9

Shortest Common Supersequence

Chapter 10

Discussion

This thesis makes several contributions to the field of....

We introduce the *conflict index*...

In Chapter X we present a algorithm...

We are not aware of any previous work that combines placement and re-embedding, even our results showed that such an approach has obvious benefits...

We have surveyed algorithms for the microarray layout problem (MLP), divided into placement, (re-)embedding, and partitioning algorithms. Because of the super-exponential number of possible layouts and the relation to the quadratic assignment problem (QAP), we cannot expect to find optimal solutions. Indeed, the algorithms we present are heuristics with an emphasis on good scalability and ideally a user-controllable trade-off between running time and solution quality, albeit without any known provable guarantees.

Among the presented approaches, two recent ones (Pivot Partitioning and Greedy+) indicate that the traditional "place first and then re-embed" approach can be improved upon by merging the partitioning/placement and (re-)embedding phases. Ongoing work will show the full potential of such combined approaches.

As a suggestion for further work, we note the needed for improving the selection of probe candidates considered for filling each spot. For example, instead using a sorted list of probes, one could use a TSP tour like the early algorithms described in Sec. 4.1. However, it is not clear if the more time-consuming TSP approach will pay off (instead, we could use this extra time to look at more candidates).

An alternative that sounds interesting would be to build some kind of "clustering" of the probes, perhaps based on a graph or a tree, in such a way that we can find similar probes more easily and spend time on candidates that are more likely to produce less conflicts.

Question: why PM and MM probes are placed together and aligned except for the middle base?

Bibliography

H. Binder and S. Preibisch. Specific and nonspecific hybridization of oligonucleotide probes on microarrays. *Biophys J*, 89(1):337–352, Jul 2005. doi: 10.1529/biophysj.104.055343. URL `http://dx.doi.org/10.1529/biophysj.104.055343`.

E. Çela. *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, 1997.

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

S. A. de Carvalho Jr. and S. Rahmann. Improving the layout of oligonucleotide microarrays: Pivot partitioning. In *Algorithms in Bioinformatics (Proceedings of WABI)*, volume 4175 of *Lecture Notes in Computer Science*, pages 321–332. Springer, 2006a. doi: 10.1007/11851561. URL `http://www.springerlink.com/content/h9r4n673058032xm`.

S. A. de Carvalho Jr. and S. Rahmann. Microarray layout as quadratic assignment problem. In D. Huson, O. Kohlbacher, A. Lupas, K. Nieselt, and A. Zell, editors, *Proceedings of the German Conference on Bioinformatics*, volume P-83 of *Lecture Notes in Informatics (LNI)*, pages 11–20. Gesellschaft für Informatik, 2006b.

W. Feldman and P. Pevzner. Gray code masks for sequencing by hybridization. *Genomics*, 23(1):233–235, 1994. doi: 10.1006/geno.1994.1482. URL `http://dx.doi.org/10.1006/geno.1994.1482`.

S. P. Fodor, J. L. Read, M. C. Pirrung, L. Stryer, A. T. Lu, and D. Solas. Light-directed, spatially addressable parallel chemical synthesis. *Science*, 251(4995):767–773, 1991.

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

E. A. Hubbell, M. S. Morris, and J. L. Winkler. Computer-aided engineering system for design of sequence arrays and lithographic masks. United States Patent number 5,856,101, Jan 1999.

A. Kahng, I. Mandoiu, P. Pevzner, S. Reda, and A. Zelikovsky. Border length minimization in DNA array design. In R. Guigó and D. Gusfield, editors, *Algorithms in Bioinformatics (Proceedings of WABI)*, volume 2452 of *Lecture Notes in Computer Science*, pages 435–448. Springer, 2002. URL `http://www.springerlink.com/content/pqp7c7emyk7gmx3u`.

A. B. Kahng, I. Mandoiu, P. Pevzner, S. Reda, and A. Zelikovsky. Engineering a scalable placement heuristic for DNA probe arrays. In *Proceedings of the seventh annual international conference on research in computational molecular biology (RECOMB)*, pages 148–156. ACM Press, 2003a. doi: http://doi.acm.org/10.1145/640075.640095.

A. B. Kahng, I. Mandoiu, S. Reda, X. Xu, and A. Z. Zelikovsky. Evaluation of placement techniques for DNA probe array layout. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design (ICCAD)*, pages 262–269. IEEE Computer Society, 2003b. doi: http://dx.doi.org/10.1109/ICCAD.2003.65.

S. Rahmann. The shortest common supersequence problem in a microarray production setting. *Bioinformatics*, 19(Suppl 2):ii156–ii161, Oct 2003.

S. Rahmann. Subsequence combinatorics and applications to microarray production, DNA sequencing and chaining algorithms. In M. Lewenstein and G. Valiente, editors, *Combinatorial Pattern Matching (CPM)*, volume 4009 of *LNCS*, pages 153–164, 2006.

S. Singh-Gasson, R. D. Green, Y. Yue, C. Nelson, F. Blattner, M. R. Sussman, and F. Cerrina. Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nat Biotechnol*, 17(10):974–978, Oct 1999. doi: 10.1038/13664. URL `http://dx.doi.org/10.1038/13664`.