Algorithms for Probe Selection and DNA Microarray Design

Sven Rahmann

Februar 2004

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

Gutachter:
Prof. Dr. Martin Vingron
Prof. Dr. Knut Reinert

1. Referent: Prof. Dr. Martin Vingron
2. Referent: Prof. Dr. Knut Reinert

Tag der Promotion:

Für Anja

Contents

List of Figures

List of Tables

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
$\mathrm{ms}^{s\mid t} = (\mathrm{ms}_i^{s\mid t})$	matching statistics of s against t
$\mathrm{ms}^{s\mid t;f}$	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; $\mathtt{MS}[i][c] = \mathrm{ms}_i^{s\mid T_c}$
(i, J)	jump at position i to level J
n, m	string lengths in Section 4.4; $\lvert s\rvert = m$, $\lvert t\rvert = n$
$\mathbb{P}$	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

$\mathbb{I}_{\{\cdot\}}$	generic notation for an indicator function
$S = (s_c)$	target sequences for all collections $c = 1, \ldots, C$
$H = (H_{ij})$	binary $m \times n$ hybridization matrix of probes vs. targets
D	design, i.e. a selection of probes; $D \subset \{1, \ldots, m\}$
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
$\mathrm{diag}(\cdot)$	diagonal matrix with specified entries
A^-	pseudo-inverse of A
$\|\cdot\|_2$	Euclidean norm of a vector; spectral norm of a matrix
$\|\cdot\|_\infty$	maximum norm of a vector
$\mathrm{cond}(A)$	condition number of A
$\mathrm{cond}_\mathrm{s}(A)$	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 $\geq \mu$
$\texttt{CSMS}[i][\mu]$	CSMS array; $\texttt{CSMS}[i][\mu] = \mathrm{csms}^{s\|t}_{i,\mu}$ 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, $i = 1..\lvert\mathcal{S}\rvert$
$\mathcal{U}$	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

DNA microarrays or DNA chips have become an increasingly important research tool in functional genomics over the last few years. Several different technology platforms exist; this thesis focuses on high-density oligonucleotide arrays.

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 1.

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, especially those of Li and Stormo (2001), 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.

A fundamental problem in microarray design and analysis is to understand the relationship between mRNA concentration (the "gene expression level") and measured signal intensity. Indeed, the relation should be approximately linear (excluding saturation effects); the key is to find the growth coefficient of this linear function. Chapter 2 provides a comprehensive discussion of this topic: We point out that the stability of the hybridization is an important factor that influences the signal strength, but it is not the only one.

For the selection of unique probes, we must ensure that each probe hybridizes to its intended target, and only to its intended target. If probes that fulfill this requirement are to be found efficiently on a large scale, we cannot evaluate the hybridization stability of each probe-target pair. Thus we introduce a rapidly computable sequence-based surrogate measure, the longest common factor (LCF), in Chapter 3. The relation between the LCF and the hybridization stability is developed from thermodynamic principles. An efficient LCF-based algorithm for evaluating the specificity of probe candidates is presented in Chapter 4. Is is based on all-against all matching statistics computation using an enhanced suffix array and an efficient representation using the concept of jumps in matching statistics, which we develop the the same chapter.

Evaluating the specificity is not enough in practical applications; several other constraints must be considered to obtain a good set of probes. Chapter 5 discusses these constraints and presents ways to include them in the probe selection process. It concludes with the outline of an efficient general-purpose selection procedure for sets of unique probes. The procedure has been implemented as the PROMIDE software package; technical remarks on its usage are given in Appendix A. PROMIDE is currently used in several chip design projects; an overview is given in Appendix B.

In Chapter 6, we look at the problems of selecting non-unique probes and decoding the probe signals into gene expression levels. Large scale quantitative analysis becomes rather impractical in this situation, but if non-unique probes are restricted to hybridize only within moderately sized sequence families, quantitative decoding is a possibility. We provide a mathematical framework based on matrix condition optimization to select a good set of probes (called a *design*) from numerous candidates.

Qualitative analysis with non-unique probes, i.e., deciding whether a certain transcript type is present (highly expressed) or absent (at most weakly expressed) is more versatile than quantitative analysis. We present an LCF-based probe candidate preselection method, and a greedy heuristic and an integer linear programming based method to select a design that is optimal in a certain sense (joint and ongoing work

with Gunnar Klau, Knut Reinert, Alexander Schliep, and Martin Vingron). A signal decoding procedure based on statistical group testing is also given.

All probe selection methodologies require that the transcriptome of the organism under consideration is known. This is not always the case and particularly interesting the case of the human transcriptome. Within the ENCODE (ENCyclopedia Of Dna Elements) project, one central question is whether all transcripts have been identified and annotated in the human genome. A promising strategy to identify transcribed regions in the genome is to use *genome tiling chips* that cover the whole genome with regularly spaced probes at a high density, say, one probe every 20 bp. Expression studies are then carried out for many different tissue types. A new transcript is identified if several consecutive probes show a signal that cannot be explained by cross-hybridization from other transcripts. The problem is that many probes are not unique, and we face a difficult decoding problem. Chapter 7 contains our contributions towards solving the challenges presented by genome tiling chips.

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. Our approach and the results are reported in Chapter 8.

A concluding discussion about the results of this thesis and an outlook into the future can be found in Chapter 9.

Publications. Parts of this thesis have been published in advance. The basic longest common factor approach from Chapter 3 — without the thermodynamic motivation given in this thesis — and a first version of the algorithm from Chapter 4 were presented at the CSB'02 conference (Rahmann, 2002) and won the best paper award. The basic procedure was later improved upon by considering and analyzing jumps in matching statistics (Rahmann, 2003a,b); these results have been integrated into Chapter 4. The results on non-unique probe selection and qualitative signal decoding from Chapter 6 were first presented at CSB'03 (Schliep, Torney, and Rahmann, 2003), and another publication is in preparation (Klau et al., 2004). The optimization of the nucleotide deposition sequence for chip production, as described in Chapter 8, was presented at ECCB'03 (Rahmann, 2003d).

This thesis also contains previously unpublished material, namely

- the conceptual discussion on signal modeling and affinity coefficient estimation in Chapter 2,
- the thermodynamic motivation of the probe unspecificity measure based on longest common factor statistics in Chapter 3,

- the systematic discussion of other constraints than specificity for probe selection in Chapter 5,
- a framework for non-unique probe selection and signal decoding for quantitative expression analysis based on matrix condition minimization in Chapter 6, and
- the material on genome tiling chips in Chapter 7.

The software described in Appendix A of this thesis can be obtained from the URL `http://oligos.molgen.mpg.de`.

Acknowledgments. This work was carried out while I was a staff scientist at the Department of Mathematics and Computer Science at the Free University of Berlin and a pre-doctoral fellow in the Computational Molecular Biology department at the Max Planck Institute for Molecular Genetics. 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 Martin Vingron for suggesting the topic, providing initial ideas, and for the opportunity to write this thesis under his guidance. Knut Reinert wrote a referee report for this thesis, for which I am very grateful.

The support of the whole computer service group at the Max Planck Institute, and in particular of Wilhelm Rüsing, 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 Verena Beier, Stefan Haas, Gunnar Klau, Stefan Kurtz, Hannes Luz, Tobias Müller, Antonio Piccolboni, Knut Reinert, Alexander Schliep, Jens Stoye, and Martin Vingron for their valuable hints and comments. In their Bachelor's theses, Christine Gräfe, Jonas Heise, and Michaela Spitzer worked on spin-off projects of this thesis; their contributions are very much appreciated. I also thank Holger Meyer, Reiner Matthiesen, and Jan Wildenhain for helpful discussions about the shortest common supersequence problem.

Anja von Heydebreck, Antje Krause, Hannes Luz, Tobias Müller, Matthias Rahmann, Stefan Röpcke, Alexander Schliep, and Christine Steinhoff read early drafts of several chapters of this thesis and helped to improve it in many ways through their comments.

Last but not least, I thank Anja Singbartl for her love and patience while I was writing this thesis. Heartfelt thanks also go to my parents and the rest of the family.

Sven Rahmann Berlin, February 2004

Chapter 1

Introduction

In this thesis, we consider computational and modeling problems that arise during design and production of high-density oligonucleotide microarrays, which have become increasingly important tools for gene expression analysis in functional genomics. To explain the principle of microarrays or "DNA Chips", we first briefly review the structure of DNA and mRNA molecules (Section 1.1). We then contrast microarrays with other techniques for gene expression analysis and compare different technology platforms for microarrays. The importance of microarrays in the field of functional genomics is pointed out (Section 1.2). High-density oligonucleotide microarrays are described in more detail; we give an overview of their production (especially the oligonucleotide probe synthesis process), the sample preparation, the hybridization reaction, and the data generation and analysis pipeline (Section 1.3). Having described the technical framework and constraints of microarrays, we formulate several algorithmic problems in chip design and oligonucleotide probe selection (Section 1.4).

1.1 The Structure of DNA, RNA and Genes

Complex organisms, such as mammals, consist of billions to trillions of cells. The genetic information that an organism needs to perform its vital functions is encoded in special macromolecules residing in the nucleus of each of the organism's cells. These molecules are called chromosomes, and each chromosome is one long molecule of deoxyribonucleic acid (DNA).

The structure of DNA was discovered 50 years ago in 1953 by Rosalind Franklin, Francis Crick, James Watson and Maurice Wilkins (Watson and Crick, 1953): DNA forms a double-helix and consists of two antiparallel complementary strands. Each strand is a directional linear polymer of four types of nucleotides or bases (adenine `A`, cytosine `C`, guanine `G`, and thymine `T`), held together by a sugar-phosphate backbone. The sugar-phosphate bonds in this backbone are phosphodiester bonds. The carbon atoms of the sugar groups are numbered 1' through 5'. A phosphodiester bond links the 5' carbon of a deoxyribose to the 3' carbon of the adjacent deoxyribose; in this

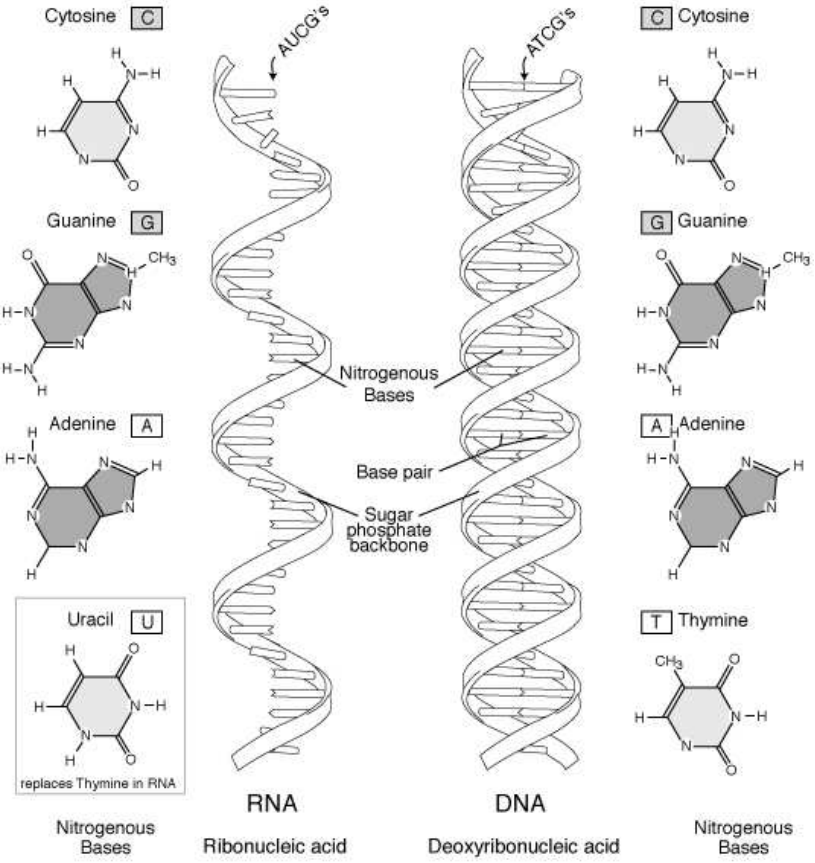


Figure 1.1: DNA and RNA bases. Adenine, cytosine, and guanine are common to both DNA and RNA. The DNA base thymine is replaced by uracil in RNA. Image by Darryl Leja, reproduced from the Talking Glossary at the National Human Genome Research Institute, USA.

way, a direction can be assigned to a DNA strand. It is customary to look at a strand in 5' → 3' orientation because DNA transcription (see below) can only proceed in this direction (from the transcript's point of view). Thus the sequence of nucleotides in a DNA strand can formally be written as a finite string over the four-letter alphabet $\Sigma_{\text{DNA}} := \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$.

In a double-stranded DNA molecule, the second strand is the first strand's *Watson-Crick complement*: `A` is the complement of `T`, `C` of `G`, and vice versa. The stability of double stranded DNA comes from the hydrogen bonds between complementary bases in opposite strands. There are three hydrogen bonds in a `G`-`C`-pair, and two hydrogen bonds in an `A`-`T`-pair, making `GC`-rich DNA more stable than `AT`-rich DNA. The hydrophobic base-pairs are on the inside of the double helix, with the hydrophilic sugar-phosphate backbone on the outside. Because of hydrophobic interactions, base pairs stack onto each other, forming a ladder going up the helix, adding to its stability.

The stacked planes are 3.4 Å apart and almost perpendicular to the backbone axis. A full turn of the helix consists of 10.5 base pairs (36 Å). The helix is right-handed and exhibits two grooves within the twisted backbone: a major and a minor groove. Sequence-specific DNA binding proteins that regulate transcription recognize their binding sites via these grooves (Griffiths et al., 2002).

The DNA contains the information needed to synthesize the proteins that control functions and build structures within cells. While only one strand is needed to encode the genetic information and is "read out" for protein synthesis, the principle of base complementarity ensures the stability of the genetic information, and its ability to replicate itself faithfully. In order to perform its various functions, the information in the genetic material must be transported to the location in the cell where it is needed. Generally, a "working copy" of a part of the DNA molecule is created for such a purpose. This process is called *transcription*.

The transcript consists of ribonucleic acid (RNA), a polynucleotide in many ways similar to DNA. However, the sugar deoxyribose in the DNA backbone is replaced by ribose in the RNA backbone, and the DNA base thymine (`T`) is replaced by uracil (`U`), so the RNA alphabet is $\Sigma_{\text{RNA}} = \{\texttt{A},\texttt{C},\texttt{G},\texttt{U}\}$. Figure 1.1 shows the differences between the DNA and RNA bases. In contrast to DNA, RNA frequently occurs as a single-stranded molecule. Ribonucleotides in the same single RNA strand base-pair with each other, forming a secondary structure. Typical structure elements are *hairpin loops*, *stems*, *bulges*, *interior loops*, and *junctions* (see Figure 1.2). From a given RNA sequence, the secondary structure can be predicted with specialized algorithms, e.g., those implemented in the MFOLD software package (Zuker, 2003) by energy minimization.

Regulatory mechanisms ensure that only those parts of the DNA that have a specific function are transcribed into RNA. Such a functional unit on a chromosome is called a *gene*. With extreme over-simplification, we can say that a gene consists of

- the 5' regulatory region containing a promoter sequence, where regulatory proteins and the transcribing enzyme (RNA polymerase) attach to the DNA,
- the transcribed region (for more details see below),
- the 3' regulatory region.

It is again the base-complementarity principle that makes transcription physically possible. The two strands of the DNA double helix separate, and one of the strands acts as a template for RNA synthesis. Free ribonucleotides that have been synthesized or recycled elsewhere in the cell form stable pairs with their complementary DNA bases in the template. The enzyme RNA polymerase attaches to the DNA and moves along it, thereby linking the free ribonucleotides to form the growing RNA molecule, which grows in the 5' $\rightarrow$ 3' direction. Thus the transcription process can be succinctly

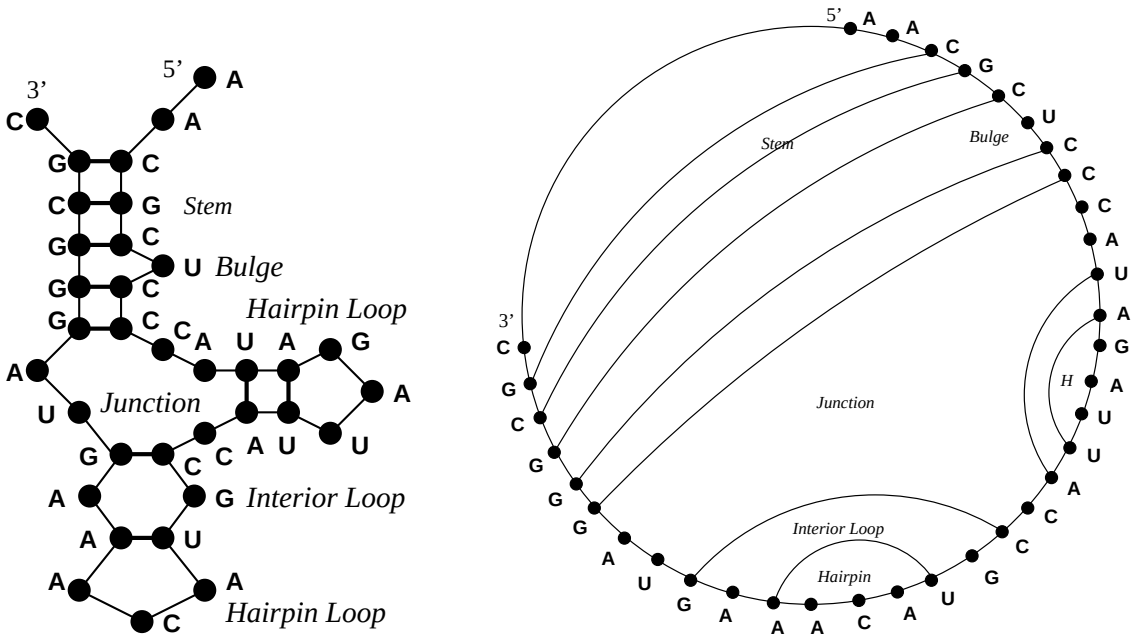


Figure 1.2: Visualization of RNA secondary structure elements in a planar graph (left) and in a circle representation (right).

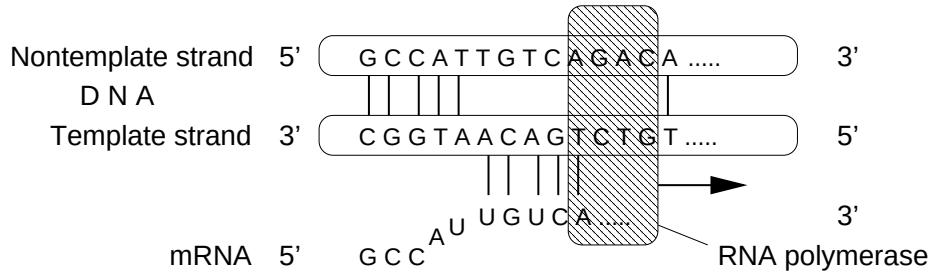
described by the formal operations 3′-A-5′ ↦ 5′-U-3′, 3′-C-5′ ↦ 5′-G-3′, 3′-G-5′ ↦ 5′-C-3′, and 3′-T-5′ ↦ 5′-A-3′.

By convention, when talking about the DNA sequence of a gene, the sequence of the *non*-template strand is used because it matches that of the mRNA sequence except for the replacement of T by U. The transcription process is visualized in Figure 1.3a.

The RNA molecule is called a *transcript* of the gene. If it is used to transfer information, thus acting as a *messenger*, we speak of an mRNA molecule. There are also RNA molecules that have an active function of their own other than transmitting information, such as transfer RNAs (tRNAs), ribosomal RNAs (rRNAs), small nuclear RNA (snRNAs), small cytoplasmic RNAs (scRNAs). In this thesis, we are mainly concerned with mRNA transcripts. The term *upstream* refers to the 5' side of a transcript, whereas *downstream* refers to the 3' side.

A primary mRNA transcript starts with a *5' untranslated region* (5' UTR). Next, between the *translation initiation site* (also called the *start codon*) and the *translational termination site* (*stop codon*), the transcript contains the *coding sequence* (CDS) that is translated into a polypeptide. In eukaryotes (see below), the coding sequence is normally interrupted by non-coding parts that are removed during post-transcriptional processing. The coding parts are called *exons*, the non-coding parts are called *introns*, and the process of intron removal is referred to as *splicing*. The transcript ends with the 3' UTR.

(a)



(b)

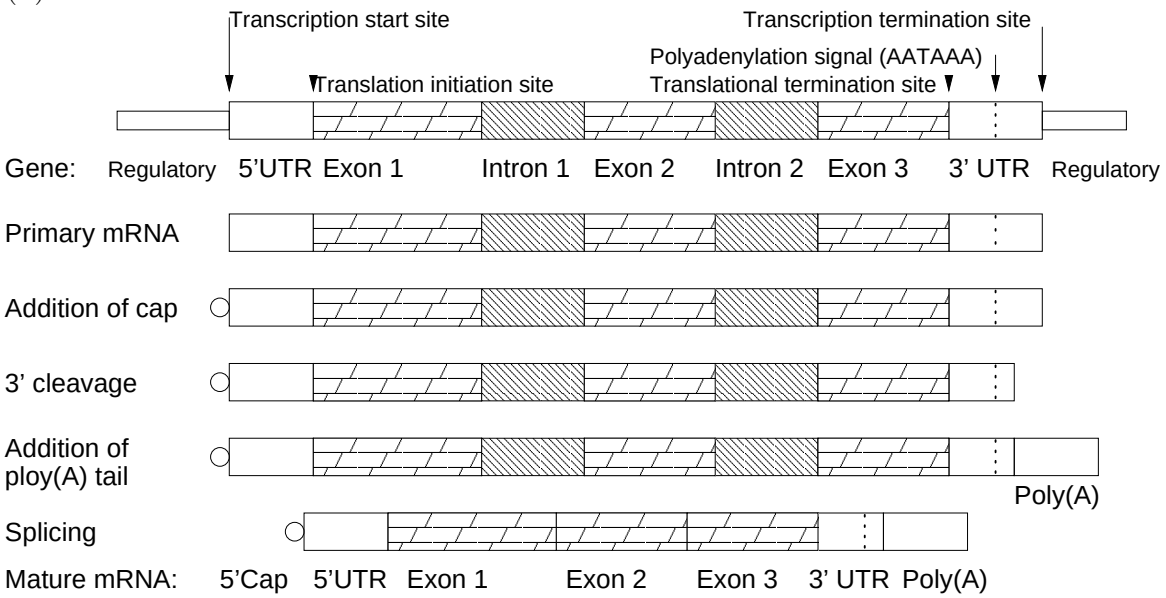


Figure 1.3: (a) Transcription. (b) Post-transcriptional processing of mRNA in eukaryotes.

While the basic principle of transcription is the same in prokaryotes (organisms whose cells do not have a nucleus) and eukaryotes (organisms whose cells do have a nucleus), there are some differences in the post-transcriptional processing of the RNA molecules. Prokaryotic mRNA molecules usually do not undergo processing. The primary eukaryotic mRNA transcript or pre-mRNA is processed as follows to give the mature mRNA (see also Figure 1.3b).

Addition of cap During transcription, a cap of a 7-methyl-guanosine residue is added to the 5' end of the transcript, linked by three phosphate groups.

3' cleavage An `AAUAAA` sequence, called a polyadenylation signal, near the 3' end of the transcript is responsible for cutting off the transcript about 20 bases further down.

Addition of poly(A) tail 150 to 200 adenine nucleotides, called a *poly(A) tail*, is appended to the cut 3' end. It is believed that the tail plays a role in the export of mature mRNA from the nucleus as well as in helping stabilize mRNA molecules by retarding their degradation in the cytoplasm.

Splicing Many eukaryotic genes contain non-coding parts called introns, which are removed in a procedure called splicing. Splicing adjoins the separate coding exons so that the mature mRNA contains a coding sequence that is completely collinear with the peptide sequence it encodes. A mechanism called *alternative splicing* allows to skip certain exons during splicing, resulting in a number of alternative transcripts for a gene.

The mature mRNA molecules are transported to cellular structures called ribosomes for translation into proteins. Proteins are directional linear biopolymers, made out of 20 different building blocks called amino acids. The *genetic code* specifies how a polypeptide is built by reading off the mRNA sequence: Each RNA triplet or *codon* (and therefore also each DNA triplet, referring to the coding strand) encodes one amino acid. Usually there is more than one codon that encodes the same amino acid; hence the genetic code is said to be degenerate.

We cannot present the intricate details of the transcription and translation process here; the reader is asked to refer to the textbooks by Alberts et al. (1994) or Griffiths et al. (2002).

1.2 Functional Genomics

The recent sequencing of the Human genome (Venter et al., 2001; Lander et al., 2001) has identified more than 30000 potential gene sequences and is considered to be a milestone for future biological and medical research. Each gene contains information to build one or more proteins; different proteins may arise from the same gene, e.g., by such mechanisms as alternative splicing or RNA editing. So far the genomic information contained in Human DNA has been mainly transferred to electronic storage media. Now the challenge is to understand the functions of the genes, their interactions, and the way how they are regulated. While biology used to be a hypothesis driven science, one of the major ideas in contemporary genomics research is to perform experiments that capture data on the "complete" state of a genomic system. This is the aim of the research field called *functional genomics*.

An important task is to gather knowledge about tissue-specific gene expression and regulation. One wants to learn, for example, how often and how frequently each kind of protein is synthesized in different cell types. Many techniques for measuring gene expression focus on the quantitation of mRNA molecules in a cell. While this information provides an important part of the picture (and indeed we will restrict

our consideration to such techniques), it should be kept in mind that other factors, such as the translation rate and the mRNA degradation rate, are also important for understanding the whole picture.

Techniques for expression analysis. Expression patterns of mRNA have been analyzed long before the era of functional genomics. Most of these experimental techniques, however, focus on a single gene or a few genes at a time. They exploit the fact that nucleic acids hybridize[1] to their complements. Some frequently used techniques are given in the following non-exhaustive list; more detailed descriptions can be found, for example, in the textbooks by Strachan and Read (2003) or Lottspeich and Zorbas (1998). The reader who is in a hurry can skip over this list and jump to the following paragraph about microarrays.

Northern blot analysis can be used to determine how strongly a gene or splice variant is transcribed in a certain tissue or under specific conditions. mRNAs are extracted from the cell sample, denatured and applied to a further denaturing agarose gel, where they undergo electrophoresis. The fragments migrate through the gel at differing speeds according to their respective sizes. The gel is placed into a salt solution (buffer) and covered by a nitrocellulose or nylon membrane and many paper towels. The RNA bands from the gel are transferred ("blotted") onto the membrane by capillary action, where they are immobilized. The membrane is bathed in a solution containing single stranded radioactively labeled DNA or RNA probes that are complementary to the targeted mRNA. The probes will hybridize stably to the targeted gene only, and unbound probe is washed off. When an X-ray image of the membrane is taken, the X-ray film is only exposed at those bands with bound probes. Comparing these bands with labeled markers reveals the number and size of the fragments in which the targeted sequences are found. The name "Northern Blotting" has been coined to contrast this technique for RNA analysis from the similar "Southern Blotting", developed by Southern (1975) for DNA analysis.

RPA (Ribonuclease protection assay) requires a radioactively labeled antisense or complementary RNA (aRNA or cRNA) probe, also called a *riboprobe*, that is complementary to the target mRNA (this riboprobe could also be used for Northern Blot analysis). It can be created by cloning the gene of interest via T7 RNA polymerase in a transcription vector containing a T7 promotor downstream of the insert. T7 is a phage (virus) that infects the bacterium *E. coli*; there are seven such viral parasites numbered T1 through T7. The T7 RNA polymerase recognizes the T7 promotor and begins transcribing the gene of interest in antisense direction. The desired riboprobe is obtained when radioactive ribonucleoside triphosphates are made available for RNA synthesis in the vector.

[1] Latin *hibrida* or *hybrida*; of animals produced from two different species. Greek *hubris*; an outrage on the person, violation. In English, the term hybrid refers to something that is made of two different things, such as a double-stranded molecule with one DNA strand and one RNA strand. Meanwhile, the verb "to hybridize" and the noun "hybridization" are increasingly used as synonyms for base-complementary binding, even for DNA-DNA and RNA-RNA bindings.

The specific riboprobe is hybridized against the total cellular RNA, and the resulting mixture of perfect RNA-RNA-hybrids and single stranded RNAs is incubated with ribonucleases which specifically hydrolyze single stranded RNA molecules, so that only perfect probe-target duplexes remain intact. These are then electrophoresed in a denaturing polyacrylamide gel, where different RNA species are separated according to their size. An X-ray image shows how much of each species was present in the sample.

RT-PCR (reverse transcription-polymerase chain reaction) is a very sensitive technique for mRNA detection and quantitation. Compared to Northern blot analysis and RNase protection assay, RT-PCR can be used to detect mRNA presence from much smaller samples, even from a single cell. First, the RNA molecules are reverse-transcribed into single-stranded complementary DNA (cDNA) using a reverse transcriptase. An oligo-(dT)-primer that binds to the poly(A) tail at the 3' end of the mRNA may be used for this step. With a sequence-specific primer, the second DNA strand is synthesized, giving double stranded DNA (dsDNA). The dsDNA molecules are amplified in several polymerase chain reaction (PCR) rounds. In each round, first single stranded DNA (ssDNA) is produced by heat denaturing the dsDNA. Then the solution is cooled, and sequence-specific primers bind to their complementary ssDNA sequences. The temperature-resistant *Taq* polymerase (from the heat-resistant bacterium *Thermus aquaticus*) replicates the ssDNA segments to which the primers have hybridized. Thus in each round, the number of dsDNA molecules increases by a factor of two (in reality, the factor is between 1.8 and 1.9) until product renaturation begins to compete with primer hybridization and the amplification rate drops quickly from exponential to linear. In order to quantify the original amount of mRNA, we need to relate the number of rounds n to the number of molecules present after round n before the phase of exponential amplification ends. An extension of RT-PCR, the so-called *real-time RT-PCR* technique, allows automatic monitoring of the amount of amplified DNA via the generation of a fluorescent signal in each round, and therefore a much more reliable quantitation.

SAGE (serial analysis of gene expression) was developed by Velculescu et al. (1995). The RNA molecules are reverse-transcribed into cDNA using a biotinylated oligo-(dT) primer. Biotin is the cofactor required by enzymes that are involved in carboxylation reactions and a frequently used linker molecule. The 3'-ends of the cDNA molecules are linked to biotin, which in turn attach to streptavidin beads. Streptavidin is a tetrameric protein that binds very tightly to biotin. The beads can be used to extract the cDNA molecules from the mixture. The cDNA is then cleaved with a frequently-cutting restriction enzyme, called the *anchoring enzyme*. This leaves a sticky `CATG` end hanging out. A double-stranded oligonucleotide adapter or linker, randomly selected from two types A and B, is attached onto each sticky end. Another enzyme, called the *tagging enzyme*, locks onto the linker, reaches downstream and cuts off a short segment of the cDNA. A collection of short tags, called *SAGE tags*, taken from each molecule, is left. The tails of type-A and type-B tags are ligated to form so-called *ditags*. The tags are amplified by several PCR rounds, the primer sequences being provided by the type-A and type-B linker sequences. Finally the linker molecules are

released, and the tags are concatemerized via their sticky ends. The concatemers are inserted into bacteria for cloning, and the clones are sequenced. Computer analysis will count the tags, determine which ones come from the same RNA molecule, and figure out which ones come from known genes and which ones are new. One of the advantages of SAGE is that it can discover new genes in organisms for which there is no complete genome. If a sequence does not match a known gene, it must come from a gene that has not been discovered before. A disadvantage of SAGE is that it is a very labor-intensive procedure.

In-situ hybridization. A radioactively or fluorescently labeled riboprobe is hybridized against RNA in tissue sections. Tissue sections are made from paraffin-embedded or frozen tissue and then mounted onto glass slides. The hybridized probe is visualized using autoradiographic or fluorescence microscopy procedures. In-situ hybridization allows to compare the expression of one gene in many different tissues.

Other techniques exist and are continuously being developed and refined. The above mentioned techniques are well established for detection and quantitation of single or few mRNA species. They require different knowledge and experimental protocols and give different information. For example, RT-PCR requires knowledge of the mRNA sequence so that one can design sequence-specific primers. SAGE does not require the sequence information and can be used to discover new transcripts. In-situ hybridization allows to monitor the activity of one gene in many different cell types or tissues.

Microarrays for functional genomics. What the above techniques do not offer is a possibility to monitor the activity of all or of a large fraction of all genes in a certain tissue, as measured by mRNA transcript abundance. This has become possible through the development of *DNA microarrays*. They promise to be an ideal research tool for functional genomics, because according to Hieter and Boguski (1997), "functional genomics is characterized through the use of high-throughput or large-scale experimental methodologies, combined with statistical and computational analysis of the results; the scope of biological investigation is expanded from single genes or proteins to the systematic study of all genes or proteins at once." Microarray experiments quickly generate massive amounts of data; and the interpretation of this data for hypothesis generation and discovery has become a large subfield within bioinformatics. The above "single-gene" methods remain important to verify the proposed hypotheses.

DNA microarrays are arrays of many DNA molecules on a quartz, glass, or nylon substrate. Because of their resemblance to microchips used in computers, they are also called *DNA chips*. The foundation of microarray technology lies in the Watson-Crick complementarity of double-stranded DNA or RNA-DNA-hybrids. DNA molecules with the known sequence of genes (or parts of it) are printed on the chips as *probes* at regularly spaced and well defined locations called *spots* or *features*. The mRNA

molecules, called *targets*[2], which are extracted from a tissue or blood sample, are prepared and labeled with a fluorescent or radioactive dye. The details of this process vary with the technology platform; there are some examples given below. The labeled targets are then allowed to hybridize to the probes on the array. Whenever the Watson-Crick complementary sequence of a probe is present in a target sequence, that target will hybridize to the probe. Unhybridized target molecules are washed off the chip, and the amount of hybridized target at each spot can be measured by the intensity of the dye or radioactivity. The idea behind this procedure is that each spot represents one gene and that the amount of hybridized target at a spot is a quantitative measure of the gene's transcript abundance in the cell sample, which is often interpreted as the gene expression level or its "activity" in the cell sample.

Microarray technologies. The above description is generic, and the details vary according to the specific type of microarray. Over the recent years, microarrays have become increasingly important research tools, and several companies offer various technology platforms; a recent listing appeared in the November 2003 issue of *Laborjournal* (Köppelle, 2003). We can distinguish between four main technology platforms of microarrays.

1. *Nylon membrane arrays* or *radioactive filters* are reviewed by Lennon and Lehrach (1991). Double-stranded PCR products from cDNA clone libraries are spotted on a nylon membrane with a spotting robot. In an experiment, sample RNAs are reverse transcribed using nucleotides labeled with ^{33}P, a radioactive phosphor isotope. The PCR products on the membrane are denatured, and the radioactive cDNA is allowed to hybridize to the complementary DNA molecules on the array. After washing, the hybridized signal on the array is analyzed by a phosphorimager. The measured signal intensity should be roughly proportional to the expression level. To compare two different cell or tissue samples, two separate arrays must be used.

2. *cDNA arrays* or *red/green arrays* were first used by Schena et al. (1995). Here PCR products are spotted on a glass slide, and the sample is labeled with a fluorescent dye. With this technique, it is customary to compare two samples (usually called the test sample and the reference sample) with two different dyes (Cyanine 3-dNTP or Cy3, and Cyanine 5-dNTP or Cy5) on the same array, so that the samples hybridize simultaneously. The hybridization signal is scanned with a fluorescent imager, which produces an image with red, green and yellow spots. A green spot indicates that the corresponding gene was more highly expressed in the test sample than in the reference sample, while a red spot indicates the opposite.

[2]Some authors, such as Schena (2002), call the molecules on the chip the targets and the molecules in the sample the probes.

3. *Polynucleotide*[3] *arrays*: For each transcript, one or more single-stranded specific 50-mers to 70-mers are synthesized separately using phosphoramidite chemistry and then spotted on a glass surface, or synthesized in situ using ink-jet technology (Hughes et al., 2001). The oligonucleotides are hybridized with fluorescently labeled sample cDNA or aRNA.

4. *Oligonucleotide arrays*: Each transcript is represented by several specific short (20–30 nt) substrings of its sequence. These oligomers can be synthesized chemically beforehand and then spotted on the chip surface, or synthesized directly on the chip with a photolithographic procedure, which allows the highest spot density of all methods (Lockhart et al., 1996; Lipshutz et al., 1999).

Spotted oligonucleotide arrays have a spot or feature size of about 50 to 300 microns[4], while photolithographically synthesized arrays are even more miniaturized with feature sizes of 15 to 30 microns. Each spot contains up to a billion (10^9) single-stranded oligonucleotides. Table 1.1 compares some relative advantages and disadvantages of PCR-product-based arrays and oligonucleotide arrays. Schena (2002) estimates that up to 2002, 65% of all microarray experiments used cDNA arrays, about 26% used oligonucleotide arrays, and the remaining 10% used other technologies. From now on, this thesis focuses on high-density 12–32 nt oligonucleotide arrays.

1.3 High-Density Oligonucleotide Microarrays

Modern oligonucleotide arrays, also called *DNA chips*, are characterized by their high number of spots or feature locations, concentrated on a small surface. They allow many gene expression measurements in parallel, and hence high-throughput generation of large-scale data. They have many applications, of which we outline a few. Young (2000) also reviews approaches to biomedical discovery with DNA arrays.

- *Gene expression measurements* are presently one of the most important uses of DNA chips. Researchers have used them to identify gene functions (Hughes et al., 2000), to define genetic signatures of diseases (among others, Golub et al., 1999), to understand genetic causes of diseases (e.g., Lock et al., 2002), and to characterize the effects of certain drugs on gene expression (Oestreicher et al., 2001). It is hoped that correlating gene expression data with other information, such as transcription factor binding site locations (e.g., see Dieterich et al., 2003), will give further insights into cellular pathways, and eventually into the "blueprints" of living organisms.

[3]Sometimes, these are called "long oligonucleotides".
[4]1 micron = 1 μm = 10^{-6} m

Table 1.1: Advantages (⊕) and disadvantages (⊖) of PCR-product-based versus oligonucleotide microarrays; adapted from Schena (2002).

	Arrays with PCR products		**Arrays with Oligonucleotides**
⊕	No sequence information required when primers binding to the vector sequence are used	⊖	Exact sequence information necessary
⊕	Long length provides much complementarity for hybridization, and hence strong signals	⊖	Reaction conditions must be optimized to get strong signals
⊕	PCR is a standard procedure and easily implemented in most laboratories	⊖	Expensive equipment or external service needed to synthesize oligonucleotides
⊖	Requires a clone library	⊕	Oligonucleotides can be synthesized as needed
⊖	Double stranded PCR-products must be denatured on the array before hybridization. Signal loss because of partial re-annealing.	⊕	Oligonucleotides are single-stranded.
⊖	Cross-hybridization occurs in families of strongly homologous genes, such as in human heat shock protein genes	⊕	High specificity of well-chosen oligonucleotides avoids cross-hybridization. Splice variants and even SNPs can be detected.

- DNA chips were originally developed for *sequencing by hybridization (SBH)*, which has never been commercially successful on a large scale. However, re-sequencing of short sequences is a possibility with DNA chips. The chip usually contains all possible 8-mers or 9-mers as probes. From the hybridization signal one learns which 8-mers occur as substrings of the sequence of interest, and from this *spectrum* one attempts to infer the original sequence (Lipshutz et al., 1994).

- A related application is the determination of *single nucleotide polymorphisms (SNPs)*, where a single base at a well-defined location differs across individuals (Cutler et al., 2001; Kozal et al., 1996). For each SNP, the DNA chip contains four probes, enumerating all four possibilities at their center location. It is also possible to *genotype* human individuals with DNA chips. Different forms of a gene are called alleles; each individual has two alleles of most genes. The genotype is called homozygous if both alleles are identical, and heterozygous if they are different. DNA chips allow the determination of the genotype (Hacia et al., 1996).

We briefly describe the life-cycle of an oligonucleotide chip from its production to the analysis of its hybridization data, taking Affymetrix's GeneChip® as an example.

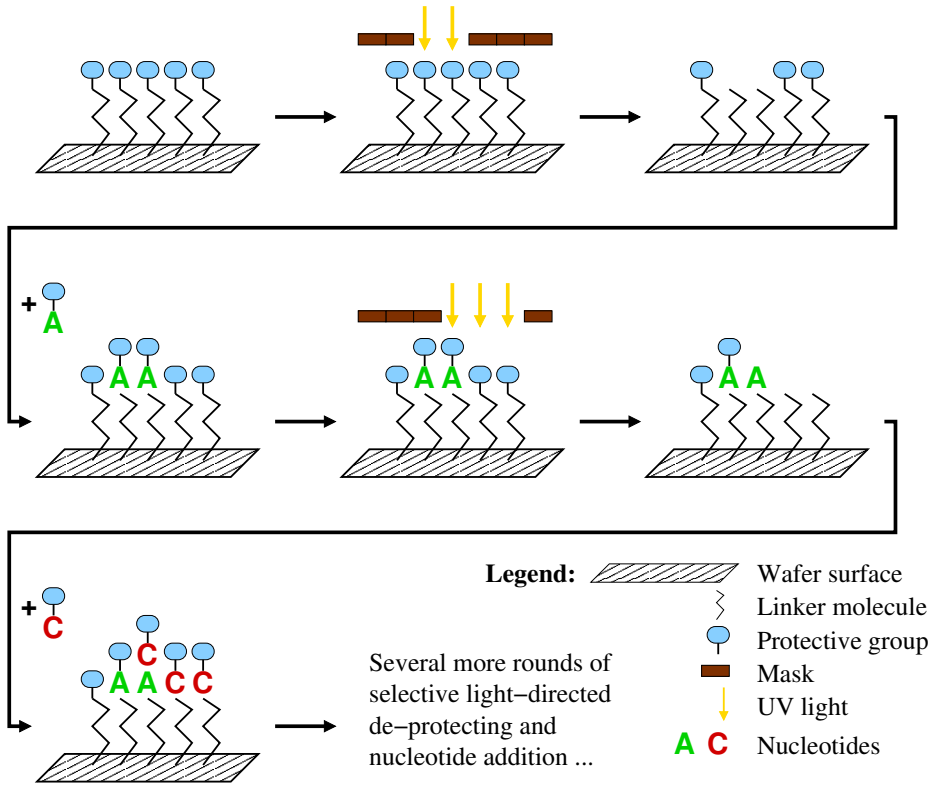


Figure 1.4: Schematic overview of photolithographic DNA microarray production. The steps of selective de-protection by ultraviolet light and nucleotide addition are repeated until all probes have reached their full length.

1.3.1 Chip Production

DNA chips are produced in a parallel process that combines photolithography and combinatorial chemistry, so that the length of the probes, and not their number, determines the number of necessary synthesis steps. Chip production begins with a quartz wafer, from which 50 to 400 arrays can be produced. The wafer is placed in a bath of silane, which reacts with the hydroxyl groups of the quartz and forms a matrix of covalently bound molecules. Linker molecules are attached to the silane matrix; these are at first chemically protected to prevent unwanted reactions.

All probes are synthesized in parallel; there are several million, up to a billion, probe molecules per spot or feature, and several thousand, up to a million, spots per chip. To define which probes will receive a nucleotide in each step, photolithographic masks with windows that correspond to the spot size (18 to 20 square microns) a placed over the wafer. When ultraviolet light is shone over the mask in the first step of synthesis, the exposed linker molecules are selectively de-protected and made available for chemical reactions. A solution containing a single type of deoxynucleotide with

a protection group is flushed over the wafer's surface. The nucleotide attaches to the activated linkers, initiating the synthesis process. In the following steps, other masks are placed over the wafer for the next rounds of de-protection and nucleotide attachment. The process is repeated until all probes reach their full length. Figure 1.4 shows the synthesis process.

The synthesis process is not entirely error-free. It happens that a nucleotide fails to attach to an unprotected partial probe. This is not a major problem because between two synthesis steps, still unprotected probes can be capped and irreversibly protected, so they cannot react with the sample later. It also happens that due to stray light, some probes are unprotected and elongated by a nucleotide when in fact they should not. This results in a small fraction of probes on each spot that do not consist of the desired sequence and are not available for hybridization with the intended target. In fact, they may hybridize better with different targets, adding to the noise level of the signal. Generally a few sample chips from each wafer are sacrificed for quality testing. Once the chip is produced and tested, it is available for a single hybridization experiment.

A different production technology is used in the geniom® one device developed by febit AG, Mannheim, Germany (Beier et al., 2001; Baum et al., 2003). Their system offers integrated chip production, hybridization experiments, and data analysis in one benchtop device. Their reaction carrier is called the DNA processor®; it consists of a glass-silicon-glass sandwich, with microchannels etched into the silicon layer. Selective de-protection of spacer molecules and partial probes is not achieved by the use of masks, but by selectively directing light with digital micromirror devices (DMDs), as originally described by Singh-Gasson et al. (1999).

1.3.2 Target Preparation

Target preparation varies according to the technology platform, the intended application, and the organism of interest. The following overview refers to eukaryotic and prokaryotic gene expression analysis; it is based on Affymetrix's GeneChip® Expression Analysis Technical Manual (Affymetrix, Inc., 2003) from the NetAffx web resource (Liu et al., 2003).

Eukaryotic target labeling. First, poly(A) mRNA is isolated from the cell or tissue sample of interest. Then the first cDNA strand, complementary to the mRNA, is synthesized by reverse transcription. The reaction is primed with an T7-oligo(dT) primer that hybridizes to the poly(A) tail of the mRNA and contains a T7-promotor at its 5'-end. Commercially available standard kits can be used for these steps. When the synthesis of the first cDNA strand is complete, the mRNA is degraded with the enzyme RNase, and the second cDNA strand is synthesized along the first one. This is

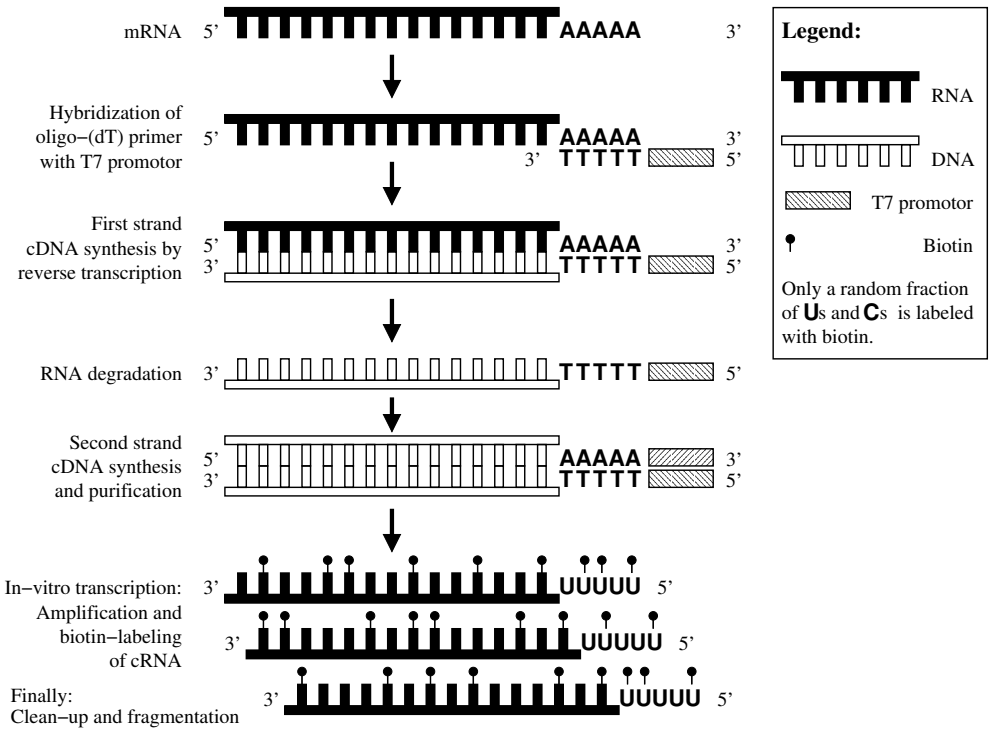


Figure 1.5: An overview of a typical eukaryotic target labeling procedure.

achieved by adding a dNTP mix and the enzymes DNA ligase and DNA polymerase I to the reaction solution. The double-stranded cDNA is washed and purified. By in-vitro transcription (IVT) of the cDNA with T7 RNA polymerase, free ribonucleotides and biotin-labeled ribonucleotides `U` and `C`, cRNA (also called antisense RNA or amplified RNA, aRNA) is produced. Because of this transcription step, a linearly amplified amount of the original amount of mRNA is produced. This amplification protocol is originally due to Van Gelder et al. (1990). The length of the in-vitro transcripts varies (approximately uniformly) between 500 nt and 1500 nt. The biotin serves a bridge molecule between the nucleotide and a fluorescent dye molecule. The resulting cRNA molecules will only be stained after the hybridization reaction (see below). The cRNA target molecules are cleaned and fragmented, and their concentration is quantified to ensure standard conditions.

Prokaryotic target labeling. Target preparation is different for prokaryotic samples because no polyadenylated mRNA molecules exist. Therefore total RNA is isolated from cells grown in culture. Only a single cDNA strand is synthesized by using random primers (usually random hexamers); this also means that not the full transcript may be reverse-transcribed. The mRNA is degraded and the cDNA purified. When the cDNA is fragmented, a biotin-labeled uracil is added at the 3'-end of each fragment.

Comparison. The main differences between eukaryotic and prokaryotic targets are as follows: Eukaryotic targets consist of cRNA fragments from transcribed poly(A)-RNA, with a random fraction of biotin-labeled `U`s and `C`s, while prokaryotic targets are cDNA fragments from total RNA, with a single biotin-labeled `U` at their 3'-end. Eukaryotic targets cover mainly the last 1000 nt near the 3'-end of the transcripts, while prokaryotic targets cover the transcripts more randomly. In both cases, however, the target sequence is the complement of the mRNA sequence (antisense), so that the probe sequence must match the mRNA sequence (sense).

1.3.3 Hybridization, Washing, and Staining

A hybridization cocktail with the fragmented target and controls is prepared and heated to a high temperature, say 99°C. The cocktail is hybridized to the microarray during an incubation cycle of several hours at a much lower temperature, say 45°C. Thus the hybridization starts at a high temperature, where fragmented targets and probes are present as single strands. The temperature gradually lowers, and the most stable perfect probe-target pairs start hybridizing first. According to Affymetrix's manual (Affymetrix, Inc., 2003), it is important for successful hybridization that the targets have been fragmented into shorter pieces of approximately 35 to 200 nucleotides. This shorter length minimizes adverse steric effects. To achieve a uniform distribution of target fragments over the whole chip surface, the chip is placed in a shaker, so that the whole mix is kept in constant motion. For the later data analysis phase, it is also important that the probe concentration on the chip is in excess of the target concentration (i.e., the overall cRNA or cDNA concentration must be within pre-defined standard limits) to avoid saturation effects.

After overnight incubation for about 16 hours, unhybridized target molecules are washed off the chip. First a non-stringent washing buffer of 6x SSPE (6-fold concentrated saline sodium phosphate buffer) is applied; then a more stringent buffer with 100 mM 2-(N-morpholino)ethanesulfonic acid (MES) and a sodium salt [Na^+] concentration of 0.1 M is used to ensure that ideally only the perfect Watson-Crick probe-target hybrids remain stably bound. Perfect results cannot be expected, however.

The remaining biotinylated targets are fluorescently stained with streptavidin-phycoerythrin (SAPE), a fluorochrome-conjugated streptavidin that binds tightly to biotin and is frequently used in indirect staining protocols. To boost the signal intensity, biotinylated streptavidin antibodies can bound to the SAPE molecules after the first round of staining. Then, in a second round, more SAPE binds to the biotin on the antibodies, amplifying the signal.

1.3.4 Data Acquisition and Analysis

After the staining phase, the first task is to measure the amount of SAPE bound at each feature. This can be accomplished with the appropriate fluorescent imaging technology, i.e., via a scan of the stained array. The result is a digital image in a standard format, say, in tagged image file format (TIFF). In this image, one needs to find the location of each feature and define which pixels constitute a spot and which pixels make up the background of the image. This complex task is performed by special-purpose image processing software. Once the feature locations are well defined, one obtains a *raw intensity value* for each feature.

Since each mRNA transcript is usually represented by several different features, i.e., different probe sequences with potentially different raw intensity values, all intensities from the same transcript are then summarized by a single *raw expression value*. In this step, one usually assumes that each probe is transcript-specific, i.e., that the measured raw intensity value is proportional to the expression level of the single target that the probe was designed for, and that there is no or negligible cross-hybridization with other targets. The GeneChip® arrays attempt to measure non-specific hybridization by using pairs of probes. A perfect match (PM) probe is always paired with a mismatch (MM) probe, which differs from the PM probe by a single base substitution at the middle (13th) position. It should be noted, however, that using such a control probe may create more problems than it solves: One must explicitly ensure that there is no other perfectly or well matching target for the MM probe.

Assuming that the raw expression values have been obtained by a reasonable aggregating procedure, they are still not comparable across different experiments. For example, in one experiment the target concentration in the sample may have been higher, and in another experiment, targets and probes may have been given more time to hybridize, and in a third experiment, the washing may have been more stringent. Therefore the expression values need to be *normalized* to make them comparable. In its most simple form, normalization consists of an affine transform $y \mapsto s \cdot y + c$ ($s > 0$, $c \in \mathbb{R}$), but more general classes of transformation can be used, for example, to stabilize the variance of the signals of high and low abundant transcripts (Rocke and Durbin, 2001; Durbin et al., 2002; Huber et al., 2002).

The combination of the two procedures, normalization and the inference of expression levels (or expression level ratios for cDNA arrays), is often referred to as *low-level analysis* of microarray data. A large number of differently motivated methods have been proposed (e.g., Chen et al., 1997; Beißbarth et al., 2000; Ideker et al., 2000; Kerr et al., 2000; Theilhaber et al., 2001; Yang et al., 2002; Huber et al., 2003). As of today, there seems to be no universally accepted best method.

Any downstream or *high-level analysis* then proceeds from the normalized expression values. An important basic procedure is the identification of differentially expressed

genes in two samples. Genes with similar expression profiles over varying conditions can be clustered to find functionally related genes or to discover cellular pathways. Previously unanalyzed tissue samples can be classified into two or several categories based on their expression profiles, with many potential applications in medical diagnostics and therapy (e.g., Golub et al., 1999; West et al., 2001; Oestreicher et al., 2001; Lock et al., 2002). Overall, the analysis of microarray data poses many algorithmic and statistical problems; it has been and still is a very active research field.

1.4 Probe Selection and Chip Design

While there exist many publications about microarray data analysis, comparatively little has been published about the design of microarrays, i.e., the selection of oligonucleotide probes and the layout of probes on the chip. A reason may be that until recently, only a limited number of pre-designed arrays was manufactured and sold by a few companies who kept their probe selection strategies as a secret. Nowadays, the probe sequences used for the Affymetrix GeneChip® arrays are published at the NetAffx website (Liu et al., 2003) and some probe selection criteria are known (Lockhart et al., 1996), but the exact selection procedure remains in the dark. Furthermore, up to now, no technology was available that would allow quick and inexpensive design of custom arrays. This is slowly beginning to change (e.g., see Hughes et al., 2001; Beier et al., 2001; Baum et al., 2003), and we expect an increasing demand for fast and reliable oligo design methodology and software over the next years.

In this section, we formulate several algorithmic problems in probe selection and microarray design, which are then treated in detail in the remainder of this thesis.

After the remarks in the previous sections, it is evident that each probe should be gene-specific, i.e., only the transcripts of a single gene should hybridize to a given probe, so the measurement taken at the probe's spot can be interpreted as the corresponding gene's expression level in the sample. Thus the main problem we are trying to solve is the following one.

Problem 1 (Main problem). Given hybridization parameters θ (e.g., temperature, salt concentration, number and density of probe molecules on the probe's spot, cRNA fragment length distribution, and other conditions specified in the experimental protocols) and a transcriptome $T = (t_1, \ldots, t_n)$ of transcript sequences, find a set of transcript-specific or *unique* probes for each transcript for quantitative expression analysis. •

There are many variations and facets of this problem, many interesting subproblems are hidden within or around this main problem. We discuss the following aspects in this thesis.

Affinity coefficients. It is important to realize that a target's cRNA or cDNA need not contain the *perfect* Watson-Crick complement of an oligo to hybridize; a near-perfect match will also suffice. Unfortunately, the extent of hybridization depends on many parameters, and the physical details of the hybridization process on oligonucleotide chips are not yet fully understood. Nevertheless, even though our results will be limited by the current lack of understanding, it is imperative that we obtain a reasonable approximate solution to the following problem, if our efforts in probe selection are to be successful.

Problem 2 (Fundamental problem). Given

- an mRNA transcript sequence t,
- an oligonucleotide probe sequence p,
- hybridization parameters θ,

find the relationship between the concentration x of the mRNA transcripts of the given kind ("expression level") and the raw intensity value y at the probe's spot. Assuming an approximately linear relationship, which especially means assuming negligible saturation effects, we need to find the proportionality constant $a = a(p, t; \theta) \geq 0$, which we call the *affinity coefficient*, such that $y = a \cdot x$. •

Indeed, Problem 2 is the fundamental problem underlying both microarray design and analysis, and is much more complicated than may be apparent from this simplified formulation. We propose several ways of attacking it in Chapter 2.

Unique probes. Assuming a basic understanding of the factors relating expression level to signal intensity, we are theoretically able to attack Problem 1 in a straightforward way.

The first step would be to identify *all* transcript-specific probes (of a certain length or length range). This is accomplished as follows. Let $p \lhd t$ express that p is a substring of t, and let $\varepsilon > 0$ (say, $\varepsilon \approx 0.1$) the lower affinity limit for transcript-specific probes. Then for each transcript t_j $(j = 1, \ldots, n)$, we find the set of probes $P_j = \{p \lhd t_j : a(p, t_j; \theta) \geq \varepsilon \text{ and } \sum_{t \neq t_j} a(p, t; \theta) \ll \varepsilon\}$. Here "$\ll \varepsilon$" should be read as "several orders of magnitude smaller than ε". Probes with specific affinity below ε or unspecific affinity above $\varepsilon/1000$, say, are discarded for safety tolerance reasons.

There are two problems with an obvious enumeration of all probe candidates based on the knowledge of affinity coefficients. First, we do *not* have (and will not have for several years to come) an exact solution to Problem 2. And second, even if we had one, it would be much too time-consuming to compute the coefficient for every probe-target pair in complex genomes, such as in the human genome. Assuming a lower bound of 30000 transcripts and about 1000 probe candidates per transcript, one

would have to check 30 million probe candidates against 30000 transcripts, resulting in 900 billion ($9 \cdot 10^{11}$) affinity computations. Some simplifying assumptions need to be made to solve this problem.

Note that we do not need to know the exact value of the affinity coefficients for making design decisions (although we might still need them for an accurate analysis). During the design phase, we are merely interested in making decisions of the type $a(p, t^*) \geq \varepsilon$ and $\sum_{t \neq t^*} a(p, t) \ll \varepsilon$. Thus we are led to the following problem.

Problem 3 (Sequence-based surrogate measure for affinity). Find a sequence-based surrogate quantity for the affinity coefficient $a(p, t; \theta)$ that, for a range of reasonable parameter settings θ, allows a quick decision $a(\cdot) \geq \varepsilon$ or $\sum a(\cdot) \ll \varepsilon$ without computing the exact $a(\cdot)$-values. •

We argue in Chapter 3 that the length of *longest common factor* (LCF) is a reasonable sequence-based measure on which design decision can be based. And even though it does not provide the exact affinity coefficient, it can always be used as a filter for more complex measures.

Problem 4 (Computation of LCF lengths). Find an efficient algorithm to compute the (relevant) LCF lengths of all probe candidates vs. all targets. •

All of Chapter 4 is devoted to the detailed description of an algorithm based on enhanced suffix arrays that computes the LCF between probe candidates and transcripts and thus ranks probe candidates according to their specificity.

It is practically more relevant not to merely enumerate all reasonable transcript-specific probe candidates, but to come back to the main problem (Problem 1) and select a *set* of 10 to 20 probes for each transcript from all candidates. Here additional considerations become important; these are discussed in Chapter 5.

Non-unique probes. An ideal set of transcript-specific (unique) probes cannot always be found. This is the case, for example, for highly homologous members of some gene families, such as the human heat shock proteins. A similar situation arises within the context of virus subtype identification or the identification of organic contaminants in water samples. In these latter problems a quantitative analysis is usually not required, but may still be desirable. Thus we are led to weaken the requirement of transcript-specific probes. Naturally, we must still take care not to choose oligonucleotides that give no information, but we can allow that probes specifically hybridize to a few transcripts with different affinity coefficients. We refer to them as *non-unique probes*. A new problem that arises in addition to probe selection is the decoding of the measured signals. The analogon to Problem 1 thus becomes

Problem 5 (Quantitative expression analysis with non-unique probes). Given hybridization parameters θ and a transcriptome $T = (t_1, \ldots, t_n)$ of sequences, find a set of (possibly non-unique) probes for each transcript, and specify an appropriate decoding procedure, such that quantitative statements about gene expression levels can be made from the observed probe signals. It is desirable to find a probe set that minimizes sources of error during decoding. •

The present reality, especially our lack of understanding of affinity coefficients, unfortunately makes large scale quantitative expression analysis with non-unique probes difficult. At least, the data that we can presently obtain with such an approach would be much less reliable than what we can get with unique probes. For the moment, it is thus of more interest to focus on qualitative applications, i.e., to make only present and absent calls for the transcripts of interest in a complex sample. It is assumed that a probe shows a positive signal if *any* of the transcripts it hybridizes to is present, that is, sufficiently highly expressed.

Problem 6 (Qualitative analysis with non-unique probes). Given a transcriptome $T = (t_1, \ldots, t_n)$ and a set of non-unique probe candidates with their respective target sets, efficiently compute a robust *design*, i.e., select a subset of the probes that allows inferring target presence or absence, even in the presence of false positive and false negative probe signals. Also specify an efficient and robust decoding procedure for this case. •

Suppose that we agree on a way of measuring a design's ability to separate different target sets. For economical reasons, it is the interesting to consider the following optimization problem.

Problem 7 (Design minimization). For a given target separation performance, find a design with a minimal number of probes from an initial set of non-unique probe candidates. •

Methods for attacking Problems 5 to 7 are described in Chapter 6. For quantitative expression analysis, we select designs by optimizing the condition of the affinity matrix associated to each design; decoding corresponds to solving a linear system of equations. For qualitative analysis, we present a greedy design heuristic and an integer linear programming based method that generates minimal designs. Probe signals are decoded by a stochastic Markov chain Monte Carlo Method.

Genome tiling chips. If unique probes are carefully designed, the inference of expression levels is relatively easy. If non-unique probes must be used, a decoding step is added to the procedure. In both cases, the whole transcriptome must be known before reliable probes can be chosen. Identifying the complete transcriptome of an

organism is difficult, however. A project called ENCODE (ENCyclopedia Of Dna Elements) is presently underway that aims to identify all functional elements of the human genome, and in particular, the transcribed regions. The genomic sequence is covered with oligonucleotide probes, e.g., by choosing one probe every 20 bp on average, as regularly as possible. Many of these probes are probably not unique and may exhibit complex cross-hybridization behavior. To identify new transcripts, we look for several consecutive probes on the genome that show a signal that cannot be explained by cross-hybridization from other transcripts. This is a complex decoding problem. Pre-selecting the probes for specificity can only help slightly in this case because every part of the genomic sequence must be covered.

Problem 8 (Transfrag identification). Given a double-stranded genomic sequence, find a relatively specific tiling probe set with prescribed minimum and maximum distances between probes. Specify an efficient and robust procedure to decode binary probe signals into locations of sufficiently long transcribed elements, called transcript fragments or *transfrags*. •

A strategy for solving Problem 8 is described in Chapter 7. Preliminary studies were carried out by Kapranov et al. (2002) on Chromosomes 21 and 22 of the human genome, but without explicitly taking cross-hybridization into account. Evidence for several new transcripts was found, but it was left unclear in how far this evidence can be explained away by cross-hybridization.

Chip production. Whatever our probe design and decoding strategy may be, once we have decided on a set of probes, and before we can actually perform experiments and decode the probe intensities, the chip must be produced. This involves further decisions:

1. A *nucleotide deposition sequence* must be found, from which all probes can be synthesized with the parallel photolithographic process described in Section 1.3.1.

2. Each probe sequence must be assigned to a location on the chip.

3. Given a deposition sequence s of length $|s|$, a sequence of $|s|$ masks must be defined. A mask specifies whether a nucleotide is added to a probe in a certain step. For a given layout and deposition sequence, several possible mask designs can exist.

Each probe is a subsequence of the deposition sequence, so the deposition sequence is a common supersequence of all probes (in contrast to a substring, a subsequence need not be contiguous). We are interested to keep the deposition sequence as short as possible.

Problem 9 (Deposition sequence minimization). For a given set of (thousands of relatively short) probe sequences, find the shortest common supersequence. •

There are two main reasons for this economy. First, a shorter sequence means shorter manufacturing times, higher throughput, and therefore better cost-effectiveness. Second, longer sequences increase the probability of errors in the manufacturing process, because masking is not perfect. Even when a probe is masked during a step, there is a slight chance that the nucleotide is appended nevertheless, finally resulting in a probe that is longer than it should be. To minimize the overall risk of errors, we must minimize the number of masked steps for each oligo, so we minimize the length of the deposition sequence.

Once the deposition sequence is fixed, production errors can be further reduced by minimizing the risk of exposure to stray light. This can be achieved by placing probes that share many masked steps next to each other, and to use the freedom in assigning probe nucleotides to deposition steps. Overall, it is attempted to minimize the total length of the border of the masks. This "border minimization problem" was recently studied by Hannenhalli et al. (2002) and Kahng et al. (2002) under the assumption that the deposition sequence is a periodic repetition of the alphabet `ACGT`. However, if the manufacturing process permits, finding a shorter deposition sequence will make the whole issue less of a problem.

We now examine the posed problems in detail. Chapters 2 to 5 present a bottom-up approach on unique probe design. The reader who wants to get to a bird's eye view quickly should start with Chapter 5 and then read the preceding chapters as necessary.

Chapter 2

Models for Feature Signal Intensity

As noted in the introduction, a major goal of probe design is to ensure transcript-specific oligonucleotides, i.e., to minimize the risk of cross-hybridization. Therefore it is important to discuss the general relationship between the quantity y that we can measure, i.e., the signal intensity at each spot or feature, and the quantity x that we want to know, i.e., the concentration of each transcript. While the basic idea underlying microarray experiments suggests that both things are one and the same, this is not true, even with the idealizing assumption of no cross-hybridization.

To set up a simple deterministic model for a single microarray experiment, we write

$$y_i = \sum_j A_{ij} \cdot x_j, \tag{2.1}$$

where y_i denotes the measured signal at spot i and x_j denotes the concentration of transcript j in the sample. The $A_{ij} \geq 0$ are called *affinity coefficients* or *intensity coefficients*. Assuming no cross-hybridization means that a single transcript $t(i)$ is associated to each spot i, so that $A_{ij} = 0$ for $j \neq t(i)$ and the model reduces to $y_i = a_i \cdot x_{t(i)}$ with $a_i > 0$ taking the role of $A_{i,t(i)}$. The model is more realistic, but considerably more complex, if general coefficients $A_{ij} \geq 0$ are allowed.

The purpose of this chapter is to shed some light on the nature of A_{ij}, a topic that is rarely discussed in the microarray analysis literature. In Section 2.1, we discuss the factors that influence an affinity coefficient: While the hybridization stability between probes and targets is an important factor, other quantities have an impact as well. Fortunately, it turns out that exact knowledge of affinity coefficients is not necessary for normalization and expression level estimation (Section 2.2), as long as sufficiently many probes with approximately the same coefficient distribution exist for each transcript (see Example 2.2). Nevertheless, it is desirable to know at least the order of magnitude of affinity coefficients, and here we follow two paths: First, we can predict the coefficients from the sequence with a thermodynamic two-state energy model (Section 2.3) that only models the hybridization strength, however. Second, we may predict the coefficients by analyzing existing intensity measurements across

several experiments (Section 2.4). We conclude this chapter by presenting a combined estimation approach.

2.1 The Nature of Affinity Coefficients

Recall that each y_i-value is a digitized representation, obtained from image analysis, of a fluorescence intensity generated by SAPE-stained biotin molecules that are attached to cRNA[1] bound to probe molecules at feature i on the chip. The cRNA was produced in several steps (reverse transcription using a T7-promotor-oligo(dT)-primer; then in-vitro transcription) from the original mRNA molecules. There are several systematic effects and several sources of stochastic error in this process. This section lists and classifies the most important ones.

- We assume that there are no saturation effects, i.e., that probe molecules are in excess and that the target concentration is within a range that avoids significant competition of targets for the same probe molecule. In fact, if we cannot ensure that this assumption holds at least approximately, not much can be done in terms of quantitative analysis for that spot.
- The overall cRNA concentration used for the hybridization influences the signal intensities in a linear way. Therefore the concentration is measured and standardized before each experiment.
- Measurements are perturbed by optical noise in the image scanning process. The noise can be a systematic general offset c, or a systematic feature-specific offset c_i included in the measurement y_i. It also has a stochastic component that we leave unspecified for the moment.
- The measurement y_i depends linearly on the number of bound SAPE-molecules, which in turn depends linearly on the number of biotin molecules attached to target fragments bound to spot i. Both dependencies vary stochastically, and we can assume that the standard deviation is proportional to y_i.
- The number of biotin molecules depends linearly on the number of `C`s and `U`s in all cRNA fragments attached to the probe molecules. For target j, this dependence is modeled by the intensity coefficient A_{ij}. Several influencing factors are apparent.

    1. It is fundamental to ask if the cRNA of transcript j has sufficient length to bind to probe i. As mentioned previously, experience shows that the in-vitro transcribed cRNAs have a varying length with an average of approximately 1000 nt; the exact length distribution depends on the protocol. If the probe

[1] For brevity, we only discuss GeneChip®-based eukaryotic expression analysis in the main text.

occurs much further than 1000 nt from the 3'-end of the mRNA transcript, there is a high change of the cRNA not containing the complementary sequence.

2. Recall that a random fraction of `C`s and `U`s in the cRNA is biotinylated. As long as the target preparation protocols are not changed, this fraction should be relatively constant because of the large number of cRNA fragments and the law of large numbers. It follows that for each transcript j that binds significantly strongly to probe i, the signal strength is proportional to a weighted fraction of `A`s and `G`s in a window around the binding site (in the DNA sequence). The further details of this relationship depend on the distribution of the target fragment length, which can vary with the experimental protocol. Stochastic noise can be assumed to be proportional to this factor.

3. The binding stability of a probe-target complex can be translated into a probability that probe and target have hybridized in a stable way after the annealing phase and are not washed off in the washing phase. Because of the overall large number of fragments, this probability immediately translates into the fraction of bound fragments. Again, stochastic noise can be assumed to be proportional to this factor.

4. As Naef and Magnasco (2003) point out, biotinylation interferes with the base-pairing ability of nucleotides. Therefore, the fraction of biotinylated nucleotides should neither be too low, as this would lead to low signal strength because of SAPE binding, nor too high, as this would also cause low signal strngth, because of reduced hybridization stability. This problem disappears in the prokaryotic labeling protocol, where a single biotinylated `U` is appended to each target fragment.

5. If a probe is highly self-complementary, it may fold back onto itself and cease to be available for hybridization. This is much more a concern with PCR-product arrays and polynucleotide arrays, but the effect cannot be completely ruled out.

6. A more serious concern is the complementarity of the cRNA transcripts. A cRNA fragment can form stable secondary structures with itself or other fragments, given a sufficiently high local sequence complementarity. It is still conceivable that a cRNA fragment that has either formed a secondary structure with itself or bound to another fragment hybridizes to a probe when the required part of the fragment is still single stranded. These possibility cannot be controlled by probe selection and chip design decisions; one can only attempt to choose hybridization conditions in such a way that the probe-target hybridization is preferred over target-target binding.

7. Another reason why a probe molecule may not be available for hybridization (with the intended target) is that some step may have failed during its synthesis process. Nucleotide addition might have failed at a particular step; in this case the molecule should be irreversibly protected and deactivated, and hence not hybridize at all. Worse is the case when additional nucleotides have been inserted due to erroneous activation by stray light. This problem usually concerns a certain fraction of all probe molecules at a spot, which depends on the masks and deposition sequence used in the synthesis process and is hard to model explicitly. It is best to ensure that this problem is minimized.

8. Any cRNA fragment can bind to at most one feature. Therefore, if there are two or more probes whose sequences are taken from physically close positions in the same target sequence, and these features are physically close to each other on the chip, they may compete for the target fragments. The best way to avoid this undesirable effect is to choose probes that are spread out over the target and do not cluster. Another good strategy is to distribute the chosen probes over the chip to avoid competition at the same location. Competition should be avoided to the maximum extent possible and as uniformly as possible for all features, because modeling it exactly difficult.

The above list suggests ways to extend Model (2.1) stochastically, and how to model the affinity matrix A. Let us focus on the affinity coefficients for now.

Claim 2.1. Ignoring the modeling of erroneous probes and of competition effects, where different features compete for the same target fragment, and assuming fixed hybridization parameters θ, the affinity matrix (A_{ij}) can in principle be computed by sequence analysis, i.e., by comparing the sequence p_i of probe i with the sequence t_j of transcript j, and potentially by comparing all t_j with each other for secondary-structure formation. ◇

In practice, this can still be difficult to impossible because we do not understand the physics of the hybridization in detail. And of course in reality, competitive effects of probes may well be present. For now, let us put these concerns aside and write

$$A_{ij} = \rho_{ij} \cdot \beta_{ij} \cdot \gamma_{ij} \cdot \sigma_{ij}. \tag{2.2}$$

In principle, all individual factors can be estimated by combining sequence analysis with appropriate physical or statistical models. They have the following meaning,

- The factor ρ_{ij} depends on the position of probe i within the mRNA sequence of transcript j, relative to its 3'-end. It can be assumed to be approximately constant, $\rho_{ij} \approx 1$, for probe locations within 1000 nt of the 3'-end, and we have

$\rho_{ij} \approx 0$ if the probe location is much further away than 2000 nt. Of course, these distances are not absolute and can change with the protocol.

- The factor β_{ij} is a factor for the binding stability between probe i and target j; it is studied in more detail in Section 2.3.
- The factor γ_{ij} models the sequence composition and is a function of the weighted fraction of `A`s and `G`s around the binding site.
- The last factor σ_{ij} denotes the fraction of probe-i-target-j-hybridizations that are *not* lost because of self-complementarity of probe i or because of the interactions of target j with itself and other targets. We may factor $\sigma_{ij} = \sigma'_i \cdot \sigma''_j$ into these two independent effects. We have $\sigma'_i \approx 1$ for probes with negligible self-complementarity and $0 < \sigma'_i \ll 1$ for highly self-complementary probes. Special purpose tools for RNA folding, such as MFOLD (Zuker et al., 1999; Zuker, 2003) can be used to predict σ'_i. An idea how probes with a small σ''_j-factor can be avoided is described in Chapter 5.

To find a compromise between realism and simplicity, we assume that the binding stability is the dominant systematic effect. That is, we assume that all probes for transcript j are well spread over the last 1000 nt of the transcript, that the probes are not significantly self-complementary, that the sequence composition of the targets is relatively homogeneous, and that their interactions do not cause extreme differences in affinity coefficients. We also assume that the coefficients are constant between different experiments carried out under the same conditions. We then have $A_{ij} \propto \beta_{ij}$, and it follows that it is worthwhile to study the stability factor in more detail; this is done in Section 2.3.

2.2 Normalization & Expression Level Estimation

The above list of systematic effects and stochastic errors suggests the following extension of model (2.1):

$$y_i = \varepsilon_i + c_i + \alpha \cdot \sum_j \exp(\eta_{ij}) \cdot A_{ij} \cdot x_j. \tag{2.3}$$

Here c_i is a deterministic per-spot bias introduced, for instance, by systematic per-spot effects of optical noise. For simplicity, it could be replaced by a global constant c. There is a stochastic additive per-spot noise term ε_i with zero expectation, which we may assume to have a Gaussian distribution because of many independent sources of error. An overall scaling factor $\alpha > 0$ adjusts for different overall cRNA concentration. There is also an interaction-specific stochastic multiplicative error term $\exp(\eta_{ij})$ which models the noise that is proportional to the true measurable signal $A_{ij}\, x_j$. Rocke and

Durbin (2001) model the multiplicative error with a log-normal distribution, i.e., they assume that η_{ij} has a Gaussian distribution with mean zero, mainly for convenience.

Application to normalization. Assuming no cross-hybridization, that is, only one target per feature, and for the moment, exactly one feature per target such that the target of feature i is $t(i) = i$, we obtain a variation of models proposed by Li and Wong (2001) and Rocke and Durbin (2001) that differs only by the explicit inclusion of the affinity coefficient a_i and the parameters c_i being constant $c_i \equiv c$. One has

$$y_i = \varepsilon_i + c + \alpha \cdot \exp(\eta_i) \cdot a_i \cdot x_i,$$

with constant variances of η and ε over all features. The model of Rocke and Durbin (2001) is useful for inter-array normalization. Setting the scale α and the offset c to pre-defined values (e.g., $\alpha = 1$, $c = 0$) for the first array, the parameters for the other arrays can be estimated from the data, assuming known intensity coefficients and that the majority of genes or some particular known genes do not change their expression significantly. If control targets of known concentrations are used on each array, the model can even be calibrated to absolute quantities. Frequently, however, one is more interested in differences or ratios (fold-changes) of expression levels than in absolute quantities. In either case, a transformation that makes the variance of y_i independent of x_i (a so-called variance-stabilizing transform) has been successfully applied. The papers by Durbin et al. (2002) and Huber et al. (2002, 2003) provide details.

We point out again that in this context, it is assumed that no significant cross-hybridization occurs and that the remaining amount can be absorbed into the offset term c and its stochastic noise ε_i. In our presentation, it is also assumed that the affinity coefficients a_i are known; they are unidentifiable and absorbed into x_i in the formulation in the above mentioned references. The reason is that these references do not explicitly consider the case that many probes measure the expression level of the same transcript.

Estimation of Expression Levels. On real oligonucleotide chips, there are several probes with different affinity coefficients per transcript. Hence when the a_i or more generally the A_{ij} are known, and we set the global scale $\alpha = 1$ and the global offset $c = 0$ without loss of generality for a single experiment, the system

$$y_i = \varepsilon_i + \sum_j \left[\exp(\eta_{ij}) \cdot A_{ij} \cdot x_j\right]$$

has more measurements y_i than unknowns x_j. This allows in principle to estimate the remaining two parameters, i.e., the variances of the η and ε terms, from the data, and to use a robust maximum likelihood approach to estimate x from y. This still remains a complex problem, however; the book of Rousseuw and Leroy (1987) contains an

arsenal of methods. As our main interest is probe selection, not data analysis, we will avoid the statistical and methodological details here, and only illustrate some general issues. Suffice it to state that "solving" the over-determined system $y = A \cdot x$ for x can be formulated as a (possibly non-smooth) minimization problem $\|y - A \cdot x\| \to \min$, where $\|\cdot\|$ denotes an appropriate norm. As the whole microarray technology contains many potential sources of errors that may lead to extreme outlier measurements at some features, such as defect or very intense spots, it is not advisable to ask for a classical least-squares solution; a more robust method should be employed. Written out, a robust version of the minimization problem could be stated as

$$\sum_i w_i \cdot \left| y_i - \sum_j A_{ij} \cdot x_j \right|^p \to \min,$$

where the $w_i > 0$ are weights that could depend on the variances of the η- and ε-terms in the statistical model, and $p \in]0, 2]$ is a robustness parameter. The choice of $p = 2$ would correspond to a weighted least-squares solution, which is not robust. Smaller values of p lead to greater robustness, and for example $p = 1$ poses the problem to minimize a weighted sum of absolute errors. The general solution of such a minimization problem, and especially with the constraints that all x_j should be nonnegative, is beyond the scope of this thesis.

Fortunately, the case of most practical interest is simple. Suppose that, for a certain target transcript, we have several signal measurements y_i from *specific* probes p_i with different affinity coefficients a_i. If we knew the coefficients, a robust way to infer the expression value x would be to minimize the sum of absolute errors $\sum_i |y_i - a_i \cdot x|$. The solution minimizing the sum of absolute errors is the median, and so we take $x = \text{median}(y_i/a_i)$. This easy solution exists because we assumed ideally specific probes and the matrix A_{ij} is a block diagonal matrix in this case. We also assumed that we know the coefficients a_i.

In practice, even if we assume that the probes are sufficiently specific, we do not know the exact values of the a_i, and a disturbing question arises: Why (or when) does quantitative gene expression analysis work in practice although we do not know affinity coefficients?

One option is to simply ignore the coefficients and take $x' = (1/a) \cdot \text{median}(y_i)$, where $a > 0$ is a "typical" affinity coefficient, i.e., a single-value approximation of the affinity coefficients of all probes. If the magnitude and distribution of the affinity coefficients for each transcript are similar, this method can produce results comparable to the above method.

Example 2.2. Suppose that there are two transcripts and three specific probes for each transcript. Further assume that we know the affinity coefficients and observe the

following signal intensities.

$$\begin{pmatrix} 1000 \\ 1200 \\ 700 \\ 120 \\ 110 \\ 90 \end{pmatrix} = \begin{pmatrix} 10 & \\ 12 & \\ 7 & \\ & 12 \\ & 11 \\ & 9 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Knowing the affinity coefficients, this over-determined system has the exact solution $x_1 = 100$, $x_2 = 10$. Not knowing the matrix and assuming an overall coefficient of $a = 1$, the medians for each transcript are $x'_1 = 1000$ and $x'_2 = 110$, which is approximately proportional to the correct solution.

The software that comes with the Affymetrix GeneChip® arrays uses a related "robust averaging" approach that differs mainly in the details. Each transcript is represented by 10 to 20 relatively specific 25-mer probes. In fact, each probe is really a probe pair, consisting of a perfect match (PM) probe and a mismatch (MM) probe that differs from the PM probe only at the middle nucleotide. It is assumed that the MM probe mainly captures non-specific hybridization and, therefore the probe pair signal is defined as the difference PM − MM. If the difference is negative, the signal is discarded. Of the remaining pairs, a weighted trimmed geometric average is taken as the expression value (Affymetrix, Inc., 2002).

We conclude that the current practice of gene expression level determination can in fact agree rather well with the "true" values that one would obtain from a full statistical model. The essential conditions for this to work are that the probes are reasonably specific, there are enough probes per transcript, and the affinity coefficients for each transcript have approximately the same distribution. In this case a sufficiently robust estimation procedure can ignore their existence. In Chapter 6, it is shown that this fortunate situation breaks down when one begins to consider non-unique probes.

2.3 Thermodynamic Hybridization Stability

We have remarked that the stability of the probe-target hybridization is one important factor influencing the affinity coefficient. Often, it is the only factor that is considered (Li and Stormo, 2001; Kaderali and Schliep, 2002), mainly in the form of the *melting temperature* of the probe-target duplex. In this section we give a brief overview over the state-of-the-art thermodynamic models being used to predict the stability expressed as Gibbs free energy. Using a two-state model, Gibbs free energy can then be converted into the stability factor β_{ij} of the affinity coefficient in Equation (2.2). We also argue

why we think that the melting temperature is a less appropriate quantity and should be given less emphasis for probe selection.

2.3.1 Basic Notions of Thermodynamics

As mentioned in Section 1.1, the backbone of a DNA or RNA molecule consists of alternating (deoxy)ribose sugars and phosphate groups. The OH groups form hydrogen bonds with water, and phosphate is always negatively charged. Therefore the backbone is very hydrophilic. The bases forming the side groups along the backbone are nearly planar and hydrophobic. The stacking interactions between the hydrophobic parallel planes in the middle of the hydrophilic backbone are one reason for the stability of helical DNA; the other reason is given by the hydrogen bonds between the base pairs (3 for a `G-C`-pair, 2 for an `A-T`-pair).

The stability of a double-stranded DNA duplex or an RNA-DNA hybrid can be expressed quantitatively in terms of the Gibbs free energy ΔG, which is introduced in the following paragraphs, following Atkins and de Paula (2002).

Internal energy. In thermodynamics, the total kinetic and potential energy of a system's molecules is called the *internal energy* U of the system. Internal energy, heat, and work are all measured in the same unit, the joule (J), which is defined as 1 J = 1 kg m^2 s^{-2}. The *molar internal energy* U_m of a substance is its internal energy per amount of substance; it is usually given in kilojoules per mole (kJ mol^{-1}) or kilocalories per mole, where 1 kcal = 4184 J exactly.

We denote by ΔU the change in internal energy when the system changes from an initial state i with internal energy U_i to a final state f with internal energy U_f. It has been found experimentally that the internal energy of a system may be changed either by doing work on the system or by heating it, and that if a system is isolated from its surroundings, then no change in internal energy takes place. These observations are summarized as the *first law of thermodynamics* that states that the internal energy of an isolated system is constant. It follows that $\Delta U = q + w$, where w denotes work done on the system and q denotes energy transfered as heat to the system. For infinitesimal quantities, we have

$$\mathrm{d}U = \mathrm{d}q + \mathrm{d}w.$$

The internal energy is a *state* function in the sense that its value depends only on the current state of the system and is independent of how that state has been reached.

Enthalpy. When the system is free to change its volume V while it is heated, some of the supplied energy is returned to the surroundings as expansion work. We thus have $\mathrm{d}U < \mathrm{d}q$ and $\mathrm{d}w > 0$. It would be desirable, however, to have a state function H

such that $\mathrm{d}H = \mathrm{d}q$. Under the assumption of constant pressure p and no additional work except expansion work, the definition $H := U + pV$ satisfies this property. H is called the *enthalpy* of the system.

> As a combination of the state functions U, p and V, H is also a state function. Obviously we have $\mathrm{d}H = \mathrm{d}U + p\,\mathrm{d}V + V\,\mathrm{d}p$. We use the relationship $\mathrm{d}U = \mathrm{d}q + \mathrm{d}w$ from above and the assumption that the system is in mechanical equilibrium with its surroundings at constant pressure p and does only expansion work; so $\mathrm{d}w = -p\,\mathrm{d}V$ and $\mathrm{d}p = 0$. It follows that $\mathrm{d}H = \mathrm{d}q - p\,\mathrm{d}V + p\,\mathrm{d}V + 0 = \mathrm{d}q$. □

Changes in enthalpy are normally reported for processes taking place under a standard set of conditions. The *standard enthalpy change* is the change in enthalpy for a process in which the initial and final substances are in their standard states (their pure forms at a specified temperature at 1 bar). For a chemical reaction with reactants and products, the *standard reaction enthalpy* $\Delta_\mathrm{r}H^\circ$ is such a change and defined as

$$\Delta_\mathrm{r}H^\circ := \sum_{\text{Products P}} \nu(\mathrm{P}) \cdot H^\circ_\mathrm{m}(\mathrm{P}) - \sum_{\text{Reactants R}} \nu(\mathrm{R}) \cdot H^\circ_\mathrm{m}(\mathrm{R}),$$

where $H^\circ_\mathrm{m}(\mathrm{J})$ is the standard molar enthalpy of substance J, and $\nu(\mathrm{J})$ is the stoichiometric coefficient of substance J in the reaction.

Hess's law states that the standard enthalpy of an overall reaction is the sum of the standard enthalpies of the individual reactions into which a reaction may be divided.

Entropy. The internal energy U is a state function that tells us which state changes are *permissible*: Only those changes may occur for which the internal energy of an isolated system remains constant. It does not tell us which changes are *spontaneous* and which ones are not. To answer the latter question, another state function has been found; it is called the *entropy* S and is a measure of the molecular disorder of a system.

The thermodynamic definition of entropy concentrates on the change in entropy $\mathrm{d}S$ that occurs as a result of a process. It is motivated by the idea that a change in the extent to which energy is dispersed in a disorderly manner depends on how much energy is transfered as heat: Heat stimulates disorderly motion; work does not change the degree of disorder. Now at a high temperature, the molecules of a system are already highly disorganized; so a small additional transfer of energy will result in a relatively small additional disorder. In contrast, at low temperatures molecules have access to far fewer energy states, and the same quantity of heat will have a pronounced effect on the degree of disorder. This suggests to define entropy changes for reversible state transitions in terms of the differential equation

$$\mathrm{d}S = \frac{\mathrm{d}q_\mathrm{rev}}{T}, \qquad \text{so} \qquad \Delta S = \int_\mathrm{i}^\mathrm{f} \frac{\mathrm{d}q_\mathrm{rev}}{T},$$

where the integral is over a reversible state change path from initial state i to final state f. It can be shown that S is indeed a state function (Atkins and de Paula, 2002, Section 4.2): The integral vanishes when $\mathrm{d}S$ is integrated over a closed path, i.e., $\oint \mathrm{d}q_{\mathrm{rev}}/T = 0$.

So far the entropy S is defined via its change; not on an absolute scale. Observations indicate that the entropy of all perfect crystalline substances is zero at $T = 0$. Therefore one adopts the convention to report entropies on the basis that $S(0) = 0$. When a substance J is in its standard state at temperature T, the *standard molar entropy* is denoted $S^{\circ}_{\mathrm{m}}(T;\mathrm{J})$. For a chemical reaction at temperature T, the *standard reaction entropy* is defined as

$$\Delta_{\mathrm{r}}S^{\circ} := \sum_{\text{Products P}} \nu(\mathrm{P}) \cdot S^{\circ}_{\mathrm{m}}(T;\mathrm{P}) - \sum_{\text{Reactants R}} \nu(\mathrm{R}) \cdot S^{\circ}_{\mathrm{m}}(T;\mathrm{R}),$$

Consider a system in thermal equilibrium with its surroundings at temperature $T = T_{\mathrm{sur}}$. A spontaneous state change of the system entails an entropy change in the system $\mathrm{d}S$ and an entropy change in the surroundings $\mathrm{d}S_{\mathrm{sur}}$. Regardless of how the change is brought about in the system, reversibly or irreversibly, we can calculate the change of entropy of the surroundings by $\mathrm{d}S_{\mathrm{sur}} = \mathrm{d}q_{\mathrm{sur}}/T_{\mathrm{sur}}$, if we assume the surroundings to be a reservoir of either constant pressure or constant volume. Because the process might be irreversible, the entropy change in the system may exceed $|\,\mathrm{d}S_{\mathrm{sur}}|$. Thus we can write

$$\mathrm{d}S \geq -\,\mathrm{d}S_{\mathrm{sur}} = \frac{\mathrm{d}q}{T},$$

because $\mathrm{d}q_{\mathrm{sur}} = -\,\mathrm{d}q$. The above inequality is known as the *Clausius inequality*.

The *second law of thermodynamics* then follows: For an isolated system, $\mathrm{d}q = 0$, and in the course of a spontaneous change, the total entropy increases: $\Delta S_{\mathrm{tot}} = \Delta S + \Delta S_{\mathrm{sur}} > 0$.

Gibbs energy. When heat is transfered between the system and the surroundings at constant pressure (as it is for most biological processes), we have $\mathrm{d}q = \mathrm{d}H$, and the Clausius inequality becomes $\mathrm{d}S \geq \mathrm{d}H/T$ or

$$\mathrm{d}H - T \cdot \mathrm{d}S \leq 0.$$

At constant enthalpy and pressure, this becomes $\mathrm{d}S \geq 0$: The entropy of the system must increase because there can be no change in entropy of the surroundings ($\mathrm{d}H = \mathrm{d}q = 0$). At constant entropy and pressure, the inequality reads $\mathrm{d}H \leq 0$: The enthalpy of the system must decrease if the entropy of the system is constant, because then it is necessary to have an increase in the entropy of the surroundings.

The above form of the Clausius inequality suggests to define a new thermodynamic quantity, the *Gibbs energy* G as

$$G := H - T \cdot S; \qquad \text{so} \qquad \mathrm{d}G = \mathrm{d}H - T \cdot \mathrm{d}S.$$

Thus at constant temperature and pressure, chemical reactions are spontaneous in the direction of decreasing Gibbs energy $\mathrm{d}G \leq 0$.

The *standard Gibbs energy of reaction* is defined in terms of the standard reaction enthalpy and entropy by

$$\Delta_\mathrm{r} G^\circ = \Delta_\mathrm{r} H^\circ - T \cdot \Delta_\mathrm{r} S^\circ.$$

It is the difference in standard molar Gibbs energies G°_m of the products and reactants in their standard states at the temperature of the reaction.

Intuitively, $\Delta_\mathrm{r} G^\circ$ can be interpreted as a difference in stability between the products and the reactants.

2.3.2 Nearest Neighbor Models

Our goal is to estimate the Gibbs energy change for perfect Watson-Crick DNA-duplex or RNA/DNA-hybrid formation in the context of a chip hybridization experiment. We consider the reaction $\mathrm{S}_1 + \mathrm{S}_2 \rightarrow \mathrm{D}$, where S_i $(i = 1, 2)$ denotes a single stranded DNA or RNA molecule, and D denotes a double stranded DNA or a hybrid molecule. Because the stability depends on stacking interactions and sequence composition, it is reasonable to attempt the prediction of the standard Gibbs energy of reaction $\Delta_\mathrm{r} G^\circ$ by a nearest-neighbor (NN) model, i.e., a model that expresses $\Delta_\mathrm{r} G^\circ$ for a duplex of n bp as a sum of $n-1$ terms for consecutive overlapping dinucleotides plus additional terms for the ends. Additionally for DNA/DNA duplexes, the model should be symmetric with the respect to the operation of taking the reverse complement.

In a review article, SantaLucia (1998) presents unified parameters for such a model and discusses the history of differently obtained parameter sets. In detail, for a DNA sequence $\mathrm{S}_1 = 5'\text{-}(s_1, \ldots, s_n)\text{-}3'$, and S_2 being the reverse complement of S_1, the model specifies that

$$\Delta_\mathrm{r} G^\circ = \left(\sum_{i=1}^{n-1} \Delta_\mathrm{r} G^\circ(s_i s_{i+1}) \right) + \Delta_\mathrm{r} G^\circ(\mathrm{init}(s_1)) + \Delta_\mathrm{r} G^\circ(\mathrm{init}(s_n)) + \Delta_\mathrm{r} G^\circ(\mathrm{sym}).$$

The last term $\Delta_\mathrm{r} G^\circ(\mathrm{sym})$ is a symmetry correction: For self-complementary oligonucleotides, it takes the value of +0.43 kcal/mol; otherwise it is zero.

Table 2.1 specifies the parameters at a sodium concentration of 1 M NaCl and at a fixed temperature of 37°C or 310.15 K. The components enthalpy $\Delta_\mathrm{r} H^\circ$ and entropy

Table 2.1: Unified NN parameters for the standard reaction enthalpy $\Delta_r H^\circ$, entropy $\Delta_r S^\circ$, and the Gibbs energy of reaction $\Delta_r G^\circ$ at 37°C and 1 M NaCl. Under a uniform random sequence model, the average Gibbs energy of reaction is −1.42 kcal/mol per dinucleotide (SantaLucia, 1998).

Dinucleotide	$\Delta_r H^\circ$ [kcal/mol]	$\Delta_r S^\circ$ [cal/(mol·K)]	$\Delta_r G^\circ$ [kcal/mol]
5′-AA-3′ ≡ 5′-TT-3′	−7.9	−22.2	−1.00
5′-AT-3′	−7.2	−20.4	−0.88
5′-TA-3′	−7.2	−21.3	−0.58
5′-CA-3′ ≡ 5′-TG-3′	−8.5	−22.7	−1.45
5′-GT-3′ ≡ 5′-AC-3′	−8.4	−22.4	−1.44
5′-CT-3′ ≡ 5′-AG-3′	−7.8	−21.0	−1.28
5′-GA-3′ ≡ 5′-TC-3′	−8.2	−22.2	−1.30
5′-CG-3′	−10.6	−27.2	−2.17
5′-GC-3′	−9.8	−24.4	−2.24
5′-GG-3′ ≡ 5′-CC-3′	−8.0	−19.9	−1.84
init(G), init(C)	0.1	−2.8	0.98
init(A), init(T)	2.3	4.1	1.03
Symmetry corr.	0	−1.4	0.43

$\Delta_r S^\circ$ are also given; thus $\Delta_r G^\circ$ can be computed for other temperatures by using the Gibbs-Helmholtz equation $\Delta G = \Delta H - T\Delta S$. Different salt concentration can be accommodated by applying a salt correction to each individual dinucleotide entropy term:

$$\Delta_r S^\circ(\text{dinucleotide}, [\text{Na}^+]) = \Delta_r S^\circ(\text{dinucleotide}) + 0.368 \cdot \ln[\text{Na}^+].$$

Similar parameters for RNA/DNA-hybrids are given by Sugimoto et al. (1995); they are shown in Table 2.2. No salt correction is given in this case.

There are some caveats:

- The NN model is only a model, not the reality. All values, especially the salt correction, are approximate, and the parameters reported by different researchers vary; for example, compare SantaLucia (1998) with Sugimoto et al. (1996, data not shown here).
- For temperatures substantially different from 37°C, results become increasingly inaccurate, because the different heat capacities of products and reactants should be considered.
- The parameters were derived by measuring the thermodynamic properties of free short oligonucleotides in solution. The conditions under which oligonucleotides attached to the chip surface hybridize to long cRNA or cDNA fragments in

Table 2.2: Thermodynamic NN parameters for RNA/DNA hybrid duplex initiation and propagation at 37°C and in 1 M NaCl buffer (Sugimoto et al., 1995). r(·) denotes RNA sequence; d(·) denotes DNA sequence.

Hybrid	$\Delta_{\mathrm{r}}H^{\circ}$ [kcal/mol]	$\Delta_{\mathrm{r}}S^{\circ}$ [cal/(mol·K)]	$\Delta_{\mathrm{r}}G^{\circ}$ [kcal/mol]
r(5′-AA-3′) d(3′-TT-5′)	−7.8	−21.9	−1.0
r(5′-AC-3′) d(3′-TG-5′)	−5.9	−12.3	−2.1
r(5′-AG-3′) d(3′-TC-5′)	−9.1	−23.5	−1.8
r(5′-AU-3′) d(3′-TA-5′)	−8.3	−23.9	−0.9
r(5′-CA-3′) d(3′-GT-5′)	−9.0	−26.1	−0.9
r(5′-CC-3′) d(3′-GG-5′)	−9.3	−23.2	−2.1
r(5′-CG-3′) d(3′-GC-5′)	−16.3	−47.1	−1.7
r(5′-CU-3′) d(3′-GA-5′)	−7.0	−19.7	−0.9
r(5′-GA-3′) d(3′-CT-5′)	−5.5	−13.5	−1.3
r(5′-GC-3′) d(3′-CG-5′)	−8.0	−17.1	−2.7
r(5′-GG-3′) d(3′-CC-5′)	−12.8	−31.9	−2.9
r(5′-GU-3′) d(3′-CA-5′)	−7.8	−21.6	−1.1
r(5′-UA-3′) d(3′-AT-5′)	−7.8	−23.2	−0.6
r(5′-UC-3′) d(3′-AG-5′)	−8.6	−22.9	−1.5
r(5′-UG-3′) d(3′-AC-5′)	−10.4	−28.4	−1.6
r(5′-UU-3′) d(3′-AA-5′)	−11.5	−36.4	−0.2
Initiation	1.9	−3.9	3.1

solution can be expected to be very different. Zhang et al. (2003) have recently proposed a tentative model for this situation.

- The NN model appears to be a reasonable approximation for perfect Watson-Crick duplexes. For duplexes with internal mismatches or loops, the NN model is not likely to yield accurate $\Delta_r G^\circ$ values because the normally parallel base-pair planes, and therefore the stacking interactions between neighbors, are disturbed in complex ways. Nevertheless, tentative parameters for single-base mismatches have been published by Allawi and SantaLucia (1997, 1998a,b,c) and Peyret et al. (1999).

2.3.3 From Gibbs Energy to Stability Factors

The question of interest is how we can relate the Gibbs energy of reaction $\Delta_r G^\circ$ to the stability factor β_{ij} in Equation (2.2), at least for perfect Watson-Crick duplexes. In (2.2), the β-factor models the probability that a cRNA or cDNA fragment of target j that is at feature location i has hybridized.

The hybridization $S_1 + S_2 \leftrightarrow D$ occurs under constant pressure and temperature and is reversible. It is assumed that a fragment close to a feature location with a matching probe has only two options: to be in the single stranded random-coil state, or to be in the more ordered hybridized state. As the forward reaction proceeds, the amount of substance of the single strands becomes smaller by $\mathrm{d}\xi$. The quantity ξ is called the *extent of reaction* and reported in moles.

While the standard molar Gibbs energy of each substance is constant, the total Gibbs energy G of the reacting system changes because the amounts of substance change. The *reaction Gibbs energy* $\Delta_r G$ is defined as the derivative to the Gibbs energy with respect to the extent of reaction. It is equal to the difference between the chemical potentials of products and reactants at the instantaneous composition of the reaction mixture, which justifies the Δ-notation. It can also be expressed in terms of the standard reaction Gibbs energy $\Delta_r G^\circ$ (Atkins and de Paula, 2002, Chapter 9):

$$\Delta_r G = \frac{\partial G}{\partial \xi} = \mu_{\text{Products}} - \mu_{\text{Reactants}} = \Delta_r G^\circ + RT \ln Q.$$

Here μ denotes chemical potential, T is the temperature in Kelvin,

$$\begin{aligned} R &= 8.3145 \text{ J K}^{-1} \text{ mol}^{-1} = 1.987 \text{ cal K}^{-1} \text{ mol}^{-1} \\ &= k \cdot N_A && \text{is the gas constant,} \\ k &= 1.38065 \cdot 10^{-23} \text{ J K}^{-1} && \text{is the Boltzmann constant,} \\ N_A &= 6.02214 \cdot 10^{23} \text{ mol}^{-1} && \text{is the Avogadro constant,} \end{aligned}$$

and Q is the *reaction quotient*, which can be approximated by the molar concentrations in this case, $Q = \frac{[\mathrm{D}]}{[\mathrm{S}_1][\mathrm{S}_2]}$. The concentrations are functions of ξ.

If $\Delta_\mathrm{r}G < 0$ at the current extent of reaction, then the forward reaction is spontaneous and the reaction is called exergonic (work-producing). If $\Delta_\mathrm{r}G > 0$, the backward reaction is spontaneous and the reaction is called endergonic (work-consuming). If $\Delta_\mathrm{r}G = 0$, the corresponding extent of reaction represents the thermodynamic equilibrium. The reaction quotient Q at equilibrium is denoted by K. One obtains thus

$$K = \exp(-\Delta_\mathrm{r}G^\circ/(RT)) = \frac{[\mathrm{D}]_\mathrm{eq}}{[\mathrm{S}_1]_\mathrm{eq}[\mathrm{S}_2]_\mathrm{eq}}. \tag{2.4}$$

As noted, a target fragment (represented here by S_1, say) can be in one of two states. The probability $\overline{\beta}$ that it is single stranded is

$$\overline{\beta} = \frac{[\mathrm{S}_1]_\mathrm{eq}}{[\mathrm{S}_1]_\mathrm{eq} + [\mathrm{D}]_\mathrm{eq}} = \frac{1}{1 + [\mathrm{D}]_\mathrm{eq}/[\mathrm{S}_1]_\mathrm{eq}} = \frac{1}{1 + [\mathrm{S}_2]_\mathrm{eq} \cdot K}.$$

Therefore the probability $\beta = 1 - \overline{\beta}$ that the fragment has hybridized is

$$\beta = \frac{[\mathrm{S}_2]_\mathrm{eq} \cdot K}{1 + [\mathrm{S}_2]_\mathrm{eq} \cdot K} = \frac{1}{1 + [\mathrm{S}_2]_\mathrm{eq}^{-1} \cdot K^{-1}} = \frac{1}{1 + [\mathrm{S}_2]_\mathrm{eq}^{-1} \cdot \exp(\Delta_\mathrm{r}G^\circ/(RT))}. \tag{2.5}$$

The basic premise was that the probe molecules S_2 are in excess so that no saturation effects occur. Thus we can treat $[\mathrm{S}_2]_\mathrm{eq}$ as a constant and assume that is still approximately equal to the probe concentration on the original chip.

Example 2.3. For $\Delta_\mathrm{r}G^\circ = -19.12$ kcal/mol $= -80000$ J/mol at $T = 45°\mathrm{C} = 318.15$ K, we have $\exp[\Delta_\mathrm{r}G^\circ/(RT)] = \exp(-30.2429) \approx 10^{-13}$. We assume a hypothetical probe concentration of 10^{-6} M and find that β is fairly close to 1: $\beta \approx 1 - 10^{-7}$.

We have seen how the stability factor β in (2.2) can be estimated in principle. It should be noted, however, that the simple two-state model (no intermediate reaction steps) may not be entirely accurate for the annealing reaction, but can still provide a reasonable approximation. The exact role of $[\mathrm{S}_2]_\mathrm{eq}$ remains somewhat mysterious as the model assumes that the molecules can freely move against each other. In the "chip reality", the targets have to find the probes. Therefore $[\mathrm{S}_2]_\mathrm{eq}$ should probably take a relatively low value, much lower than the true concentration at the feature location. We also do not know the correct $\Delta_\mathrm{r}G^\circ$ values (see the caveats in Section 2.3.2) for array hybridizations yet, but this problem is likely to be addressed in the near future (Zhang et al., 2003). In (2.5), we have ignored the possibility of multiple binding sites in the same target. Recall that the targets are fragmented. If the same binding site exists twice in the same target at distant positions, two times as many fragments may hybridize and β_{ij} should be multiplied by 2. This effect rarely occurs in practice, however, and we will ignore it here (but see also Section 3.1).

2.3.4 Melting Temperature

The fundamental relation (2.4) can be developed in another direction. Instead of fixing the reaction temperature, we can look for the temperature T_{M}, called the *melting temperature*, at which half of the potential duplexes have hybridized and the other half is still in the form of single strands when equilibrium is reached, such that $[\mathrm{D}]_{\mathrm{eq}}/[\mathrm{S}_1]_{\mathrm{eq}} = 1$.

Substituting the relationship $\Delta_{\mathrm{r}}G^\circ = \Delta_{\mathrm{r}}H^\circ - T\Delta_{\mathrm{r}}S^\circ$ into (2.4) and solving for T, we obtain

$$T = \frac{\Delta_{\mathrm{r}}H^\circ}{\Delta_{\mathrm{r}}S^\circ - R \cdot \ln([\mathrm{D}]_{\mathrm{eq}}/([\mathrm{S}_1]_{\mathrm{eq}}[\mathrm{S}_2]_{\mathrm{eq}}))}.$$

As $[\mathrm{D}]_{\mathrm{eq}} = [\mathrm{S}_1]_{\mathrm{eq}}$, the melting temperature is

$$T_{\mathrm{M}} = \frac{\Delta_{\mathrm{r}}H^\circ}{\Delta_{\mathrm{r}}S^\circ + R \cdot \ln([\mathrm{S}_2]_{\mathrm{eq}})}. \tag{2.6}$$

If C_{target} and C_{probe} denote the respective initial molar target and probe concentrations and probes are in excess, we have $[\mathrm{S}_2]_{\mathrm{eq}} = C_{\mathrm{probe}} - C_{\mathrm{target}}/2$, because half of the target fragments have hybridized. If probe and target concentrations are approximately equal and we set $C_{\mathrm{tot}} := C_{\mathrm{target}} + C_{\mathrm{probe}} \approx 2C_{\mathrm{target}}$, then $[\mathrm{S}_2]_{\mathrm{eq}} \approx C_{\mathrm{tot}}/4$. However, since we assume that $C_{\mathrm{probe}} \gg C_{\mathrm{target}}$, we can usually set $[\mathrm{S}_2]_{\mathrm{eq}} \approx C_{\mathrm{probe}}$ and use the molar probe concentration in (2.6).

Le Novère (2001) provides a software tool called MELTING and a web service for the computation of melting temperatures and enthalpy and entropy values for a variety of parameter sets.

Melting temperatures should not be a basis for probe selection. Although we can in principle compute T_{M}, we discourage using melting-temperature-based approaches for probe selection. This is in contrast to other work (e.g., Li and Stormo, 2001; Kaderali and Schliep, 2002), and also to our own early work (Rahmann, 2002). The reason is not that T_{M} might depend on the target concentration (as we have just shown, this effect is not so pronounced). Instead, we feel that for chip design, the melting temperature is relatively insignificant. Kaderali and Schliep (2002) suggested that probes should be selected such that they share similar T_{M} values, but from the present chapter we have learned that a more sensible goal is to aim for approximately constant affinity coefficients $A_{i,t(i)}$ for all probes i. If they are all equal and $A_{i,j} = 0$ for targets $j \neq t(i)$, knowledge of them is not required to decode probe signals; they can then be treated as a constant coefficient that is removed during global normalization. We have also seen that T_{M} has no direct relation to A_{ij}, but $\Delta_{\mathrm{r}}G^\circ$ is related to A_{ij} via β_{ij} and Equations (2.2) and (2.5).

2.4 Empirical Determination of Affinity Coefficients

We have seen that knowing affinity coefficients is useful for low-level data analysis, especially for inferring a gene expression level from signal measurements of various probes. We have discussed one key component of affinity coefficients, the probability β of probe-target hybridization in detail, and we have learned (a) that it is hard to predict accurately because we lack valid parameters for RNA or DNA hybridization to DNA probes attached to a chip surface, and (b) that the stability is only one parameter influencing an affinity coefficient. It is unclear whether a systematic modeling approach will eventually be able to predict the coefficients accurately; this has not been attempted so far.

At present, it appears simpler to predict the coefficients from microarray experiments, either controlled ones with known target concentrations (so-called *spike-in* experiments), or by exploiting the fact that relative affinity levels can be inferred by examining data from many published datasets.

Spike-in experiments. Recall the basic noise-free model (2.1) with $y = A \cdot x$, where y contains m observed probe signals, x contains $n \leq m$ expression values, and A is the $m \times n$ affinity matrix. If we set $x := e_j$, the j-th unit vector, then y becomes the j-th column of A. Doing this for $j = 1, \ldots, n$, we successively obtain the whole matrix A. On the experimental side, we would have to create a sample containing only transcript j at fixed unit concentration and let it hybridize to the array. This would have to be repeated for every transcript; and for accuracy several repetitions should be used. The disadvantage is that these experiments are expensive and the sample preparation is time-consuming, so it would be too costly to systematically use them to obtain all affinity coefficients.

If it is known that each probe i is highly specific to only one transcript $t(i)$, the knowledge that $A_{ij} \approx 0$ for $j \neq t(i)$ can be used, and the experiment can be made with an artificial sample that contains each transcript in unit concentration, i.e., that corresponds to $x = (1, 1, \ldots, 1)$. Then the non-zero affinity coefficients can be found in y: We have $y_i = \sum_j A_{i,j} = c + A_{i,t(i)}$ with a constant $c > 0$ modeling global unspecific hybridization. It is close to impossible, however, to create artificial samples that contain each transcript in the same known concentration.

Inferring affinity coefficients from existing datasets. Thinking more practically, every microarray experiment with the same chip type and under the same conditions provides some information about the relative affinity coefficients. We still assume that each probe is highly specific. Consider the index set $I_j = \{i : t(i) = j\}$ of those probes belonging to the same transcript j, and the corresponding observations y_{I_j}. Since they share the same expression level x_j, which can be estimated by a robust

averaging procedure as outlined in Section 2.2, the relative size of the $A_{ij} \approx y_i/x_j$ ($i \in I_j$) can be determined, except for a global scaling factor. In reality, not all probes are very specific, and unavoidable errors in the estimation of x_j would propagate into the estimated $A_{i,t(i)}$. This is especially severe for low-expressed transcripts: If $x_j \approx 0$, but $y_i \gg 0$ due to noise, A_{ij} is easily overestimated by several orders of magnitude.

Statistical approaches that infer affinity coefficients from published experimental data have been pioneered by Li and Wong (2001) and attracted more interest recently (Irizarry et al., 2003a,b). Zhang et al. (2003) have recently constructed a phenomenological NN model from such estimates by assuming $A_{ij} \approx \beta_{ij}$ in (2.2) and using (2.5) in reverse. Mei et al. (2003) from Affymetrix® are taking a similar approach. Wang et al. (2003) used the model of Li and Wong and the known structures of 21 alternatively spliced genes to infer the mixture of splice variants present in a sample by estimating the appropriate probe-variant affinity coefficients.

For the probe selection process, however, the empirical models are less useful: We need to ensure that $A_{ij} \approx 0$ for $j \neq t(i)$ before producing the chip and running experiments. When we later find that the data gives high-variance estimations for x_j and inconsistent estimates for $a_i = A_{i,t(i)}$ because we overlooked some $A_{ij} \gg 0$, it is too late. Our answer is to keep it simple and develop a sequence-based measure to predict probe specificity in Chapter 3.

A combined approach. To conclude this chapter, we propose a new approach for estimating arbitrary affinity coefficients. We do not assume that each probe has only one perfectly matching target; thus we may have that $A_{ij} \gg 0$ for several j in every row. Our approach combines inference from data and modeling from first principles.

Consider again the basic model (2.1). We extend it to $K > 1$ experiments performed with the same protocol so that the affinity coefficients remain constant. We observe m probe signals in every experiment. It is assumed that these intensities are globally normalized (any simple method will do for this purpose) so that the orders of magnitude of the intensities are comparable across experiments. We collect the observations in an $m \times K$ matrix $Y = (Y_{ik})$, with Y_{ik} being the observed signal at feature i in experiment k. The unknown expression levels are collected in an $n \times K$ matrix $X = (X_{jk})$, where X_{jk} is the expression level of transcript j in experiment k. It is assumed, of course, that $m \gg n$. The $m \times n$ affinity coefficients $A = (A_{ij})$ are also unknown. Except for measurement errors and noise, the equation

$$Y = A \cdot X$$

should now hold.

On the right side we face $mn+nK$ unknowns; on the left side we have mK (normalized) observations. The equation is highly non-linear because both A and X are unknowns.

We only have a chance of robustly fitting both A and X in the general case if $mK \gg mn + nK$, or equivalently, $n \ll mK/(m+K)$, or $K \gg mn/(m-n)$. Because of the non-linearity, a unique solution cannot be guaranteed, however, and different choices of A and X may lead to small residuals $\|Y - AX\|$. Here $\|\cdot\|$ denotes a generic norm-like one-dimensional measure of difference that we do not further specify. Any implementation of the method we sketch here would of course have to make a specific choice at this place.

The key idea is to use reasonable prior knowledge about the affinity coefficients to develop an initial guess A^0. The longest common factor approach developed in subsequent chapters of this thesis provides a reasonable starting point. The basic idea is that a long common substring (factor) of probe i and target j is an indicator for high $A_{ij} \gg 0$, whereas if no sufficiently long substring exists, one may assume that $A_{ij} \approx 0$. If we trust this initial guess, we can assume A^0 as known for the moment obtain a usual linear system $Y = A^0 \cdot X^0$ which we can "solve" for the initial expression level X^0 estimate by minimizing $\|Y - A^0 \cdot X^0\|$ under the side condition that $X^0 \geq 0$ (no negative expression levels).

We then alternate in re-estimating A and X for a fixed number of steps or until convergence is observed. Starting with step $s = 1$, we repeat:

- Find $A^s \geq 0$ such that $\|Y - A^s \cdot X^{s-1}\| \to \min$.
- Find $X^s \geq 0$ such that $\|Y - A^s \cdot X^s\| \to \min$.

If we allow fully general $A \geq 0$ and assume that there are, say 10 probes per target, so $m = 10n$, we need $K > 10n^2/(9n) = 1.11n$ experiments. If however we determine initially that some affinity coefficients A_{ij} are insignificant and then force them to remain zero in every step of the iteration, the number of unknowns can be considerably reduced. Assume that the number of non-zero affinity coefficients per probe is bounded by a constant $c > 1$. Then there are mK observations for only $mc + nK$ unknowns, and we only need $K > mc/(m-n)$ experiments. If for instance also $m = cn$, then the condition becomes $K > c/(1 - 1/c)$, so $2c$ to $3c$ experiments should always provide a reasonable data basis.

The approach we described can be used in an incremental way: As more data becomes available for a chip type, more information is gathered about A, and even past analyses can be gradually improved.

It remains yet to be seen whether the idea, as it is sketched here, can be efficiently implemented and produces the expected results. It should be noted that for large scale projects $m = 100000$, $n = 10000$, $c = 10$ on average could be possible parameters. The resulting minimization problems are large, and specialized methods may be required to deal with large sparse matrices. It would be very interesting to apply the method to situations where non-unique probes must be used for the analysis; such as in the inference of mixtures of alternatively spliced transcripts (e.g., Wang et al., 2003).

Chapter 3

The Longest Common Factor Approach

Given the difficulties in affinity coefficient estimation, but also for efficiency reasons, we propose a sequence-based surrogate measure. Decisions about probe selection can then be made from this surrogate instead of from the unknown affinity coefficients. We introduce and motivate the *longest common factor* (LCF) measure in Section 3.1 and derive probe unspecificity measures from it in Section 3.2. The LCF approach is compared to other approaches in Section 3.3.

3.1 Definition and Motivation

We propose to exploit the correlation between the affinity coefficient $a(p, t)$ of probe p and target t and the length of the *longest common factor* of p and t. In this context, *factor* is a synonym for *substring*. We prefer the word factor to better distinguish it from a subsequence: A factor is contiguous, while a subsequence need not be. For example, `AAT` is a subsequence, but not a substring of `ACAGT`, and `CAG` is both a substring and a subsequence of `ACAGT`.

Basics. We write $s \lhd t$ if s is a factor of t; the cases that s is empty or that $s = t$ are allowed. The length of a string s is denoted by $|s|$.

Definition 3.1 (Longest common factor). A *common factor* of two strings p and t is a string s with both $s \lhd p$ and $s \lhd t$. A common factor is a *longest common factor* if no longer common factor exists. We write

$$\mathrm{lcf}(p, t) := \max\{|s| : s \lhd p \text{ and } s \lhd t\}$$

for the *length* of the longest common factor of p and t.

A probe physically hybridizes to a cRNA or cDNA target sequence that contains the complement of p, not p itself. However, *we adopt the convention to describe target sequences by the target's mRNA sequence in the DNA alphabet, such that a probe for t is in fact a substring of t.*

Motivation. The choice of the longest common factor as a stability surrogate is motivated by the following physical evidence.

- Every sufficiently stably hybridized probe-target pair needs a stable core for the hybridization to start. Differences in the probe-target alignment correspond to destabilizing irregularities in the normally parallelly stacked planes of basepairs.
- A non-perfect hybridization can be interpreted as consisting of several stable perfect sub-matches, interrupted by destabilizing mismatches or indels. Even if the whole imperfect duplex is relatively stable under hybridization conditions, it is more easily de-stabilized under stringent washing conditions. The two-state model (Section 2.3.3) is much less applicable to non-perfect hybridizations; many possible intermediate states exist between the single strands and the duplex.
- In general, duplexes with the same number of (non-consecutive) matches can have very different stabilities (see Figure 3.1). The stacking interactions between consecutive matching basepairs are an important factor. Therefore it seems more reasonable to consider the length of consecutive matches instead of the number of total matches.
- On average, but not in all cases, the most stable perfect sub-duplex is the longest perfect sub-duplex. A common base corresponds to 2.5 hydrogen bonds on average when a uniform base composition is assumed.
- Pozhitkov and Tautz (2002) report that, if a mismatch is introduced into a perfect probe-target duplex, the resulting stability loss depends on the position of the mismatch; it is most pronounced when the mismatch occurs at the middle position of the probe and drops when the mismatch position moves closer to either end of the probe. Zhang et al. (2003) also suggest a position-dependent model. The longest common factor automatically takes such a dependence into account: If a mismatch occurs in the middle, both sub-matches have only about half the length of the probe. Mismatches near the ends leave one longer perfect sub-match.

We therefore use the working hypothesis that the longest perfect match, i.e., longest common substring, of probe and target provides the dominating contribution to the hybridization stability $\Delta_{\mathrm{r}}G^{\circ}$. This is also a practical assumption. The nearest neighbor model and the two-state assumption, as discussed in Section 2.3.3, are essentially applicable only to perfect matches. We will also see in Chapter 4 that the LCF and further LCF-based quantities can be computed efficiently.

```
ATCTCCACCCGTTGTTCAT      ATCTCCACCCGTTGTTCAT      ATCTCCACCCGTTGTTCAT
||| || ||| |||| |||      ||||||||||||||||      |||||||||| ||||||||
ATCACCTCCCTTTGTCCAT      ATCTCCACCCGTTGTCAGG      ATCTCCACCCTTTGTTCAT
```

(a) lcf = 4, lcf^1 = 8, matches = 15, len = 19

(b) lcf = 15, lcf^1 = 16, matches = 15, len = 19

(c) lcf = 10, lcf^1 = 19, matches = 18, len = 19

Figure 3.1: Lengths of longest common factors (lcf values), longest common factors allowing one mismatch (lcf^1), and number of matches between imperfect probe-target alignments of length 19.

There are some natural disadvantages of the basic LCF measure.

- The longest match need not be the most stable one: The base composition of the probe-target match is ignored and replaced by an average. We show in Section 4.6 how to efficiently re-incorporate the sequence information by exploiting the concept of *jumps in matching statistics*, introduced in Section 4.4.
- The LCF measure must be expected to systematically underestimate stability because it considers only one match (the longest perfect one), even when there are several (long perfect and imperfect) ones for the same probe-target combination. This effect also arises because the target is fragmented; so one target molecules can in fact hybridize several times to one spot if each fragment contains a long match. Because fragmentation is random, this effect is hard to quantify systematically.

The second point suggests that even if accuracy is of prime importance, the LCF can still be used as a filter to quickly reduce the number of probe candidates that must be considered: If already $\mathrm{lcf}(p,t)$ is high, but probe candidate p is not intended for target t, the stability need not be evaluated and p can be immediately discarded.

LCF with one mismatch. To heuristically counter-act systematic underestimations, we generalize the longest common factor to contain mismatches. In practice, we will always restrict ourselves to the case of at most one mismatch.

Definition 3.2 (Factors with at most f mismatches). We write $s \lhd_f t$ if s is a factor of t with at most f mismatches, i.e., if there exists a position i in t and an index set F with $|F| \leq f$, such that $s_j = t_{i+j}$ for all $j \in \{0, 1, \ldots, |t|-1\} \setminus F$.

Definition 3.3 (Longest common factor with at most f mismatches). A *common factor with at most f mismatches* of two strings p and t is a string s with both $s \lhd_f p$ and $s \lhd_f t$. It is a *longest common factor with at most f mismatches* if no longer common factor with at most f mismatches exists. We write

$$\mathrm{lcf}^f(p,t) := \max\{|s| : s \lhd_f p \text{ and } s \lhd_f t\}$$

for the *length* of the longest common factor with at most f mismatches of p and t.

Figure 3.1 provides some examples of different possible $(\mathrm{lcf}, \mathrm{lcf}^1)$-combinations. Including both $\mathrm{lcf} \equiv \mathrm{lcf}^0$ and lcf^1 in our considerations has two beneficial effects. First, it provides a chance to assess two different potential binding sites in the same target if the positions of the LCF and the LCF with one mismatch are different (case not shown). Second, it provides a foot in the door towards stability assessment of non-perfect hybridizations.

How should we compare, for instance, two probe candidates p_1, p_2 of length 19 that have $(\mathrm{lcf}(p_1,t), \mathrm{lcf}^1(p_1,t)) = (10, 19)$ as in Fig. 3.1c (similar to MM probes on Affymetrix GeneChips®) and $(\mathrm{lcf}(p_2,t), \mathrm{lcf}^1(p_2,t)) = (15, 16)$ as in Fig. 3.1b, respectively? In other words, how can we combine the two-dimensional quantity $(\mathrm{lcf}, \mathrm{lcf}^1)$ into a one-dimensional quantity lcf^* in order to exploit the linear order on the real numbers? We cannot offer a unique answer to this question, but two approaches suggest themselves.

1. We set $\mathrm{lcf}^*(p,t) := \max\{\mathrm{lcf}(p,t),\ \mathrm{lcf}^1(p,t) - y\}$, where $y > 0$ is an average penalty term for the internal mismatch. If, for instance, we assume that three additional matching bases are required to "outweigh" the mismatch ($\mathrm{lcf}^1 \geq \mathrm{lcf} + 4$), we could set $y := 3.5$. In the above example, this results in $\mathrm{lcf}^*(p_1,t) = 15.5$ and $\mathrm{lcf}^*(p_2,t) = 15$.

2. We use a (weighted) average $\mathrm{lcf}^*(p,t) := \alpha \cdot \mathrm{lcf}(p,t) + (1-\alpha) \cdot \mathrm{lcf}^1(p,t)$ with a weighting factor $\alpha \in [0,1]$. If we place twice more importance of lcf than on lcf^1, i.e., set $\alpha := 2/3$, we obtain $\mathrm{lcf}^*(p_1,t) = 13$ and $\mathrm{lcf}^*(p_2,t) = 15.3$.

The methods disagree when ranking the two examples, but so do experts. Both approaches share the property that $\mathrm{lcf}^*(p,t) \geq \mathrm{lcf}(p,t)$ and can potentially correct for the systematic stability underestimation of lcf.

The development in the following sections is presented for lcf, but can be immediately applied to lcf^* without changes. Thus lcf^* should be seen as an attempt to bias-correct the basic lcf measure.

3.2 LCF-Based Unspecificity Measures

To identify target-specific probes, the obvious procedure now is to compute $\mathrm{lcf}(p,t)$ for all probe candidates p and all targets $t \in T$, where T denotes an arbitrary enumeration of the targets or transcripts. For a given probe candidate p, one obtains the vector of longest common factors with each target, the *LCF vector* $\mathrm{LCF}(p \,|\, T) := (\mathrm{lcf}(p,t))_{t \in T}$.

LCF Statistics. To assess the (un-)specificity of p, it is unnecessary to know *which* transcripts have high or low lcf values with p. The information can therefore be compressed into a vector that merely *counts* the number of transcripts that share a certain lcf value with p. Additionally, the exact lcf value is irrelevant if it is small, e.g., less than half of the probe length, say. Therefore we fix an integer $\Delta > 0$ and consider lcf values below or equal to $|p| - \Delta$ as equivalent to zero.

Definition 3.4 (LCF statistics). The *LCF statistics* of width Δ for probe p against transcriptome T are defined as the vector $\mathrm{LCFS}(p \,|\, T; \Delta)$ with

$$\mathrm{LCFS}(p \,|\, T; \Delta)_\delta := \#\{t \in T : \mathrm{lcf}(p, t) = |p| - \delta\} \qquad (\delta = 0, \ldots, \Delta - 1).$$

When p, T or Δ are obvious from the context, we simply write LCFS_δ or $\mathrm{LCFS}(p \,|\, T)_\delta$ for the δ-component of the LCF statistics.

If probe p^* is to be specific for target t^*, we know that we must have $p^* \lhd t^*$, or equivalently $\mathrm{lcf}(p^*, t^*) = |p^*|$, and $\mathrm{lcf}(p^*, t)$ must be small for all $t \neq t^*$. Thus we should have $\mathrm{LCFS}_0 = 1$ because of t^*. The next few components should vanish, and only for δ approaching Δ, we would accept to see some $\mathrm{LCFS}_\delta > 0$.

Conceptually the parameter Δ has no special meaning as long as it is not too small. In practice, however, its choice is crucial for efficient methods that compute $\mathrm{LCFS}(p|T; \Delta)$ for many probe candidates. If it can be determined early that $\mathrm{lcf}(p, t) \leq |p| - \Delta$, then the exact value of $\mathrm{lcf}(p, t)$ need not be computed. This observation can save tremendous amounts of computational resources. Therefore it seems desirable to choose Δ as small as possible. Care must be taken, however, not to ignore significantly long LCFs which could indicate a potential for cross-hybridization. Our experiences suggest that it is desirable to choose $\Delta > |p|/2$ and that it appears reasonable to set $\Delta := 16$ for 25-mer probes, such that lcf values below or equal to 9 need not be computed. The computational method is explained in Chapter 4.

Probe Unspecificity. The LCF statistics provide a multidimensional measure of unspecificity for each probe. It is desirable to combine the information into a one-dimensional quantity U by which all probe candidates for a transcript can then be ranked. We now derive such a measure using the signal intensity model from Chapter 2.

Consider a hypothetical spike-in experiment where the expression level x_j of each transcript j is constant; so $x = (1, 1, \ldots, 1)$ except for the common constant. The cRNA or cDNA target fragments are uniformly distributed over the chip, and we assume that a unit amount of each type is available for hybridization with each probe type. Probe candidate p_i has a certain intended target $t(i)$, or when non-unique probes are allowed (Chapter 6), a set $T(i) \subset T$ of intended targets within the transcriptome T. The undesired part of the signal observed at p_i is then given by $\sum_{j \in T \setminus T(i)} A_{i,j} \cdot 1$,

where $A_{i,j} = a(p_i, t_j; \theta)$ is the affinity coefficient under the hybridization conditions described by θ.

Ignoring that other factors than the hybridization probability $\beta_{i,j}$ contribute to $A_{i,j}$ in (2.2), we thus define preliminarily the unspecificity of p_i in T by

$$U'(p_i \,|\, T) := \sum_{j \in T \setminus T(i)} \beta_{i,j}. \tag{3.1}$$

Recall that $\beta_{i,j}$ is a function of the hybridization "stability", which we have chosen to approximate by the Gibbs energy of the longest perfect match (common factor) of p_i and t_j. Given appropriate nearest neighbor parameters, we can find the average energy $\gamma := \mathbb{E}[\Delta_{\mathrm{r}} G^\circ] < 0$ per common dinucleotide under a given base composition. Then the Gibbs energy of the hybridization of p_i with an unintended target t_j differs from the energy of the hybridization of p_i with its intended targets by an average amount of $\delta_{i,j} \cdot \gamma$, where $\delta_{i,j} = |p_i| - \mathrm{lcf}(p_i, t_j)$.

From (2.5), we see that the average hybridization strength β decreases with increasing δ:

$$\beta(\delta) = \frac{1}{1 + [\mathrm{S}_2]_{\mathrm{eq}}^{-1} \cdot \exp((G - \delta\gamma)/(R\tau))}. \tag{3.2}$$

Here $G \ll 0$ denotes the Gibbs energy of the perfect duplex, $[\mathrm{S}_2]_{\mathrm{eq}}$ is a probe concentration term as in (2.5), and τ is the temperature in Kelvin (we use τ here instead of T to avoid confusion with the transcriptome T). As in Section 2.3.3, R denotes the gas constant.

We can simplify the presentation of the above equation by setting $b := -\gamma/(R\tau) > 0$ and $\zeta := -G/(R\tau) + \ln[\mathrm{S}_2]_{\mathrm{eq}}$, so that

$$\beta(\delta) = \frac{1}{1 + \exp(b \cdot \delta - \zeta)}. \tag{3.3}$$

Since $b > 0$, $\beta(\delta)$ decreases with increasing δ. The parameters b and ζ depend on the reaction temperature τ, the perfect duplex stability G, and the nearest neighbor model parameters (including salt concentration).

We will usually try to ensure that all selected probes share approximately the same stability G, i.e., we prescribe a certain narrow stability range for probe candidates. This is also true if we consider probes of different lengths: A candidate can be extended until it enters the prescribed stability range. Therefore the parameters b and ζ are constants within one microarray experiment.

The interpretation of ζ remains somewhat mysterious because it contains the probe concentration term $[\mathrm{S}_2]_{\mathrm{eq}}$. which should typically be chosen much lower than the true concentration because the probes are attached to the chip (cf. Section 2.3.3). We can

take a more practical approach, however. In reality, under optimized conditions that balance strong hybridization of perfect duplexes against high specificity with respect to non-perfect duplexes, not all target fragments in the vicinity of a feature location will hybridize. If the probes are well chosen and have the same affinities, we may assume that for $\delta = 0$, always the same constant fraction $z < 1$ hybridizes. Solving $1/(1 + e^{-\zeta}) = z$ results in $\zeta = \ln(z/(1 - z))$. Especially, for $z = 0.5$, the constant ζ vanishes. The following table shows some useful values.

z	ζ
0.5	0
0.9	$\ln(9) = 2.1972$
0.99	$\ln(99) = 4.5941$
0.999	$\ln(999) = 6.9068$
$1 - 10^{-k}$	$\ln(10^k - 1) \approx k \cdot 2.3 \quad (k \gg 0)$

Suppose that ζ is small in (3.3) and δ is sufficiently large. Then the influence of the constant 1 in the denominator of $\beta(\delta)$ becomes negligible and we can approximate $\beta(\delta)$ by

$$u(\delta) := e^{-b\cdot\delta+\zeta} \qquad (\delta \text{ large}). \tag{3.4}$$

If we use this approximation for smaller values of δ, we can only overestimate the cross-hybridization probability β, which is not a bad thing. Therefore the simplified quantity $u(\delta)$ version can be used wherever this appears more convenient. Note that, apart from ζ, both β and u contain only one parameter $b := -\gamma/(R\tau) > 0$ that depends on the average energy $\mathbb{E}[\Delta_r G^\circ] < 0$ per common dinucleotide and the reaction temperature τ.

Example 3.5. For the purpose of illustration, we assume that $z = 0.9$, so $\zeta = 2.1972$ from the table above. For 25-mers at 45°C = 318.15 K with a Na^+-concentration of 0.075 M (i.e., $\ln([Na^+]) = -2.6$), and using the parameters of SantaLucia (1998) from Table 2.1, we find $b = 1.4741$. Figure 3.2 shows β on a log-scale as a function of δ.

We now come back to the definition of the unspecificity $U'(p_i \,|\, T)$ in (3.1). Replacing $\beta_{i,j}$ by its approximation $u(\delta_{i,j})$ from (3.4), where $\delta_{i,j} = |p_i| - \mathrm{lcf}(p_i, t_j)$, we obtain a computationally convenient definition of *probe unspecificity* U.

$$\begin{aligned} U(p_i \,|\, T) :&= \sum_{j\in T\setminus T(i)} u(\delta_{i,j}) \;=\; \sum_{j\in T\setminus T(i)} e^{-b\cdot\delta_{i,j}+\zeta} \\ &= \sum_{\delta=0}^{\Delta-1} e^{-b\cdot\delta+\zeta} \cdot \mathrm{LCFS}'(p_i \,|\, T; \Delta)_\delta \end{aligned} \tag{3.5}$$

In (3.5), we have aggregated all targets j with the same value of δ; these are counted by the δ-th component of the longest common factor statistics $\mathrm{LCFS}(p_i \,|\, T; \Delta)$. The

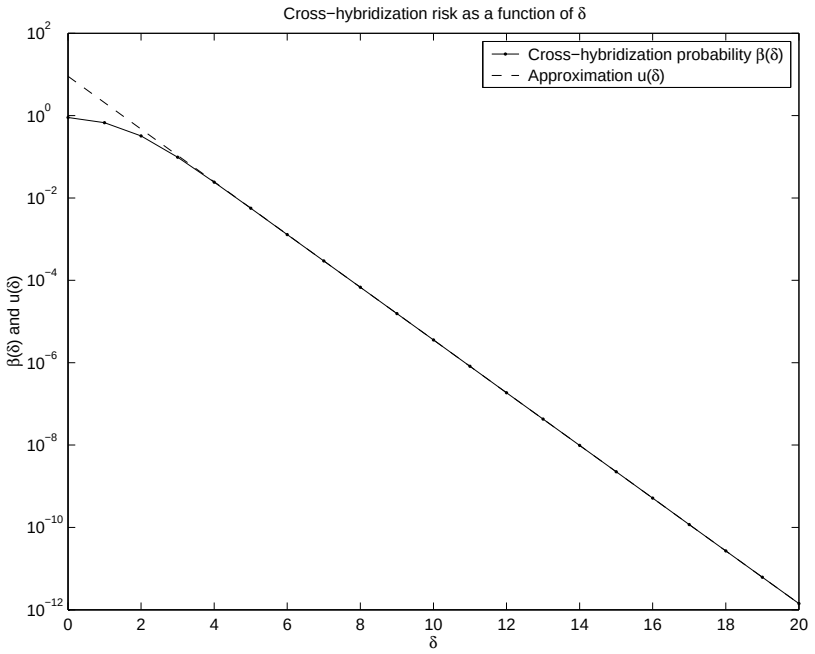


Figure 3.2: Cross-hybridization probability β log-scale as a function of the difference δ between probe length and longest common factor of probe and unintended target. The parameters are given in Example 3.5.

prime-notation LCFS$'$ denotes that for $\delta = 0$, we subtract the number $|T(i)|$ of intended targets for p_i, usually a single one. Technically, $U(p_i \,|\, T)$ depends on the value of Δ, but if Δ is large enough, the terms with $\delta \approx \Delta$ only contribute an exponentially small additional amount to U; so we shall drop the dependence on Δ in the notation.

The threshold for U below which we can consider a probe to be sufficiently specific may depend on the requirements imposed by the intended application. Generally, if $U(p_i \,|\, T) < 10^{-4}$, we can assume that we have found a sufficiently specific probe. Note that this means that the individual $u(\delta_{i,j})$-terms in $U(p_i \,|\, T)$ must all be considerably smaller than 10^{-4}. Therefore it does not matter whether the decision is based on the more exact value β from (3.3) or on its approximation $u(\delta)$; so we have chosen the simpler functional form $u(\delta)$.

Summary. The final definition of probe unspecificity $U(p \,|\, T)$ provides an LCF-based way to assess the cross-hybridization risk of a probe candidate p. Its form is intuitive: With decreasing LCF length, the risk drops exponentially from a certain length difference on. The measure also considers *all* unintended targets; not only the most stable one. This makes the assessment more robust. Only the LCF statistics are required for its computation, not the individual values $\mathrm{lcf}(p, t)$ for all targets t.

While the relation between hybridization stability and lcf or lcf* is not simple or even one-to-one, the longest common factor approach can provide a simple and robust sequence-based approximation; no knowledge of energy parameters is necessary as soon as the parameter b has been agreed upon and ζ has been fixed. For 25-mer design, the parameters from Example 3.5, $b = 1.47$ and $\zeta = 2.2$, can be used and agree reasonably well with experience: For any probe p, a non-intended target t with $\mathrm{lcf}(p, t) \geq 17$, that is $\delta \leq 8$, leads to unspecificity $\gtrsim 10^{-4}$ and is thus borderline.

3.3 Relation to Other Approaches

We originally introduced the LCF approach at the CSB'02 conference (Rahmann, 2002), but the notion of unspecificity was only defined heuristically. In the previous section, we presented a thermodynamic motivation of the unspecificity's exponential decay rate with increasing LCF difference δ. In particular, the rate b in (3.4) has a physical interpretation.

Before we move on to show how the LCF statistics can be efficiently obtained in the next chapter, we briefly compare the LCF approach against other methods for oligonucleotide selection that have been proposed in the current scientific literature. We point out that the LCF approach is mostly designed for short oligonucleotides. For polynucleotides (50–70-mers), it is less reasonable that only the longest contiguous match provides the dominating contribution to hybridization stability. As we shall see, software based on the LCF approach can outperform the other methods presented here by several orders of magnitude in term of speed.

- Primer design programs such as PRIDE (Haas et al., 1998) are based on similar ideas as oligonucleotide probe design programs (counting mismatches, etc.), but they only need to find one primer or primer-pair within a well-defined sequence region (specificity would still be checked against the whole transcriptome), decreasing the need for computational efficiency. Also, different side conditions must be taken into account. Generally, primer design methodologies cannot be expected to scale up to whole genome short oligonucleotide probe design without modification.

- "ProbeSelect" by Li and Stormo (2001) use a specificity prediction based on sequence landscapes (Levy et al., 1998) and edit distance (Myers, 1999) between probe and target. The assumption is that the probe-target affinity can potentially be high when an optimal local unit-cost alignment (penalizing each mismatch, insertion, and deletion by -1) results in more than 4 differences; such probe candidates are then discarded. Stabilities and melting temperatures are computed for the remaining candidates using MFOLD (Zuker et al., 1999; Zuker, 2003). Overall, the approach is relatively slow; Verena Beier reports that the

software designs oligos for predicted *Neurospora crassa* genes (15.5 Mbp) within one week (personal communication). The method is applicable to both short and long oligonucleotides, but does not attempt to model affinity coefficients.

- "Probesel" by Kaderali and Schliep (2002) is similar in spirit. It directly attempts to find an alignment with the maximal melting temperature using a heuristic dynamic programming algorithm. The focus on melting temperature can be problematic, as discussed in Section 2.3.4. This approach is also relatively slow and appears to have been tested only on small datasets. It is reported that the selection of unique probes for 58 HIV-1 subtypes (total length 600 kbp) takes about nine hours.

- Pozhitkov and Tautz (2002) focus on species identification with oligonucleotide arrays and therefore restrict their probes and targets to small fractions of the genome, such as the ribosomal RNA genes. They suggest an ad-hoc stability function that depends on the position within the probe. For each probe, only non-intended targets that share an alignment with the intended target with more than 90% sequence similarity are explicitly checked for cross-hybridization. Finding one probe within a database of 16000 small subunit sequences took 1.5 hours.

- Rouillard et al. (2002) focus on the design of long oligonucleotides (50-mers). The increased length allows them to find and exclude possible cross-hybridization candidates with BLAST. They emphasize the requirement that the oligos do not form a secondary structure. This issue is not considered by the other programs. Oligos for the 6343 yeast (*Saccharomyces cerevisiae*) ORFs (a total of 9.5 Mbp) are selected within a day.

- Lipson et al. (2002) measure the specificity of 30-mer probes by the minimum Hamming distance between the probe and all targets. They also provide a statistical model to predict the minimal reasonable probe length for unique probes in a given transcriptome.

- Sun and Lee (2003) design fixed-length polynucleotides (50-mers) and use the local Hamming distance between probe and target as a specificity criterion. Their paper describes an efficient filtering algorithm for quickly deciding whether the Hamming distance is larger than a given threshold.

- Wang and Seed (2003) reject a 70-mer probe candidate when any non-intended target shares at least a contiguous 15-mer with the probe. A hash table is used to quickly locate all 10-mer occurrences; common 15-mers are then found as two-overlapping 10-mers with the correct offsets. Their work comes most closely to the LCF approach, but lacks the robustness provided by LCF statistics. We also believe that the LCF approach is more suited for short oligonucleotides.

The LCF approach offers several features that other approaches do not offer:

- It allows ranking probe candidates by their (un-)specificity based on cross-hybridization probabilities obtained from a physical model, distilled into the constant b in Equation (3.5).
- Whereas other methods base their decisions only on the most stable cross-hybridization site the whole transcriptome, the longest common factor statistics $\mathrm{LCFS}(p \mid T; \Delta)$ provides a more robust measure.
- It avoids the imprecise computation of melting temperatures and works with the more plausible affinity model 2.2.
- Single mismatches in otherwise perfect probe-target duplexes are automatically position-dependent with the LCF approach, which is consistent with experimental observations; methods that count the total number of matches (i.e., consider the Hamming distance) do not have this property.
- Last but not least, the LCF approach is the only one that is fast enough to allow whole genome projects (see also Chapter 7).

Chapter 4

Efficient Computation of Longest Common Factor Statistics

This chapter presents efficient algorithms for the computation of longest common factor statistics, the key component for evaluating the unspecificity of probe candidates, as defined in (3.5). We begin with some preliminaries in Section 4.1, and introduce *enhanced suffix arrays*, the data structure on which our method is based, in Section 4.2. LCF statistics are computed in several steps; in the first step, *matching statistics* of all targets against all transcript collections are found. We present the first efficient suffix-array based algorithm for this all-against-all procedure (Section 4.3). To reduce the considerable output size, we exploit the structure of matching statistics and introduce the concept of *jumps* (Section 4.4) and provide an algorithm to obtain the desired LCF statistics from the jump lists (Section 4.5). We then describe two variations of the algorithm that trade precision for speed in both ways (Section 4.6). The chapter concludes with computational results on probe selection for all open reading frames of *Saccharomyces cerevisiae* (baker's yeast).

4.1 Preliminaries

4.1.1 Targets, Transcripts, and Collections

So far, we have used the terms *target* and *transcript* quite synonymously; they were sequences for which probes are to be designed. At this point, however, it is convenient to make a finer distinction between these terms, and to further introduce *transcript collections*.

Target sequences or **targets** are those sequences which furnish the probe candidates. In other words, every probe candidate is a substring of at least one target. We generally denote a target sequence by s.

Transcript sequences or **transcripts** are the sequences against which probe specificity must be checked. Transcripts are denoted by t.

Transcript collections are simply sets of transcripts. We introduce them to be able to model a larger variety of probe design applications in practice within the same basic framework (examples follow). Thus the transcriptome T does not consist of individual transcripts, but of transcript collections, which in turn consist of one or more transcripts. The number of collections is denoted by C; the number of transcripts in collection c is denoted by N_c.

$$T = (T_1, \ldots, T_C), \qquad T_c = \{t_{c,1}, \ldots, t_{c,N_c}\}$$

The notation $t \in T$ means that there exists an index $c \in \{1, \ldots, C\}$ such that $t \in T_c$.

A target s typically has the property that every substring p of s is also a substring of some transcript $t \in T$: $(p \lhd s) \implies (\exists t \in T : p \lhd t)$.

To assess the unspecificity of a probe candidate p, we do not compute the longest common factor length $\mathrm{lcf}(p, t)$ for every transcript $t \in T$, but only for every collection,

$$\mathrm{lcf}(p, T_c) := \max_{t \in T_c} \mathrm{lcf}(p, t).$$

This additional level of abstraction can be ignored in standard applications where each transcript forms its own collection. Even if this is not the case, one can represent T_c as a single string by concatenating the individual transcripts with a special separator character `$`.

$$T_c = t_{c,1}\$t_{c,2}\$ \ldots \$t_{c,N_c}$$

Every target s is associated to a single collection $c = c(s)$. Typically, for every probe candidate $p \lhd s$, we expect that $\mathrm{lcf}(p, T_{c(s)}) = |p|$, i.e., that p occurs in $T_{c(s)}$. This would increase $\mathrm{LCFS}(p \,|\, T)_0$ by 1, but of course, this contribution is ignored for the unspecificity value $U(p \,|\, T)$, as explained subsequently to Equation (3.5).

The usefulness of the above distinctions becomes evident from the following two examples.

1. Frequently, probes are only required for a small subset of all transcripts. Additionally, the probes should be taken from approximately the last 1000 bases of the 3'-end of each transcript (see Section 1.3.2). The targets then consist only of the 1000-nt-suffixes of those transcripts for which probes are required. Of course, specificity must be checked against *all* transcripts. For safety reasons, the full-length transcripts are taken, and each transcript is placed in its own collection.

2. In large and highly homologous gene families, such as the human heat shock protein family, it may be impossible to find specific probes for each transcript. Instead, it is desired to find *family-specific probes*, i.e., probes that occur in each family member but not in other transcripts. Assume that the family members are organized in a collection T_c. It is a non-trivial task to construct the target sequence s. It should consist of several separated strings with the property that every substring (without a separation character) in s is a substring of every $t \in T_c$, and not too far away from the 3'-end.
3. Organizing large gene families into collections also has the advantage that fewer lcf values need to be computed.

4.1.2 Key Observations

Potentially, *every* substring p of a given target s is a probe candidate for s. In practice, length and stability ranges will be prescribed. More details are given in Chapter 5. Here the key is to recognize that probe candidates can start and end at every position in s and be of varying length. Generally, the set of probe candidates to consider can be larger than $|s|$, but assumed to be bounded by a constant times $|s|$.

Assume that there are n targets of constant length, say, 1000 nt, each giving rise to 2000 probe candidates of average length 25. Assume further that there are C collections, each represented by a string of 10000 nt. Thus $2000 \cdot n \cdot C$ longest common factor values must be computed. To compute $\mathrm{lcf}(p, T_c)$, a naive algorithm might attempt to find the longest match at every combination of starting position in p and starting position in T_c. These are $25 \cdot 10000$ combinations, and we may assume that the average number of comparisons at each combination is 2. Thus $\mathrm{lcf}(p, T_c)$ is determined with half a million matching operations. Altogether, we thus require $n \cdot C \cdot 10^9$ matching operations. Assuming $n = 10000$ targets and $C = 10000$ collections, this leads to 10^{17} (one hundred quadrillion) matching operations. Probe design would be only possible on petaflop supercomputers.

Of course, a lot of work is superfluous in the naive algorithm. The following observations are helpful.

- Probe candidates overlap with each other, and the lcf-values of two probes that share a long stretch of common sequence are highly correlated.
- As we already noted, very short matches are irrelevant, and their exact length need not be determined.

Index data structures for strings are of great utility here. The first observation suggests to use suffix trees, and the second observation suggests to use a lookup or hash table for words of a certain minimum length. Suffix trees have an excellent reputation for their usefulness in such problems, but not such a good one for their high memory

requirements (but see Kurtz, 1999, for memory reduction techniques). Fortunately, *enhanced suffix arrays* (Abouelhoda et al., 2002a) provide all of the desired features at reasonable memory requirements, and are even well suited for use with secondary memory. We use enhanced suffix arrays, which we introduce in the next section, to compute *matching statistics* (Sections 4.3 and 4.4), an intermediate quantity from which we obtain the lcf-values for LCF statistics (Section 4.5).

4.2 Enhanced Suffix Arrays

4.2.1 The Basic Suffix Array

The *basic* suffix array as introduced by Manber and Myers (1993) is an index of the alphabetically ordered suffixes of a string. The concept of *enhanced* suffix arrays is formally introduced in a paper by Abouelhoda et al. (2002a), but the use of additional tables was already proposed in Manber and Myers' paper to accelerate lookup operations in the suffix array. The concept is quite versatile. Additional tables may be added as needed according to the particular application.

Let $\Sigma := \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ be the DNA alphabet. We use the character $\mathtt{\$} \notin \Sigma$ as an end-marker and as a sequence separator when two or more sequences are concatenated. We use $\mathtt{X}$ as a wildcard character, that is, to represent a sequence character not in Σ, such as an undetermined DNA base like $\mathtt{N}$. We use the character order $\mathtt{A} < \mathtt{C} < \mathtt{G} < \mathtt{T} < \mathtt{X} < \mathtt{\$}$, with the additional property that $\mathtt{X}$ and $\mathtt{\$}$ do not match themselves ($\mathtt{\$} \neq \mathtt{\$}$, $\mathtt{X} \neq \mathtt{X}$). A string of length q is called a *q-gram*. We write Σ^q for the set of all q-grams and Σ^+ for the set of all non-empty strings consisting of characters from Σ. A *prefix* of a string is a factor that starts at the beginning of the string, and a *suffix* is a factor that ends at the end of the string. We use subscripts to refer to the individual characters of a string s: $s = (s_0, \ldots, s_{|s|-1})$. We also write $s_{p_1..p_2}$ for the factor $(s_{p_1}, \ldots, s_{p_2})$, and $s_{(p)} := s_{p..|s|-1}$ for the suffix starting at position p. (In this section, s is an arbitrary string, not necessarily a target sequence, and p denotes a position in s, not a probe. The index i generally serves as an index into the suffix array.)

Definition 4.1 (Suffix array). Let $s' = (s_0, \ldots, s_{n-1}) \in (\Sigma \cup \{\mathtt{X}\})^n$ be a string of length n. We set $s := s'\mathtt{\$}$. The suffix array of s is an array $\mathtt{pos} = (\mathtt{pos}[0], \ldots, \mathtt{pos}[n])$ of length $n+1$ that contains the starting positions of the suffixes of s in alphabetical order, $s_{(\mathtt{pos}[0])} < s_{(\mathtt{pos}[1])} < \cdots < s_{(\mathtt{pos}[n])}$ (see Figure 4.1).

The suffix array can also be defined for a set of sequences: One concatenates the sequences, separating them by the separator $\mathtt{\$}$. The fact that two separators or wildcards never compare as equal can lead to ambiguity in the definition of the suffix array:

Pos. p	s_p
0	C
1	A
2	C
3	A
4	C
5	X
6	A
7	C
8	C
9	$

$\Longrightarrow$

Rank i	pos$[i]$	$s_{(\texttt{pos}[i])}$
0	1	ACACXACC$
1	6	ACC$
2	3	ACXACC$
3	0	CACACXACC$
4	2	CACXACC$
5	7	CC$
6	4	CXACC$
7	8	C$
8	5	XACC$
9	9	$

Figure 4.1: Basic suffix array pos of $s =$ CACACXACC$. On the left side, s is shown. On the right side, the suffixes have been alphabetically sorted, assuming A < C < X < $. The suffixes are only shown to make the sorted order apparent; only pos forms the suffix array.

When comparing ACGT$A and ACGT$T, the first four characters match, but the result of the comparison may be either ACGT$A<ACGT$T or ACGT$T<ACGT$A. Note that the resulting inequality is of no relevance, and both orderings yield a correct suffix array for our purposes.

4.2.2 Enhancing the Basic Array

The basic suffix array is augmented with additional information. Recalling that very short common factors are irrelevant to the problem of probe selection, we decide on a threshold $q > 0$ such that common factors shorter than length q can be considered as equivalent to zero. Because of the alphabetical order, suffixes that share a common prefix are found adjacent to each other in the suffix array. Especially, suffixes that start with the same q-gram form a contiguous region of the suffix array, a so-called *q-bucket*. The formal definitions follow.

Definition 4.2 (Longest common prefix). The *longest common prefix* of two strings s and t is the longest string that is both a prefix of s and t. We write $\mathrm{lcp}(s,t) := \max_{0 \le p < |s|,|t|}\{p : s_{0..p-1} = t_{0..p-1}\}$ for its length.

Definition 4.3 (Longest common prefix (lcp) array). Let $s' = (s_0, \ldots, s_{n-1}) \in (\Sigma \cup \{\texttt{X}\})^n$, $s := s'\texttt{\$}$, and pos be the suffix array of s. Then lcp is an array of length $n+1$, where lcp$[i]$ is the length of the longest common prefix of $s_{(\texttt{pos}[i-1])}$ and $s_{(\texttt{pos}[i])}$ (for $i > 0$) and lcp$[0] = 0$.

Note the typographical distinction between the lcp-array and the lcp-function of two strings: lcp$[i] = \mathrm{lcp}(s_{(\texttt{pos}[i-1])}, s_{(\texttt{pos}[i])})$. We emphasize that the separator $ and the

wildcard `X` do not match anything, not even themselves. Therefore $\mathrm{lcp}(\texttt{ACGT}, \texttt{ACGA}) = 3$, but $\mathrm{lcp}(\texttt{ACXT}, \texttt{ACXA}) = 2$.

Definition 4.4 (q-bucket). A *q-bucket* of `pos` is an interval $[l, r]$ such that $\texttt{lcp}[l] < q$, $\texttt{lcp}[r+1] < q$ and $\texttt{lcp}[i] \geq q$ for all $i = l+1, \ldots, r$.

Thus the suffixes $s_{(\texttt{pos}[l])}, s_{(\texttt{pos}[l+1])}, \ldots, s_{(\texttt{pos}[r])}$ all start with the same q-gram (string of length q). They may even share a longer common prefix, but the suffixes $s_{(\texttt{pos}[l-1])}$ and $s_{(\texttt{pos}[r+1])}$ start with a different q-gram. It is convenient to have the starting points of all q-buckets available.

Definition 4.5 (Bucket (`bck`) array). Define a code $\langle a \rangle$ for each letter $a \in \Sigma$ as follows: Let $\langle \texttt{A} \rangle := 0$, $\langle \texttt{C} \rangle := 1$, $\langle \texttt{G} \rangle := 2$, and $\langle \texttt{T} \rangle := 3$. For a q-gram $Q = (Q_0, \ldots, Q_{q-1}) \in \Sigma^q$, let $\langle Q \rangle := \sum_{i=0}^{q-1} 4^{q-1-i} \langle Q_i \rangle$. Let s be a string with suffix array `pos`. For $\gamma = 0, \ldots, 4^q - 1$, let $\texttt{bck}[\gamma]$ be the left end l of the q-bucket $[l, r]$ of the q-gram Q with $\langle Q \rangle = \gamma$ if $Q \lhd s$, and set $\texttt{bck}[\gamma] := \infty$ (or any special value not else used) otherwise.

Note that we do not store the left endpoints of buckets of q-grams containing wildcards or separators in `bck`; the code $\langle Q \rangle$ is only defined for q-grams $Q \in \Sigma^q$. The above code definition has the advantage that it is easily updated when moving from one q-gram to the next one along a sequence.

Lemma 4.6. *For a string $s \in \Sigma^n$ of length n, let $Q(p) := s_{p..(p+q-1)}$ be the q-gram starting at position p ($p = 0, \ldots, n-q$). Then for $p \in [1, n-q]$,*

$$\langle Q(p) \rangle = 4 \cdot (\langle Q(p-1) \rangle \bmod 4^{q-1}) + \langle s_{p+q-1} \rangle.$$

This update can be implemented by bitwise operations since the DNA alphabet size is a power of two.

For probe design, we will build an enhanced suffix array of the entire transcriptome T, which consists of several collections T_c, which in turn consist of transcripts $t_{c,k}$. We represent T as a single long string that concatenates all $t_{c,k}$, not necessarily sorted by collection. We need to be able to identify the collection associated to the i-th suffix in alphabetical order, i.e., the collection of the transcript sequence that occupies position $\texttt{pos}[i]$ in T.

Definition 4.7 (Collection number (`cl`) array). Let $\texttt{cl}[i]$ be the collection number c of the transcript in which the suffix $T_{(\texttt{pos}[i])}$ starts.

Stefan Kurtz (personal communication) pointed out that in practice, the `cl`-array may be computable on the fly with a binary search in a list of start- and end-points of each transcript, and with a table that associates the running number of each transcript to

its collection number. While this modification slows down the lookup operation from constant time to time that increases with the logarithm of the number of transcripts, the memory for the `cl`-array can be saved. Only memory proportional to the number of transcripts instead of the total length of the transcripts is needed to emulate the `cl`-array in this way.

A full example of an enhanced suffix array built from strings from two collections is given in Figure 4.2.

p	s_p
0	C
1	A
2	C
3	A
4	C
5	$
6	A
7	C
8	C
9	$

$\Rightarrow$

i	pos$[i]$	$s_{(\text{pos}[i])}$	lcp$[i]$	cl$[i]$
0	1	ACAC$...	0	103
1	6	ACC$	2	27
2	3	AC$...	2	103
3	0	CACAC$...	0	103
4	2	CAC$...	3	103
5	7	CC$	1	27
6	4	C$...	1	103
7	8	C$	1	27
8	5	$...	0	103
9	9	$	0	27

$+$

γ	bck$[\gamma]$
$\langle$AA$\rangle = 0$	∞
$\langle$AC$\rangle = 1$	0
$\langle$CA$\rangle = 2$	3
$\langle$CC$\rangle = 3$	5

Figure 4.2: Enhanced suffix array of the concatenation s of `CACAC$` and `ACC$` from collections 103 (positions 0–5) and 27 (positions 6–9), respectively. Because two separators never compare as equal, their order is not important, and we have `lcp`$[7] = 1$ (not 2) and `lcp`$[9] = 0$. The bucket array `bck` was built for a q-gram length of $q = 2$ on the alphabet $\{$`A`, `C`$\}$ with $\langle$`A`$\rangle = 0$ and $\langle$`C`$\rangle = 1$. As `AA` occurs nowhere in s, we have `bck`$[\langle$`AA`$\rangle] = \infty$.

4.2.3 Enhanced Suffix Arrays vs. Suffix Trees

Suffix arrays are closely related to suffix trees. The definition of suffix trees, methods for their construction, and many applications are described extensively by Gusfield (1997).

The suffix tree of a string s of length $|s| = n$ is a rooted directed tree with n leaves numbered 0 to $n - 1$, which correspond to the suffixes of s. Each internal node, other than the root, has at least two children and each edge is labeled with a non-empty substring of s. No two edges out of a node have edge-labels beginning with the same character. The concatenation of the edge-labels on the path from the root to each leaf p exactly spells out the suffix $s_{(p)}$. Therefore each internal node u can be identified with the common prefix of all suffixes below u. A fundamental property is that the suffix tree of a string of length n can be represented with $O(n)$ space and built in $O(n)$ time, as first shown by Weiner (1973). A simpler algorithm was given by Ukkonnen (1995). A key component of the algorithm is the use of *suffix links* that directly link

each internal node u to the internal node that corresponds to $u_{(1)}$, i.e., to u without its first character.

From a suffix tree, the basic suffix array can be obtained by a "lexical" depth-first traversal. Therefore the suffix array can also be built in $O(n)$ time. Because of the space requirements of suffix trees (about 20 bytes or 5 integers per input character; Kurtz, 1999), however, it is more practical to build the suffix array directly, i.e., without building the tree as an intermediate step. Manber and Myers gave an algorithm that builds the `pos` and `bck` arrays in $O(n \log n)$ time using a linear amount of additional space. The `vmatch` implementation by Kurtz (2002) requires no additional space, is asymptotically slower in the worst case, but competitive in practice. The year 2003 brought no small surprise when three groups (Kim et al., 2003; Ko and Aluru, 2003; Kärkkäinen and Sanders, 2003) showed simultaneously that the *direct construction of suffix arrays is possible in linear time*; the approach of Kärkkäinen and Sanders uses a ternary divide-and-conquer approach and is probably the simplest one. Before then, it was assumed that the suffix links are an essential component for all linear-time algorithms, and these are not represented in suffix arrays.

As discovered by Kasai et al. (2001), the `lcp` array can be built from `pos` and its inverse `rank` in $O(n)$ time. It can be interpreted as an "encoding" of the tree structure.

From `rank` (which is trivially computable from `pos` in linear time by `rank`[`pos`$[i]] \leftarrow i$ for all i), the `cl` array is also built in $O(n)$ time by a linear traversal of the concatenated sequence: `cl`[`rank`$[p]] \leftarrow$ collection of the transcript that spans position p.

Enhanced suffix arrays have many practical advantages over suffix trees. Most importantly, they do not require implementing data structure for trees, but can be stored as flat arrays. This automatically makes them easier to handle and less costly in terms of memory. Additionally, they can be used independently of the method with which they have been generated; the user need not adopt the creating program's internal tree structure.

Initially it appeared that some problems can be solved more efficiently with suffix trees than with suffix arrays, but recently increasingly more direct array methods are found that are as efficient as or sometimes even more efficient than the tree methods. For example, determining whether u is a substring of s can be answered in $O(|u|)$ time (independently of $|s|$!), given a suffix tree of s by attempting to following the path labeled u from the root as deep as possible into the tree. If and only if this is possible for $|u|$ steps, u is a substring of s. Given a suffix array of s, one would perform a binary search which takes up to $O(|u| \log |s|)$ time. Using an appropriately sized bucket array, this can be reduced to expected $O(|u|)$ time, with additional tables, i.e., by further enhancing the array, the problem, can also be solved in $O(|u|)$ worst case time (Abouelhoda et al., 2002b; Sim et al., 2003).

The "bucket scan" algorithm that we present below is one of the few examples that is much more naturally and efficiently implemented with suffix arrays than with trees.

4.3 Matching Statistics

4.3.1 Definitions

Instead of looking at each probe in a target separately, we compare the whole target s to each transcript (or collection) t. For every $i = 0, 1, \ldots, |s| - 1$, we find the longest string that *starts at position* i in s and occurs *somewhere* in t. The resulting match lengths are called the *matching statistics* of s against t (Chang and Lawler, 1994).

Definition 4.8 (Matching statistics). For strings s and t, the *matching statistics* of s against t are $\mathrm{ms}^{s|t} = \mathrm{ms} = (\mathrm{ms}_0, \ldots, \mathrm{ms}_{|s|-1})$, where ms_i is the length of the longest prefix of $s_{(i)}$ that occurs somewhere in t.

Since we are also interested in longest common factors with mismatches, we generalize the definition of matching statistics as follows.

Definition 4.9 (Matching statistics with f mismatches). For strings s and t, the *matching statistics allowing f mismatches* of s against t are $\mathrm{ms}^{s|t;f} = (\mathrm{ms}_0^f, \ldots, \mathrm{ms}_{|s|-1}^f)$, where ms_i^f is the length of the longest prefix of $s_{(i)}$ that occurs somewhere in t with at most f mismatches.

Example 4.10. Suppose that $s = \texttt{GAATACT}$ and $t = \texttt{ATACGACT}$. We find $ms^{s|t;0} = (2, 1, 4, 3, 3, 2, 1)$. For instance, the third value, 4, is obtained by finding the longest prefix of $s_{(2)} = \texttt{ATACT}$ that matches somewhere in t. It is `ATAC`, a string of length 4. For one mismatch, we have $\mathrm{ms}^{s|t;1} = (4, 3, 5, 4, 3, 2, 1)$. The first value, 4, results from the match `GAAT` versus `GACT` at the end of t. Trivially $\mathrm{ms}_k^{s|t;f} \geq \mathrm{ms}_k^{s|t;0} + f$, unless the end of either string is hit.

For probe design, the exact value of certain matching statistics is irrelevant. Even without specifying the exact probe lengths yet, we know that the length of all probes is bounded by some constant, say 32; so there is no need to determine the exact value of matching statistics above 32. Also, small values of matching statistics are unimportant; the only relevant piece of information is that they are smaller than a certain threshold. Let $0 < R^0_{\min} \leq R^1_{\min} \leq R_{\max}$ be three positive integers that determine the range of relevant matching statistics, $[R^0_{\min}, R_{\max}]$ for ms, and $[R^1_{\min}, R_{\max}]$ for ms^1. We assume that the bucket prefix length q of the suffix array has been chosen such that $q \leq R^0_{\min}$. It is also reasonable to ensure that for given width Δ of longest common factor statistics LCFS, we have $R^0_{\min} \leq |p| - \Delta + 1$ for all probes p. Thus if $|p_{\min}|$ and $|p_{\max}|$ are the respective lengths of the shortest and the longest probe candidate to be considered, we need the following chain of inequalities to hold:

$$q \leq R^0_{\min} \leq |p_{\min}| - \Delta + 1 \leq R^1_{\min} \leq |p_{\min}| \leq |p_{\max}| \leq R_{\max}.$$

For example, for 25-mer probes, we take $\Delta = 16$, implying that common factors down to length 10 are counted in the final longest common factor statistics. Then one could take $q = R^0_{\min} = 10$, $R^1_{\min} = 13$, and $R_{\max} = 25$. Now the computational problem is as follows.

> For a given target s and C transcript collections T_c ($c = 1, \ldots, C$), where T_c is represented as a single string, compute approximations $\texttt{MS}[i][c]$ and $\texttt{MS}^1[i][c]$ ($i = 0, \ldots, |s| - 1$; $c = 1, \ldots, C$) to the matching statistics $\mathrm{ms}_i^{s|T_c}$ and $\mathrm{ms}_i^{s|T_c;1}$ such that

$$
\begin{array}{rcll}
\texttt{MS}[i][c] & = & \mathrm{ms}_i^{s|T_c} & \text{if } \mathrm{ms}_i^{s|T_c} \in [R^0_{\min}, R_{\max} - 1], \\
\texttt{MS}[i][c] & < & R^0_{\min}, & \text{if } \mathrm{ms}_i^{s|T_c} < R^0_{\min}, \\
\texttt{MS}[i][c] & \geq & R_{\max} & \text{if } \mathrm{ms}_i^{s|T_c} \geq R_{\max}; \\
\texttt{MS}^1[i][c] & = & \mathrm{ms}_i^{s|T_c;1} & \text{if } \mathrm{ms}_i^{s|T_c;1} \in [R^1_{\min}, R_{\max} - 1], \\
\texttt{MS}^1[i][c] & < & R^1_{\min} & \text{if } \mathrm{ms}_i^{s|T_c;1} < R^1_{\min}, \\
\texttt{MS}^1[i][c] & \geq & R_{\max} & \text{if } \mathrm{ms}_i^{s|T_c;1} \geq R_{\max}.
\end{array}
\tag{4.1}
$$

4.3.2 The Bucket Scan Algorithm

We present an efficient algorithm to compute the MS-array according to (4.1) using an enhaced suffix array of the entire transcriptome T concatenated to a single long string t. For each starting position i of the target s, we proceed as follows.

- We first obtain the code γ of the q-prefix $Q(i)$ of $s_{(i)}$, $\gamma = \langle Q(i) \rangle = \langle s_{i..(i+q-1)} \rangle$. According to Lemma 4.6, this code can be updated in constant time as i increases. (If $Q(i)$ is not a valid DNA q-gram, because it contains a separator or a wildcard, then γ is not defined. There is nothing to do and we may skip ahead to the next valid q-gram in the sequence.)
- The code γ is used as an index to the bucket array; let $r := \texttt{bck}[c]$. Unless the bucket is empty ($r = \infty$), we perform a *bucket scan* to find $\texttt{MS}[i][c]$ for all collections c. The bucket scan is shown in detail in Figure 4.3. After the bucket scan, we move to the next position $i + 1$ of the target sequence s.

To understand the bucket scan in detail, note two things: First, the index variable r is incremented from its starting value (the left end of the bucket) until the right end of the bucket is reached (that is, until $\texttt{lcp}[r] < q$ for the first time). Second, the variable μ keeps track of the longest common prefix of $s_{(i)}$ and $t_{(\texttt{pos}[r])}$ as r scans over the bucket.

In line 1, we know already that $s_{(i)}$ and $t_{(\texttt{pos}[r])}$ have a common prefix of at least q characters, but we need to check for a longer match of length $\mu \geq q$ by comparing additional characters (at most $R_{\max} - q$ because matching statistics above $R_{\max}$ are

The Bucket Scan

Input: Target sequence s; Current position i;
Transcriptome T (concatenated from C collections) and its
Enhanced suffix array `pos`, `cl`, `lcp`;
Starting position r of the q-bucket of $s_{i..i+q-1}$ ($r = \texttt{bck}[\langle s_{i..(i+q-1)}\rangle]$);
Maximum relevant matching statistic $R_{\max}$;
Initially $\texttt{MS}[i][c] = 0$ for all $c = 1, \ldots, C$

Output: Approximate matching statistics $\texttt{MS}[i][c]$ from (4.1) for all $c = 1, \ldots, C$

1. $\mu \leftarrow q + \min(\mathrm{lcp}(s_{(i+q)}, t_{(\texttt{pos}[r]+q)}),\ R_{\max} - q)$
2. $\texttt{MS}[i][\texttt{cl}[r]] \leftarrow \mu$
3. $r \leftarrow r + 1$
4. **while** $(\texttt{lcp}[r] \geq \mu)$
5. **if** $(\texttt{lcp}[r] = \mu)$ **then**
   $\mu \leftarrow \mu + \min(\mathrm{lcp}(s_{(i+\mu)}, t_{(\texttt{pos}[r]+\mu)}),\ R_{\max} - \mu)$
6. $\texttt{MS}[i][\texttt{cl}[r]] \leftarrow \mu$
7. $r \leftarrow r + 1$
8. **while** $(\texttt{lcp}[r] \geq q)$
9. **if** $(\texttt{lcp}[r] < \mu)$ **then** $\mu \leftarrow \texttt{lcp}[r]$
10. **if** $(\texttt{MS}[i][\texttt{cl}[r]] < \mu)$ **then** $\texttt{MS}[i][\texttt{cl}[r]] \leftarrow \mu$
11. $r \leftarrow r + 1$

Figure 4.3: The *bucket scan* algorithm constitutes the core of all-against-all matching statistics computation.

irrelevant). In line 2, the matching statistics for the collection belonging to position $\texttt{pos}[r]$ are set. As r scans over the bucket, there are two phases. First, there is a phase of increasing μ (lines 4–7); additional matches can only be obtained when the current value of μ equals the `lcp`-value at the current position. The phase ends once $\texttt{lcp}[r]$ drops below μ, because then in line 4,

$$\begin{aligned} \mathrm{lcp}(s_{(i)}, t_{(\texttt{pos}[r])}) &\leq \mathrm{lcp}(t_{(\texttt{pos}[r-1])}, t_{(\texttt{pos}[r])}) = \texttt{lcp}[r] \\ &< \mathrm{lcp}(t_{(i)}, s_{(\texttt{pos}[r-1])}) = \mu, \end{aligned}$$

so to maintain the loop invariant $\mu = \mathrm{lcp}(s_{(i)}, t_{(\texttt{pos}[r])})$, μ must decrease. In the second phase (lines 8–11), μ decreases until it drops below q. In lines 6 and 10, the `MS` entry for the collection $\texttt{cl}[r]$ corresponding to position $\texttt{pos}[r]$ is set; in the second phase, we must explicitly ensure that we do not decrease a possibly existing value of $\texttt{MS}[i][\texttt{cl}[r]]$.

After the bucket scan, we have $\texttt{MS}[i][c] = 0$ if and only if collection c never appears in the bucket (i.e., $\mathrm{ms}_i^{s|T_c} < q \leq R^0_{\min}$). Otherwise

$$\texttt{MS}[i][c] = \min(R_{\max}, \max_k \mathrm{lcp}(s_{(i)}, (T_c)_{(k)})),$$

where k is any position in the concatenated background string that belongs to collection T_c. Thus $\texttt{MS}[i][c]$ contains a correct value in the sense of (4.1).

Computing Matching Statistics with one Mismatch. In principle, we compute $\texttt{MS}^1[i][c]$ in the same way as $\texttt{MS}[i][c]$. The only difference is that instead of a single bucket scan, $3R_{\max}+1$ bucket scans are necessary for each position i, because we scan with the exact substring $s_{i,\ldots,i+R_{\max}-1}$, and with each of the $3R_{\max}$ strings obtained by applying each possible mismatch type at each possible position. The value $\texttt{MS}^1[i][c]$ is set to the maximum of all $3R_{\max}+1$ values of $\texttt{MS}[i][c]$ obtained in this way.

Even though this is a brute force approach to compute ms^1, it is surprisingly fast because a bucket scan takes an unnoticeable amount of time if the value of bucket prefix length q is chosen appropriately.

4.3.3 Analysis and Choice of the Bucket Prefix Length

Define L as the length of the longest target sequence, C as the number of different collections, and n as the total length of all transcripts in all collections (the length of the suffix array).

As presented, the algorithm has one major problem: The MS array needs $L \cdot C$ bytes (e.g., 10 MB for $L = 1000$ and $C = 10000$), and only initializing it to zero requires $O(LC)$ time for *every* target. Since small matching statistics are not recorded, however, the resulting array is sparse anyway. We can also exploit that consecutive matching statistics are not independent. A sparse representation that exploits this property is thus desirable. We provide a solution in Section 4.4 where we describe how to represent a vector of matching statistics as a *jump list*. Conceptually this does not change the algorithm, only the memory representation of MS is changed. The average length of a jump list also increases linearly with L, and we need 3 bytes per entry (one short integer for the position number and one byte for the jump level; see Section 4.4). Experiments on real DNA sequences show that the average list length is much shorter than L, approximately $L/250$, so the space requirement reduces to about $L/80$ bytes per collection. If carefully implemented, the re-initialization of the C jump lists for each target can be done in $O(C)$ total time. For matching statistics with one mismatch, jumps become more frequent, but this can be compensated by setting $R^1_{\min}$ higher than $R^0_{\min}$. We found that for $R^1_{\min} = R^0_{\min} + 3$, the jump lists have about the same size.

A central point in the performance of the algorithm is the choice of the bucket prefix length q. Since matching statistics smaller than q are reported as zero, q must be sufficiently small so that no relevant long matches are lost. On the other hand, q should be as large as possible to keep the buckets small. When we choose $q := \lfloor \log_4(n/128) \rfloor$,

the expected occurrence of each q-gram ranges between 128 and 512, thus the buckets have constant expected size. When n is large enough and the typical probe length is more than twice the value of q, there are no problems with this choice. In practice, $q = 10$ works well for the design of 25-mers. Choosing $q = \Theta(\log n)$ always yields an expected running time of $O(|s|)$ for the matching statistics algorithm for a target sequence s, independently of C. The reason is that only a single q-bucket, which has constant average size, is considered for each position i of s.

The memory requirements of the algorithm are dominated by those of the suffix array. We need $n + 1$ integers ($4n + 4$ bytes) for each of `pos` and `cl`, 4^q integers (at most $n/32$ bytes when q is chosen as above) for `bck`, and $n + 1$ bytes for `lcp` (we do not need to know the exact `lcp`-value when it is above 255). The sequence itself requires n bytes. Here we disregard memory needed to store sequence identifiers and other administrative information. Thus the total memory requirement for the suffix array is slightly more than $10n$ bytes. For the human transcriptome, as available from the GeneNest database (Haas et al., 2000) of size 550 MB, this amounts to 5.5 GB. If the `cl`-array is computed on the fly, as described above, we save $4n$ bytes, reducing the memory requirement to slightly more than $6n$ bytes, i.e., 3.3 GB for the GeneNest transcripts. This option should only be used when memory is more valuable than time, as we found that the computation time increases by a factor of approximately 3.3 in this case.

An issue to be considered in the future is the size of a basic integer. We have worked with a basic integer type of 4 bytes, restricting the total background sequence length to $2^{32} - 1$ at most. During the next years, we will be confronted with ever larger genomes and must move to 8-byte integers; hence memory requirements will roughly double.

4.3.4 The Role of the Enhanced Suffix Array

Would it have been possible to implement the same algorithm with a suffix tree instead of an enhanced suffix array? We do not see a way that would be as efficient as our solution.

An asymptotically optimal algorithm to compute matching statistics of s against *one* string was given by Chang and Lawler (1994): To compute $\mathrm{ms}_0^{s|t}$, start matching the prefix of s in the suffix tree of t by walking down from the root until no further matches are possible. The number μ of matches is exactly $\mathrm{ms}_0^{s|t}$. To compute $\mathrm{ms}_1^{s|t}$, use the suffix links to move to the position in the tree that corresponds to $s_{1..\mu-1}$ and try to obtain additional matches from there until no more matches are possible. This gives $\mathrm{ms}_1^{s|t}$. The procedure is repeated for every position $i < |s|$ of s. Since every character of s is matched only once against some character in t and following the suffix links

is possible in $O(|s|)$ overall time, the matching statistics are computed in $O(|s|)$ time once the suffix tree of t is built.

To compute matching statistics of s against *many* sequences or collections, we may either use a separate suffix tree for each of the C collections (requiring $O(C|s|)$ total traversal time), or one generalized suffix tree of all sequences, in which each leaf is annotated with the collection number. However, to compute $\texttt{MS}_{i,c}$ for all c with this method, we must examine the set of leaves not only below the maximally matching prefix of $s_{(i)}$ in the tree, but also of all shorter prefixes, since the different collections have different matching statistics with s. The need to look at this leaf set results in a large time overhead. The faster alternative to annotate each internal node with the set of collection numbers that occur in the leaves below it must be discarded because of the $O(C)$ memory requirement per node.

In principle the bucket scan algorithm can be understood as a traversal of only the leaf level of the generalized suffix tree, and only below the required q-prefix. The key is that this procedure is more simply and more efficiently implemented with the enhanced suffix array traversing the suffix tree. Additional advantages gained by using the array are the lower space requirements and a much better cache performance due to linear array scans instead of the random memory access caused by traversing a tree structure.

4.4 Jumps in Matching Statistics

4.4.1 Motivation and Definition

As noted in Section 4.3.3, the `MS` array requires considerable $O(CL)$ space and also $O(CL)$ time for its initialization to zero for each target. Additionally, it contains redundant information because successive matching statistics are not independent: From the definition of matching statistics, one property follows immediately.

Lemma 4.11 (Suffix property of matching statistics). *Consecutive matching statistics decrease by at most one:*

$$\mathrm{ms}_i^{s|t} \geq \mathrm{ms}_{i-1}^{s|t} - 1 \quad \textit{for all } i = 1, \ldots, |s| - 1.$$

Proof. The inequality holds, because a prefix of $s_{(i-1)}$ of length m matching at some position j in t implies a matching prefix of $s_{(i)}$ of length $m-1$ at position $j+1$ in t. Of course, there can be a longer match of $s_{(i)}$ elsewhere in t. □

The suffix property leads to the concept of *jumps* in matching statistics.

Definition 4.12 (Jumps in matching statistics). We say that a *jump* occurs at position $i > 0$ in $\mathrm{ms}^{s|t}$ if and only if $\mathrm{ms}_i^{s|t} \neq 0$ and $\mathrm{ms}_i^{s|t} > \mathrm{ms}_{i-1}^{s|t} - 1$. At the beginning of s ($i = 0$), there is always a jump unless $\mathrm{ms}_0^{s|t} = 0$. When there is a jump at position i, we call $J := \mathrm{ms}_i^{s|t}$ the *jump level*.

Lemma 4.11 and Definition 4.12 can also be stated for matching statistics with f mismatches $\mathrm{ms}^{s|t;f}$ without any changes.

A jump is formally written as a pair (i, J) of the jump's position and level. Then a vector of matching statistics can be succinctly represented by list of jumps. Since there are relatively few jumps compared to the string length, it pays to store lists of jumps instead of the complete matching statistics.

Using jump lists in the bucket scan. The bucket scan algorithm (Figure 4.3) was presented using an array $\mathtt{MS}[i][c]$ of matching statistics. Lines 2, 6 and 10 are easily modified not to store the value directly into an array, but to update a jump list. A straightforward approach is to keep a jump list for each collection T_c and to check whether the current value of μ during the bucket scan results in a jump at position i by comparing it to $J - d$, where J is the level of the most recent jump in collection T_c, and d is its distance to position i.

The jump lists can be organized in a way that minimizes their lengths. For example, even though all jumps to levels $\geq q$ are detected during the bucket scans, we only need to store those to levels $\geq R_{\min}^0$ in the jump lists. On the other hand, m consecutive jumps at positions $i, i+1, \ldots i+m-1$ to levels all $\geq R_{\max}$ can be stored as a single jump at position i to level $R_{\max} + m - 1$.

4.4.2 Stochastic Properties of Jumps

We shall estimate the frequency of jumps to level L in a (non-uniform) i.i.d. random text model, where each character $c \in \Sigma$ receives a probability π_c. We generate two independent i.i.d. strings s and t of lengths $|s| = m$ and $|t| = n$ according to π. We use $\mathbb{P}$ as a generic probability measure for events in this text model. Especially, we write $\mathbb{P}(w) := \prod_{k=0}^{|w|-1} \pi_{w_k}$ for the probability that the word w is generated in $|w|$ steps. Let us define p as the probability that two random characters match, $p := \sum_{c \in \Sigma} \pi_c^2$. We also set $q := 1 - p$.

Frequency of jumps. Consider the random number of occurrences K_L of a random string of length L in t and its expectation $\mathbb{E}[K_L]$.

Lemma 4.13 (Expected number of occurrences of a random word).

$$\mathbb{E}[K_L] = (n - L + 1) \cdot p^L.$$

Proof. We have

$$\begin{aligned} \mathbb{E}[K_L] &= \sum_{i=0}^{n-L+1} \sum_{w \in \Sigma^L} \mathbb{P}(W = w, \text{ and } w \text{ occurs at position } i \text{ in } t) \\ &= (n - L + 1) \cdot \sum_{w \in \Sigma^L} (\mathbb{P}(w))^2 \;=\; (n - L + 1) \cdot p^L, \end{aligned}$$

because the characters of w are independent. □

Exact statements about the whole distribution of K_L are more difficult, because we need to take possible self-overlaps (also known as autocorrelations or period sets) of all words w into account. This is theoretically possible, and an efficient algorithm to enumerate all autocorrelations for a given word length L is described by Rivals and Rahmann (2001). Rahmann and Rivals (2003) prove for uniform π that the Poisson approximation $P(K_L = 0) \approx e^{-\mathbb{E}[K_L]}$ is highly accurate.

Both intuition and simulations suggest that K_L has approximately a Poisson distribution with parameter $\lambda := \mathbb{E}[K_L]$ when π is close to uniform and $\mathbb{E}[K_L] \ll n$. Unless π is close to a Dirac distribution, a typical randomly drawn word W has no or low self-overlap, and therefore its K_L occurrences in t are likely non-overlapping and hence independent. The condition $\mathbb{E}[K_L] \ll n$ ensures that the occurrences are sufficiently rare, so it is unlikely that two occurrences are adjacent. Thus approximately

$$\mathbb{P}(K_L = k) = e^{-\lambda} \cdot \lambda^k / k!, \tag{4.2}$$

where $\lambda = \mathbb{E}[K_L] = (n - L + 1) \cdot p^L$ by Lemma 4.13.

Now we compute the expected number E_L of jumps in $\mathrm{ms}^{s|t}$ to jump level L. Consider an arbitrary but fixed position $1 \le i \le m - L - 1$; we shall estimate the probability ρ_L that a jump to level L occurs at position i in s.

Lemma 4.14. *We have*

$$\rho_L = e^{-\lambda \cdot (1 - q^2)} - e^{-\lambda}, \tag{4.3}$$

where $\lambda = (n - L + 1) \cdot p^L$ *as before.*

Proof. A jump to level L occurs at i if and only if the following three conditions hold.

1. The L-gram $w := s_{i..i+L-1}$ occurs in t at least once.
2. No occurrence of w in t is followed by the letter s_{i+L}.

3. No occurrence of w in t is preceded by the letter s_{i-1}.

We condition on the number k of occurrences of w in t and find that the probability that the second and third conditions hold, given that $K_L = k$, is just q^{2k} since we assume that the letters following and preceding each occurrence are independent (the occurrences are not adjacent). Thus we have approximately

$$\begin{aligned}\rho_L &= \sum_{k\geq 1} \mathbb{P}(K_L = k) \cdot q^{2k} \;=\; e^{-\lambda} \cdot \sum_{k\geq 1} (\lambda q^2)^k / \, k! \\ &= e^{-\lambda} \cdot (\exp(\lambda q^2) - 1),\end{aligned}$$

as claimed. □

Theorem 4.15. *An estimate of the expected number E_L of jumps in $\mathrm{ms}^{s|t}$ with jump level L is given by*

$$E_L = (m - L - 1)\, e^{-\lambda\cdot(1-q^2)} \;+\; 2\, e^{-\lambda p} \;-\; (m - L + 1)\, e^{-\lambda},$$

where $\lambda = (n - L + 1) \cdot p^L$.

Proof. The expectation E_L is

$$E_L = \sum_{i=0}^{m-L} \mathbb{P}(\text{Jump to level } L \text{ at } i).$$

As stated in Lemma 4.14, this probability is ρ_L for $i = 1, \ldots, m - L - 1$. For the L-grams at each boundary of s, either the second or third condition in the proof of Lemma 4.14 is automatically satisfied. Instead of ρ_L we obtain a probability of $\rho'_L = e^{-\lambda p} - e^{-\lambda}$. Summing $(m - L + 1)\rho_L + 2\rho'_L$ yields the stated result. □

Let us point out again that Theorem 4.15 provides only an approximation of E_L, based on the assumption that occurrences of typical length-L words are relatively rare and non-overlapping and hence Poisson-distributed. The approximation is likely to break down when highly periodic words become more probable, that is, when π is far from the uniform distribution on Σ, or when L becomes too short in comparison to the string length n such that λ approaches n.

All calculations may be generalized to the case where we allow f mismatches. The main problem is to determine the distribution of K_L, the number of occurrences with f mismatches of a random L-gram in t. For increasing values of f, occurrences become more frequent, so the Poisson approximation eventually breaks down. Therefore we shall restrict the discussion to the case of a single mismatch and proceed as in the previous section.

Theorem 4.16. *The expected number E_L of jumps to level L in the matching statistics with one mismatch* $\mathrm{ms}^{s|t;1}$ *is approximately given by*

$$E_L = (m - L + 1)\,(e^{-\lambda\cdot(1-q^2)} - e^{-\lambda}),$$

where $\lambda = (n - L + 1) \cdot L \cdot p^{L-1} \cdot q$.

Proof. The expected number of 1-mismatch-occurrences of a random L-gram in a string of length n is $\lambda := \mathbb{E}[K_L] = (n - L + 1) \cdot L \cdot p^{L-1} \cdot q$. Assuming that Poisson approximation is reasonable, the rest of the computation remains unchanged, except that we completely ignore edge effects this time. □

Length of the longest common factor. Let $E_L^+ := \sum_{\ell \geq L} E_\ell$ be the expected number of jumps to level at least L. Consider the length L^* where $E_{L^*}^+ \geq 1$ but $E_{L^*+1}^+ < 1$. Thus in typical cases, we observe jumps to level L^*, but not to or above level $L^* + 1$. Therefore L^* is a good estimate of the typical *length of the longest common factor* $\mathrm{lcf}(s, t)$ of s and t.

Necessary conditions for $E_L^+ \approx 1$ are $\lambda \ll 1$ and that L is small compared to both m and n when m and n are of comparable magnitude. Thus for the case of no mismatches, we may approximate $\lambda \approx np^L$, and $e^{-\lambda x} \approx 1 - \lambda x$. Therefore $E_L \approx mq^2\lambda \approx mnq^2p^L$. In the case of one mismatch, we obtain the approximations $\lambda \approx nLqp^{L-1}$ and $E_L \approx Lmnq^3p^{L-1}$. By geometric summing, it follows that

$$\begin{aligned} E_L^+ &\approx mnqp^L \quad \text{(exact matches)}, \\ E_L^+ &\approx Lmnq^2p^{L-1} \quad \text{(one mismatch)}. \end{aligned} \tag{4.4}$$

While substring lengths and L^* are integers, the functional form of E_L^+ is also defined for non-integer values of L. Therefore we define the *center* of the distribution of the lcf as the value L° where $E_{L^\circ}^+ = 1$. Note that the center is neither the mean nor the median, nor any other moment of the lcf distribution.

Theorem 4.17. *The center L° of the lcf of two random strings with lengths m and $n = \Theta(m)$ generated with close-to-uniform character distribution π on Σ is*

$$L^\circ \approx \frac{\ln(mn) + \ln q}{\ln(1/p)}. \tag{4.5}$$

This simplifies to $L^\circ \approx \log_{|\Sigma|}(mn)$ when π is uniform and $|\Sigma| \gg 1$.

For the length of the longest common factor with one mismatch lcf^1*, the center L° is the solution of the equation*

$$mnq^2 \cdot L^\circ \cdot p^{L^\circ - 1} = 1. \tag{4.6}$$

Proof. This follows directly from the functional form of E_L^+ above. □

While this result is approximate, it agrees with more precise results given by Waterman (1995) and Abbasi (1997) when $m = n$, π is uniform, and the strings are considered as cyclic (so there are no edge effects). In this case it has been shown that for any $a \geq 1$, $\mathbb{P}[|\mathrm{lcf}(s,t) - \log_{|\Sigma|} n^2| > a] \leq 18|\Sigma|^{-a}/n^2$; so $\mathrm{lcf}(s,t)$ is indeed tightly concentrated around $\log_{|\Sigma|} n^2$. Even though Equations (4.5) and (4.6) do not make a precise statement, they give us an intuitive idea about the longest common factor in the non-uniform Bernoulli model, for unequal string lengths, and allowing mismatches.

Table 4.1: Comparison of the observed (O_L) and expected (E_L) number of jumps to level L in $\mathrm{ms}^{s|t}$ and $\mathrm{ms}^{s|t;1}$ for 100 four-letter strings s of length 61140 against 10 four-letter strings t of length 1048576.

	ms^0			ms^1		
L	O_L	E_L	$\mathbb{P}(\mathrm{lcf} = L)$	O_L	E_L	$\mathbb{P}(\mathrm{lcf}^1 = L)$
7	0.000	0.000	0.000	0.000	0.000	0.000
8	56.472	56.050	0.000	0.000	0.000	0.000
9	9560.324	9550.611	0.000	0.000	0.000	0.000
10	17059.882	17063.854	0.000	0.048	0.123	0.000
11	7222.149	7223.793	0.000	1284.051	1646.841	0.000
12	2064.799	2064.855	0.000	15475.076	16479.856	0.000
13	533.840	533.867	0.000	13440.482	13654.596	0.000
14	134.500	134.593	0.000	5020.893	5039.977	0.000
15	33.599	33.719	0.000	1471.427	1471.212	0.000
16	8.531	8.434	0.067	401.045	401.500	0.000
17	2.148	2.109	0.418	107.368	107.308	0.000
18	0.528	0.527	0.357	28.614	28.451	0.000
19	0.132	0.132	0.115	7.517	7.511	0.084
20	0.034	0.033	0.032	2.058	1.977	0.445
21	0.007	0.008	0.007	0.483	0.519	0.323
22	0.004	0.002	0.004	0.134	0.136	0.115
23	0.000	0.001	0.000	0.033	0.036	0.029
24	0.000	0.000	0.000	0.004	0.009	0.004

Simulations. To check the accuracy of our estimations, we created a set S of 100 DNA sequences ($|\Sigma| = 4$) of length 61140 (60 Kb), and a set T of 10 DNA sequences of length 1048576 (1 Mb). We counted the number of jumps in the matching statistics $\mathrm{ms}^{s|t}$ of each sequence pair $(s,t) \in S \times T$, averaged the observed numbers over all 1000 pairs, and compared the averages with the expected numbers E_L. We did the same comparison for the matching statistics with one mismatch. The results are shown in Table 4.1. Additionally, we found the lcf for each of the 1000 string pairs. Its empirical distribution is shown in the same table.

The observed jump counts agree well with the approximated expected counts for the matching statistics allowing no mismatches. The center of the longest common factor distribution is 17.75, and most of the probability mass is indeed found on the lengths 17 and 18. The mean of the empirical distribution is 17.66. The comparison for $\mathrm{ms}^{s|t;1}$ shows, however, that the approximation is less good than for $\mathrm{ms}^{s|t}$, and this becomes worse for decreasing L. The approximate lcf-center is at 20.73, and the empirical mean is at 20.57.

4.5 Obtaining LCF Statistics from Jumps in Matching Statistics

So far we have a way to obtain $\mathrm{ms}^{s|t}$ and $\mathrm{ms}^{s|t;1}$ for a target sequence s against each transcript t (or transcript collection). Probe candidates are certain substrings $p \lhd s$; they are specified by a starting position i and a length L. We are interested in obtaining $\mathrm{lcf}(p, t)$ from $\mathrm{ms}^{s|t}$.

4.5.1 Matching Statistics of Substrings

From the definition of matching statistics, it is immediate that the length of longest common factor between p and t is equal to the maximum value of the matching statistics between p and t,

$$\mathrm{lcf}(p, t) = \max \mathrm{ms}^{p|t}.$$

At the moment, we know the matching statistics of s against t, not those of p against t. Therefore we need to discuss the general connection between $\mathrm{ms}^{s|t}$ and $\mathrm{ms}^{u|t}$ when $u \lhd s$ (during this general discussion, we use the generic notation u instead of the probe-specific p). Our main interest lies in the connection between the *jump lists* of $\mathrm{ms}^{s|t}$ and $\mathrm{ms}^{u|t}$.

Lemma 4.18 (Matching statistics of substrings). *Let $u := s_{i..i+L-1}$, be the length-L substring of s that starts at position i. Then*

$$\mathrm{ms}_k^{u|t} = \min(\mathrm{ms}_{i+k}^{s|t},\, L - k) \qquad (k = 0, \ldots, L-1).$$

Proof. A match starting at position k in u initially consists of the same characters as a match starting at position $i+k$ in s. It may be truncated by the remaining substring length, which is $L - k$ for the suffix starting at position k in u. □

Lemma 4.19 (Jumps in matching statistics of substrings). *Jumps in* $\mathrm{ms}^{u|t}$ *can occur at position* $k = 0$*, and otherwise at most at those positions* $0 < k < L$ *where a jump occurs in* $\mathrm{ms}^{s|t}$ *at position* $i + k$*. If there is a jump at position* $k = 0$*, its jump level is*

$$J_0 = \min(\max\{j - d, 0\}, L).$$

If there is a jump at position $k > 0$*, its jump level is*

$$J_k = \mathrm{ms}_k^{u|t} = \min(\mathrm{ms}_{i+k}^{s|t},\ L - k).$$

Proof. There is always (unless u_0 does not occur at all in t) a jump at $k = 0$ in u (the start of the substring), even when there is no jump at position i in s. Its jump level $J_0 = \mathrm{ms}_0^{u|t}$ is obtained by looking at the closest jump to the left of i in s. Assume it occurs at position $i - d$ for some $d \geq 0$ and its level is j. Because there are no other jumps between $i - d$ and i, we know that $\mathrm{ms}_i^{s|t} = j - d$ unless this is below zero. It follows that $J_0 = \min(\max\{j - d, 0\}, L)$.

For $0 < k < L$, note that a jump at position k in $\mathrm{ms}^{u|t}$ implies a jump at position $i+k$ in $\mathrm{ms}^{s|t}$, because $\mathrm{ms}_k^{u|t} > \mathrm{ms}_{k-1}^{u|t} - 1$ is equivalent to $\min(\mathrm{ms}_{i+k}^{s|t},\ L-k) > \min(\mathrm{ms}_{i+k-1}^{s|t},\ L-k+1) - 1$, which implies $\mathrm{ms}_{i+k}^{s|t} > \mathrm{ms}_{i+k-1}^{s|t} - 1$. (The converse implication does not hold, as shown by the following example.) The statement about the jump level follows from the previous lemma. □

Example 4.20. For $s =$ `GAATACT` and $t =$ `ATACGACT`, $\mathrm{ms}^{s|t} = (2, 1, 4, 3, 3, 2, 1)$. We now focus our attention on the substring $u =$ `AATA` of s. We obtain the following jumps (non-jumps are marked with a dot).

Position i in s	0	1	2	3	4	5	6
Jumps in $\mathrm{ms}^{s\|t}$	2	.	4	.	3	.	.
Jumps in $\mathrm{ms}^{u\|t}$		1	3	.	.		
Position k in u		0	1	2	3		

There is an initial jump to level 1 at $k = 0$ (derived from the jump to level 2 at $i = 0$). The jump to level 3 at $k = 1$ is the jump to level 4 at $i = 2$ restricted to the remaining substring length. Finally, the jump to level 3 at $i = 4$ does not become a jump at $k = 3$, because the value of $\mathrm{ms}_3^{u|t} = 1$ is already implied by the previous jump.

Definition 4.21 ($[i, L]$-jump-set). We define the $[i, L]$-jump-set of $\mathrm{ms}^{s|t}$ as the set consisting of the jumps between positions $i+1$ and $i+L-1$ (inclusive), and the closest jump to the left of $i + 1$ at position $i - d$ for the appropriate distance $d \geq 0$.

From Lemma 4.19, we deduce

Corollary 4.22. Let u be the substring of length L of s that starts at position i. To obtain the jumps in $\mathrm{ms}^{u|t}$, it suffices to look at the $[i, L]$-jump-set of $\mathrm{ms}^{s|t}$.

It is easy to see that all relationships continue to hold when jumps in matching statistics with f mismatches are considered. Thus we obtain the following result.

Theorem 4.23. *Let u be a substring of s starting at position i. The longest common substring of u and t with at most f mismatches is the maximal level of all jumps in $\mathrm{ms}^{u|t;f}$. It can be computed from the $[i, |u|]$-jump set of $\mathrm{ms}^{s|t;f}$.*

Proof. By definition, $\mathrm{lcf}(u,t) = \max_{i=0,\dots,|u|-1} \mathrm{ms}_i^{u|t;f}$. The maximum must occur at a jump; therefore it is equal to the maximal jump level in $\mathrm{ms}^{u|t;f}$. The last statement now follows from Corollary 4.22. □

4.5.2 Jump List Processing

We come back to the problem of obtaining LCF statistics for all probe candidates from the same target s. The probes can be succinctly described by specifying s and a list P with a pair of starting position and length (i, L) for each probe, i.e., $P = (p_1, \dots, p_N) = ((i_1, L_1), \dots, (i_N, L_N))$ with $0 \le i_1 \le i_2 \le \cdots \le i_N < |s|$ and $i_j + L_j \le |s|$ for all j. Note that it is allowed that several probes start at the same position in s.

Assume that the jump list for the matching statistics of s against target collection T_c is written as $J = ((i'_1, J_1), (i'_2, J_2), \dots)$ with $0 = i'_1 < i'_2 < \dots$. (If in fact there is no jump at position 0, we introduce the artificial jump $(0, 0)$ for convenience.)

We move through P from left to right and maintain the jump set that belongs to the current probe. For probe j, the jump set is $\{(i'_\alpha, J_\alpha), (i'_{\alpha+1}, J_{\alpha+1}), \dots, (i'_\beta, J_\beta)\}$ with

$$\begin{aligned} \alpha &= \max\{a : i'_a \le i_j\}, \\ \beta &= \max\{b : i'_b < i_j + L_j\}. \end{aligned} \tag{4.7}$$

- For the first probe candidate p_1, we find α and β by scanning through the list of i'-values from left to right to satisfy (4.7). From the jump levels in the so determined $[i_1, L_1]$-jump-set and using Lemma 4.19, we find $\mathrm{lcf}(p_1, T_c)$.
- For each successive probe candidate p_j, we increase α or β (or both) from their current values until (4.7) is again satisfied and proceed as before.

The jump sets are typically very small (say, 2 to 3 jumps per probe candidate); therefore this is a rapid procedure. Figure 4.4 illustrates the method.

It is possible to determine $\mathrm{lcf}^1(p_j, T_c)$ similarly from the jump list of $\mathrm{ms}^{s|t;1}$. If desired, the lcf and lcf^1 values can be combined into $\mathrm{lcf}^*(p_j, T_c)$ at this point (cf. the discussion at the end of Section 3.1).

Position i, i'	00	01	02	03	04	05	06	07	08	09	10	11	12
Jump level	**4**	**7**	.	.	.	**5**	.	.	.	**6**	.	.	.
Index α, β	1	2				3				4			
$p_1 = (0, 6)$	▬	▬	▬	▬	▬	▬							
Adj. levels	**4**	**5**				**1**							
$\alpha =$	⇑1												
$\beta =$						⇑3							
Position i, i'	00	01	02	03	04	05	06	07	08	09	10	11	12
$p_2 = (2, 7)$			▬	▬	▬	▬	▬	▬	▬				
Adj. levels		(**7**)	6			**4**							
$\alpha =$		⇑2											
$\beta =$						⇑3							
Position i, i'	00	01	02	03	04	05	06	07	08	09	10	11	12
$p_3 = (7, 6)$								▬	▬	▬	▬	▬	▬
Adj. levels						(**5**)		3		**4**			
$\alpha =$						⇑3							
$\beta =$										⇑4			

Figure 4.4: Conversion of jump lists of matching statistics to lcf values. There are four jumps, indexed 1–4, at positions 0, 1, 5, and 9 to levels 4, 7, 5, and 6, respectively: $J = ((0, 4), (1, 7), (5, 5), (9, 6))$. There are three probe candidates. For p_1, we have $\alpha = 1$, $\beta = 3$, and the adjusted jump levels within p_1 are 4, 5, and 1 by Lemma 4.19. The lcf value is the maximum 5 of these levels. For p_2, the jump set begins to the left of the probe ($\alpha = 2$); the jump level at the left end of p_2 is reduced to 6 (see Lemma 4.19).

As we move through the jump list of each collection c, the unspecificity U for each p_j from (3.5) and the appropriate LCF statistic are incremented. Of course, the collection to which the current target is associated is skipped, so that U is not erroneously increased. It is assumed that initially U and all LCF statistics are zero. Let δ denote the difference between the probe length $|p_j|$ and the lcf or lcf* value. Then for $\delta < \Delta$,

$$\begin{aligned} U(p_j \,|\, T; \Delta) &\leftarrow U(p_j \,|\, T; \Delta) + e^{-b \cdot \delta + \zeta}, \\ \mathrm{LCFS}(p_j \,|\, T; \Delta)_\delta &\leftarrow \mathrm{LCFS}(p_j \,|\, T; \Delta)_\delta + 1. \end{aligned}$$

If a non-integer value of lcf* is used to obtain δ, it can be rounded up to obtain a conservative integer index for LCFS.

4.6 Trading Speed for Precision

The method that we presented in Sections 4.3 through 4.5 is sufficiently fast even for large scale probe design projects in practice. It also has additional desirable features.

- Jump lists of matching statistics need only be computed once for all target-collection-pairs and can then be stored on disk. If the probe selection criteria are later changed (e.g., to a different length range), only the part of Section 4.5 must be re-done.

- All sufficiently large lcf values ($\mathrm{lcf}(p, T_c) \geq R^0_{\min}$) are computed before they are aggregated into the LCF statistics. This allows, for example, to obtain detailed information about which collections c are almost certain to cross-hybridize (say, when $\mathrm{lcf}(p, T_c) \geq |p| - 3$).

- It is also the individual knowledge of lcf values that allows their combination with lcf^1 to obtain adjusted values lcf^*.

If time is at a premium, however, the individual lcf values may not be of interest, and it is more important to obtain the worst case or *maximal* lcf value for each probe candidate quickly. This goal can be achieved by a minor modification of the bucket scan algorithm, but it yields a less robust unspecificity measure. On the other hand, if precise estimates are more important than short running times, the jump list processing in Section 4.5.2 can be modified to directly estimate the hybridization probability β from the Gibbs energy $\Delta_r G^\circ$ by (2.5). We now provide details on these two variations of the basic probe selection method.

Finding the maximal lcf value. If the two-dimensional array $\mathtt{MS}[i][c]$ is replaced by a one-dimensional vector $\mathtt{MS}[i]$ — no jump lists are required in this case because the amount of data is reduced by a factor of c—, the bucket scan algorithm is easily modified to return the maxima of matching statistics over all collections. It is important, though, that the target's own collection c_s is not considered, because otherwise the returned maximum would always be the maximum possible value $R_{\max}$.

The only required changes to the algorithm in Figure 4.3 are to replace each of the following lines.

- Lines 2 and 6 become: **if** $(\mathtt{cl}[r] \neq c_s)$ **then** $\mathtt{MS}[i] \leftarrow \mu$

- Line 10 becomes: **if** $(\mathtt{MS}[i] < \mu)$ and $\mathtt{cl}[r] \neq c_s)$ **then** $\mathtt{MS}[i] \leftarrow \mu$

For a probe p starting at position i in target s, we obtain the maximal longest common factor across collections $\ell(p \,|\, T) := \max_{c \neq c_s} \mathrm{lcf}(p, T_c)$ (within the range of interest) from maximal matching statistics across collections stored in $\mathtt{MS}[i]$ (within the range

of interest) from the relationship

$$\begin{aligned}
\ell(p \,|\, T) &= \max_{c \neq c_s} \mathrm{lcf}(p, T_c) \\
&= \max_{c \neq c_s} \max_{k=0..L-1} \mathrm{ms}_k^{p|T_c} \\
&= \max_{c \neq c_s} \max_{k=0..L-1} \min\{\mathrm{ms}_{i+k}^{s|T_c}, L-k\} \qquad \text{(by Lemma 4.18)} \\
&= \max_{k=0..L-1} \min\{(\max_{c \neq c_s} \mathrm{ms}_{i+k}^{s|T_c}), L-k\} \\
&= \max_{k=0..L-1} \min\{\mathtt{MS}[i+k], L-k\}.
\end{aligned}$$

The main advantage of this method lies in its considerably reduced memory requirements: Only one instead of C vectors of matching statistics are stored for each target; jump lists are not required. The unspecificity (3.5) of probe p_i becomes

$$U(p_i \,|\, T) = \exp(-b \cdot [|p_i| - \ell(p_i \,|\, T)] + \zeta);$$

it cannot discriminate between a single or several transcripts with long lcf values.

Using jumps to compute energy-based probe unspecificity. The LCF-based unspecificity measure (3.5) has the inherent disadvantage that all common factors of the same length are treated equally, independently of their sequence composition.

Recall that we arrived at (3.5) after several approximations. The most principled definition would have been to set $U(p_i \,|\, T) = \sum_{j \in T \setminus T(i)} A_{ij}$. In (3.1), we approximated A_{ij} by the hybridization probability β_{ij}, which we obtained by (2.5) from the Gibbs energy of the longest common factor of p_i and transcript t_j. Only in (3.3) and (3.4) we replaced the specific energy contribution of a dinucleotide by an average value.

Given enough computational resources, we can take one step back and use the unspecificity definition $U'(p_i \,|\, T) = \sum_{j \in T \setminus T(i)} \beta_{i,j}$ from (3.1), where we now express β_{ij} through the difference $\Delta\Delta G_{ij}$ between the standard Gibbs energy of hybridization $\Delta_\mathrm{r} G_i^\circ$ of p_i to its intended targets and the energy $\Delta_\mathrm{r} G_{ij}^\circ$ of p_i to transcript t_j. In other words, $\Delta\Delta G_{ij} := \Delta_\mathrm{r} G_{ij}^\circ - \Delta_\mathrm{r} G_i^\circ > 0$ is a more precise measure of the stability difference between the intended hybridization and the potential cross-hybridization with transcript j than the averaged $\delta \cdot \gamma$ with $\delta = |p_i| - \mathrm{lcf}(p_i, t_j)$ in (3.2). It follows that we can replace (3.5) by

$$U'(p_i \,|\, T) = \sum_{j \in T \setminus T(i)} \frac{1}{1 + \exp[\Delta\Delta G_{ij}/(R\tau) - \zeta]} \;\leq\; \sum_{j \in T \setminus T(i)} \exp(-\Delta\Delta G_{ij}/(R\tau) + \zeta),$$

where R, τ, and ζ are as in Section 3.2.

It remains to compute $\Delta\Delta G_{ij} = \min\{\Delta_r G^\circ(s) : s \lhd p_i \text{ and } s \lhd t_j\} - \Delta_r G^\circ(p_i)$ efficiently. We evaluate the Gibbs energy only for perfect matches, but the longest match need not be the most stable one: A long AT-rich common factor is less stable than a slightly shorter GC-rich common factor. Because of monotonicity, however, the "most stable common factor" s minimizing $\Delta_r G^\circ(s)$ must still begin at a *jump* in the matching statistics $\mathrm{ms}^{p_i|t_j}$. Therefore we only need to modify the jump list processing (Section 4.5.2) slightly: Instead of using the $[i, L]$-jump-sets to determine longest common factors, we use them to minimize $\Delta_r G^\circ(s)$ by evaluating the Gibbs energy of the longest perfect match at every jump. Computational studies show that with the energy-based unspecificity evaluation, probe selection takes 2 to 3 times as long as with LCF-statistics-based selection.

4.7 Probe Selection for Saccharomyces cerevisiae

We applied probe selection based on longest common factor statistics and on stability difference ($\Delta\Delta G$) estimates to all 6342 open reading frames (ORFs) of *Saccharomyces cerevisiae* (baker's yeast) and their reversed complements (12684 sequences, about 18 Mbp). All computations were carried out on a Compaq AlphaServer ES45 with four 64-bit 1 GHz processors and 27 GB of main memory (only 250 MB were used). Programs were compiled with the native C compiler with all speed optimizations.

The preprocessing took 38 minutes (35 seconds for the creation of the enhanced suffix array, and 37 minutes for the jump lists of all matching statistics). The values of $18 \cdot 10^6 \cdot 12684 \approx 240 \cdot 10^9$ matching statistics are evaluated during this time. Storing these in the straightforward way using 1 byte per value would take $2 \cdot 240$ GB for ms^0 and ms^1. Our jump list format uses 3 bytes per jump, and we only store jumps to levels of at least 10 for ms^0 and to at least 13 for ms^1. The resulting file requires only 2.4 GB, only 0.5% compared to the naive approach. This corresponds to about $800 \cdot 10^6$ jumps overall. The estimate E_L^+ from (4.4) predicts $2.3 \cdot 10^6$ jumps to levels 10 or above for exact matching statistics ms^0, and $1.4 \cdot 10^6$ jumps to level 13 or above for ms^1. The observed number of jumps exceeds the predicted number by a factor of more than 200, because the yeast genome is not a random text: Local similarities are frequent, and jumps to medium levels occur more often than in random texts. Nevertheless, the use of jump lists saves 99.5% of the required space.

We selected oligos under the following constraints: length between 19 and 21 bp; standard Gibbs energy of intended hybridization $\Delta_r G^\circ$ between -19.8 and -18.8 kcal/mol at 45°C (318 K) and a salt concentration of 0.075 M NaCl. We attempted to provide the 20 best probes for every ORF and its reversed complement. In 514 out of 12684 cases, even the best probe is not satisfactory, either because of lack of suitable candidates in very short ORFs, or because of high similarity to other sequences.

The longest common factor based selection took 138 minutes, and the selection based on $\Delta\Delta G$ took 317 minutes. When the oligo candidates are restricted to a fixed length of 20, the times drop to 57 minutes and 127 minutes, respectively. The energy-based approach is slower by a factor of only 2.0 to 2.5, but is naturally more accurate.

Chapter 5

Unique Probe Selection

In Chapters 2 through 4, we have described in a bottom-up way how the specificity of probe candidates can be efficiently estimated by a sequence-based quantity, the longest common factor statistics; see Equation (3.5). We have also discussed ways of trading computational speed for precision (Section 4.6). Of the factors that influence the affinity coefficient A_{ij} in (2.2), we have mainly discussed the hybridization probability factor β_{ij}. In summary, we have proposed solutions to Problems 2 to 4 from the introduction, but we have not yet said how to solve the main probe selection problem (Problem 1) as a whole, i.e., how to find a good *set* of probes, when several candidates are pre-selected and ranked by their (un)specificity. In the present chapter, we thus define criteria and propose methods for *probe set* selection (Sections 5.1 and 5.2, respectively). The chapter concludes with the presentation of a general workflow description for unique probe selection (Section 5.3). Our software PROMIDE is discussed in more technical terms in Appendix A.

5.1 Criteria for Probe Set Selection

From the discussion in Section 2.2 and in particular from Example 2.2, we conclude that to minimize variance in expression level estimates in the case of unique probes, affinity coefficients should have the same order of magnitude and approximately the same distribution across the probes of all targets. It would be even better if they were as constant as possible. Specificity selection has ensured that the hybridization probability factor β_{ij} satisfies $\beta_{ij} \gg 0$ for the intended target j of probe i and $\beta_{ij} \approx 0$ for all other transcripts j in (2.2). We face the dilemma that we cannot exactly quantify the contribution of the other factors in (2.2), but a reasonable approach is to avoid probes spanning target positions that show a risk of producing extreme factors.

In particular, assuming sufficient specificity, selected probes should

(a) be located in a stretch of sequence with "normal" base composition to avoid excessively many or few bound biotin molecules in the fragments that hybridize

to it (modeled by the γ-factor in (2.2)). This requirement can actually be addressed before the probe selection process by pre-filtering target sequences for low complexity regions and masking problematic regions.

(b) be located close to the 3'-end of the target to ensure a constantly strong signal, even under slight variations of the biotin-labeling step (the ρ-factor in (2.2); see also Section 1.3.2). In principle it suffices to keep targets reasonably short. A problem arises when the transcription stop site is not known; resources like GeneNest (Haas et al., 2000) are helpful in this situation. An alternative to cutting the target at a fixed position is to integrate the 3'-end distance into a position-specific preference factor (see also the next requirement).

(c) not be selected from a part of the target where the cRNA fragments are likely to bind to other cRNA fragments instead of the intended probe. This requirement corresponds to the σ''-part of the σ-factor in the discussion following (2.2). We introduce a position-specific *preference factor* f_k to model both the ρ- and σ''-factors in (2.2) of probes that span position k. Details are given below.

(d) not be highly self-complementary (the σ'-part of the σ-factor in the discussion after (2.2)). Probe self-complementarity is only a problem in extreme cases, and it suffices to check for self-complementarity at the latest step, i.e., when post-processing the final candidate list. This procedure is described in Section 5.2.

As before, we assume that the effect of erroneous probes on the chip can be ignored (not all probes are synthesized perfectly and thus some may have the wrong sequence; we present a method for minimizing one source of error in Chapter 8). We also assume that cRNA fragments do not compete for probes because probes are in excess. The converse effect that probes compete for cRNA fragments is kept small by distributing the probes for the same target uniformly across the chip. Additionally, and to increase the general robustness of the design, we may add the following item to the list of requirements: Selected probe should

(e) be uniformly distributed over the target sequence. Indeed, high-quality probe candidates tend to cluster together: If a probe p starting at position k satisfies all criteria, a probe starting at position $k+1$ or $k-1$ is likely to share some of p's properties. Candidate post-processing therefore should include a *de-clustering* step; it is described in Section 5.2.

Computing preference factors. Every position in a target sequence s_j receives a preference factor f that incorporates the distance from the 3'-end and models the risk that the base at this position form a basepair with another base in a molecule *of the same target type* j.

- The position-specific factor ρ should be approximately equal to 1 for positions less than 1000 nt away from the 3'-end and then decrease exponentially fast for larger distances d. A suitable function is given by

$$\rho(d) = \frac{1}{1 + \exp((d - 1000) \cdot \mathit{drop}/100)},$$

  which depends on a tunable parameter *drop* that controls how quickly ρ drops to the right of 1000. Figure 5.1 shows the behavior of $\rho(d)$ for different values of *drop*. The choice of $\mathit{drop} = 2$ appears reasonable, because it marks probes that are 1500 nt away from the 3'-end as "borderline" ($\rho \approx 10^{-4}$).

- We need to estimate the probability that the base at a given distance d from the 3'-end is available for hybridization with a probe on the chip, i.e., the probability that it is unpaired in a sufficiently stable secondary structure that the cRNA target forms with itself. An estimate of this probability is given by the so-called "S-num value" introduced by Jaeger et al. (1989, 1990). Given several predicted optimal and suboptimal foldings of the target, S-num(d) is defined as the fraction of foldings in which the base at distance d is unpaired. The estimations can be carried out, for example, with the MFOLD software (Zuker et al., 1999; Zuker, 2003).

Combining these pieces of information, we define the preference factor for distance d from the 3'-end as

$$f(d) := \rho(d) \cdot \text{S-num}(d).$$

Avoiding unavailable target bases. The preference factor does not contain the risk that a target base forms a basepair with a base in a target molecule *of a different target type*. Let us call such a target base *unavailable*. The following argument shows that special consideration of unavailable bases is unnecessary and can be accounted for by a simple trick already during LCF statistics computation.

The idea is to include the reverse complement of each target sequence (i.e., the antisense sequence) together with the target sequence (sense) in the same collection and compute LCF statistics against such extended collections. Assume that a certain target sequence j has unavailable bases because the antisense fragments tend to bind to bases in an antisense fragment of another target sequence j'. This means that the sequence of the antisense fragment in j' is identical to the sense sequence of j and hence to the probe sequence. Therefore this probe candidate receives a high lcf value with the collection of j', provided that the antisense sequence of j' has been included.

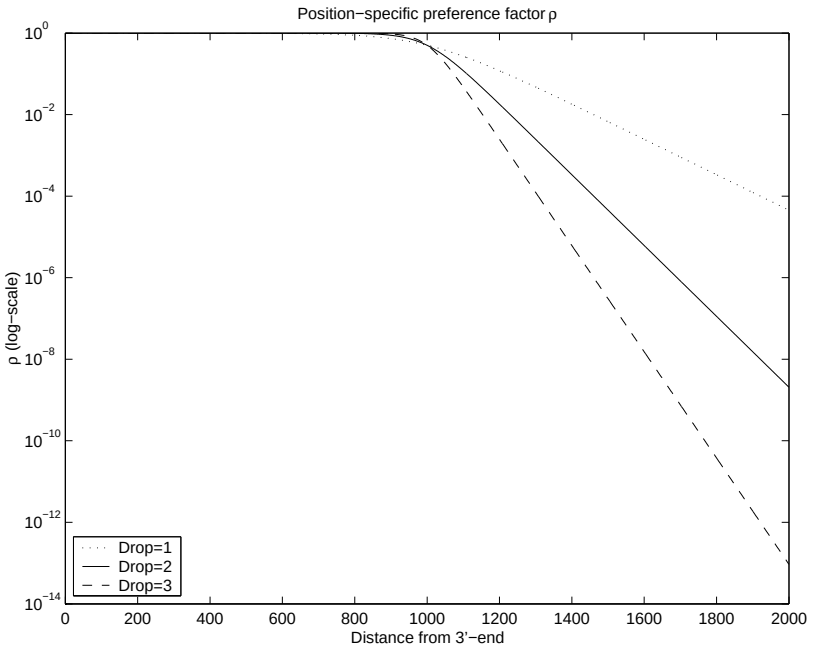


Figure 5.1: Position preference $\rho(d)$ as a function of the distance from the 3'-end d.

5.2 Obtaining the Final Probe Set

After candidate selection, we have a list of probes candidates $P = (p_i)_{i=1..m}$. Each probe is specified by its starting position in the target, expressed as the distance d_i from the 3'-end, and its length ℓ_i, i.e., $p_i = (d_i, \ell_i)$.

Each probe comes with a hybridization probability estimate $\beta_i \approx 1$ (with respect to its intended target), an unspecificity measure U_i (i.e., total cross-hybridization risk with respect to unintended targets). At this time, we remove probe candidates with $U_i \geq \overline{U}$, where $\overline{U}$ is the unspecificity threshold. For example $\overline{U} = 10^{-4}$ means that modulo approximations, the total probability that a probe candidate cross-hybridizes to any non-intended target is four orders of magnitude smaller than the hybridization probability to its intended target, assuming equal target concentrations. Less specific probes for which $U > \overline{U}$ are not further considered as candidates.

Probe p_i also has its preference factor $f_i \leq 1$ that is an aggregate measure (e.g., the median) of all preference factors $f(d_i), \ldots, f(d_i + \ell_i - 1)$ at positions spanned by p_i. At this time, we also assess the self-complementarity of p_i as the hairpin formation probability h_i, defined via the energy of the most stable secondary structure of p_i that contains a stem-hairpin-loop (see Figure 1.2). It is computable with MFOLD, for example. Spitzer (2003) provides a thermodynamically motivated alignment algorithm to estimate h_i.

For the purposes of candidate ranking, we compute a *badness* measure B_i of each candidate that includes the unspecificity U_i, the hairpin formation probability h_i, and the preference factor f_i. Excellent probes candidates have $U_i \approx 10^{-6}$, $h_i \approx 0$, and $f_i \approx 1$. Bad candidates have $U_i \approx 1$, $h_i \approx 1$, and $f_i \approx 0$. Therefore the badness of probe i is defined as

$$B_i := \frac{U_i}{f_i \cdot (1 - h_i)},$$

because it avoids that good f_i and h_i values decrease U_i. It follows that $B_i \geq U_i$.

We assume that the candidate list is sorted by badness $B_1 < B_2 < \cdots < B_m$. The task is to select a set of M probes from the candidates. The following method takes care of the remaining probe distribution considerations with a de-clustering approach. The method maintains and dynamically updates a modified badness value B'_i for each probe, which is initially set to B_i. Probes are successively removed from the initial candidate list P and placed in the final selection S until M probes have been selected or we run out of candidates. De-clustering of candidates is achieved by increasing the modified badness value for all probes that begin close to probes that are already in the selection.

Final Probe Set Selection by De-Clustering

0. $S \leftarrow ()$; $P = (p_i)$; $B'_i \leftarrow B_i$ for all i
1. **while** ($|S| < M$ and $|P| > 0$)
2. Find the probe p_i in P with minimal B'_i-value
3. Update B'_i:
4. Determine $k \in S$ be such that $|d_i - d_k| \rightarrow \min$
5. $B'_i \leftarrow B_i + \mathcal{B}(|d_i - d_k|)$
6. **if** B'_i changed **then** go back to step 2.
7. **warn if** B'_i exceeds a threshold $\overline{B}'$
8. Remove p_i from P; add p_i to S.
9. Now S contains the selected probes ranked by their final B'_i.

The function $\mathcal{B}(\delta)$ determines as how much worse a probe candidate is considered for ranking purposes when a nearby probe (δ nt away) has already been selected. The function values must be small in absolute magnitude in order not to prefer low-quality probes too much and drop to zero quickly for increasing distances. An obvious choice is $\mathcal{B}(\delta) := 10^{-\delta}$. Warnings are given when the modified badness exceeds a pre-defined threshold $\overline{B}'$, such as 10^{-1}. It must then be decided whether these probes should be used to obtain a full set of M probes or whether it is better to discard them and accept a smaller set. This choice will depend on the intended application and the size of the project.

For small-scale projects, users might want to hand-inspect examine all reasonable candidates with their B_i values or individual U_i, h_i and f_i values anyway. In larger

projects, picking the M probes with the lowest B_i values may suffice, even if some of them overlap.

To estimate the affinity coefficient of $p_i \in S$ with respect to its intended target, we may assume that $a_i \approx \beta_i \cdot (1 - h_i) \approx 1$ because of the selection criteria. Refined estimation methods were discussed in Chapter 2.

5.3 Workflow

A typical session of probe selection for gene expression proceeds as enumerated below. Note that, while steps 1. and 2. are essentially independent of the probe selection algorithm, they are important and complex tasks themselves and may in fact take more time than the rest of the process. The other steps are relatively specific to the longest common factor approach and to the PROMIDE software package.

1. Obtain the transcriptome of the organism, e.g., as a file in FASTA format where each transcript has its unique identifier. Choose and specify transcript collections if desired; these are additionally marked by unique collection identifiers.

2. Specify target sequences. Targets should be kept relatively short and restricted to the vicinity of the 3'-end of transcripts (about 1000 nt). Not all transcripts need to be targets, but each target is usually obtained from a single transcript collection to which it is then associated. For example, a target might be a part of the consensus sequence of a strongly homologous gene family. Heise (2003) describes a way to generate targets containing only probe candidates that occur in all members of a given sequence family.

3. Process target sequences for quality. Mask sequence parts of low complexity or highly biased sequence composition to avoid extreme γ-factors in (2.2). Compute the preference factor $f(d)$ for each position that comprises a factor for the 3'-end distance d and for the risk of intra-target basepair formation at this position.

4. Create the enhanced suffix array (Section 4.2) of the entire transcriptome. If desired, the reversed complements of the target sequences can be included to ensure that no probes at unavailable target bases are selected (see paragraph "Avoiding unavailable target bases" in Section 5.1). PROMIDE uses the external program `mkvtree`, part of Stefan Kurtz's `vmatch` software package, to build the array.

5. Compute jump lists of matching statistics of all targets against all transcript collections (Sections 4.3 and 4.4). This concludes the pre-processing steps. Note that, up to this point, it was not necessary to specify any desired probe properties.

6. Specify the experimental conditions (the "hybridization parameters" θ that enter the calculations are the temperature and the sodium ion concentration) and the desired probe properties, i.e., their length range and Gibbs energy range under the given conditions). It requires some experimentation to find combinations that work well. We suggest to obtain reasonable values from experimental protocols. For the GeneChip® arrays, the probe length is 25, the temperature is 318 K, and the Na^+-concentration in the washing phase is 0.1 M, such that $\ln([Na^+]) = -2.3$. For average probes of average `GC`-content, typical $\Delta_r G^\circ$ values are around -23.2 kcal/mol. A program that computes the average Gibbs energy of random probes with given length and `GC`-content under specified hybridization conditions was developed by Gräfe (2003). It makes sense to allow variable length probes, but to ensure a relatively tight range of stability.

7. For each target, compute the longest common factor statistics and unspecificity values (Section 3.2) of all probe candidates that satisfy the specified criteria.

8. For each target, select a final probe set of desired size (Section 5.2). Spitzer (2003) has written a program that selects a final probe set from candidates generated by PROMIDE.

If several probe sets with different properties are desired, only the last two steps need to be repeated for each set. This is an advantage of separating the computation of matching statistics from the computation of longest common factors in the PROMIDE implementation.

Chapter 6

Non-Unique Probe Selection and Signal Decoding

6.1 Preliminaries

Experience shows that it is difficult to find sufficiently unique probes for transcripts from families with many similar sequences. For example, for some of the yeast ORFs (Section 4.7), we did not find unique probes because of their high similarity to other ORFs. Kaderali and Schliep (2002) report similar problems on a different set of sequences. Therefore, it is reasonable to allow *non-unique* probes, i.e., probes that potentially hybridize to more than one target in a controlled way.

Intuitively, we expect that designing a chip with non-unique probes should be easier than designing a chip with unique probes. However, we have to state clearly what we mean by non-unique probes.

Definition 6.1 (Non-unique probe). Let $T = (T_1, \ldots, T_C)$ be a transcriptome with transcript collections T_c ($c = 1, \ldots, C$). Let $S = (s_1, \ldots, s_C)$ be the set of target sequences, where s_c is the (possibly empty) target sequence for collection c. It is assumed that every substring of s_c not containing wildcards or separators is also a substring of T_c. Let θ denote hybridization conditions, and let $a(p, t; \theta)$ denote the affinity coefficient between p and t under θ. A probe p_i is called a *non-unique probe* for (S, T) if there exist an index set[1] $T(i) \subset \{1, \ldots, C\}$ with $|T(i)| \geq 1$ (the intended targets[2]) and an affinity threshold $\varepsilon > 0$ (say, $\varepsilon \approx 0.1$), such that

- $a(p_i, s_j) \geq \varepsilon$ for all $j \in T(i)$, and
- $U(p_i \,|\, T) \ll \varepsilon$, where $U(p_i \,|\, T)$ was defined in Equation (3.5).

[1] The index set $T(i)$ bears no relation to the transcript collection T_c.

[2] It follows that non-unique probes are in fact *not necessarily unique* probes and that unique probes are included among non-unique probes with $|T(i)| = 1$.

Note the asymmetry in the definition: A non-unique probe must show a consistent signal for a set of *targets*, but no signal for the remaining set of *transcripts* (or collections). This is a safety precaution: Suppose that there are two transcript collections T_1 and T_2, each with a single transcript. Suppose further that probe p occurs near the 3'-end of T_1, so it is part of target s_1, but also occurs far away from the 3'-end of T_2, so it is not part of target s_2. Depending on the parameters of the IVT labeling reaction, T_2 may or may not contribute a signal to p in this case. We would not know how to interpret the signal of p: Should it be interpreted as a sum of expression levels of T_1 and T_2, or does it only represent the expression level of T_1? The correct interpretation may vary with the labeling protocol and cannot be determined in advance. However, a consistent signal over all intended targets is required. Additionally, while the affinity coefficients may differ for the same probe from target to target, we must be able to separate the intended signal from the remaining noise. For these reasons, non-unique probes are not much more frequently found than unique probes.

Experience shows that for quantitative expression analysis, non-unique probes are useful mainly to discriminate between members of highly homologous gene-families where not sufficiently many unique probes are found for each target. In particular, non-unique probes are useful for inferring mixtures of splice variants of the same gene (Wang et al., 2003).

Non-unique probes are more versatile for qualitative analyses, i.e., for detecting the presence or absence of certain transcripts or genomic sequences. Possible applications are SNP detection (Kozal et al., 1996; Cutler et al., 2001), species barcoding (Hebert et al., 2003a), e.g., virus subtyping (Rash and Gusfield, 2002; Wang et al., 2002), and detection and identification of pathogens in water samples (e.g., Loy et al., 2002).

In both quantitative and qualitative analysis, probe candidate selection proceeds in the same way and needs only to be slightly modified from the selection process for unique probes; it is summarized in the paragraph below. The final *probe set* selection, however, becomes more complex and is differently motivated for quantitative and qualitative analysis. The additional complexity arises from the fact that probe signals do not correspond to single expression levels, but are mixtures of these. Therefore probe set selection must ensure the *decodability* of the signals into expression levels. Further details follow in Sections 6.2 and 6.3.

Probe candidate selection. As before, for each probe candidate p_i, the longest common factor statistics $\mathrm{LCFS}(p_i \,|\, T)$ are computed. Suppose it is found that $\mathrm{LCFS} = (k, 0, 0, 0, 0, 0, 0, u, v, w, \ldots)$ with $k \geq 1$ and $u, v, w, \cdots \geq 0$. This indicates that the probe candidate is relatively specific for a certain set of k transcripts (for unique probes, we would require that $k = 1$). Now it additionally remains to check that p_i occurs indeed in k different targets (and that these targets are associated to the same k collections T_j with $\mathrm{lcf}(p_i, T_j) = |p_i|$). This could be accomplished with an

additional suffix array for all targets, or by simpler methods such as a hash-table or a Boyer-Moore search. Since LCFS-filtered non-unique probe candidates are much rarer than all potential probe candidates, time efficiency is not primarily important in this step.

Let $\sigma(i)$ denote the number of different targets with an occurrence of p_i. If the verification shows that $\sigma(i) < k = \mathrm{LCFS}(p_i \,|\, T)_0$, this is an indication that p_i should not be chosen because it could potentially give a signal for k different transcripts, but is not guaranteed to. Of course, this checking step can be skipped if $S = T$. Note that if the target sequences are properly chosen, it should not happen that $\sigma(i) > k$ because it is assumed that each substring of a target also occurs in the associated transcript collection.

Experience and the hybridization probability computations in Section 2.3.3 show that a good signal is still obtained at probe p_i with targets t for which $\mathrm{lcf}(p, t) = |p| - 1$. Therefore one can relax the above requirements slightly and use the following LCF-statistics-based definition of non-unique probes.

Definition 6.2 (Non-unique probe, LCFS-based). Let $T = (T_1, \ldots, T_C)$ be a transcriptome with collections t_c $(c = 1, \ldots, C)$.[3] Let $S = (s_c)$ be the set of target sequences, where s_c is the target sequence for collection c. A probe p_i is called a *non-unique probe* for (S, T) if

$$\mathrm{LCFS}(p_i \,|\, T) = (\sigma_0(i), \sigma_1(i), \overbrace{0, \ldots, 0}^{\mu}, u, v, \ldots)$$
$$\mathrm{LCFS}(p_i \,|\, S) = (\sigma_0(i), \sigma_1(i), \underbrace{0, \ldots, 0}_{\mu}, u', v', \ldots)$$

with $\sigma_0(i) > 0$, $\sigma_1(i) \geq 0$, $u, u', v, v', \cdots \geq 0$, and μ is sufficiently large ($\mu = 6$ or $\mu = 7$ is safe in practice). Note that it is required that $\sigma_0(i)$ and $\sigma_1(i)$ take the same value in both LCF statistics.

The candidates determined in this way form an affinity matrix (A_{ij}) with $0 \ll A_{ij} \leq 1$ for $j \in T(i)$ and $A_{ij} \approx 0$ for $j \notin T(i)$. Here $T(i)$ denotes the set of intended targets for probe i ($\mathrm{lcf}(p_i, T_j) \geq |p_i| - 1$). Similarly we write $P(j)$ for the set of probes hybridizing to target j.

As with unique probes, we find that good non-probe candidates frequently occur in clusters in the target sequences; probes in the same cluster tend to have the same properties. If this happens and enough candidates are available, only one candidate from each cluster is selected using the de-clustering method described in Section 5.2 in order to prune the candidate matrix.

[3] In some applications, t_c may in fact be a whole (virus) genome, so T is not a transcriptome but a set of genomes. We will stick to the original terminology, however.

A final probe set is obtained by choosing rows of this matrix according to certain optimality criteria. The exact criteria for quantitative and qualitative analysis are discussed in Sections 6.2 and 6.3, respectively.

6.2 Design and Decoding for Quantitative Analysis

From the m probe candidates whose affinity values form the m rows of the affinity matrix A, we would like to select at most $\mu \leq m$ rows. We write H for the *hybridization matrix*

$$H_{ij} := \mathbb{I}_{\{A_{ij} \not\approx 0\}} = \begin{cases} 1 & \text{if } j \in T(i), \\ 0 & \text{otherwise.} \end{cases}$$

In other words, H is a binary approximation of A.

We denote the index set of the chosen rows by D for *design*[4]. We have $D \subset \{1, 2, \ldots, m\}$ and $|D| \leq \mu$. Let A^D and H^D denote the matrices obtained from A resp. H by removing all rows whose index is not in D.

The requirements on D are that the equation $y = A^D \cdot x$ must be stably and robustly solvable for the n expression levels x, given the $|D|$ probe signals y. More precisely, we require the following properties.

- Each target is covered by a minimum number $\mathcal{M} \geq 1$ of probes, i.e., $\sum_i H^D_{ij} \geq \mathcal{M}$ for all j.
- On average, each target is covered by $\mathcal{A}$ probes, i.e., $\sum_{i,j} H^D_{ij} \geq n \cdot \mathcal{A}$. It follows that in a valid design, on average each of the $|D|$ probes hybridizes to at least $n\mathcal{A}/|D|$ targets.
- The minimization problem $\|y - A^D \cdot x\| \to \min$ has a unique solution, implying that the matrix A^D has full rank n and that necessarily $|D| \geq n$.
- Small errors in A or in y do not strongly influence the solution x.

The last requirement is particularly important for a robust design; it is formalized by the notion of the *condition* of a matrix. Often the condition is only considered for square matrices; here we require a more general notion that also applies to rectangular $m \times n$ matrices. It can be defined via the singular value decomposition and the pseudoinverse A^- of a matrix A.

[4]We are talking about chip designs, not combinatorial designs.

6.2.1 Matrix condition numbers

We denote the set of real-valued $m \times n$ matrices by $\mathbb{R}^{m\times n}$, the transpose of a matrix $A \in \mathbb{R}^{m\times n}$ is denoted by $A^\mathsf{T} \in \mathbb{R}^{n\times m}$, and the $n \times n$ identity matrix is written as I_n.

Definition 6.3 (Singular value decomposition). Suppose that $A \in \mathbb{R}^{m\times n}$ has rank $r \leq \min\{m, n\}$. A *singular value decomposition* (SVD) of A consists of a triple (U, Σ, V) such that $A = U\Sigma V^\mathsf{T}$, $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal ($U^\mathsf{T}U = UU^\mathsf{T} = \mathrm{I}_m$; $V^\mathsf{T}V = VV^\mathsf{T} = \mathrm{I}_n$), and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min\{m,n\}}) \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix, where $\sigma_1 \geq \cdots \geq \sigma_r > 0 = \sigma_{r+1} = \cdots = \sigma_{\min\{m,n\}}$ are called the *singular values* of A.

One can show that the singular values are uniquely determined by A, and that they are equal to the square roots of the eigenvalues of the positive semidefinite matrices $A^\mathsf{T}A$ or AA^T (Golub and van Loan, 1996). The singular value decomposition provides a convenient way to define the *pseudo-inverse* A^- of A which can be used to directly solve the over-determined system $y = A \cdot x$ as a least squares minimization problem $\|y - A \cdot x\|_2^2 = \sum_i (y_i - \sum_j A_{ij} \cdot x_j)^2 \to \min$.

Definition 6.4 (Pseudo-inverse). Let A be an $m \times n$ matrix of rank $r \leq \min\{m, n\}$ with SVD $A = U\Sigma V^\mathsf{T}$. Define the $n \times m$ matrix Σ^- as the diagonal matrix $\Sigma^- := \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0)$. Then $A^- := V\Sigma^-U^\mathsf{T} \in \mathbb{R}^{n\times m}$ is called the *pseudo-inverse* of A. It has the property that $\|y - A \cdot x\|_2^2 \to \min$ is solved by $x = A^- \cdot y$ (Golub and van Loan, 1996).

Apart from solving the least-squares minimization problem, the pseudo-inverse has many useful properties. One can show, for example that if $y \in \mathbb{R}^m$ is not in the range of A, then $y_A := AA^-y$ is the orthogonal projection of y on the range of A.

Now we are ready to define the condition $\mathrm{cond}(A)$ of a rectangular matrix $A \in \mathbb{R}^{m\times n}$ of full rank n (see also Golub and van Loan, 1996, p.230). We assume that for vectors, $\|\cdot\|$ denotes the Euclidean norm and for matrices, it denotes the associated spectral norm.

Definition 6.5 (Condition). Let A be an $m \times n$ matrix of full rank $n \leq m$, and let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$ be the singular values of A. Then the condition of A is defined as

$$\mathrm{cond}(A) := \sigma_1/\sigma_n.$$

If A does not have full rank n, then $\sigma_n = 0$, and we set $\mathrm{cond}(A) := +\infty$.

The condition measures how strongly changes in the measurement y influence the solution x of the minimization problem $\|y - A \cdot x\| \to \min$, and even how changes in A affect the solution. For example, the following results are known.

Lemma 6.6. *Let A be an $m \times n$ matrix of full rank $n \leq m$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$. Let $y \simeq A \cdot x$ denote that x solves $\|y - A \cdot x\| \to \min$. Let x and Δx be such that $y \simeq A \cdot x$ and $y + \Delta y \simeq A \cdot (x + \Delta x)$. Then*

$$\frac{\|\Delta x\|}{\|x\|} \leq \operatorname{cond}(A) \cdot \frac{\|\Delta y\|}{\|y_A\|},$$

where y_A is the projection of y on the range of A as above. If A varies additionally, such that $y + \Delta y \simeq (A + \Delta A) \cdot (x + \Delta x)$ and $\|\Delta A\| < \sigma_n$, then

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A)\,\|\Delta A\|/\|A\|} \cdot \left[\frac{\|\Delta A\|}{\|A\|}\left(1 + \operatorname{cond}(A)\frac{\|y - y_A\|}{\|y_A\|}\right) + \frac{\|\Delta y\|}{\|y_A\|}\right].$$

More details and pointers to a proof of Lemma 6.6 can be found, for example, in the book by Werner (1992). The definition of the condition of A was based on the Euclidean norm mainly out of convenience because it can be derived from the largest and smallest nonzero singular value of A. However, every pair of vector norms induces a pair of matrix norms, which implicitly defines a condition number. Fortunately, all these norms are equivalent, i.e., within constant factors of each other, in finite dimensional spaces.

As defined, the condition of a matrix focuses on relative errors in the vector norms. This can be problematic if the components of the observations range over several orders of magnitude. Therefore we also consider the *Skeel condition number* of A, which allows componentwise bounds.

Definition 6.7 (Skeel condition number). Let A be an $m \times n$ matrix of full rank $n \leq m$, and let A^- be its pseudo-inverse. We denote by C_A the $n \times n$ matrix given by the product of the element-wise absolute values of A^- and A: $C_A := |A^-| \cdot |A| \geq 0$. The *Skeel condition number* of A is defined as $\operatorname{cond}_{\mathrm{s}}(A) := \max_i \sum_j C_{A;i,j}$ (maximal row sum of C_A).

Lemma 6.8. *Under the conditions of Lemma 6.6, we have*

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq \operatorname{cond}_{\mathrm{s}}(A) \cdot \max_i \frac{|\Delta y_i|}{|y_{A;i}|},$$

where $\|x\|_\infty$ denotes the maximum norm $\|x\|_\infty := \max_j |x_j|$.

A development from first principles leading to Lemma 6.8 can be found, for example, in the textbook by Deuflhard and Hohmann (2002).

The Skeel condition is invariant under row-wise rescaling: $\operatorname{cond}_{\mathrm{s}}(RA) = \operatorname{cond}_{\mathrm{s}}(A)$ for any non-singular diagonal matrix R. In particular, $\operatorname{cond}_{\mathrm{s}}(R) = 1$, whereas $\operatorname{cond}(R) = \max_i |R_{ii}| / \min_i |R_{ii}|$.

The bounds in Lemma 6.8 concern the components of y, but the norm of x. Unfortunately there is no easily computable universal number C with $\max_j (|\Delta x_j|/|x_j|) \leq C \cdot \max_i (|\Delta y_i|/|y_i|)$ independently of x and y.

6.2.2 Condition optimization

Given the significance of the condition number of a matrix, it is desirable to select the design D in such a way that $\mathrm{cond}(A^D)$ or $\mathrm{cond}_\mathrm{s}(A^D)$ is minimized, as this promises that y can be reliably decoded even in the presence of errors in A and measurement errors in y. It should be noted that we use either condition number (which is in both cases based on the concept of the pseudo-inverse) only as a *design principle*. For decoding the observed data y, we would not solve $y = A \cdot x$ in a least-squares sense, but in a more robust way, e.g., by minimizing the sum of absolute errors.

Naturally, the choice of minimizing either $\mathrm{cond}(A)$ or $\mathrm{cond}_\mathrm{s}(A)$ when selecting a design makes a difference. Intuitively, minimizing $\mathrm{cond}_\mathrm{s}(A)$ aims at a block diagonal matrix with entries of possibly different magnitude, while minimizing $\mathrm{cond}(A)$ also aims at entries of equal magnitude. Both objectives are reasonable; and we will follow them in parallel in this section.

We assume that the affinity matrix that consists of all candidates has full rank n, and that the associated hybridization matrix H satisfies the coverage constraints $\min_j \sum_i H_{ij} \geq \mathcal{M}$ and $\sum_{i,j} H_{ij} \geq n\mathcal{A}$.

We let $\mathcal{D} := \{D \subset \{1, 2, \ldots, m\} : |D| \leq \mu,\ \mathrm{rank}(A^D) = n,\ \min_j \sum_i H^D_{ij} \geq \mathcal{M},\ \sum_{i,j} H^D_{ij} \geq n\mathcal{A}\}$ denote the set of admissible designs under the side conditions. It is assumed that $\mathcal{D}$ is not empty; otherwise we have to increase μ or decrease $\mathcal{M}$ or $\mathcal{A}$. The combinatorial problem is to minimize $\mathrm{cond}(A^D)$ or $\mathrm{cond}_\mathrm{s}(A^D)$ among all $D \in \mathcal{D}$.

A greedy heuristic. To our knowledge, the *condition optimization problem by row selection* has not been posed in the mathematical literature before, although other condition optimization problems have been examined by Elsner et al. (1994, 1995). It is a difficult problem because the singular values of two matrices A^D and A^{D-i}, where $D - i := D \setminus \{i\}$, are not related in an obvious way (although inequalities could be derived from the Courant-Fischer min-max Theorem), and also because the landscape of admissible designs potentially has many local minima. In spite of these difficulties, we propose a greedy heuristic to obtain a good admissible design.

The procedure is shown in Figure 6.1. It starts with a full design (which is assumed to satisfy the coverage constraints) and iteratively removes a single row from the design. The row i^* is chosen in such a way that the remaining design $D - i^*$ still satisfies the coverage constraints and that $\mathrm{cond}(A^{D-i^*})$ is minimized among all $\mathrm{cond}(A^{D-i})$ for $i \in D$ (lines 5–6). If the resulting design is admissible (i.e, if it is small enough), it is

Greedy Condition-based Design
Input: An $m \times n$ affinity matrix A and hybridization matrix H
Note: Here cond can be either cond or cond_s.
1. $D \leftarrow \{1, 2, \ldots, m\}$
2. $B \leftarrow \emptyset, \quad C \leftarrow +\infty$
3. **while** $(|D| > n)$
4. $\quad c \leftarrow +\infty, \quad i^* \leftarrow 0$
5. $\quad$ **for each** $i \in D$
6. $\quad\quad$ **if** $(\min_j \sum_{i'} H^{D-i}_{i'j} \geq \mathcal{M})$ **and** $(\sum_{i',j} H^{D-i}_{i'j} \geq n\mathcal{A})$ **and** $(\text{cond}(A^{D-i}) < c)$ **then** $c \leftarrow \text{cond}(A^{D-i}),\ i^* \leftarrow i$
7. $\quad$ **if** $i^* = 0$ **then break**
8. $\quad D \leftarrow D - i^*$
9. $\quad$ **if** $(|D| \leq \mu)$ **and** $(c < C)$ **then** $B \leftarrow D,\ C \leftarrow c$
10. **if** $(B = \emptyset)$ **then** $B \leftarrow D,\ C \leftarrow c$
Output: Best found design B with condition $C = \text{cond}(A^B)$

Figure 6.1: Greedy heuristic for condition-based design.

compared against the current best admissible design B (line 9). This is repeated until the design size equals the number of targets (line 3) or no smaller design satisfying the coverage constraints can be found (line 7). In line 10, if $B \neq \emptyset$, then B contains an admissible approximation to the best design. If $B = \emptyset$, the current design D is taken as a non-admissible approximation: D then contains more than μ probes, but no further rows can be removed without violating the coverage constraints.

Example 6.9. Consider the affinity matrix A and the associated hybridization matrix H in Table 6.1. The full matrix has conditions $\text{cond}(A) = 2.2963$ and $\text{cond}_s(A) = 3.4664$. Results of the search for designs of maximum size 12 resp. 6 with coverage 1 resp. 3 by exhaustive search and the greedy heuristic are also shown in Table 6.1.

- For $\mathcal{M} = \mathcal{A} = 1$, the freedom to choose only one probe per target leads to the choice of *unique* probes because the identity matrix has the smallest possible condition. The greedy heuristic is able to find this optimum only for the Skeel condition.
- For $\mu = 6$ and $\mathcal{M} = \mathcal{A} = 3$, the requirement to reduce the number of probes from 12 to 6 but to keep the minimum coverage at 3 leads to a slight increase in condition compared to the full matrix, and the greedy heuristic finds this optimum. The optimal Skeel condition drops slightly compared to the full matrix, but the greedy heuristic does not find this solution.

Performance. For small affinity matrices (15×3 and 18×4), we can obtain optimal solutions by exhaustive search and assess the performance of the greedy heuristic. The following behavior is typical; see Figure 6.2.

Table 6.1: An affinity matrix A with 12 non-unique probes for 4 targets and the corresponding hybridization matrix H. Below the matrices, the search results for certain designs by both exhaustive search and the greedy heuristic are shown.

$$A = \begin{pmatrix} 0.100066 & 0.200061 & 0.100017 & 0.000011 \\ 0.100046 & 0.000006 & 0.000048 & 0.000084 \\ 0.300092 & 0.000063 & 0.200016 & 0.100010 \\ 0.100063 & 0.100067 & 0.300100 & 0.000041 \\ 0.000059 & 0.000013 & 0.100052 & 0.000056 \\ 0.000031 & 0.000061 & 0.100096 & 0.100015 \\ 0.200093 & 0.200063 & 0.000034 & 0.000060 \\ 0.100046 & 0.000045 & 0.100068 & 0.200086 \\ 0.000025 & 0.100064 & 0.000097 & 0.000081 \\ 0.200009 & 0.000028 & 0.000080 & 0.100061 \\ 0.000064 & 0.000041 & 0.000076 & 0.100034 \\ 0.000006 & 0.100011 & 0.000061 & 0.300088 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Design	Full	Optimal	Greedy	Optimal	Greedy
Results		$\mu = 12$, $\mathcal{M} = \mathcal{A} = 1$		$\mu = 6$, $\mathcal{M} = \mathcal{A} = 3$	
D	$\{1..12\}$	$\{2, 5, 9, 11\}$	$\{1..7, 9, 10, 12\}$	$\{1, 3, 4, 7, 10, 12\}$	
cond	2.2963	1.0026	2.1840	2.3864	
D	$\{1..12\}$	$\{2, 5, 9, 11\}$		$\{1, 4, 6, 9, 10, 12\}$	$\{1, 2, 4, 6, 11, 12\}$
cond_s	3.4664	1.0041		3.1770	3.8347

- For both cond and cond_s, the optimal affinity matrix with m' removed rows has a smaller condition than the full design when m' is small.
- Initially, i.e., for small m', the greedy solution tends to follow the optimal solution until the optimal solution becomes better. It may or may not happen that the solutions coincide again for some larger m'. This may occur, for example, if the optimal designs for m' and $m' + 1$ removed rows are disjoint.
- Removing many rows from A eventually leads to an increase in condition and Skeel condition for both optimal and greedy solution.
- Selecting a random full-rank submatrix with m' removed rows leads to an increase in condition and Skeel condition.
- Optimal and greedy design need not be of the same size; i.e., the minima of the condition curves can occur at different values of m'. However, the shape of the curve (first decreasing condition, then increasing condition) is similar.

For 100 random 18×4 affinity matrices generated according to the protocol shown in Figure 6.3, mean and median condition numbers are shown in Table 6.2. In this setting,

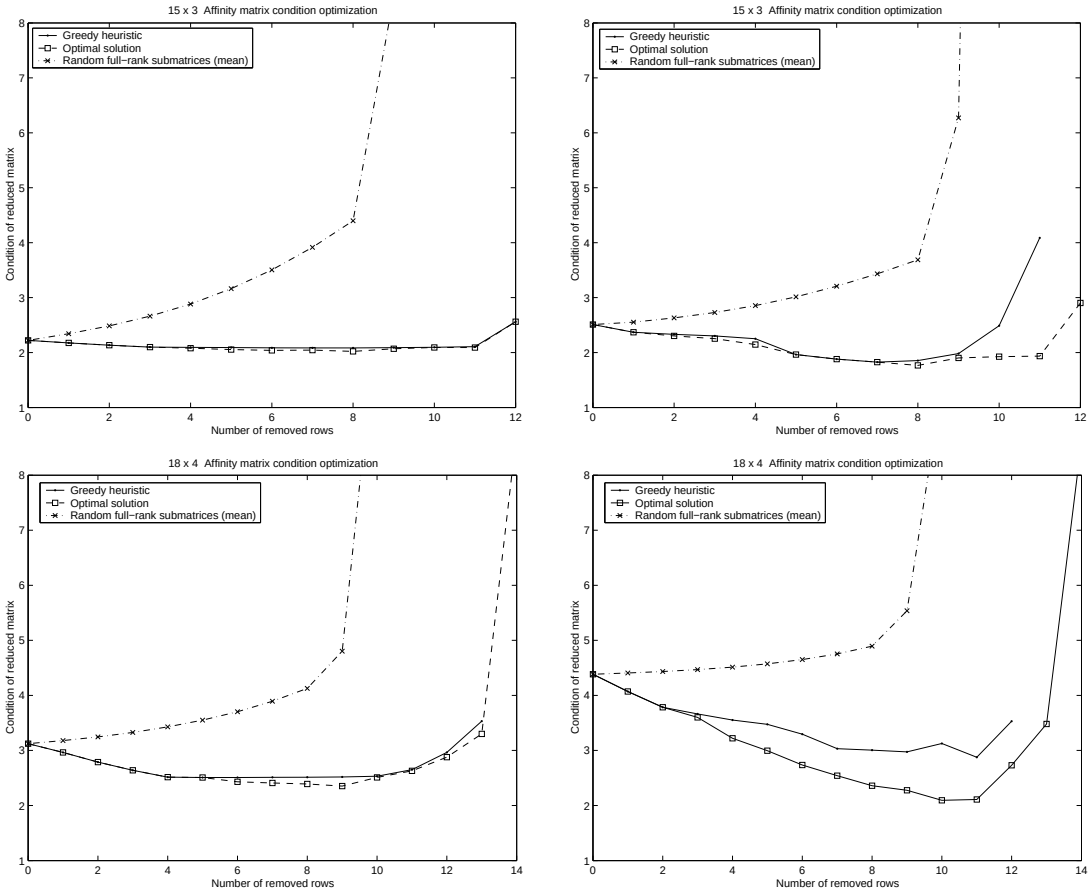


Figure 6.2: Changes in condition (left) and Skeel condition (right) for typical random 15×3 (top) and 18×4 (bottom) affinity matrices with minimum and average coverage $\mathcal{M} = \mathcal{A} = 3$ when probes are removed from the full design.

the greedy heuristic finds the optimal design in 29% of the cases when optimizing $\mathrm{cond}(A^D)$ and in 31% of the cases when optimizing $\mathrm{cond}_\mathrm{s}(A^D)$.

We cannot assess the performance of the greedy heuristic compared to the optimum for large matrices. However, we can see how much the condition is reduced by the greedy heuristic compared to the full design: The lower part of Table 6.2 shows that Skeel condition numbers are higher than classical condition numbers, but also that the greedy heuristic is successful at reducing them by about 37%, whereas the classical condition is only reduced by 16%.

Practical considerations. The approach to optimize the condition of the design is based on the intuitive idea that signal decoding, i.e., solving the over-determined linear system $y = A \cdot x$ for x should be robust against small errors in A and should not magnify measurement errors in the data y. As a consequence, (row permutations of) block diagonal matrices B made out of 1-column blocks are optimal with $\mathrm{cond}_\mathrm{s}(B) = 1$.

```
function A = randommatrix(m,n,minc,lambda)
Create a random m · n affinity matrix with minimum coverage minc.
Random elements have an exponential(lambda) distribution.

Cover each target j = 1,...,n by minc probes.
A = zeros(m,n);
for j=1:n
     i=randperm(m);
     A(i(1:minc),j) = 1+floor(-log(rand(minc,1)));
end

Use each probe at least once.
j=floor(n*rand(m,1))+1;
for i=1:m
     A(i,j(i))=A(i,j(i))+1;
end

Shoot random (floor(exponential(lambda))-distributed) elements into A.
B = (rand(m,n)<=0.3) .* floor(-log(rand(m,n))/lambda);
A = max(A, B);

Scale A and add noise:
Intended probe-target pairs have affinity coefficients 0.1 to 1.
Noise affinity coefficients are ≤ 10^-4.
A = A/10 + (1E-4)*rand(m,n);
A = min(A,ones(m,n));
```

Figure 6.3: MATLAB code for generating $m \times n$ affinity matrices with given minimum coverage for each target.

In practice, it is not sufficient to select a block-diagonal submatrix of A (if it exists), because coverage constraints must be observed. Therefore we generally need to use heuristics to select a design with small condition. Our simulations on small matrices show that the greedy heuristic performs reasonably well. Additionally, it is easy to implement and runs quickly.

Minimizing the condition requires that the full affinity matrix is known. This is usually not the case before a chip with the corresponding probes has been produced and one has had a chance to infer the relative coefficients from experimental data (see also Section 2.4).

In practice, we can assume that probes are non-unique only within a small group of targets, such as the splice variants of a single gene or within a gene family. The Skeel condition minimization problem can be solved independently for each family, because $\mathrm{cond_s}(\mathrm{diag}(A_1, A_2)) = \max\{\mathrm{cond_s}(A_1), \mathrm{cond_s}(A_2)\}$. This allows to solve several small

Table 6.2: Top: Condition numbers for reduced random 18×4 affinity matrices. Bottom: Condition numbers for 200×20 matrices.

18×4 $\mathcal{M} = \mathcal{A} = 3$, $\mu = 7$	Full	Greedy	Optimal
Mean cond	3.26	2.66 (−18%)	2.26 (−30%)
Median cond	3.01	2.38 (−21%)	2.15 (−28%)
Mean cond_s	3.71	2.90 (−22%)	2.39 (−36%)
Median cond_s	3.67	2.57 (−30%)	2.30 (−37%)

200×20 $\mathcal{M} = \mathcal{A} = 5$, $\mu = 50$	Full	Greedy
Mean cond	3.42	2.86 (−16%)
Median cond	3.39	2.85 (−16%)
Mean cond_s	8.31	5.25 (−37%)
Median cond_s	8.22	5.22 (−37%)

problems, potentially by exhaustive search for very small families, instead of one large problem with thousands of targets and tens of thousands of probes.

In summary, condition minimization appears impractical to design a large scale chip from scratch, but it provides a way to optimize existing chips for quantitative gene expression analysis of target groups with high sequence similarity.

6.3 Design and Decoding for Qualitative Analysis

Within this section, we focus not on quantification, but on robust presence or absence calls of targets with non-unique probes. Potential applications include qualitative detection of expressed genes or gene variants from certain families (Wang et al., 2003), species identification (Hebert et al., 2003a,b), and virus typing or subtyping (Rash and Gusfield, 2002; Wang et al., 2002).

It is assumed that for each probe, a signal is either observed (1) or not observed (0); there are no quantitative levels in between. In the absence of unique probes, we may hope that each target sequence gives rise to a different *combination* of probe signals. The vector of probe responses to a target sequence has also been called a target's *barcode*, and recently it has been proposed that all animal life can be barcoded with probes from the mitochondrial cytochrome *c* oxidase subunit I gene (Hebert et al., 2003a).

Theoretically, n probes give rise to 2^n different barcodes and a million of different targets should be identifiable using only 20 different probes. In practice, the problem

is more difficult:

- More than one target may be present in a complex sample subjected to the identification procedure. In this case, we observe the logical `or` of the probe signals we would obtain from the individual targets. Section 6.2
- False positive and false negative probe signals must be expected due to experimental error and overall noise level.

Previous probe selection approaches have often neglected these issues and focused on combinatorial and information theoretic methods that aim at minimizing the number of probes required to identify a single target in the absence of errors (Herwig et al., 2000; Borneman et al., 2001; Rash and Gusfield, 2002).

We take a different approach based on *statistical group testing* that avoids the above shortcomings (Section 6.3.1). The overall procedure is as follows.

1. We assume that we have pre-selected suitable probe candidates, where the usual constraints for oligo design apply (recall Section 6.1).
2. From these candidates, we generate a group testing design, i.e., we pick a subset of probes that allows discrimination between as many (small-sized) target sets as possible. Note that in theory, adding a probe to a design never decreases the design's ability to separate two target sets. Therefore, using *all* candidates guarantees the best possible separation properties of the design (the design quality is naturally limited by the properties of the best candidate probes). In practice, however, the number of candidate probes is very large, and several probes might hybridize to the same target sets. Therefore adding a probe does not necessarily contribute new information, and we would waste spots on the chip with uninformative oligos. Selecting a smaller design may allow use of a smaller chip and considerable reduction of cost. Furthermore, in practice, candidates are ranked according to quality (see the previous section), and we want to include lower-quality probes only when absolutely necessary in order to keep the noise level for the decoding procedure as low as possible. In short, while a small design is preferable, it would be misleading to minimize the number of probes as the only objective. Design strategies are described in Section 6.3.2.
3. The decoding procedure to infer the presence of target sequences uses a Bayesian framework. It is based on Monte Carlo Markov chain sampling and explicitly allows false positive and false negative experimental errors (Section 6.3.3).

An evaluation of the design and decoding steps on real and artificial datasets are presented in Section 6.3.4.

The main difference between the quantitative design and decoding procedure in Section 6.2 and the qualitative procedure in this section are thus as follows. In the quantitative setting, signal decoding is equivalent to solving a linear system and a

good design is characterized by an affinity matrix with small (Skeel) condition number. In the qualitative setting, decoding involves disentangling an `or`-ed Boolean signal; so a good design is characterized by the separability of many target sets (see Section 6.3.2).

6.3.1 Group Testing

Group testing is a general procedure applicable whenever a large population of individuals has to be subjected to the same test. A general introduction to the field is given by Du and Hwang (2000). The idea is to group objects and test groups instead of individuals. A group tests positive when at least one individual within the group tests positive. Every experiment involving group testing requires a design, i.e., a definition of the groups (every individual can belong to many groups), and an corresponding decoding procedure to infer the status of individuals from the status of groups. When we select the groups a priori we have a *non-adaptive* group test. In contrast, in *adaptive* group testing the groups are chosen iteratively, using the results from previous tests to guide selection of groups for the next iteration. When the test results are exact, that is error-free, we speak about *combinatorial* group testing; in the presence of errors one has the harder *statistical* variant of the problem.

Group testing is most successful in general, whenever only few individuals are expected to test positive, because we can use large groups and hence need only a few tests to infer the status of every individual. When the proportion of positive individuals is large, nothing is to be gained in comparison to individual testing.

Successful applications are from medical diagnostics, including screening draftees for syphilis during World War II, the first recorded application (Du and Hwang, 2000), and from industrial quality assurance. In molecular biology, group testing has been applied to the problem of screening DNA clone libraries for sequence tagged sites to aid in the construction of physical maps (Barillot et al., 1991; Bruno et al., 1995; Knill et al., 1996).

Group Testing Issues for Microarrays. In the microarray setting we propose to use a statistical, non-adaptive group testing scheme. The target sequences we intend to identify correspond to individuals. Potential groups, which can be freely chosen in the general setting, are specified here by a non-unique probe which hybridizes to a set of target sequences. The goal is to devise a group testing design which covers each target with a certain number of probes and allows identification of several targets simultaneously while using a reasonably small total number of probes. We encounter several novelties that are not present in other group testing settings.

- *Constrained assignment of individuals to groups.* In contrast to a medical screening setting, we cannot arbitrarily assign individuals to groups. Groups (sets of target sequences) are always defined by an oligo that occurs in all sequences of the group. This restriction is the most important difference between DNA array group testing and "classical" group testing designs.

- *Cross-hybridization.* Even though we allow non-unique probes, the cross-hybridization problem does not disappear. Assume that a probe p_i occurs in all target sequences of a set $T(i)$, and also approximately (but not exactly) matches another target $j \notin T(i)$. The hybridization behavior of j with respect to p depends on many parameters and will vary from experiment to experiment. To keep error rates as low as possible, it is preferable to discard probes whose hybridization behavior is unclear, as stated in Section 6.1.

- *Comparatively high error rates.* Even if we avoid potential cross-hybridization problems, false positive (a probe giving a signal when it should not) and false negative errors (a probe not showing a signal when it should) with rates of up to 5% must be anticipated.

- *Moving targets.* In reality, target sequences are not static objects. They undergo mutations, recombine, or are altered in other ways. Covering a target with many probes adds robustness to the target identification; allowing these probes to be non-unique helps to keep the required probe number low.

6.3.2 Generating Designs

This section describes a fast heuristic to find a good group testing design D, i.e., to select rows of the full $m \times n$ probe-target hybridization matrix H. We also present an optimal design method based on integer linear programming (ILP). We use the notation introduced in Section 6.1; however, T will also be used as a set of target indices.

The primary design goal is to be able to distinguish between most (ideally all) target sets whose size is not too large. The restriction to relatively small sets is both realistic and required by the method. For example when identifying the HIV subtype(s) present in a blood sample, it can be expected that the patient is only infected with one or few subtypes. When detecting pathogens in water samples, it is hoped that none are present; if more than one or a few are found, additional tests are requires in any case. Because of the logical `or`-operation inherent in group testing, the whole procedure is only effective if few positive individuals (targets) are present.

Definition 6.10 (d-separability of target sets). • Let S be a set of target sequence indices. We say that a probe p *hybridizes to the set* S when p hybridizes

to at least one target in S. By $P(S)$ we denote the set of all probe indices hybridizing to S, i.e., $P(S) := \bigcup_{j \in S} P(j)$.

- Now let S and T be two different target sets. Probe p_i *separates* S and T if $i \in P(S) \Delta P(T)$, i.e., if p_i hybridizes to either S or T, but not to both (Δ denotes symmetric set difference).
- The target sets S and T are d-separable if at least d probes separate them, i.e., if $|P(S) \Delta P(T)| \geq d$.

Example 6.11. Suppose we have d *unique* probes for each target. Then two target sets S and T with $|S \Delta T| = c$ are $(c \cdot d)$-separable, because the signal of d different probes differs for each target in $S \Delta T$. Of course, we do not generally have unique oligos to choose from and are restricted to the available candidate probes.

Example 6.12. Consider the hybridization matrix H in Table 6.1 with $m = 12$ probes and $n = 4$ targets. If targets 1 and 2 (but not targets 3 and 4) are present in a sample, we see a signal at probes $1, 2, 3, 4, 7, 8, 9, 10, 12$. If targets 2 and 3 are present, we see a signal at probes $1, 3, 4, 5, 6, 7, 8, 9, 12$. Thus there are 4 probes that separate $S = \{1, 2\}$ from $T = \{2, 3\}$; these are $P(S) \Delta P(T) = \{2, 5, 6, 10\}$. Note that there is one more probe (number 7) that separates target 1 from target 3 alone: $P(\{1\}) \Delta P(\{3\}) = \{2, 5, 6, 7, 10\}$.

As mentioned earlier, the primary design goal is best achieved by choosing *all* available probe candidates. However, the space on a DNA chip is limited and a chip with fewer probes will be less expensive. Therefore we are interested in minimizing the number of probes in a design for given lower bounds on target coverage γ and target set separability σ for all targets sets of small cardinality. This problem is NP-complete, as can be seen by a reduction from the set cover problem.

A greedy heuristic. We propose the following greedy design heuristic.

1. We add probes until every target is covered by at least γ probes, i.e., every singleton target set $\{j\}$ is γ-separated from the empty set $\{\}$, by calling SEPARATE$(\{j\}, \{\}, \gamma)$ for all $j = 1, \ldots, n$ (see Figure 6.4 for a description of SEPARATE).
2. We ensure that all pairs of targets are separated by at least σ oligos by calling SEPARATE$(\{j\}, \{j'\}, \sigma)$ for all $1 \leq j < j' \leq n$.
3. Since there can be several hundreds to thousands of targets, it would take too much time to systematically ensure σ-separation for all larger target sets up to a certain cardinality. Instead, we randomly pick a number N of additional pairs of target sets S and T and σ-separate them by a calling SEPARATE(S, T, σ). The size distribution of the sets we pick follows the distribution of the number

SEPARATE(S, T, d)
Add probes to the current partial design D to d-separate S and T.
1. $C \leftarrow P(S)\Delta P(T)$ (*all separating probes*)
2. Partition C into $C = C_D \cup C'$, where $C_D \leftarrow C \cap D$, and $C' \leftarrow D \setminus C$.
   (C' *contains the separating probes not yet included in* D)
3. **if** $|C_D| \geq d$ **then return** (*nothing to do*)
4. **if** $|C'| < (d - |C_D|)$ **then warn** "Can only $(|C_D| + |C'|)$-separate S and T"
5. Add the $d - |C_D|$ most unique probes from C' to D

Figure 6.4: The SEPARATE procedure ensures d-separation of target sets S and T by adding probes to an existing partial design D.

of targets present in a typical sample (cf. the cardinality prior in Section6.3.3). The parameter N can be chosen according to the time available to refine the design. As an example, this step takes about 5 minutes for 600 targets and 14000 probe candidates with $N = 500000$.

A call to the procedure SEPARATE(S, T, d) in Figure 6.4 ensures d-separation of S and T by adding appropriate additional probes to the current partial design D or produces a warning if the candidate set allows only d'-separation for some $d' < d$.

The idea behind the heuristic is as follows. First the easy problem of covering each target is solved by examining the targets in the given order and adding probes to the design to cover the current target. Unique probes are preferred (followed by probes that hybridize to 2, 3, ... targets). Although this may lead to the inclusion of more probes than necessary for target coverage, the hope is that choosing unique probes in this step will lead to much fewer inclusions during the subsequent target and target set separation steps.

Optimal designs by integer linear programming. The heuristic may choose too many probes in the sense that the same target coverage and target set separability might be achieved with fewer probes. As mentioned, the optimization problem is NP-complete by reduction from Set-Cover, but is still feasible even for designs with several hundreds of targets and thousands of probe candidates. We formulate it as a variation of the Set-Cover integer linear program (ILP).

We represent the design $D \subset \{1, \ldots, m\}$ by a binary column vector $\delta = (\delta_i) \in \{0, 1\}^m$ where $\delta_i = 1$ is equivalent to $i \in D$. We also write H^δ for H^D.

Let h_j denote the j-th column of H, and define the column vector $z^{S,T} = (z_i^{S,T})$ by

$$z_i^{S,T} := \mathbb{I}_{\{i \in P(S)\Delta P(T)\}} \qquad (S, T \subset \{1, \ldots, n\}).$$

In other words, $z_i^{S,T}$ indicates whether probe candidate i separates target sets S and T. The separability of S and T is thus given by $\sum_i z_i^{S,T}$. Note that the j-th column of H can be written as $h_j = z^{\{\},\{j\}}$. Further, let $\prec$ denote an arbitrary linear ordering on the target sets that respects set cardinality, i.e., $S \preceq T \iff |S| \leq |T|$.

The basic ILP now looks as follows.

$$\begin{aligned}
&\text{Minimize } \sum_{i=1}^{m} \delta_i \\
&\text{such that } \delta \in \{0,1\}^m, \\
&\qquad \delta^{\mathsf{T}} \cdot h_j \geq \gamma \qquad \text{for all } j = 1,\ldots,n, \qquad (6.1)\\
&\qquad \delta^{\mathsf{T}} \cdot z^{S,T} \geq \sigma \qquad \text{for all } S \prec T \subset \{1,\ldots,n\},\ 1 \leq |S| \leq |T| \leq B. \qquad (6.2)
\end{aligned}$$

The constant B denotes the maximal relevant target set size. If we only demand σ-separation of all pairs of singleton targets ($B = 1$), then (6.2) can be written as $\sum_i \delta_i \cdot |H_{i,j} - H_{i,j'}| \geq \sigma$ for all $\binom{n}{2}$ pairs (j, j') with $1 \leq j < j' \leq n$. In other words, the Hamming distance between any two columns of the design matrix H^δ must be at least σ.

Depending on the probe candidate set and the magnitude of γ and σ, the ILP may not have a feasible solution: There might only be $d < \sigma$ probe candidates in the full matrix H that separate targets j and j'. This problem can be addressed by adding a sufficiently large number $L := n \cdot \max\{\gamma, \sigma\}$ of "virtual unique probes" that are chosen only if it is impossible to achieve sufficient coverage or separation with the existing candidates. This is achieved by setting their coefficient in the objective function to a large number $C > n$. We denote the usage indicator of the L virtual probes by $\delta^+ = (\delta_\ell^+)$. The extensions of the vectors h_j and $z^{S,T}$ are similarly denoted by h_j^+ and $z^{+,S,T}$. The ILP then becomes

$$\begin{aligned}
&\text{Minimize } \sum_{i=1}^{m} \delta_i + C \cdot \sum_{\ell=1}^{L} \delta_\ell^+ \\
&\text{such that } \begin{pmatrix} \delta \\ \delta^+ \end{pmatrix} \in \{0,1\}^{m+L}, \\
&\qquad \begin{pmatrix} \delta \\ \delta^+ \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} h_j \\ h_j^+ \end{pmatrix} \geq \gamma \qquad \text{for all } j = 1,\ldots,n, \\
&\qquad \begin{pmatrix} \delta \\ \delta^+ \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} z^{S,T} \\ z^{+,S,T} \end{pmatrix} \geq \sigma \quad \text{for all } S \prec T \subset \{1,\ldots,n\},\ 1 \leq |S| \leq |T| \leq B.
\end{aligned} \qquad (6.3)$$

The number of inequalities in (6.2) or (6.3) is exponential in B: There are $N_B := \sum_{k=1}^{B} \binom{n}{k} = O(n^B)$ target sets of cardinality between 1 and B, and $\binom{N_B}{2}$ such pairs of

target sets to be separated. For $B = 1$, this results in approximately $n^2/2$ inequalities. Comparisons between the greedy heuristic and the ILP approach are presented in Section 6.3.4.

For $B = 2$, about $(n^4 + 2n^3 + 2n^2)/8$ inequalities must be satisfied, which for $n = 400$ targets amounts to $3.216 \cdot 10^9$ inequalities. Thus it is hardly practical to even formulate the ILP for $B > 2$ and moderately small n. However, if several probes are relatively unique, that is, if they hybridize to only a few targets, many of the inequalities in (6.3) may already be satisfied after target pair separation and need not be considered. Therefore Gunnar Klau and Knut Reinert have proposed to start with an optimal design that satisfies the target pair separation ($|S| = |T| = 1$) inequalities, and to find violated separation inequalities for larger sets dynamically with a cutting plane approach by solving a different ILP. If a violated inequality is detected, it is added to the ILP, which is then solved for a new design δ. These steps are repeated until no more violated inequalities for target sets of size at most B are found. This idea is being pursued by Gunnar Klau (see Klau, Rahmann, Schliep, Vingron, and Reinert, 2004).

6.3.3 Decoding

We assume that a heuristic or optimal design D has been chosen and from now on only consider the the design matrix H^D. Thus in this section, m is the number of probes in the design and smaller than the number of candidates. After observing probe signals, we face the problem of inferring which targets are present in the sample.

We know H^D and observe the probe signal vector $y = (y_1, \ldots, y_m) \in \{0, 1\}^m$ for the probes $p_1, \ldots, p_m$.

We use a probabilistic model allowing false positive and false negative errors to describe the dependencies between the unknown true target presence vector $x = (x_j) \in \{0, 1\}^n$ and the observations y. Somewhat abusing notation, we use $\mathbb{P}$ to denote both a probability measure and a distribution: $\mathbb{P}(x \,|\, y)$ is the conditional distribution of x given y, but also the probability that the random variable represented by x takes the value x; and $\mathbb{P}(x_j = 1)$ denotes the probability that $x_j = 1$. We have

$$\mathbb{P}(x \,|\, y) \propto \mathbb{P}(x, y) = \mathbb{P}(x) \cdot \mathbb{P}(y \,|\, x).$$

$\mathbb{P}(x)$ denotes the prior probability of target set $X = \{j : x_j = 1\}$ before observing the probe signals. We assume that it depends on the cardinality $|X| = \sum_j x_j$ and independently also on the relative prevalence $\rho_j > 0$ of each target $j \in X$ in the overall

target population:

$$\mathbb{P}(x) \;\propto\; c(|X|) \cdot \prod_{j \in X} \rho_j \;=\; c(\textstyle\sum_j x_j) \cdot \displaystyle\prod_{j=1}^{n} \rho_j^{x_j}.$$

Here $c = (c(0), \ldots, c(n)) \geq 0$ is a vector that contains the relative prior probabilities of different target set sizes. It is assumed that most of the mass in concentrated at low set cardinalities. Depending on the application, one could also use a prior that includes correlations between targets. If no prior information is available, however, all prevalences ρ_j can be taken to be equal and independent. We do not need to know the normalizing constant for $\mathbb{P}(x)$, as will become evident shortly.

To compute the likelihood $\mathbb{P}(y \,|\, x)$, we assume that false negative and false positive perturbations of the true observation associated to x act independently on each probe. Let f_0 be the false negative error rate per hybridization, and let f_1 be the false positive error rate per probe.

If probe p_i hybridizes to no target in X, we have $\mathbb{P}(y_i = 1 \,|\, x) = f_1$ (the false positive rate) and $\mathbb{P}(y_i = 0 \,|\, x) = 1 - f_1$. If probe p_i hybridizes to $k > 0$ targets in X, i.e., if $|X \cap T(i)| = k > 0$, we only observe no signal if all k hybridizations fail and we do not have a false positive probe: $\mathbb{P}(y_i = 0 \,|\, x) = f_0^k \cdot (1 - f_1)$. We set

$$k_i(x) := |X \cap T(i)| \;=\; \sum_j x_j \cdot \mathbb{I}_{\{j \in T(i)\}}$$

$$\text{and} \quad \lambda_i(x) := \mathbb{P}(y_i = 1 \,|\, x) \;=\; 1 - f_0^{k_i(x)} \cdot (1 - f_1),$$

and obtain

$$\mathbb{P}(y \,|\, x) = \prod_{i=1}^{m} \lambda_i(x)^{y_i} \cdot (1 - \lambda_i(x))^{1 - y_i}. \tag{6.4}$$

There are alternative ways of specifying a likelihood model. For instance, we could simply use a simple per-probe false negative rate instead of a per-hybridization rate. This would lead to $\lambda_i(x) := 1 - f_0^{\mathbb{I}_{\{k_i(x) \neq 0\}}} \cdot (1 - f_1)$, which equals f_1 if $k_i(x) = 0$, and $1 - f_0 + f_0 f_1$ if $k_i(x) \geq 1$.

Putting everything together, we find the posterior probability

$$\mathbb{P}(x \,|\, y) = Z_y \cdot c(\textstyle\sum_j x_j) \cdot \displaystyle\prod_{j=1}^{n} \rho_j^{x_j} \cdot \prod_{i=1}^{m} \lambda_i(x)^{y_i} \cdot (1 - \lambda_i(x))^{1 - y_i}, \tag{6.5}$$

where Z_y is the normalizing constant that turns $\mathbb{P}(x \,|\, y)$ into a proper probability distribution for observation y.

We are interested in

1. the marginals $\mathbb{P}(x_j = 1 \mid y)$ of the posterior for all $j = 1, \ldots, n$,
2. the posterior target set size distribution $\mathbb{P}(|X| = k \mid y)$ for small $k \geq 0$, and
3. the vector x maximizing $\mathbb{P}(x \mid y)$.

The first two problems can be solved by constructing an ergodic Markov chain with $\mathbb{P}(\cdot \mid y)$ as its stationary distribution and sampling instances of x from $\mathbb{P}(x \mid y)$, e.g., by Gibbs sampling. Estimates of the marginal probabilities and the target set size are then obtained as ergodic averages of the sample components and sample sizes. Details are given below.

The second problem can be solved by a simulated annealing variation of the sampling method (see below). However, finding the exact maximum can be both difficult and uninformative, because several $x \in \{0,1\}^n$ may explain the observations well. Also, if a distinguished maximum exists, it becomes evident from the posterior marginal probabilities. If there are several local maxima, there will be a difference between the average posterior target set size $|X|$ and the number of x_j with relatively high posterior probability of being present.

Sampling from the posterior. As the observation y is fixed, we set $\pi(x) := \mathbb{P}(x \mid y)$, and we assume that $\pi(x) > 0$ for all x, even though it is reasonable that $\pi(x) \approx 0$ for large cardinality target sets x. We use the Metropolis-Hastings framework (see Hastings, 1970; Gilks et al., 1996, Chapter 1) to define a Markov chain with stationary distribution $\pi(x)$, taking $x = (0, 0, \ldots, 0) \in \{0,1\}^n$ as the start state.

Being in state x, a new state z is proposed from a proposal distribution $q(z \mid x)$ and then accepted with probability

$$\alpha(x, z) = \min\left\{1, \frac{\pi(z) \cdot q(x \mid z)}{\pi(x) \cdot q(z \mid x)}\right\}.$$

The transition kernel of the Markov chain is thus defined by $P_{x,z} = q(z \mid x) \cdot \alpha(x, z)$ for $x \neq z$ and $P_{x,x} := 1 - \sum_{z \neq x} P_{x,z}$. The states z for which $P_{x,z} > 0$ are said to form the neighborhood of x. For convenience, we require that the neighborhood relation is symmetric. Now it can be seen why π needs only to be known up to a constant: it only appears as a ratio in $\alpha(x, z)$.

The proposal distribution q must be chosen in such a way that in some number of iterations t^*, every state z can be reached from any state x, i.e., we require that there exists t^* such that $P^{t^*} > 0$. It then follows that the constructed Markov chain is irreducible and aperiodic and hence ergodic with a unique stationary distribution that is equal to π. Independently of the start state, the state distribution converges exponentially fast to π (e.g., Gilks et al., 1996, Chapters 3 and 4). The convergence rate depends on the structure of π and q; it is related to the second largest absolute

eigenvalue $\lambda < 1$ of P. It can be shown that $|(P^t)_{xz} - \pi(z)| \leq \text{const} \cdot \lambda^t$ (e.g. Billingsley, 1995). It is difficult to obtain a tight bound on λ in practice, however; recall that P is a $2^n \times 2^n$ matrix in this setting. Experience shows that if the states are well connected, if there exist no "bottlenecks" and if the start state is not extremely improbable under π, then convergence can be expected to occur after a small multiple of n steps.

We define the neighborhood of x as $\mathcal{N}(x) := \{z : 0 \leq |\sum_j (z_j - x_j)| \leq 1 \text{ and } 0 \leq \sum_j |z_j - x_j| \leq 2\}$. In other words, $x \in \mathcal{N}(x)$ and otherwise $z \in \mathcal{N}(x)$ if z differs from x by exclusion of one target or inclusion of one additional target, or if z and x have the same number total number of targets, but in both x and z at most one target is present that is absent in the other state. In x, each neighbor is proposed with the same probability $1/|\mathcal{N}(x)|$. It is easy to see that if $\sum_j x_j = k$, then $|\mathcal{N}(x)| = 1 + k + (n-k) + k(n-k) = (k+1)n - k^2 + 1$. With this choice of the proposal distribution, every state is reachable from any state in n steps with positive probability, and the chain is well connected.

An alternative is given by the Gibbs sampler, where each iteration consists of n steps, and in step j, only the j-th component of x is updated; a new value from $\{0, 1\}$ is proposed according to the full conditional distribution $\mathbb{P}(x_j \,|\, y, (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n))$. For this step, the new components $x_1, \ldots, x_{j-1}$ and the old components $x_{j+1}, \ldots, x_n$ are used. The proposed value of x_j is always accepted, but it can be equal to the old value with high probability. In each step, x either does not change, or its number of 1-components changes by one. Alexander Schliep implemented Gibbs-sampling based decoding in the Markov Chain Pool Decoder (MCPD; Schliep, 1998), which was successfully used in several projects (Knill, Schliep, and Torney, 1996; Schliep, Torney, and Rahmann, 2003).

In both the neighborhood-based Metropolis-Hastings algorithm and the Gibbs sampler, the quantities of interest can be computed by averaging appropriate functions of the sample x for $M \gg 0$ iterations after a sufficiently long burn-in time t_0 to achieve convergence of the chain to π. Let $x^{(t)}$ denote the sample at time t. The distribution of the target set cardinality $\mathbb{P}(|X| = k \,|\, y)$ is estimated by

$$\hat{\mathbb{P}}(|X| = k \,|\, y) = \frac{1}{M} \cdot \sum_{t=t_0}^{t_0+M-1} \mathbb{I}_{\{\sum_j x_j^{(t)} = k\}},$$

and $\pi(x_j = 1)$ is estimated by

$$\hat{\mathbb{P}}(x_j = 1 \,|\, y) = \frac{1}{M} \cdot \sum_{t=t_0}^{t_0+M-1} x_j^{(t)}.$$

Alternatively, $\pi(x_j = 1)$ can be estimated whenever x_j is updated by using the full conditional distribution $\mathbb{P}(x_j = 1 \,|\, y, (x_{j'})_{j' \neq j})$. Again, these estimates are averaged

over many sampling steps. The use of full conditional distributions usually leads to a smaller variance in the estimation (see Gilks et al., 1996, Chapters 1 and 5).

Simulated annealing. As noted earlier, the best way to find target sets with high posterior probability is to form combinations of the targets with high marginal posterior. Usually there will be only few candidate sets, and these can be examined exhaustively. In the case of several likely present targets, an alternative to find the best set is given by simulated annealing (Kirkpatrick et al., 1983). Here the posterior distribution $\pi(x) = \mathbb{P}(x \mid y)$ of the Markov chain is modified by a temperature parameter T. The probability $\pi_T(x)$ is interpreted as the function of the "energy" of state x via $\pi_T(x) \propto \exp(-E(x)/k \cdot T)$, where k is the Boltzmann constant. In contrast to the molar energy considered in Section 2.3.3, the energy $E(x)$ considered here is an abstract quantity). We may as well change the scale of the temperature to units of $1/k$ and set $E(x) := -\log(\pi(x))$, such that $\pi \equiv \pi_1$, and

$$\pi_T(x) = \frac{1}{Z(T)} \cdot \pi(x)^{1/T},$$

where $Z(T) = \sum_z \pi(z)^{1/T}$ is the normalization constant. High T lets $\pi_T(x) \to 1/Z(T)$ for all x and hence flattens the distribution. Lowering $T \to 0$ amplifies differences in π such that in the limit only the state x^* with maximal $\pi(x^*)$ has nonzero probability: $\pi_{T \to 0}(x^*) \to 1$. The idea of simulated annealing is to start at a high temperature where the chain can move freely and has a stationary distribution close to the uniform distribution, and then to lower the temperature gradually in such a way that the chain has sufficient time to adjust to its new equilibrium π_T. If the cooling rate is chosen properly, the chain will become restricted to x^* as $T \to 0$. The problem encountered in practice is to determine the correct cooling rate to avoid ending up in local maxima without a chance to escape again (Laarhoven and Aarts, 1987).

6.3.4 Evaluation

We evaluate the greedy design heuristic and the ILP design on two artificially evolved sequence families and on a real test set of 679 28S rDNA sequences of organisms present in the Meiobenthos[5].

Meiobenthic test set. Markmann (2000) provides an initial set of 1230 28S rDNA sequences from meiobenthic organisms. The set contains redundancies and many close homologs. To reduce the level of redundancy we use the `blastclust` software from

[5] *Meiobenthos*: Benthic organisms (animals or plants) whose shortest dimension is less than 0.5 mm but greater than or equal to 0.1 mm. *Benthic:* dwelling on, or relating to, the bottom of a body of water.

NCBI[6] to cluster sequences in the date set which share at least 99% sequence identity over at least 99% of their length. Given the average sequence length of about 676 nucleotides this corresponds to about 7 mismatches between clustered sequences on average. The 149 clusters containing two or more sequences represent about 56% of all sequences. For each of those clusters we pick an arbitrary representative (the first one in the `blastclust` output). This procedure results in a test set consisting of 679 sequences.

Artificial sequence families. To generate artificial data that closely models homologous sequence families, we use REFORM (Random Evolutionary FORests Model; see Appendix A.2 and Rahmann, 2003c), which we developed as a general-purpose tool for sequence evolution modeling. It allows to define arbitrary sets of evolutionary trees ("evolutionary forests") with either random or pre-defined root sequences. The sequences are evolved from the root through internal nodes to the leaves along the branches of the tree for a time proportional to the branch lengths, and may consist of several segments. For each segment it is possible to specify a separate evolutionary model or only separate relative speeds or speed distributions.

The nucleotide substitution model is specified as an evolutionary Markov process (EMP), e.g., the simple Jukes-Cantor model (Jukes and Cantor, 1969) that assigns equal probabilities to all mutation types. Alternatively it can be specified as any valid rate matrix $Q = (Q_{ij}) \geq 0$ with $Q_{ii} = -\sum_{j \neq i} Q_{ij}$ generating an ergodic time-continuous Markov process (e.g., Müller and Vingron, 2000). For $i \neq j$, Q_{ij} is the instantaneous mutation rate $i \to j$, where i and j are different nucleotides, and $|Q_{ii}|$ then measures the overall mutation rate away from i. Branch lengths are measured as percent of expected mutations (PEM). If π is the unique stationary distribution associated to Q, i.e., if $\pi \cdot Q = 0$, and if $\sum_i \pi_i \cdot |Q_{ii}| = 0.01$, then Q is said to be calibrated to 1 PEM and the transition kernel (conditional mutation probability matrix) for a branch of length t is given by $\exp(tQ)$, where $\exp(\cdot)$ denotes the matrix exponential.

An indel model is placed on top of the substitution process by specifying a deletion rate, an insertion rate, an indel length distribution, and a nucleotide distribution of inserted residues. During sequence evolution along a branch, the probability of deleting one or several characters at each position of the parent sequence is given by the product of the branch length, the relative speed for the current segment, and the deletion rate. The length of the gap is then drawn from the specified gap length distribution. A similar rule is applied to inserts. Substitutions are only computed for non-deleted positions, but inserts can follow immediately after deletions.

For our experiments, we use two different models (see Figure 6.5) and generate five independent test sets from each model.

[6] `http://www.ncbi.nlm.nih.gov/BLAST/`

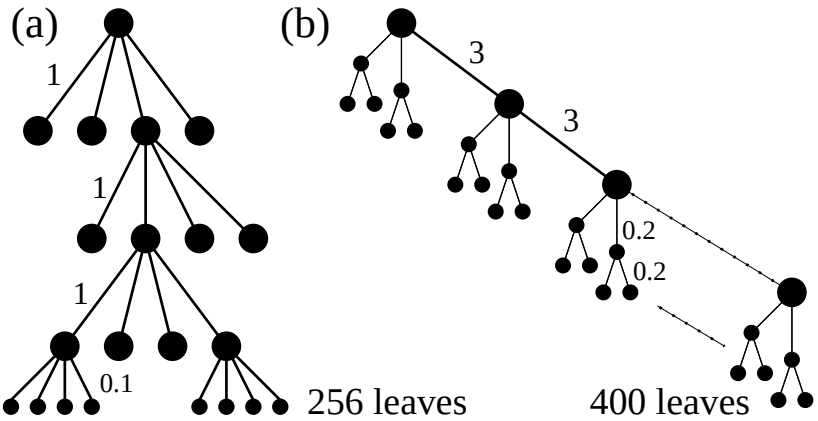


Figure 6.5: The two evolutionary tree models used for sequence family generation. Branch lengths are indicated next to sample branches. The 256 resp. 400 leaf sequences were taken as family members. In (a), not all children of the nodes are shown.

(a) The first model produces a family of 256 sequences of average length 1000 nt. The root consists of a random 1000 nt sequence with uniform nucleotide distribution and splits into 5 segments with relative evolutionary speeds of 0.9 0.95, 1, 1.05, and 1.1. Substitutions are generated according to the Jukes-Cantor model. The global delete and insert rates are set to 0.005, and the distribution of the gap lengths is given by the probability vector proportional to $(8, 1, 4, 2, 1, 0.5, 0.25, 0.125, \ldots)$. Inserted residues are drawn from the uniform distribution. The tree has three levels of internal nodes below the root for a total of $4 + 16 + 64 = 84$ internal nodes. Starting with the root, each internal node has 4 children. The distance between adjacent nodes corresponds to $t = 1$ PEM. Each internal node on the third level has four leaf children at a distance of $t = 0.1$ PEM for a total of 256 leaves with different distances to each other (0.2, 2.2, 4.2, ...). The leaf sequences are subsequently used for probe candidate selection.

(b) In the second model, all global parameters are as in the first model, and the sequences consist of a single segment of average length 1000 nt. The topology differs considerably from the first model: The tree consists of a linear chain of 100 internal nodes (including the root) 3 time units apart; two "cherries" with branch lengths of 0.2 are attached to each internal node (Figure 6.5b) for a total of 400 leaves.

Probe candidates and designs. To generate probe candidates for each of the 10 families (5 instances of each model), we use our own software PROMIDE (Appendix A.1). Non-unique probe candidates are selected to be between 19 and 21 nt long with a Gibbs free energy of -20 to -19.5 kcal/mol at 40°C and $[Na^+] = 0.075$ M according to the Nearest Neighbor model with parameters from Table 2.1 (SantaLucia, 1998).

Table 6.3: For each artificial dataset (a)1 to (b)5 and for the Markmann (2000) meiobenthic data (M), the table shows the number n of targets, the number #cand of probe candidates, and the number of probes m chosen by the greedy design heuristic and the ILP approach, using singleton pair separation only. Percentages represent the number of selected probes in relation to the number of probe candidates. The probe ratio $m_{\text{Greedy}}/m_{\text{ILP}}$ and the ratio $t_{\text{Greedy}}/t_{\text{ILP}}$ of the required design time are also shown.

Set	n	#cand	Greedy m		ILP m		m ratio	t ratio
(a) 1	256	2786	1163	(42%)	503	(18%)	2.31	0.23
(a) 2	256	2821	1137	(40%)	519	(18%)	2.19	0.21
(a) 3	256	2871	1175	(41%)	516	(18%)	2.28	0.25
(a) 4	256	2954	1169	(40%)	540	(18%)	2.17	0.17
(a) 5	256	2968	1175	(40%)	504	(17%)	2.33	0.24
(b) 1	400	6292	1908	(30%)	879	(14%)	2.17	0.02
(b) 2	400	6283	1885	(30%)	938	(15%)	2.01	0.02
(b) 3	400	6311	1895	(30%)	891	(14%)	2.13	0.06
(b) 4	400	6223	1888	(30%)	915	(15%)	2.06	0.02
(b) 5	400	6285	1876	(30%)	946	(15%)	1.98	0.07
(M)	679	15139	3851	(25%)	3158	(21%)	1.22	0.08

When filtering for specificity, we ensure that each candidate p_i satisfies Definition 6.2 on page 95 with the assumption that the transcripts are equal to the targets, and with the constraints that $\sigma_1(i) = 0$ and $\mu = 2$. This corresponds to a stringent washing procedure (normally $\sigma_1 > 0$ would be allowed at the price of a higher value of μ in Definition 6.2). Probe candidates are de-clustered before the final probe set selection.

Designs with minimum coverage 10 and minimum target pair separation 5 are generated with the greedy heuristic (a part of our PROMIDE software package) and by solving the ILP with CPLEX (ILOG, Inc., 1987–2004). The CPLEX ILP formulation was implemented by Gunnar Klau (Vienna University of Technology). The results are shown in Table 6.3. Naturally, the heuristic runs faster, but it also generates a design that is often larger by a factor of more than 2 than the optimal design found with the ILP approach. The difference between the designs is smaller for Markmann's dataset (M), because not all sequences in this set are close homologs. Since the absolute running times are within the range of 50 to 1700 seconds, the ILP approach is quite practical.

The heuristic attempts to "looks ahead", i.e., it starts by choosing the most unique probes when covering and separating targets. While this cannot result in a minimal design for target pair separation, fewer probes need to be added for separating larger of target sets. In contrast, if more inequalities are added to the ILP, i.e., if B is increased in (6.3), it is likely that the optimal solution changes and grows with every additional inequality (presently not implemented). This behavior is also reflected in

the fact that the greedy design, which guarantees only singleton target pair separation, actually separates larger target sets quite well (see decoding results below). The ILP design has difficulties in dealing with larger target sets because of its minimality. We conjecture that the heuristic and the ILP design will approach each other in size and performance as constraints for larger target set separation are added.

Simulated microarray experiments. To simulate noisy microarray probe signals, we first choose a target set size k and then create a test set randomly according to the relative prevalences ρ_j for each target j. We used uniform prevalences and created 100 test sets of each cardinality from 1 to 5 for each of the experiments listed in Table 6.3. Noisy probe signals are generated according to (6.4) with per-probe false negative and false positive rates of 0.01 for one evaluation run and of 0.05 for a second run. The noisy signals are used as input for the decoder.

Decoder. Alexander Schliep adapted the Markov Chain Pooling Decoder (MCPD; Schliep, 1998) that implements the Gibbs sampler to the problem at hand. We checked the speed of convergence of the Markov chain by performing 100 runs each for 100 artificial inputs, and by computing the standard deviation of the marginal probabilities for a number of time points. Inspection of the convergence data guided our choice of using $t_0 = 5000$ burn-in and 50000 total steps. On an AMD Athlon XP 2100 Linux machine, a decoding run needs about 15 seconds CPU time. The target set cardinality prior used for decoding purposes was set to the probability distribution proportional to $c = (c(k))_{k\geq 0} := (0.001, 0.5, 1, 1, 0.5, 0.25, 0.1, 0.05, 0.025, \ldots)$, where subsequent entries decrease exponentially fast.

Results. The result of the decoding is a sorted list of the most probable true positive targets. For the decoded noisy microarray results of each experiment in Table 6.3, the ranks of the true positives were, and it was noted how many of the k true positives are found among the top 1, 2, 3, 4, 5, 10, and 20 ranking targets. Tables 6.4–6.6 show the results for each of the three models (M), (a), and (b), averaged over the 5 sequence sets for models (a) and (b) and the above mentioned 100 repetitions of each experiment. A design is ideal if the proportion of the k true positive targets among the k targets with highest posterior probability is approximately equal to 1.

Clearly the success rate degrades as k grows since the designs only guarantee that pairs of singleton target sets are separable. Even for small k, maximal performance cannot be expected for two reasons. First, in the the number of true positive probes is vastly outnumbered by the number of false positive ones even in the presence of only small error rates. Second, the decoding procedure is stochastic and hence not guaranteed to give a perfect result. The simulations were carried out by Gunnar Klau and Dietmar Ebner (Vienna University of Technology).

The ILP design, which for models (a) and (b) contains less than half as many probes as the heuristic design, still has excellent decoding capabilities; sometimes it even appears to be slightly better than the heuristic because of stochastic fluctuations. For sets with five true positive targets, the decoding capability of the ILP design is significantly worse than that of the heuristic. This can be explained by the fact that only pairs of singleton target sets are guaranteed to be separable. The ILP produces a minimal solution with this property, while the heuristic includes more probes, and especially more "unique" probes, i.e., probes that hybridize to fewer targets on average, and hence have better overall separation capabilities.

Interestingly, for 4 or 5 true positive targets, decoding results are better with error levels of 5% than with 1%. This is not an error; it is explained by the cardinality prior c that favors sets with 2 and 3 targets. The higher error rates allow the Markov chain to explore larger sets more often, and generally lead to a higher mixing rate of the chain.

Table 6.4: Decoding results for model (M). The numbers have the same meaning as in Table 6.5 on page 122.

Model (M), Heuristic Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	0.96	0.98	1.00	1.00	1.00	1.00	1.00
2	—	0.95	0.98	0.99	1.00	1.00	1.00
3	—	—	0.96	0.98	0.99	1.00	1.00
4	—	—	—	0.88	0.92	0.96	0.96
5	—	—	—	—	0.83	0.93	0.93

Model (M), ILP Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	0.89	0.96	0.98	0.99	1.00	1.00	1.00
2	—	0.92	0.97	0.99	0.99	1.00	1.00
3	—	—	0.91	0.97	0.98	0.99	0.99
4	—	—	—	0.88	0.93	0.97	0.97
5	—	—	—	—	0.75	0.89	0.90

Model (M), Heuristic Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	0.96	0.99	0.99	0.99	0.99	1.00	1.00
2	—	0.95	0.98	0.98	0.99	0.99	0.99
3	—	—	0.94	0.97	0.99	0.99	0.99
4	—	—	—	0.92	0.97	0.99	0.99
5	—	—	—	—	0.87	0.95	0.95

Model (M), ILP Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	0.92	0.95	0.98	1.00	1.00	1.00	1.00
2	—	0.87	0.94	0.97	0.98	0.99	0.99
3	—	—	0.89	0.95	0.98	1.00	1.00
4	—	—	—	0.89	0.96	0.99	0.99
5	—	—	—	—	0.85	0.95	0.95

Table 6.5: Decoding results for model (a), averaged over the 5 sequence families (a)1 through (a)5. Row k refers to sets with k true positives. Column "top T" refers to the fraction of true positives found among the T targets with highest posterior probabilities. All fractions are averaged over 100 simulations. Example: In target sets with $k = 5$ true positives, on average 63% of these true positives are found among the 5 most probable targets, and 88% of them are found among the 10 or 20 most probable targets. (with false positive and false negative error rates of 1%).

Model (a), Heuristic Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	0.99	1.00	1.00	1.00	1.00
3	—	—	0.98	0.98	0.99	0.99	0.99
4	—	—	—	0.91	0.92	0.98	0.98
5	—	—	—	—	0.68	0.88	0.88

Model (a), ILP Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	1.00	1.00	1.00	1.00	1.00
3	—	—	0.91	0.93	0.93	0.97	0.97
4	—	—	—	0.71	0.74	0.86	0.86
5	—	—	—	—	0.45	0.64	0.64

Model (a), Heuristic Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	1.00	1.00	1.00	1.00	1.00
3	—	—	0.99	0.99	0.99	1.00	1.00
4	—	—	—	0.95	0.96	0.98	0.98
5	—	—	—	—	0.78	0.91	0.91

Model (a), ILP Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.98	0.99	0.99	0.99	1.00	1.00
3	—	—	0.96	0.97	0.98	0.99	0.99
4	—	—	—	0.84	0.87	0.92	0.92
5	—	—	—	—	0.62	0.77	0.77

Table 6.6: Decoding results for model (b), averaged over the 5 sequence families (b)1 through (b)5. The numbers have the same meaning as in Table 6.5.

Model (b), Heuristic Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	1.00	1.00	1.00	1.00	1.00	1.00
3	—	—	0.99	0.99	0.99	1.00	1.00
4	—	—	—	0.91	0.93	0.98	0.98
5	—	—	—	—	0.66	0.88	0.88

Model (b), ILP Design, 1% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	1.00	1.00	1.00	1.00	1.00
3	—	—	0.96	0.97	0.98	0.98	0.98
4	—	—	—	0.80	0.82	0.91	0.91
5	—	—	—	—	0.51	0.70	0.70

Model (b), Heuristic Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	1.00	1.00	1.00	1.00	1.00
3	—	—	0.99	0.99	0.99	0.99	0.99
4	—	—	—	0.96	0.97	0.98	0.98
5	—	—	—	—	0.81	0.92	0.92

Model (b), ILP Design, 5% Error Rate							
k	top 1	top 2	top 3	top 4	top 5	top 10	top 20
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	—	0.99	1.00	1.00	1.00	1.00	1.00
3	—	—	0.97	0.98	0.99	0.99	0.99
4	—	—	—	0.93	0.94	0.97	0.97
5	—	—	—	—	0.75	0.85	0.85

Chapter 7

Design and Analysis of Genome Tiling Chips

In this chapter, we show how the LCF approach and MCMC-based decoding can be adapted to genome tiling chips, which are useful for determining the transcriptome of an organism (Problem 8 in Chapter 1).

With genome-tiling chips, a whole genome's euchromatin (technically speaking, a repeat-masked version of the genome) is covered with probes at relatively regular intervals; so there is considerably less freedom in choosing specific probes. There is also no a-priori concept of target or transcript, since the transcriptome is considered as unknown. Nevertheless, we can *evaluate* the quality of all possible probes[1] by computing approximate LCF lengths against all 25-mers within the genome and consequently move probes a few nucleotide positions upstream or downstream to obtain higher average specificity. Section 7.1 describes how to adapt the LCF approach to genome tiling chips.

The objectives of genome tiling chips are large-scale transcript identification and gene finding, including both non-coding transcripts (e.g., tRNA and rRNA genes) and protein-coding genes. Recall from Figure 1.3 that eukaryotic protein-coding genes consist of several exons, interrupted by introns. Therefore a transcript does not necessarily occupy a contiguous region in the genome, but breaks up into several transcript fragments or *transfrags*. Section 7.2 shows how experiments with genome tiling chips can be decoded to discover previously unknown transfrags.

Using genome-tiling chips, Kapranov et al. (2002) discovered new transcriptional activity in chromosomes 21 and 22 of the human genome, but their analysis did not explicitly take possible cross-hybridization into account; so some of their observations might be "explained away" in this manner. Our purpose is to provide a framework for choosing good probes for genome tiling chips and to explore the limits of the decodability of these experiments.

[1] Referring to Affymetrix GeneChip® technology, we restrict the discussion in this chapter to 25-mers.

7.1 Probe Evaluation and Selection

Since we have no a-priori definition of targets and transcripts and must deal with large amounts of data, we modify the longest common factor approach to find relatively specific probes. Our intention is to compare all 25-mers in the euchromatin sequence s of the human genome (technically, we take one repeat-masked strand of each chromosome; approximately $1.5 \cdot 10^9$ nt) against all 25-mers in the whole genomic sequence t (both strands; about $6 \cdot 10^9$ nt). As a first proposal, suppose that we would like to compute longest common factor statistics of width $\Delta = 14$ at single nucleotide resolution, i.e.,

$$\mathrm{LCFS}(i)_\delta := |\{j : \mathrm{lcf}(s_{i..i+24}, t_{j..j+24}) = 25 - \delta\}| \qquad (\delta = 0, \ldots, 13)$$

for all starting positions i in s such that $s_{i..i+24}$ is a 25-mer probe candidate. However, the structure of this problem is different from the one studied in Chapter 4; we now have billions of "transcript collections" (all 25-mers in t), which additionally overlap. This has two consequences. First, since every probe candidate of s is also a 25-mer of t, we already expect $\delta + 1$ "transcripts" with LCF length $25 - \delta$. Second, it would be wasteful to process all "transcripts" separately. Therefore, we propose a modified specificity measure for this application; it is based on cumulative statistics of matching statistics (CSMS).

Definition 7.1 (Cumulative statistics of matching statistics). For two strings s, t define the matrix of *cumulative statistics of matching statistics* $\mathrm{csms}^{s|t} = (\mathrm{csms}^{s|t}_{i,\mu})$ of s against t by

$$\mathrm{csms}^{s|t}_{i,\mu} := |\{j : \ \mathrm{lcp}(s_{(i)}, t_{(j)}) \geq \mu\}| \quad (i = 0, \ldots, |s| - 1, \ \mu \in [R^0_{\min}, R_{\max}], \ R_{\max} \geq 25).$$

Recall that $\mathrm{lcp}(s_{(i)}, t_{(j)})$ denotes the longest common prefix of the suffix of s starting at position i and the suffix of t starting at position j. Thus, $\mathrm{csms}^{s|t}_{i,\mu}$ denotes the number of times that $s_{i..i+\mu-1}$ occurs in t.

A bucket scan algorithm for CSMS. For genome tiling chips, the bounds of interest for μ are $[R^0_{\min}, R_{\max}] = [12, 25]$ (these are $\Delta = 14$ values). For μ in this range and fixed i, all $\mathrm{csms}^{s|t}_{i,\mu}$ are easily computed by a modification of the bucket scan algorithm in Figure 4.3.

To initialize the computation for position i, we set $\texttt{CSMS}[i][\mu] \leftarrow 0$ for all μ. Lines 2, 6, and 10 of the bucket scan algorithm are replaced by

$$\texttt{CSMS}[i][\mu] \leftarrow \texttt{CSMS}[i][\mu] + 1.$$

The `cl` array is then no longer required for the algorithm (nor does it exist for genome tilting chips). From the discussion of the algorithm in Section 4.3.2, it is evident that upon termination of the bucket scan, we have $\texttt{CSMS}[i][\mu] = |\{j : \ \mathrm{lcp}(s_{(i)}, t_{(j)}) = \mu\}|$. To obtain *cumulative* statistics, we need to form cumulative sums:

for $\mu = R_{\max} - 1$ **downto** $R^0_{\min}$
    $\texttt{CSMS}[i][\mu] \leftarrow \texttt{CSMS}[i][\mu] + \texttt{CSMS}[i][\mu + 1]$

Thereafter we have $\texttt{CSMS}[i][\mu] = \mathrm{csms}^{s|t}_{i,\mu}$ for $\mu \in [R^0_{\min}, R_{\max}]$, as desired.

Unspecificity evaluation. For the specificity of the probe candidate starting at position i, only the vectors $\mathrm{csms}^{s|t}_i, \ldots, \mathrm{csms}^{s|t}_{i+24}$ are of interest. By the suffix property of matching statistics, we have

$$\mathrm{csms}^{s|t}_{i,\mu} \geq \mathrm{csms}^{s|t}_{i-1,\mu+1}.$$

In other words, there are at least as many matches of length $\geq \mu$ starting at position i as there are matches of length $\geq \mu + 1$ starting at position $i - 1$. If we find more than this minimum number of matches, we know that these must be due to jumps. Let us call them *unexpected*.

Definition 7.2 (Unexpected CSMS). For the probe candidate starting at position i, the *unexpected* CSMS $\sigma^i := (\sigma^i_{k,\mu})$ are defined by

$$\sigma^i_{k,\mu} := \begin{cases} \mathrm{csms}_{i,\mu} & \text{if } k = 0 \text{ and } \mu \in [R^0_{\min}, 25], \\ \mathrm{csms}_{i+k,\mu} - \mathrm{csms}_{i+k-1,\mu+1} & \text{if } 0 < k \leq 25 \text{ and } \mu \in [R^0_{\min}, 25 - k], \\ 0 & \text{if } 0 < k \leq 25 \text{ and } \mu > 25 - k. \end{cases}$$

Note that σ^i_k is the null vector for $k > 25 - R^0_{\min}$.

To obtain a surrogate $L_{i,\delta}$ for longest common factor statistics, we first sum the unexpected CSMS over all positions k; this corresponds to the total number of unexpected matches of length $\geq \mu$ in $s_{i..i+24}$,

$$S_{i,\mu} := \sum_{k=0}^{25-R^0_{\min}} \sigma^i_{k,\mu}.$$

Now S is still a cumulative quantity; so we take first-order differences. Additionally. we express the quantity L_i as a function of $\delta = 25 - \mu$.

$$L_{i,\delta} := \begin{cases} S_{i,25} & \text{if } \delta = 0, \\ S_{i,25-\delta} - S_{i,25-(\delta-1)} & \text{if } \delta > 0. \end{cases}$$

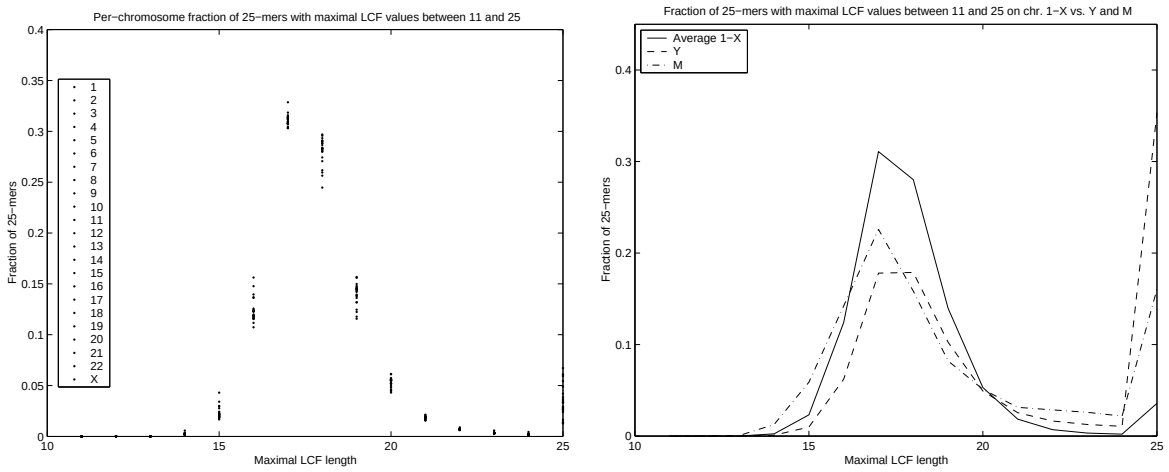


Figure 7.1: The distribution of the maximal LCF length is almost constant across all human chromosomes, except the Y chromosome and the mitochondrial DNA (possibly due to small size effects).

It should be noted that $L_{i,0} \geq 1$ because of $s_{i..i+24}$ itself; this match must be subtracted when computing the unspecificity according to (3.5), for example. We will assume that this has been done.

Results. Using the method as described above, we computed $L_i = (L_{i,\delta})$ for all 25-mers in the repeat-masked human genome. On a Compaq AlphaServer ES45 with four 64-bit 1 GHz processors (only one processor was in fact used) and 27 GB of main memory, the whole computation, including building the suffix arrays for all chromosomes, took approximately two days.

To visualize the obtained information, we consider $M = (M_i)$ with $M_i := 25 - \min\{\delta : L_{i,\delta} > 0\}$, i.e., the maximal LCF length of $s_{i..i+24}$ with other (non-overlapping) 25-mers in the genome. Interestingly, the distribution of M is relatively constant across all chromosomes (except the Y chromosome), with a peak at 17 and 18 nt, as shown in Figure 7.1. The distribution differs for the Y chromosome with almost 35% of the probe candidates having full-length matches elsewhere in the genome. We do not have an explanation for this effect. The distribution for the mitochondrial "chromosome" is also shown but is hardly comparable because of its relatively small size.

Figure 7.2 shows the cumulative distribution function of M, averaged over Chromosomes 1 through X. For example, About 15% of all 25-mers have a maximal LCF length of at most 16. Such probes, however, tend to occur in clusters, and not the whole genome can be tiled with them. As a consequence, we should expect a considerable level of cross-hybridization for genome tiling chips and be prepared to use non-unique probes.

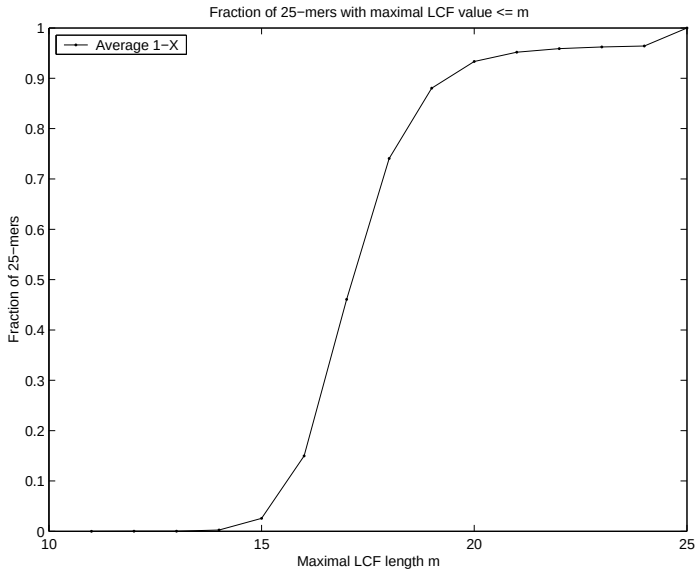


Figure 7.2: Cumulative distribution function of the maximal LCF length in the human genome. About 15% of all 25-mer probe candidates have a maximal LCF length of at most 16.

7.2 Decoding

Consider the illustration in Figure 7.3. The location of the true transfrags is unknown; thus we tile the genome with several (partially overlapping) hypothetical transfrags of constant length L and distances D, in such a way that each hypothetical transcript spans several probes. Intuitively, if (almost) all probes belonging to one of the hypothetical transfrags show a signal, we infer that the corresponding transfrag is a true one, unless the probe signals can be explained otherwise, i.e., by cross-hybridization or as false positive signals.

The more probe locations are spanned by hypothetical transfrags, the higher is our confidence in the predicted transfrags for constant error rates. To increase the confidence, we can thus either use a dense probe tiling or restrict the prediction to relatively long transfrags. For efficient decoding, it is desirable to keep the number of hypothetical transfrags as low as possible, but doing so also decreases the resolution. For example, in Figure 7.3 the left true transfrag is hard to detect because it spans only three probes and additionally occurs between two hypothetical transfrags, which have only a small overlap. Choosing a larger overlap would lead to many more (non-independent) hypothetical transfrags, increasing the problem size and the modeling complexity. On the experimental side, it is of course desirable to increase the probe number as far as possible for maximal resolution (i.e., use every 25-mer as a probe so that two consecutive probes overlap by 24 nt). On the other hand, longest common factor analysis

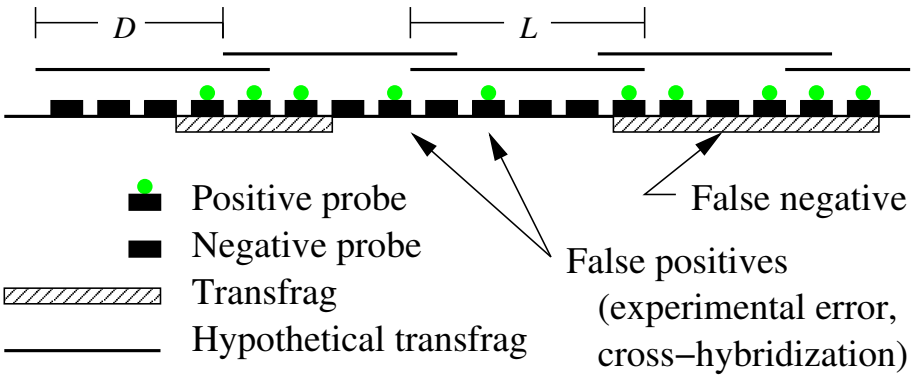


Figure 7.3: Transfrag discovery with genome tiling chips. Observed probe signals are to be explained with the presence or absence of hypothetical transfrags of length L. The distance between the start position of two consecutive transfrags is D.

may reveal that some probes are uninformative, because they have a high probability of cross-hybridization with other hypothetical transfrags. Additionally, the budget of such a large-scale project will only allow a limited number of chips and feature locations per chip, which dictates the probe spacing.

Studies on the human genome (Lander et al., 2001) have shown that the length distribution of internal exons of protein-coding genes peaks between 50 and 200 nt; therefore we may choose $L = 100$ and $D = 80$, say. It should be noted that the decoding can be repeated with arbitrary parameters of D and L once the experimental results are available. The size of one strand of the repeat-masked human genome is approximately $1.5 \cdot 10^9$ nt, resulting in $18.74 \cdot 10^6$ hypothetical transfrags and in $75 \cdot 10^6$ probes for a probe distance of 20 nt (so consecutive probes overlap by 5 nt). These numbers have to be doubled to account for the second strand. In this section we present a feasibility study of transfrag detection on a slightly smaller scale ($1 \cdot 10^6$ probes for 250000 transfrags).

Formally, we assume that the microarray experiments result in binary probe observations $y = (y_i)_{i=1,\ldots,m}$. For each of the n hypothetical transfrags $j = 1, \ldots, n$, we know which probes show a signal if the transfrag is expressed; let $P(j) \subset \{1, \ldots, m\}$ be this probe set for hypothetical transfrag j. Note that $P(j)$ typically contains more than the few probes spanned by t_j: it contains all probes i with high LCF value $\mathrm{lcf}(p_i, t_j)$. Our goal is to infer the status $x = (x_j) \in \{0, 1\}^n$ of each hypothetical transfrag $j = 1, \ldots, n$.

In principle, the decoding proceeds according to the Gibbs sampling procedure described in Section 6.3. We can exploit additional prior knowledge in this situation, however: We know some (but possibly not all) true transcripts and their locations by mapping known EST sequences to the genome (e.g., Mott, 1997). This knowledge allows to estimate the false negative rate f_0 of the probe signals, it enables us to fix

some $x_j = 1$ that can then be removed from the sampling procedure, and it helps to define the prior distribution on all possible solutions x.

We need to agree on a way to map the known true transfrags to present hypothetical transfrags; this can be difficult, e.g., for the left true transfrag in Figure 7.3. A reasonable strategy is to declare a hypothetical transfrag as present if more than half of its range is spanned by a true transfrag. Probes that are inconsistent with this decision are then treated as false positives or false negatives.

We thus assume that $x_j = 1$ for $j \in K \subset \{1, \ldots, n\}$, where K is the index set of hypothetical transfrags known to be present. Let $P_1 := \bigcup_{j \in K} P(j)$ be the probes whose signal is expected and explained by the presence of the known transcripts. Then we estimate the false negative rate f_0 by $|\{i \in P_1 : y_i = 0\}|/|P_1|$. The false positive rate is harder to estimate because we do not know the additionally present transfrags. It depends strongly on how we define $P(j)$ based on LCF statistics. If for example, $P(j) = \{i : \mathrm{lcf}(p_i, t_j) = |p_i|\}$, the probe sets are relatively small, but a considerable fraction of observed probe signals will be false positives because of unspecific hybridization. If on the other hand, we set $P(j) = \{i : \mathrm{lcf}(p_i, t_j) \geq |p_i| - 1\}$, the false positive rate is lower at the price of generally increased decoding complexity and a larger false negative rate.

Let us set $M := \{1, \ldots, n\} \setminus K$; so M contains the indices of transfrags whose status is unknown. For the prior distribution of a configuration x, we use a distribution that slowly decaying with the number of additional transfrags,

$$\mathbb{P}(x) \propto q^{e(x)} \qquad (q < 1;\ \ e(x) := \textstyle\sum_{j \in M} x_j).$$

The parameter q is chosen from the observation that the total prior probability of a solution x with e ones among the $|M|$ unknown components is proportional to $\binom{|M|}{e} \cdot q^e$, which takes its maximum at $e^* \approx (q|M| - 1)/(q + 1)$. If $e^* \ll |M|/2$ additional transfrags are expected, we thus set $q := (e^* + 1)/(|M| - e^*)$.

For simplicity, we work with a basic per-probe error model. Let $Y := \{i : y_i = 1\}$ denote the probes where a signal is observed, and let $P(x) := \bigcup_{j:\, x_j = 1} P(j)$ denote the probes where a signal is expected for solution x.

Using the notation of Section 6.2, we have the posterior probability

$$\pi(x) \equiv \mathbb{P}(x \,|\, y) \ \propto\ q^{e(x)} \cdot (1 - f_0)^{|Y \cap P(x)|} \cdot f_0^{|P(x) \setminus Y|} \cdot (1 - f_1)^{m - |Y \cup P(x)|} \cdot f_1^{|Y \setminus P(x)|},$$

where $e(x) := \sum_{j \in M} x_j$. Note that $P(x)$ is always a superset of P_1. The Markov chain moves as described in Section 6.2, paragraph "Sampling from the posterior (Gibbs sampler)".

Sampling efficiency is improved by noting that the likelihood of a solution x, in which hypothetical transfrags with no or very few positive probes are present, will be low

because of the $f_0^{|P(x)\setminus Y|}$ term and because the prior prefers sparse solutions. We may thus fix $x_j := 0$ for $j \in N$, where N is the set of absent transfrags determined by this filtering step. Sampling is now restricted to coordinates from $J := M \setminus N = \{1, \ldots, n\} \setminus (K \cup N)$.

Simulations. To assess the feasibility of genome-wide decoding in the presence of high error rates, we performed large scale simulations with $n = 250000$ hypothetical transfrags and $m = 10^6$ probes; we assume that each transfrag is covered by 5 probes on average and that there is an overlap of one probe between two consecutive transfrags. Further we assume that on average $\chi = 3\%$ of the transfrags also hybridize to probes intended to cover a different transfrag; unspecific-cross hybridization is additionally modeled by a large false positive error rate (see below). This setting is devised to simulate on a smaller scale (total "genome" size 20 million nt) the problems that are expected to occur during decoding of experiments on the human genome. Transfrags (multi-exon genes) are distributed randomly in the genome; because of the small genome size, we arranged a relatively high transcript density of about 7% of the genome. A random 50% of the transcriptome is assumed to be known beforehand; the goal is to correctly infer the remaining 50% from noisy probe signals with realistically high and per-probe error rates ($f_0 = 10\%$, $f_1 = 15\%$).

Of the 250000 hypothetical transfrags, on average 8750 (half of 7%) are known to be present; on these, we are able to estimate the false negative probe rate reliably. Pre-filtering (at most one of the five consecutive probes spanning a transfrag shows a signal) results in about 188000 transfrags to be declared as absent; this step considerably reduces the problem complexity and leads only to a small fraction of erroneous decisions: For 5-fold covered transfrags, it is improbable that at most one probe shows a signal if the transfrag is truly present (the probability is $(f_0)^5 + \binom{5}{1} \cdot (f_0)^4 \approx 1/2000$). On the filtered transfrags, we estimate the false positive rate f_1, but it is clear that this estimate is biased and that the true rate needs to be higher. Our simulations show that good results are obtained when the estimated rate is multiplied by 1.5.

Pre-filtering leaves the status of about $|J| \approx 53000$ transfrags to be sampled; their indices are denoted by the set J. We initially estimate the number of present transfrags by the number of transfrags e^* for which at least three covering probes show a signal, thus overestimating the true number. We set the prior parameter $q := (e^*+1)/(|M| - e^*) \approx 0.45$, as described above. This decision biases the samples towards containing too many transfrags, but it can be compensated later by a strict posterior threshold.

For Gibbs sampling, we choose 100 burn-in iterations (each consists of $|J|$ single-coordinate steps) and 4100 iterations in total; we observed that even reducing these parameters to 50 burn-in steps and 1000 total steps did not lead to differences in the results. During each iteration after the burn-in phase, the $|J|$ full conditional

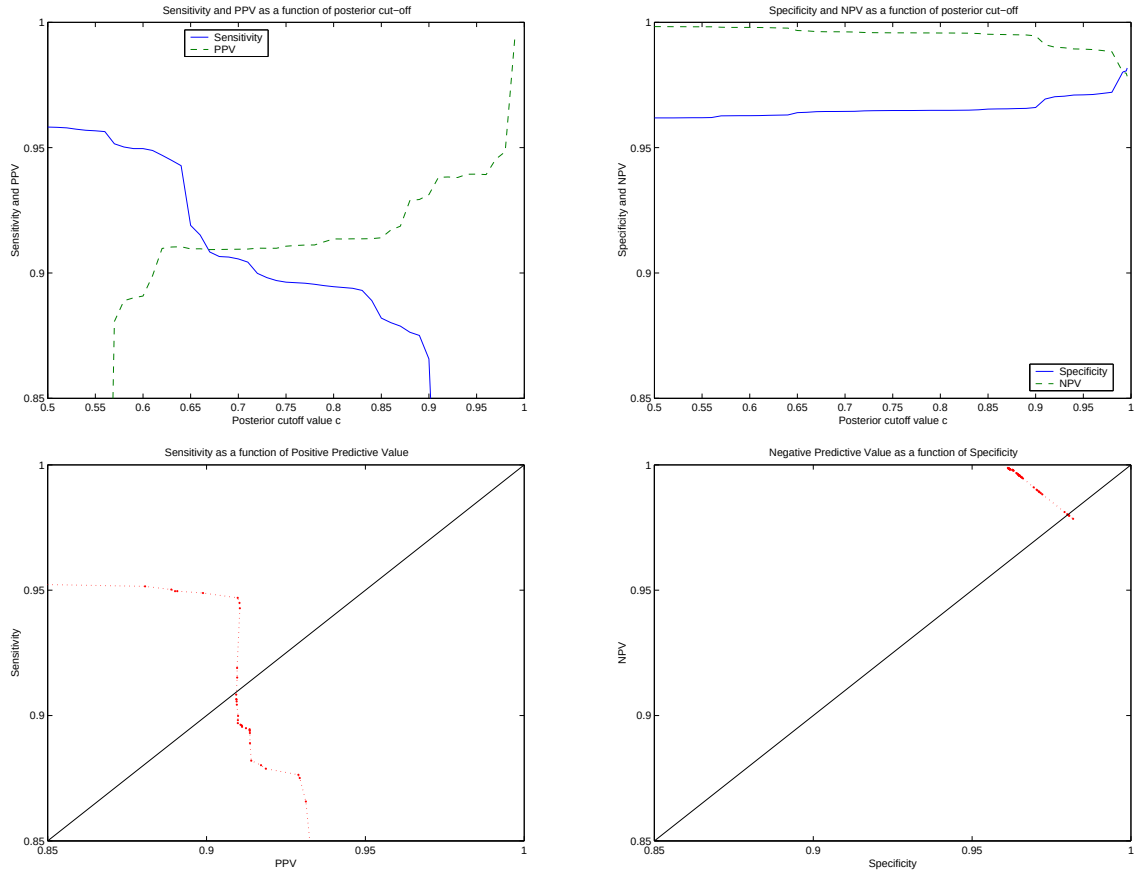


Figure 7.4: Decoding results for genome tiling chips when 3% of all transfrags have probes that hybridize to other transfrags as well. Results of one representative simulation are shown; they are virtually identical across repetitions with the same parameters.

probabilities $\mathbb{P}(x_j = 1 \,|\, y, (x'_j)_{j' \neq j})$ are estimated for $j \in J$ and averaged over all iterations to estimate the marginals $\pi(x_j = 1)$.

Transfrags with high marginal estimates $\pi(x_j = 1) \geq c$ are declared as present and called *positives*; the remaining transfrags are called *negatives*. These terms and the following discussion only refer to the transfrags $j \in M$ of unknown status. A positive transfrag is a *true positive* (TP) if it represents a real transfrag; otherwise it is a *false positive* (FP). A negative transfrag is a *true negative* (TN) if it represents no real transfrag; a real transfrag that is not recognized is a *false negative* (FN). As we are working with simulated data, we can determine all of the following characteristics as a function of the threshold c.

- The *sensitivity* is defined as the fraction of true positives among all real transfrags (TP/(TP+FN)).
- The *positive predictive value* (PPV) is the fraction of true positives among all declared positives (TP/(TP+FP)).

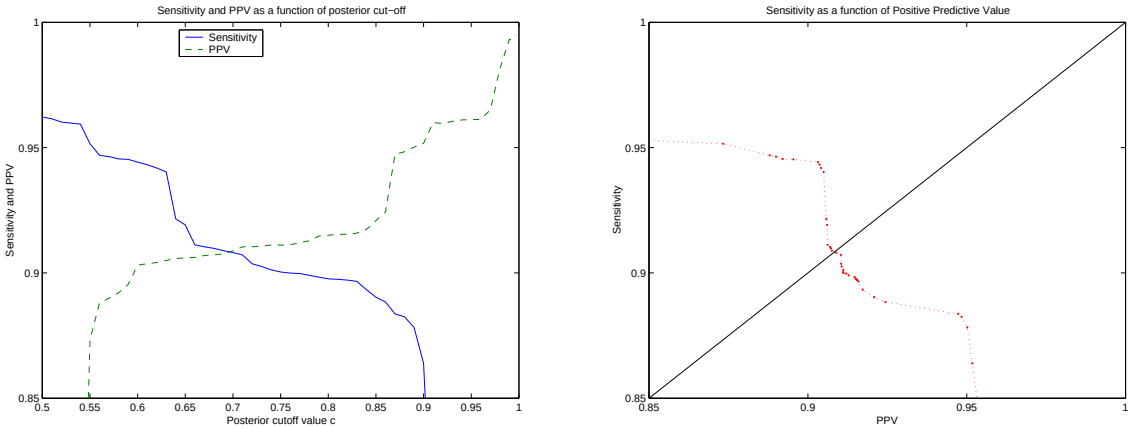


Figure 7.5: Decoding results for genome tiling chips when 20% of all transfrags have probes that hybridize to other transfrags as well.

- The *specificity* is the fraction of true negatives among all declared negatives (TN/(TN+FP)).
- The *negative predictive value* (NPV) is the fraction of true negatives among all declared negatives (TN/(TN+FN)).

Figure 7.4 shows that in the setting we described, a posterior cut-off of about 0.9 plays a critical role with a pronounced drop in sensitivity for larger thresholds. Choosing the threshold at about $c \approx 0.64$ leads to a specificity of 94% and a PPV of 91%, but there are drops in sensitivity for slightly larger c and in PPV for slightly smaller c. In practice, one does not know the optimal threshold. For cut-offs between 0.65 and 0.85, results are relatively stable and achieve a sensitivity and PPV of about 90% at the same time (i.e., 90% of the true transfrags are found and 90% of the found transfrags are correct). This is a very good result for the high error rates we consider. Naturally, the problem of finding the true negative transfrags is easier because of their larger number. Over the whole range of thresholds, we achieve a specificity and PPV above 96%.

We repeated the same analysis in a more difficult situation where $\chi = 20\%$ (instead of 3%) of the transfrags hybridize to one or several additional probes beyond those that are intended to cover them. Figure 7.5 shows that the results change only marginally; the false positive errors are still dominating the non-unique probe effect. Therefore no problem is expected for decoding the human transcriptome, even if many probes hybridize to multiple transfrags in the human genome.

In the previous experiments, we estimated the true false positive and false negative rates f_1 and f_0 rather well, but what happens, for example, if f_1 is not heuristically corrected by a factor of 1.5, as described above? Figure 7.6 shows that the range of posterior cutoff values that result in both 90% sensitivity and PPV has moved to between 0.8 and 0.95.

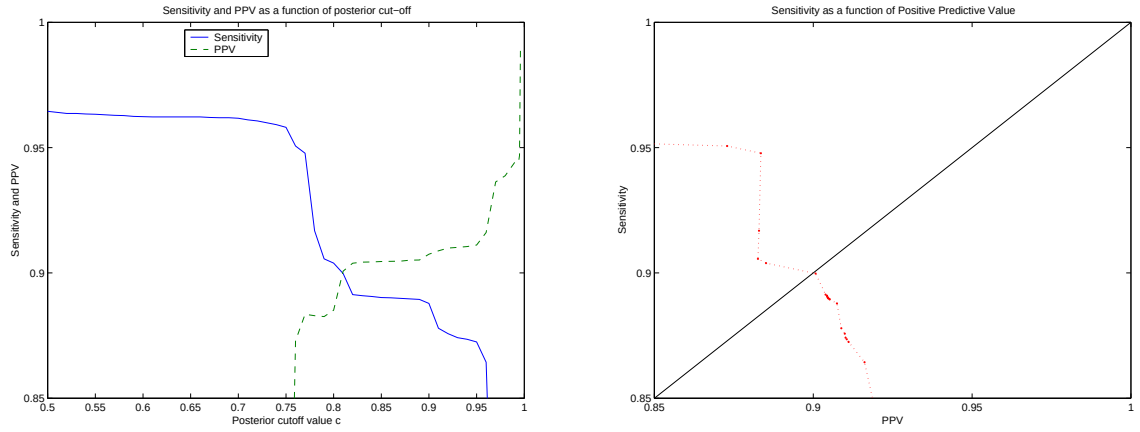


Figure 7.6: Decoding results for genome tiling chips when the false positive error rate is underestimated: An estimate of 10% instead of the true 15% is assumed.

In practice, we do not know the true error rates, and we also do not know the optimal threshold. However, our simulations indicate that a threshold of $c \approx 0.85$ is a robust choice.

7.3 Summary

The design and analysis of genome tiling chips poses many challenges, such as finding a balance between the experimental costs and high resolution, or between the probe coverage of transfrags and the decoding complexity. Many overlapping hypothetical transfrags would require a model that additionally takes correlations between consecutive transfrags into account. For our simulation studies, decoding runs were in the range of hours, but even without model refinement, we estimate that the required time for one decoding run on the human genome would be about one week.

Our studies also show that, if there are unknown transcribed regions hidden in the genome, we have a good chance to detect them with well designed genome tiling chips. Additionally, what we detect has a good chance of being correct, even with the (realistically) high and unknown error levels that we assumed. Our results are encouraging for large scale projects such as the ENCODE project; and we expect that it will lead to many new insights concerning the human transcriptome.

Chapter 8

Optimization of the Deposition Sequence for Chip Production

After treating probe selection and signal decoding in various frameworks in the previous chapters, we now take a look back at the chip production phase.

Recall the photolithographic probe synthesis process described in Section 1.3.1. All probes are synthesized on the chip in parallel on a nucleotide-by-nucleotide basis. In each synthesis step, the same nucleotide is appended to all probes that have been selectively activated to receive it. Activation occurs by exposure to light, enabling the chemical synthesis reaction. Thus each synthesis step is succinctly described by

1. a nucleotide (a character from {A,C,G,T})

2. a *mask*, that is, an index set of probes that do not grow during this step, or alternatively, the complementary index set indicating the activated probes to which the nucleotide is appended.

The sequence of nucleotides used in the chip synthesis process is called the *deposition sequence*. Each probe is a subsequence of the deposition sequence, so the deposition sequence is a common supersequence of all probes. In the introduction, we have given several reasons why the deposition sequence should be kept as short as possible. Therefore the aim of this chapter is to solve Problem 9, the *shortest common supersequence problem* (SCSP) in a microarray production setting, where the SCSP has several characteristics which distinguish it from previously studied SCSPs in other settings: We have many short sequences (10000 to 100000, recently up to a million sequences of 20 to 30 base pairs), and the alphabet is small (four nucleotides). Previous studies considered the problem with reversed dimensions, i.e., with fewer but much longer sequences.

8.1 Previous Work and our Approach

Known heuristics for the SCSP (see Irving and Fraser, 1995; Jiang and Li, 1995) are

- ALPHABET, the obvious factor-$|\Sigma|$ approximation: Let π be a permutation of the characters in Σ and let L be the length of the longest input sequence. Let s be the sequence obtained by L-fold repetition of π. Clearly s is a supersequence of every sequence of length $\leq L$ over Σ. Also, the SCS must have length at least L; therefore s is within a factor of $|\Sigma|$ of the optimum. As our statistical analysis will show, this method is hard to beat for many input sequences of constant length. Therefore and because it is easily implemented, it is the current standard for microarray production.

- Clustering approaches that replace a pair of sequences by one of their shortest common supersequences in every step until only one sequence remains. Irving and Fraser (1995) describe, for example, heuristics called TOURNAMENT and GREEDY that fall into this category. These algorithms do not have a guaranteed performance ratio.

- Algorithms that move a front through all sequences from left to right. In each step, the method picks the next character of the supersequence based on the partial sequences to the right of the front. Then the front advances in those sequences that can use the picked character. A classical example is MM, the Majority Merge algorithm, that appends the most frequent symbol at the current front position.

More recently, several publications on the SCSP have appeared proposing new heuristics. For example, character selection in Majority Merge can be improved by considering the remaining string length in each step. Combining heuristics with genetic algorithms has been shown to give good performance (Branke et al., 1998).

None of the above methods has been designed for or applied to the microarray production setting with thousands of sequences. The running time of clustering methods depends at least quadratically on the number of input sequences, which is prohibitive. Therefore we use ALPHABET to obtain an initial supersequence, and develop *refinement methods* that improve an existing supersequence by local modifications. We begin by stating the required definitions. Then we examine the efficiency of the so-called ALPHABET-LEFTMOST heuristic in the microarray setting, which defines the baseline for potential improvements. We continue by describing our refinement methods for supersequences, and conclude with computational experiments and a summary of our findings.

8.2 Definitions

Let Σ be a finite alphabet of size $\sigma = |\Sigma|$ (in the microarray production setting, $\sigma = 4$). In the following, we consider strings over Σ. When s is a string, we write $|s|$ for its length and use subscripts to refer to the individual characters: $s = (s_1, \ldots, s_{|s|})$.

Definition 8.1 (Subsequence; Supersequence). A string s' is a *subsequence* of s when s' can be obtained by deleting some (possibly zero or all) characters from s without changing the order of the remaining characters. We also say that s is a *supersequence* of s' in this case. When I is a subset of $\{1, \ldots, |s|\}$, we write s_I for the subsequence $s' = (s_i)_{i \in I}$.

Definition 8.2 (Common Supersequence). A string s is a *common supersequence* of a set $\mathcal{S}$ of strings if each string in $\mathcal{S}$ is a subsequence of s.

We use the term *step* to refer to a position within a supersequence, while we reserve the term *position* for positions within sequences from $\mathcal{S}$.

Definition 8.3 (Embedding). Let $\mathcal{S}$ be a set of m strings, $t \in \mathcal{S}$, and let s be a common supersequence of $\mathcal{S}$ of length n.

An *embedding* of t in s is a binary vector $e_t = (e_{t,1}, \ldots, e_{t,n}) \in \{0,1\}^n$ or equivalently the index set $I_t := \{j \,|\, e_{t,j} = 1\} \subset \{1, \ldots, n\}$, such that $s_{I_t} = t$. Note that for any embedding I_t we have $|I_t| = |t|$.

An *embedding matrix* of the set $\mathcal{S}$ in s is a binary $m \times n$-matrix E whose i-th row is an embedding of the i-th string of $\mathcal{S}$ in s.

Definition 8.4 (Leftmost Embedding). An embedding e_t of t in s is called the *leftmost embedding*, when $\sum_{j' \le j} e_{t,j'} \ge \sum_{j' \le j} e'_{t,j'}$ for all other embeddings e'_t and all steps j. Intuitively, t uses the characters of s as soon as possible from left to right.

An embedding matrix E is called the *leftmost embedding matrix*, when every row is the leftmost embedding of its associated string.

Definition 8.5 (Productivity of Steps). Let E be an embedding matrix of the sequence set $\mathcal{S} = \{t_1, \ldots, t_m\}$ in the supersequence s. We say that sequence t_i is *masked* in step j and that step j is *unproductive for sequence* t_i when $e_{t_i,j} = 0$. Sequence t_i is *unmasked* in step j and step j is *productive for sequence* t_i when $e_{t_i,j} = 1$.

Sequence t_i is *complete in step* j when $\sum_{j' < j} e_{t_i,j'} = |t_i|$; otherwise it is *incomplete* (it might be completed in step j; then it is complete in step $j+1$). The *completion step* of sequence t_i is the largest j where t_i is incomplete.

The *productivity* of step j is the number of sequences for which step j is productive in relation to the number of incomplete sequences in this step. A step with productivity zero is called *unproductive*. An unproductive step can be safely removed from a supersequence.

Let us illustrate these definitions with an example.

Example 8.6. We consider a set $\mathcal{S} = \{t_1, t_2, t_3\}$ of three probes of length 7.

- $t_1 =$ `ACGTTAG`
- $t_2 =$ `CGAGTCA`
- $t_3 =$ `AGAGCAG`

One possible deposition sequence (supersequence) for $\mathcal{S}$ is obtained by cycling through the alphabet as long as necessary: $s =$ `ACGTACGTACGTACG` suffices, as shown by the following embedding matrix.

s	A	C	G	T	A	C	G	T	A	C	G	T	A	C	G
e_{t_1}	1	1	1	1				1	1		1				
e_{t_2}		1	1		1		1	1		1			1		
e_{t_3}	1		1		1		1			1			1		1
										1	1	1	1	1	1
Step	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5

Sequence t_1 completes in step 11; thus it is complete in all steps ≥ 12. It is masked in steps 5, 6, 7, 10, 12, 13, 14, and 15. There are three unproductive steps for each sequence. Overall, there are three unproductive steps; these are 6, 12, and 14 and can be removed from s. The embedding of t_3 in s can be either described by the binary vector e_{t_3} in the third row of the given binary matrix, or equivalently by the index set of the seven unmasked steps $I_{t_3} = \{1, 3, 5, 7, 10, 13, 15\}$.

8.3 Alphabet-Leftmost and its Stochastic Properties

The following method, called Alphabet-Leftmost, will be used as a baseline to compare other methods against. The input set $\mathcal{S}$ consists of m strings over Σ, the i-th string having length L_i.

Alphabet-Leftmost

1. Let π be any fixed permutation of the letters in Σ; let L be the length of the longest input string in $\mathcal{S}$.
2. Set $s' := \pi^L$. Compute the leftmost embedding matrix E of $\mathcal{S}$ in s'.
3. Remove all unproductive steps from s' to obtain the result s, and let $\mathcal{U} := |s|$.

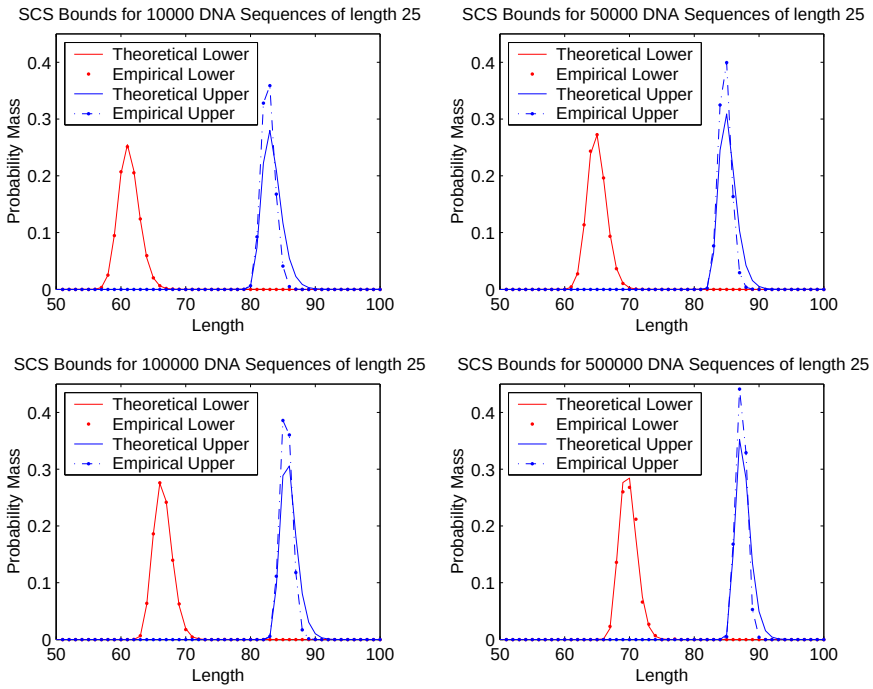


Figure 8.1: Empirical distribution of the lower and upper bounds $\mathcal{L}$ and $\mathcal{U}$ as found by random simulations ("Empirical Lower", "Empirical Upper"), and theoretically computed distribution of $\mathcal{L}$ and C; the latter serves as an approximation to $\mathcal{U}$ ("Theoretical Lower", "Theoretical Upper"). The four panels show the distributions for 10K, 50K, 100K, and 500K DNA 25-mers, respectively.

Running Alphabet-Leftmost gives us an upper bound $\mathcal{U}$ on the length of the SCS. When the alphabet is small, we may run Alphabet-Leftmost on every permutation of Σ and pick the best bound.

A lower bound is also easily obtained. For $i = 1, \ldots, m$ and $x \in \Sigma$, let $N_i(x)$ denote the number of occurrences of character x in the i-th sequence, and define $N(x) := \max_{i=1,\ldots,n} N_i(x)$. Clearly, every common supersequence must contain at least $N(x)$ occurrences of x. Thus a lower bound on its length is given by $\mathcal{L} := \sum_x N(x)$. The typical range of $\mathcal{L}$ and $\mathcal{U}$ for π =(`A,C,G,T`) and 10000 to 500000 uniformly drawn 25-mers is shown in Figure 8.1.

The rest of this section contains an analysis of the distribution of $\mathcal{L}$ and $\mathcal{U}$ under a uniform random input model. This model is appropriate for the microarray production SCSP because the oligo selection process will usually force a uniform distribution on the nucleotides because departures from uniformity increase the average similarity between oligos and hence the risk of cross-hybridization.

Distribution of the Lower Bound $\mathcal{L}$. Since $N_i(x)$ is the number of occurrences of character x in the i-th sequence, $N_i(x)$ has a Binomial distribution with parameters L_i and $1/\sigma$. Furthermore, for fixed x, the N_i are independent. It follows that

$$\begin{aligned}\mathbb{P}(N_i(x) \le k) &= \sum_{j=0}^{k} \binom{L_i}{j} \cdot \sigma^{-j} \cdot (1-1/\sigma)^{L_i-j}, \\ \mathbb{P}(N(x) \le k) &= \prod_{i=1}^{m} \mathbb{P}(N_i(x) \le k).\end{aligned}$$

Recall that $\mathcal{L} := \sum_x N(x)$. While the random variables $N(x)$ are not completely independent, they can be treated as such in good approximation when the alphabet size is small compared to the number of oligos. Intuitively, the value of $N(x)$ has little influence on the other values $N(y)$ for $y \neq x$ (unless $N(x)$ is small), because the maxima over i of $N_i(x)$ and $N_i(y)$ can stem from many different sequences. Thus we may well approximate the distribution of $\mathcal{L}$ by the convolution of the distributions of $(N(x))_{x\in\Sigma}$. Note the exact agreement of this theoretical distribution with the empirical distribution in Figure 8.1.

Approximate Distribution of the Upper Bound $\mathcal{U}$. For $k = 1, \ldots, L_i$, let $W_{i,k}$ be the number of consecutive unproductive steps between the $(k-1)$-st and k-th productive step for sequence i. Since the sequence characters are independent and uniformly distributed and the deposition sequence is periodic, the $W_{i,k}$ are independent and uniformly distributed on $\{0,1,2,3\}$. We write the completion step of sequence t_i as

$$C_i = L_i + \sum_{k=1}^{L_i} W_{i,k}.$$

From this representation, we may compute the distribution of C_i explicitly by L_i-fold convolution of the uniform distribution on $\{0,1,2,3\}$, or approximate it by a Gaussian distribution with mean $2.5\,L_i$ and Variance $1.25\,L_i$ with the Central Limit Theorem. We set $C := \max_i C_i$ and compute its distribution with the product formula $\mathbb{P}(C \le c) = \prod_{i=1}^{m} \mathbb{P}(C_i \le c)$. For m probes of the same length L, using the normal approximation results in

$$\mathbb{P}(C \le c) \approx \left[\Phi\left(\frac{c + 0.5 - 2.5\,L}{\sqrt{1.25\,L}}\right)\right]^m,$$

where Φ denotes the cumulative distribution function of the standard Normal distribution, and adding 0.5 to c represents a continuity correction.

Note that $\mathcal{U} \le C$. Both C and $\mathcal{U}$ do not count removed unproductive steps at the end of the periodic deposition sequence, but C does count internal unproductive steps

that are removed by Alphabet-Leftmost and hence not counted in $\mathcal{U}$. When the number of sequences is large, internal unproductive steps are rare, and will only occur near the end of the deposition sequence when only few incomplete sequences are left. The plots in Figure 8.1 show the discrepancy in the right tail between the distribution of $\mathcal{U}$ obtained by simulations and the distribution of C, which we use as an approximation to $\mathcal{U}$.

8.4 Refining Supersequences

The results of the previous section show that there is only a limited potential for improvement in comparison to Alphabet-Leftmost. Additionally, we cannot allow the search to take more than a few minutes if we want the method to be practical in custom microarray production where each chip has its unique design. Therefore we are considering methods that refine existing supersequences and whose execution time is easily controlled. Their aim is to increase overall productivity and reduce the number of low-productivity steps.

8.4.1 Supersequence Editing

First, we consider three *edit operations* on the supersequence $s = (s_1, \ldots, s_{|s|})$.

- Delete(k): We shorten the supersequence by removing s_k. Every sequence whose embedding uses step k must be re-embedded, and its new embedding may not complete in the shorter supersequence. In such a case we have to append characters to the supersequence until each sequence is properly embedded. A Delete-operation may thus shorten the supersequence by one character (when all sequences can be embedded without appending characters), leave the length of the supersequence unchanged (when exactly one character is appended), or even lengthen the supersequence. In the latter case, using the delete operation may not be advisable.

- Insert(k, a): We insert character a before step k (assuming $s_k \neq a$; otherwise we may equivalently insert before step $k+1$) and compute a new leftmost-embedding of all sequences whose first productive step after $k-1$ appends an a. While the Insert-operation lengthens the supersequence at first, it may result in one or more unproductive steps near the end of the supersequence. These steps are then removed, potentially shortening the supersequence by several characters.

- Twiddle(k) for $k = 1, \ldots, |s| - 1$: We switch the characters s_k and s_{k+1} (assuming they are not equal) and re-embed every sequence that was unmasked in step $k+1$. This reduces the completion time of some sequences, while increasing

s	A	C	G	T	A	G	T	A	C	G	A	G
e_{t_1}	1	1	1	1			1	1		1		
e_{t_2}		1	1		1	1	1		1		1	
e_{t_3}	1		1		1	1			1		1	1
										1	1	1
Step	1	2	3	4	5	6	7	8	9	0	1	2

(a) ⇓

s^+	A	C	G	T	A	G	T		C	G	A	G
$e^+_{t_1}$	1	1	1	1			1				1	1
$e^+_{t_2}$		1	1		1	1	1		1		1	
$e^+_{t_3}$	1		1		1	1			1		1	1

Original supersequence and embedding

(b) ⇓

s^+	A	C	G	T	A	G	T	A	C	A	G	G
$e^+_{t_1}$	1	1	1	1			1	1			1	
$e^+_{t_2}$		1	1		1	1	1		1	1		
$e^+_{t_3}$	1		1		1	1			1	1	1	

Figure 8.2: Effects of supersequence edit operations on the sequences of Example 8.6, with non-productive steps in the supersequence s already removed. The INSERT operation is not illustrated. (a) Effect of the DELETE operation at step 8: Sequence t_1, which was unmasked in this step must be re-embedded, making step 10 unproductive. Thus two characters (A and G) can be removed from the supersequence. (b) Effect of the TWIDDLE operation at step 10: All three sequences must be re-embedded to the right of step 10, making step 12 unproductive, which can then be removed.

the completion time of others. A TWIDDLE-operation may shorten or lengthen the supersequence by several characters, depending on the circumstances.

The edit operations are visualized in Figure 8.2. They can be applied with different strategies. Two immediate examples for a strategy are the following ones.

- MCMC: We define a quality function for every supersequence. This can be a function that increases exponentially with decreasing sequence length, and that is otherwise high when the minimum productivity among all steps is small, as such a step may potentially be removed. Given such a quality function, we propose an edit operation randomly and accept it with a probability that corresponds to

the quality ratio of new and old supersequence (if the quality improves, the edit operation is always accepted). This procedure is repeated several times.

- BEST: We evaluate all $5|s| - 1$ possible edit operations and pick one that leads to the best quality. This procedure is iterated until no further improvement is possible.

Our experiments showed that the second strategy finds a local quality maximum after only a few iterations. The MCMC strategy explores the sequence space more thoroughly, but does not usually find shorter supersequences in a reasonable amount of time.

8.4.2 Suffix Enumeration

Once we find that edit operations do not improve the supersequence further, we switch to a branch-and-bound method that enumerates additional sequences for a given given amount of time. Assume that the shortest currently known supersequence s^* has length ℓ. We want to find out whether there are supersequences of length $\ell - 1$ or shorter. In principle, we could enumerate every sequence of length $\ell - 1$, but this is computationally prohibitive.

Consider any k-prefix ($k \leq \ell - 1$) of a potential supersequence. We can partially leftmost-embed all sequences into this prefix and obtain the lower bound $\mathcal{L}$ (see above) on the *remaining* supersequence length by adding the maximal letter counts for the remaining partial sequences. Whenever $k + \mathcal{L} \geq \ell$, we can immediately move to the next k-prefix, or $(k-1)$-prefix if all k-prefixes have been evaluated. Only if $k + \mathcal{L} < \ell$, we must consider all possible extensions of the current prefix, as they may still lead to a supersequence of length less than ℓ. Two modifications make this approach more efficient in practice.

First, we can sometimes improve the lower bound as follows. We find the remaining partial sequence t_A with the highest count of As $N'(\mathtt{A})$, and similarly partial sequences $t_\mathtt{C}, t_\mathtt{G}, t_\mathtt{T}$ with C-, G-, and T-counts $N'(\mathtt{C})$, $N'(\mathtt{G})$, and $N'(\mathtt{T})$. Instead of simply computing $\mathcal{L} = \sum_x N'(x)$, we compute the length $\mathcal{L}'$ of a shortest common supersequence of $t_\mathtt{A}, t_\mathtt{C}, t_\mathtt{G}, t_\mathtt{T}$ by four-dimensional dynamic programming. This is possible because the partial oligo sequences are short. Clearly $\mathcal{L}' \geq \mathcal{L}$, and generally the bound is improved by one or two characters.

Second, we do not enumerate the sequences alphabetically. We begin with the $\ell - 1$-prefix of s^*, and change the character at step k only after all suffixes of length $\ell - k - 1$ have been explored. The sequences we can explore in this way all share a common prefix whose length depends on the permitted running time. In practice, we feel that 5 to 10 minutes are a reasonable time; sometimes this suffices to shorten the supersequence by one additional character.

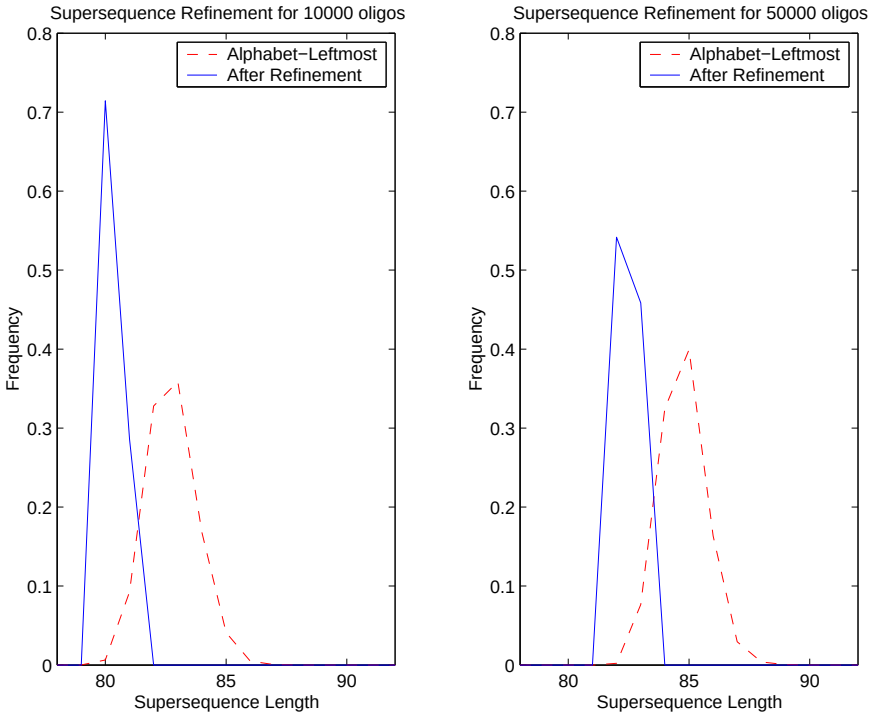


Figure 8.3: Effects of supersequence refinement for 10000 and 50000 random uniformly distributed 25-mers, respectively.

8.5 Computational Experiments

To measure the impact of our heuristics in a practically relevant setting, we created several sets of 10000 and 50000 uniformly distributed 25-mers. For each of the 24 permutations of the alphabet's characters, we construct a supersequence by running Alphabet-Leftmost, and attempt to shorten it using the edit-strategy Best.

With one of the shortest sequences found in this process, we run the branch-and-bound suffix enumeration for 5 minutes. The results are as follows: On average, the supersequence is shortened from 82.7 by 2.4 to 80.3 characters for 10000 oligos, and from 84.8 by 2.3 to 82.5 characters for 50000 oligos. The distribution of the supersequence length after Alphabet-Leftmost and after supersequence refinement are shown in Figure 8.3.

8.6 Summary

The stochastic analysis of Alphabet-Leftmost shows that the supersequence optimization potential is small in the microarray production setting, especially since the

lower bound is generally not tight. We have shown that it is possible to shorten the supersequence on average by 2.4 characters for 10000 sequences and 2.3 characters for 50000 sequences using simple edit operations and a branch-and-bound suffix enumeration method. Even this seemingly small improvement is important for error reduction during oligo synthesis. For chips with 10000 25-mers, the number of masked steps, and hence the potential number of erroneous synthesis steps due to stray light, is reduced from 57.7 to 55.3 (by 4.2%), which is a respectable result for a few minutes of computation time. Further error reduction techniques (such as synthesizing oligos with similar embeddings next to each other, as described by Kahng et al. (2002), for example) will be easier to apply after shortening the deposition sequence.

Other approaches like Greedy or Tournament that do not attempt to refine an existing supersequence but build one from the individual sequences fail in the microarray production setting, generally leading to supersequences much longer than the lower bound (data not shown). This is not surprising because these heuristics were designed with the idea in mind that we need to integrate relatively few but longer and rather similar sequences into one supersequence.

An interesting variation of the SCSP for microarrays was recently suggested by Alain Guénoche (personal communication; submitted work): Assume that we would like to have each probe represented at two spots for safety and repeatability reasons. To guard against failure of certain synthesis steps, it would be best if the two instances of each probe are synthesized by disjoint sequences of steps. Therefore we are interested in an algorithm that decides whether a probe is 2-embeddable, i.e., can be embedded twice into a given supersequence with disjoint sets of steps, and in a method to find the shortest common supersequence of a set of probes such that each probe is 2-embeddable. Guénoche provides a dynamic programming algorithm to solve the first problem.

Chapter 9

Discussion and Outlook

This thesis makes several contributions to the field of probe selection and microarray design. We classify them into conceptual contributions, algorithms, stochastic results, and software packages.

Conceptual contributions. We introduce the *longest common factor (LCF) approach* for short oligonucleotide probe selection. It is based on the assumption that the affinity coefficient A_{ij} between probe i and target j can be approximated as a function of the length of the longest common substring of probe i and target j.

It is clear that the LCF length cannot predict the true affinity coefficients exactly. In Chapter 2 we present a factorization of the affinity coefficient into different components and discuss methods for avoiding extreme cases. Although the exact factors depend on the technology platform, in principle the same issues arise for all high-density oligonucleotide arrays. We have not found a systematic discussion of affinity coefficient modeling in the existing literature, and we believe that our contribution about the nature of affinity coefficients in Section 2.1 can be serve as a general basis for both refined probe selection methods and microarray data analysis.

The LCF approach is compared to other methods in Section 3.3, and we shall not repeat this discussion here. We point out, however, that the LCF approach is appropriate for short oligonucleotides because any stable hybridization needs a stable core and it is unlikely that there are several strong stable cores in oligonucleotides shorter than 30 nt in length. With longer polynucleotides, the probability that a probe has several stable hybridization possibilities increases, and there is less reason to believe in the validity of the LCF approach. Therefore the focus of this thesis is on short probes, such as 25-mers on the GeneChip® arrays. In contrast to other sequence-based quantities, such as the minimal edit distance between a probe and any substring of a target, the LCF length has the advantage of being efficiently computable, which allows genome-wide chip designs in large genomes, such as the genome tiling chips described in Chapter 7. Compared to the method published by Li and

Stormo (2001), the LCF approach is faster by two orders of magnitude even on small genomes (baker's yeast; see Section 4.7).

We are not aware of any previous work that distinguishes conceptually between *targets*, *transcripts*, and *target collections*, even though this distinction has obvious benefits, as discussed in Section 4.1. Most other approaches also do not consider probe similarity *statistics* with respect to all transcript collections, such as longest common factor statistics. Instead, other methods reduce the similarity information to the maximal value, which is misleading when many nearly equivalent high values exist.

Efficient computation of LCF lengths is based on vectors of matching statistics (Chapter 4). To store and process these efficiently, we introduce the concept of *jumps in matching statistics* (Section 4.4) and analyze their properties. We prove relationships between jump sets and LCF lengths. The utility of jumps extends beyond reduced memory requirements for storing matching statistics and the benefit of fast LCF computation: Let p and t be strings, and suppose that $f : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is any real-valued function of two strings of the form

$$f(p,t) = \max_{p' \lhd p;\, p' \lhd t} g(p').$$

Examples used in this thesis are $f(p,t) := \mathrm{lcf}(p,t)$ with $g(p') := |p'|$ and $f(p,t) := \max_{p' \lhd p;\, p' \lhd t} (-\Delta_{\mathrm{r}} G^{\circ}(p'))$, where $\Delta_{\mathrm{r}} G^{\circ}(p')$ is the standard Gibbs energy of duplex formation of p' according to a Nearest Neighbor Model with zero or low initiation terms (e.g., those from Tables 2.1 and 2.2 in Section 2.3.2). For all f of the mentioned form, it suffices to evaluate $g(p')$ for substrings p' that begin at jumps in $\mathrm{ms}^{p|t}$ (see also Section 4.6); this leads to considerable savings in the evaluation of $f(p,t)$.

In the existing literature, little has been said about using non-unique probes for gene expression analysis or species identification. Recall that we define non-unique probes in a way that still enforces specificity, but allows the set of intended targets to contain more than one element (Definition 6.2). Allowing non-unique probes complicates both the *design* of a chip and the *analysis* of probe signals; a *decoding* step is required to obtain expression values or presence/absence calls from the measured signal intensities. In fact, a simple form of decoding is also required with unique probes; it consists of the normalization and robust averaging steps described in Section 2.2. Quantitative analysis with non-unique probes requires robustly solving a linear system; this leads to the design objective of *minimizing the condition* of the affinity matrix. For qualitative analysis (de-OR-ing), we have adapted and modified an MCMC-based decoding method previously used in clone library screening and proposed (joint work with Alexander Schliep). A good chip design is characterized by the probes' ability to *separate* many target sets, as formalized in Definition 6.10, and a small number of probes. Both objectives proposed in this thesis, condition minimization under coverage constraints and probe number minimization under separation constraints, are similar in spirit, but formally very different: Condition minimization is a highly non-linear problem, while

minimizing the number of probes can formulated as an integer linear program (joint work with Gunnar Klau and Knut Reinert).

We have shown by a simulation study that a simple Markov chain Monte Carlo based decoding algorithm can be used to infer the presence of previously unknown transfrags in the human genome quite reliably, even in the presence of high error rates and cross-hybridization (Section 7.2). To decode real genome tiling microarray experiments efficiently, the method may still need some tweaking and algorithm engineering, or simply more computational resources.

This thesis, especially Chapter 6, also shows that microarray design and analysis are more closely related than one generally believes, because the availability of unique probes hides most of the problems. Today's commercial microarrays are certainly designed with probe uniqueness in mind, but low-level data analysis turns out to be very complicated, as indicated by the wealth of papers published on this subject over the last few years (see Section 1.3.4 for a selection). If the nature of affinity coefficients was better understood — and we hope that Section 2.1 will be helpful in this respect — and all probes for a target could be selected to have the same affinity coefficient, analysis of microarray experiments would be greatly simplified. There are still many sources of variation and possibilities for errors, but one major reason for different probe signals would be eliminated (cf. Example 2.2). At present, we cannot predict affinity coefficients directly from the sequence, but the analysis of existing experimental data helps to build stochastic models. In Section 2.4, we propose a procedure that estimates both the affinity coefficients and the expression levels using both experimental data and sequence information. We expect that combined models of this kind will become increasingly useful in the near future, but their value has yet to be established.

Summarizing, the effort spent on chip design has considerable implications for the process of low-level data analysis. This is obvious for chips with non-unique probes, but also true for current commercial designs with unique probes.

Algorithms. We present an efficient algorithm to compute all-against-all matching statistics of two set of strings (targets vs. transcript collections) with upper and lower bounds in Chapter 4. This problem has not been specifically considered before, and the obvious generalizations of Chang and Lawler's matching statistics algorithm using either many suffix trees or one large generalized suffix tree would be less efficient, especially in terms of memory requirements, but also as as far as running time is concerned (see also Section 4.3.4). The recently developed *enhanced suffix array* data structure (Abouelhoda et al., 2002a) is crucial for the efficiency of our algorithm, whose core is the *bucket scan* from Figure 4.3. The concept of jumps that we introduce in Section 4.4 is also instrumental for efficient later processing of matching statistics. Interestingly, Abouelhoda et al. (2002a) bet that for any algorithm based on suffix trees, there exists an algorithm based on enhanced suffix arrays that runs as efficiently (or even more so

in practice because of better cache performance), even though enhanced suffix arrays lack some features of suffix trees, such as the suffix links. Although this statement has yet to be proved or disproved, our all-against-all matching statistics algorithm provides a (rare?) example of a method that works much better in practice and is also more easily implemented with suffix arrays than with suffix trees, even though the tree's suffix links were an important component of the original method by Chang and Lawler. Bucket scans are so fast that (bounded) longest common factors with one mismatch can be found with a brute-force method, enumerating all possibilities. To our knowledge, the bucket-scan-based LCF approach is at present the only method fast enough to be used in whole genome applications, such as for the optimization of genome tiling chips presented in Chapter 7.

In Chapter 6, we present two heuristic methods. The first one aims to minimize the condition of a submatrix of an affinity matrix obtained by row removal. Our analysis on small matrices shows that the heuristic finds the optimal solution in 30% of the cases and generally performs very well, even if the global optimum is not found. The second heuristic aims at selecting a minimal set of probes for target coverage and target set separation. The key idea of the heuristic is to "look ahead" and use the most unique probes first, as they will have the best separation properties for large target sets. Optimally small probe sets can be selected by an integer linear programming formulation, which suffers from the potentially exponential number of constraining inequalities, however; see the discussion in Section 6.3.4. If only pairs of singleton target sets are separated, the heuristic designs contains twice as many probes as the optimal designs, but also show better decoding performance for larger target sets (Tables 6.4 to 6.6).

In chapter 8, we consider the *shortest common supersequence problem* for many short DNA sequences (i.e., strings over a small alphabet). Whenever this problem was considered previously, the dimensions were reversed, and experimental results show that the known heuristics developed for relatively few, but longer, strings over potentially large alphabet work poorly in this setting. We have therefore developed a new algorithm that *refines* an existing supersequence by edit operations; an initial guess is obtained by the Alphabet-Leftmost heuristic, which turns out to be difficult to beat by more than 3 characters for large scale designs.

Stochastic results. We obtain the probability distribution of upper and lower bounds on the length of the shortest common supersequence in typical microarray production settings. There is a gap of approximately 20 nt between the considered upper and lower bound. Computational experiments suggest that the upper bound is tighter than than the lower bound, which is only based on the number of occurrences of each letter in every sequence, and we expect that the lower bound can be improved by considering longer subsequences. Very recent work by Salomaa (2003), the Cauchy inequality for subsequences, may provide a way to strengthen the lower bound considerably.

In Section 4.4, we present a statistical analysis of the frequency of jumps in random strings and derive a result about the expected length of the longest common factor of two random strings. While our approach does not yield precise concentration inequalities as the work of Waterman (1995) and Abbasi (1997), it provides a generalization of their results to strings of unequal length and with non-uniform character distribution.

Software packages. The LCF approach is implemented in the PROMIDE software package, which academic users may obtain free of charge by signing a license agreement. PROMIDE consists of several programs written in C and PERL; Appendix A.1 describes them in more detail.

To generate artificial families of similar sequences that can serve as test data for non-unique probe design methodologies, the REFORM software packages was developed. It also serves as a general-purpose tool for modeling sequence evolution and can be used for teaching phylogeny and evolution classes. In comparison to the existing software ROSE (Stoye et al., 1998), specifying a sequence family requires a little less work with ROSE, but REFORM offers much more flexibility. More information is given in Appendix A.2.

Outlook. To conclude this discussion, we shall attempt to predict some future developments in microarray design.

As microarray technology becomes more and more common, costs will further drop, increasing the possibilities for custom designed arrays to answer specific questions. There will be an increasing need for efficient and reliable custom chip design software. PROMIDE certainly is efficient, but to make it more widely available, it would certainly need a graphical user interface.

We hope that the design and analysis of microarrays will soon merge in such a way that custom designed arrays will make data analysis as simple as possible and that the analysis can in fact be based on the design methodology. Until better models to predict affinity coefficients from sequence similarity are developed, our proposal to estimate affinity coefficients and expression levels simultaneously by combining LCF information and experimental results (Section 2.4) may prove useful. Work aiming to improve affinity modeling has been recently published (Zhang et al., 2003), but it is mainly phenomenological and does not incorporate the various factors that make up the measurable signal strength, such as those described in Section 2.1. Interestingly, Affymetrix® (Mei et al., 2003) has also begun to base new chip designs on estimates of probe affinity coefficients from existing arrays. We expect increasing interest in this topic in the future.

As mentioned in the introduction, measuring the abundance of mRNA transcripts if often only a makeshift solution when one is really interested in the activity of certain proteins. Factors such as RNA degradation and translation rate can heavily distort the relationship between gene expression and corresponding protein activity. We expect that we will see increasingly more technologies that focus directly on the protein level in the future.

Some of the methods in this thesis may be applicable to proteomics as well. Today, the presence of certain peptides can be sensitively detected via their masses using mass spectrometry technology. In complex samples, several proteins may be present whose enzymatic digestion results in several peptide fragments with approximately the same mass. To infer the set of present proteins in a sample, decoding techniques similar to those described in Section 6.3 may prove useful. In other words, while single fragment masses are not a sufficiently unique description to identify a single protein, combinations of mass peaks may be usable to infer the presence of a combination of proteins.

We are certain that the refinement of DNA microarray technology and the development of new technologies for functional genomics and proteomics will continue to pose new computational challenges.

Bibliography

S. Abbasi. Longest common consecutive substring in two random strings. DIMACS Technical Report, DIMACS, October 1997.

M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *Proceedings of the Second International Workshop on Algorithms in Bioinformatics (WABI 2002)*, volume 2452 of *LNCS*, pages 449–463. Springer, 2002a.

M. I. Abouelhoda, E. Ohlebusch, and S. Kurtz. Optimal exact string matching based on suffix arrays. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 2476 of *LNCS*, pages 31–43. Springer, 2002b.

Affymetrix, Inc. *Microarray Suite (MAS) Version 5.0 User's Guide*. Santa Clara, CA, U.S.A., 2002. Document #701088, Rev. 1.

Affymetrix, Inc. *GeneChip® Expression Analysis Technical Manual*. Santa Clara, CA, U.S.A., 2003. Document #701021, Rev. 4.

B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell*. Garland Publishing, New York, 3rd edition, 1994.

H. T. Allawi and J. SantaLucia. Thermodynamics and NMR of internal G-T mismatches in DNA. *Biochemistry*, 36:10581–10594, 1997.

H. T. Allawi and J. SantaLucia. Nearest neighbor parameters for internal G-A mismatches in DNA. *Biochemistry*, 37:2170–2179, 1998a.

H. T. Allawi and J. SantaLucia. Nearest neighbor parameters of internal A-C mismatches in DNA. *Biochemistry*, 37:9435–9444, 1998b.

H. T. Allawi and J. SantaLucia. Thermodynamics and NMR of internal C-T mismatches in DNA. *Nucleic Acids Research*, 26(11):2694–2701, 1998c.

P. Atkins and J. de Paula. *Physical Chemistry*. Oxford University Press, 7th edition, 2002.

E. Barillot, B. Lacroix, and D. Cohen. Theoretical analysis of library screening using a N-dimensional pooling strategy. *Nucleic Acids Research*, 19(22):6241–6247, 1991.

M. Baum, S. Bielau, N. Rittner, K. Schmid, K. Eggelbusch, M. Dahms, A. Schlauersbach, H. Tahedl, M. Beier, R. Guimil, M. Scheffler, C. Hermann, J. M. Funk, A. Wixmerten, H. Rebscher, M. Honig, C. Andreae, D. Buchner, E. Moschel, A. Glathe, E. Jager, M. Thom, A. Greil, F. Bestvater, F. Obermeier, J. Burgmaier, K. Thome, S. Weichert, S. Hein, T. Binnewies, V. Foitzik, M. Muller, C. F. Stähler, and P. F. Stähler. Validation of a novel, fully integrated and flexible microarray benchtop facility for gene expression profiling. *Nucleic Acids Research*, 31(23):e151, 2003.

M. Beier, M. Hausch, M. Müller, C. Stähler, F. Stähler, and P. F. Stähler. An integrated approach to gene analysis. *Innovations in Pharmaceutical Technology*, December 2001.

T. Beißbarth, K. Fellenberg, B. Brors, R. Arribas-Prat, J. M. Boer, N. C. Hauser, M. Schneideler, J. D. Hoheisel, G. Schütz, A. Poustka, and M. Vingron. Processing and quality control of DNA array hybridization data. *Bioinformatics*, 16(11):1014–1022, 2000.

P. Billingsley. *Probability and Measure*. John Wiley & Sons, New York, 3rd edition, 1995.

J. Borneman, M. Chrobak, G. D. Vedova, A. Figueroa, and T. Jiang. Probe selection algorithms with applications in the analysis of microbial communities. *Bioinformatics*, 17(Suppl. 1):S39–S48, 2001.

J. Branke, M. Middendorf, and F. Schneider. Improved heuristics and a genetic algorithm for finding short supersequences. *OR Spektrum*, 20:39–45, 1998.

W. J. Bruno, D. J. Balding, E. H. Knill, D. Bruce, C. Whittaker, N. Doggett, R. Stallings, and D. C. Torney. Design of efficient pooling experiments. *Genomics*, 26:21–30, 1995.

W. Chang and E. L. Lawler. Sublinear expected time approximate string matching and biological applications. *Algorithmica*, 12:327–344, 1994.

Y. Chen, E. R. Dougherty, and M. L. Bittner. Ratio-based decisions and the quantitiative analysis of cDNA microarray images. *Journal of Biomedical Optics*, 2:364–374, 1997.

E. Coward, S. A. Haas, and M. Vingron. SpliceNest: visualization of gene structure and alternative splicing based on EST clusters. *Trends in Genetics*, 18(1):53–55, 2002.

D. J. Cutler, M. E. Zwick, M. M. Carrasquillo, C. T. Yohn, K. P. Tobin, C. Kashuk, D. J. Mathews, N. A. Shah, E. E. Eichler, J. A. Warrington, and A. Chakravarti. High-throughput variation detection and genotyping using microarrays. *Genome Research*, 11(11):1913–1925, 2001.

P. Deuflhard and A. Hohmann. *Numerische Mathematik I.* Walter de Gruyter, 3rd edition, 2002.

C. Dieterich, R. Herwig, and M. Vingron. Exploring potential target genes of signaling pathways by predicting conserved transcription factor binding sites. *Bioinformatics*, 19(Suppl. 2):ii50–ii56, 2003.

D. Du and F. K. Hwang. *Combinatorial Group Testing and Applications.* World Scientific, 2nd edition, 2000.

B. P. Durbin, J. S. Hardin, D. M. Hawkins, and D. M. Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18(Suppl. 1): S105–S110, 2002. Proceedings of ISMB 2002.

L. Elsner, C. He, and V. Mehrmann. Minimizing the condition number of a positive definite matrix by completion. *Numerische Mathematik*, 69(1):17–24, 1994.

L. Elsner, C. He, and V. Mehrmann. Minimizing the norm, the norm of the inverse and the condition number of a matrix by completion. *Numerical Linear Algebra with Applications*, 2(2):155–171, 1995.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice.* Chapman & Hall, 1996.

G. H. Golub and C. F. van Loan. *Matrix Computations.* The Johns Hopskins University Press, 3rd edition, 1996.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

C. Gräfe. Computersimulation von Energie-Eigenschaften von Oligonukleotiden unter verschiedenen Rahmanbedingungen. Bachelor's Thesis, Bioinformatics Program, Freie Universität Berlin, July 2003.

A. J. F. Griffiths, W. M. Gelbart, R. C. Lewontin, and J. H. Miller. *Modern Genetic Analysis – Integrating Genes and Genomes.* W. H. Freeman and Company, 2nd edition, 2002.

D. Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, Cambridge, 1997.

S. A. Haas, T. Beißbarth, E. Rivals, A. Krause, and M. Vingron. GeneNest: automated generation and visualization of gene indices. *Trends in Genetics*, 16(11):521–523, 2000.

S. A. Haas, M. Vingron, A. Poustka, and S. Wiemann. Primer design for large scale sequencing. *Nucleic Acids Research*, 26:3006–3012, 1998.

J. G. Hacia, L. C. Brody, M. S. Chee, S. Fodor, and F. S. Collins. Detection of heterozygous mutations in BRCA1 using high density oligonucleotide arrays and two-colour fluorescence analysis. *Nature Genetics*, 14(4):441–447, 1996.

S. Hannenhalli, E. Hubbell, R. Lipshutz, and P. A. Pevzner. Combinatorial algorithms for design of DNA arrays. In J. Hoheisel, editor, *Chip Technology*. Springer, 2002.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

P. D. N. Hebert, A. Cywinska, S. L. Ball, and J. R. deWaard. Biological identifications through DNA barcodes. *Proceedings of the Royal Society of London B*, 270:313–321, 2003a.

P. D. N. Hebert, S. Ratnasingham, and J. R. deWaard. Barcoding animal life: cytochrome *c* oxidase subunit 1 divergences among closely related species. *Proceedings of the Royal Society of London B (Suppl.)*, 270:S96–S99, 2003b.

J. Heise. Selection of family-specific probes for microarrays. Bachelor's Thesis, Bioinformatics Program, Freie Universität Berlin, October 2003.

R. Herwig, A. O. Schmitt, M. Steinfath, J. O'Brien, H. Seidel, S. Meier-Ewert, H. Lehrach, and U. Radelof. Information theoretical probe selection for hybridisation experiments. *Bioinformatics*, 16(10):890–898, 2000.

P. Hieter and M. Boguski. Functional genomics: it's all how you read it. *Science*, 278: 601–602, 1997.

W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differenatial expression. *Bioinformatics*, 18(Suppl. 1):S96–S104, 2002. Proceedings of ISMB 2002.

W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Parameter estimation for the calibration and variance stabilization of microarray data. *Statistical Applications in Genetics and Molecular Biology*, 2(1):Article 3, 2003. `http://www.bepress.com/sagmb/vol2/iss1/art3`.

T. R. Hughes, M. Mao, A. R. Jones, J. Burchard, M. J. Marton, K. W. Shannon, S. M. Lefkowitz, M. Ziman, J. M. Schelter, M. R. Meyer, S. Kobayashi, C. Davis, H. Dai, Y. D. He, S. B. Stephaniants, G. Cavet, W. L. Walker, A. West, E. Coffey, D. D.

Shoemaker, R. Stoughton, A. P. Blanchard, S. H. Friend, and P. S. Linsley. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nature Biotechnology*, 19(4):342–347, 2001.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, et al. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

T. Ideker, V. Thorsson, A. F. Siegel, and L. E. Hood. Testing for differentially expressed genes by maximum-likelihood analysis of microarray data. *Journal of Computational Biology*, 7:805–818, 2000.

ILOG, Inc. CPLEX. `http://www.ilog.com/products/cplex`, 1987–2004.

R. A. Irizarry, B. M. Bolstad, F. Collin, L. M. Cope, B. Hobbs, and T. P. Speed. Summaries of Affymetrix GeneChip® expression measures. *Nucleic Acids Research*, 31(4):Article e15, 2003a.

R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Anntonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high-density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003b.

R. W. Irving and C. B. Fraser. Approximation algorithms for the shortest common supersequence. *Nordic Journal of Computing*, 2:303–325, 1995.

J. A. Jaeger, D. H. Turner, and M. Zuker. Improved predictions of secondary structures for RNA. *Proceedings of the National Academy of Sciences of the U.S.A.*, 86:7706–7710, 1989.

J. A. Jaeger, D. H. Turner, and M. Zuker. Predicting optimal and suboptimal secondary structure for RNA. *Methods in Enzymology*, 183:281–306, 1990.

T. Jiang and M. Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal of Computing*, 24(5):1122–1139, 1995.

T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, 1969.

L. Kaderali and A. Schliep. Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics*, 18(10):1340–1349, 2002.

A. B. Kahng, I. I. Mandoiu, P. A. Pevzner, S. Reda, and A. Z. Zelikovsky. Border length minimization in DNA array design. In R. Guigó and D. Gusfield, editors, *Proceedings of WABI 2002*, volume 2452 of *LNCS*, pages 435–448. Springer, 2002.

P. Kapranov, S. E. Cawley, J. Drenkow, S. Bekiranov, R. L. Strausberg, S. P. A. Fodor, and T. R. Gingeras. Large-scale transcriptional activity in chromosomes 21 and 22. *Science*, 296(5569):916–919, 2002.

J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. In *Proceedings of the 13th International Conference on Automata, Languages and Programming (ICALP)*, volume 2719 of *LNCS*, pages 943–955. Springer, 2003.

T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In A. Amir and G. M. Landau, editors, *Proceedings of the 12th Symposium con Combinatorial Pattern Matching (CPM)*, volume 2089 of *LNCS*, pages 181–192. Springer, 2001.

M. K. Kerr, M. Martin, and G. A. Churchill. Analysis of variance for gene expression microarray data. *Journal of Computational Biology*, 7(6):819–837, 2000.

D. K. Kim, J. S. Sim, H. Park, and K. Park. Linear-time construction of suffix arrays. In *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 2676 of *LNCS*, pages 186–199. Springer, 2003.

S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

G. W. Klau, S. Rahmann, A. Schliep, M. Vingron, and K. Reinert. Optimal robust non-unique probe selection using integer linear programming. Submitted manuscript, 2004.

E. H. Knill, A. Schliep, and D. C. Torney. Interpretation of pooling experiments using the Markov chain Monte Carlo method. *Journal of Computational Biology*, 3(3): 395–406, 1996.

P. Ko and S. Aluru. Space efficient linear time construction of suffix arrays. In *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 2676 of *LNCS*, pages 200–210. Springer, 2003.

W. Köppelle. Produktübersicht Microarrays: Geschmacklose Chips. *Laborjournal*, 10, November 2003.

M. Kozal, N. Shah, N. Shen, R. Yang, R. Fucini, T. C. Merigan, D. D. Richman, D. Morris, E. Hubbell, M. Chee, and T. R. Gingeras. Extensive polymorphisms observed in HIV-1 clade B protease gene using high-density oligonucleotide arrays. *Nature Medicine*, 2(7):753–759, 1996.

S. Kurtz. Reducing the space requirement of suffix trees. *Software - Practice and Experience*, 29(13):1149–1171, 1999.

S. Kurtz. *Construction and Application of Virtrual Suffix Trees*. Technische Fakultät, Universität Bielefeld, 2002. Technical Manual. Available at `http://www.vmatch,de`.

P. J. M. V. Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Number 37 in Mathematics and Its Applications. Kluwer, 1987.

E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. Errata in *Nature* 411(6838):720 and *Nature* 412(6846):565.

N. Le Novère. MELTING, computing the melting temperature of nucleic acid duplex [*sic*]. *Bioinformatics*, 17(21):1226–1227, 2001. WWW server available at `http://bioweb.pasteur.fr/seqanal/interfaces/melting.html`.

G. G. Lennon and H. Lehrach. Hybridization analyses of arrayed cDNA libraries. *Trends in Genetics*, 7(10):314–317, 1991.

S. Levy, L. Compagnoni, E. W. Myers, and G. D. Stormo. Xlandscape: the graphical display of word frequencies in sequences. *Bioinformatics*, 14:74–80, 1998.

C. Li and W. H. Wong. Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. *Proceedings of the National Academy of Sciences of the U.S.A.*, 98(1):31–36, 2001.

F. Li and G. Stormo. Selection of optimal DNA oligos for gene expression analysis. *Bioinformatics*, 17(11):1067–1076, 2001.

R. J. Lipshutz et al. Advanced DNA sequencing technologies. *Current Opinion in Structural Biology*, 4:376–380, 1994.

R. J. Lipshutz, S. P. Fodor, T. R. Gingeras, and D. J. Lockhart. High density synthetic oligonucleotide arrays. *Nature Genetics*, 21:20–24, 1999.

D. Lipson, P. Webb, and Z. Yakhini. Designing specific oligonucleotide probes for the entire *S. cerevisiae* transcripome. In *Proceedings of WABI 2002*, volume 2452 of *LNCS*, pages 491–505. Springer, 2002.

G. Liu, A. E. Loraine, R. Shigeta, M. Cline, J. Cheng, V. Valmeekam, S. Sun, D. Kulp, and M. A. Sinai-Rose. NetAffx: Affymetrix probesets and annotations. *Nucleic Acids Research*, 31(1):82–86, 2003.

C. Lock, G. Hermans, R. Pedotti, A. Brendolan, et al. Gene-microarray analysis of multiple sclerosis lesions yields new targets validated in autoimmune encephalomyelitis. *Nature Medicine*, 8(5):500–508, 2002.

D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14(13):1675–1680, 1996.

F. Lottspeich and H. Zorbas, editors. *Bioanalytik*. Spektrum Akademischer Verlag, 1998.

A. Loy, A. Lehner, N. Lee, J. Adamczyk, H. Meier, J. Ernst, K.-H. Schleifer, and M. Wagner. Oligonucleotide microarray for 16S rRNA gene-based detection of all recognized lineages of sulfate-reducing prokaryotes in the environment. *Applied and Environmental Microbiology*, 68(10):5064–5081, 2002.

U. Manber and G. W. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

M. Markmann. *Entwicklung und Anwendung einer 28S rDNA-Sequenzdatenbank zur Aufschlüsselung der Artenvielfalt limnischer Meiobenthosfauna im Hinblick auf den Einsatz moderner Chiptechnologie*. PhD thesis, University of Munich, 2000.

R. Mei, E. Hubbell, S. Bekiranov, M. Mittmann, F. C. Christians, M. M. Shen, G. Lu, J. Fang, W. M. Liu, T. Ryder, P. Kaplan, D. Kulp, and T. A. Webster. Probe selection for high-density oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the U.S.A.*, 100(20):11237–11242, 2003.

R. Mott. EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Computer Applications in the Biosciences*, 13(4):477–478, 1997.

T. Müller and M. Vingron. Modeling amino acid replacement. *Journal of Computational Biology*, 7(6):761–776, 2000.

E. W. Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM*, 46(3):395–415, 1999.

F. Naef and M. O. Magnasco. Solving the riddle of the bright mismatches: Labeling and effective binding in oligonucleotide arrays. *Physical Review E*, 68:Article 011906, 2003.

Neurospora Sequencing Project Version 3. Neurospora sequencing project version 3. Whitehead Institute / MIT Center for Genome Research, 2002. `http://www-genome.wi.mit.edu`.

J. L. Oestreicher, I. B. Walters, T. Kikuchi, P. Gilleaudeau, J. Surette, U. Schwertschlag, A. J. Dorner, J. G. Krueger, and W. L. Trepicchio. Molecular classification of psoriasis disease-associated genes through pharmacogenomic expression profiling. *The Pharmacogenomics Journal*, 1(4):272–287, 2001.

N. Peyret, P. A. Seneviratne, H. T. Allawi, and J. SantaLucia. Nearest neighbor thermodynamics and NMR of DNA sequences with internal A-A, C-C, G-G, and T-T mismatches. *Biochemistry*, 38:3468–3477, 1999.

A. E. Pozhitkov and D. Tautz. An algorithm and program for finding sequence specific oligo-nucleotide probes for species identification. *BMC Bioinformatics*, 3(9), 2002. `http://www.biomedcentral.com/1471-2105/3/9`.

S. Rahmann. Rapid large-scale oligonucleotide selection for microarrays. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference (CSB'02)*, pages 54–63. IEEE, 2002. An abstract of this work also appeared in *Proceedings of WABI 2002*, LNCS vol. 2452, p. 434. Springer, 2002.

S. Rahmann. Fast and sensitive probe selection for DNA chips using jumps in matching statistics. In *Proceedings of the 2nd IEEE Computational Systems Bioinformatics (CSB'03) Conference*, pages 57–64. IEEE, 2003a.

S. Rahmann. Fast large scale oligonucleotide selection using the longest common factor approach. *Journal of Bioinformatics and Computational Biology*, 1(2):343–361, 2003b.

S. Rahmann. REFORM (Random Evolutionary FORests Modeling software). Available at `http://www.molgen.mpg.de/~rahmann`, 2003c.

S. Rahmann. The shortest common supersequence problem in a microarray production setting. In *Proceedings of the 2nd European Conference in Computational Biology (ECCB 2003)*, volume 19 Suppl. 2 of *Bioinformatics*, pages ii156–ii161, 2003d.

S. Rahmann and E. Rivals. On the distribution of the number of missing words in random texts. *Combinatorics, Probability and Computing*, 12:73–87, 2003.

S. Rash and D. Gusfield. String barcoding: Uncovering optimal virus signatures. In *Proceedings of RECOMB 2002*, pages 254–261. ACM Press, 2002.

E. Rivals and S. Rahmann. Combinatorics of periods in strings. In P. Orejas, P. G. Spirakis, and J. van Leuween, editors, *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP 2001)*, volume 2076 of *LNCS*, pages 615–626. Springer, 2001.

D. M. Rocke and B. Durbin. A model for measurement error for gene expression analysis. *Journal of Computational Biology*, 8(6):557–567, 2001.

J.-M. Rouillard, C. J. Herbert, and M. Zuker. OligoArray: Genome-scale oligonucleotide design for microarrays. *Bioinformatics*, 18(3):486–487, 2002.

P. J. Rousseuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.

A. Salomaa. The formal language theory column: Counting (scattered) subwords. In *Bulletin of the European Association for Theoretical Computer Science No. 81*. EATCS, October 2003.

J. SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Sciences of the U.S.A.*, 95:1460–1465, 1998.

M. Schena. *Microarray Analysis*. Wiley-Liss, 2002.

M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 1995.

A. Schliep. The markov chain pooling decoder. Los Alamos National Laboratories; now available at `http://algorithmics.molgen.mpg.de/MCPD/`, 1998.

A. Schliep, D. C. Torney, and S. Rahmann. Group testing with DNA chips: Generating designs and decoding experiments. In *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference (CSB 2003)*, pages 84–93. IEEE, 2003.

J. S. Sim, D. K. Kim, H. Park, and K. Park. Linear-time search in suffix arrays. In *Proceedings of the 14th Australasian Workshop on Combinatorial Algorithms*, pages 139–146, 2003.

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. *Nature Biotechnology*, 17:922–926, 1999.

E. M. Southern. Detection of specific sequences among DNA fragments separated by gel electrophoresis. *Journal of Molecular Biology*, 98(3):503–517, 1975.

M. Spitzer. Post-processing of selected oligos ordered by their specificity. Bachelor's Thesis, Bioinformatics Program, Freie Universität Berlin, July 2003.

J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163, 1998.

T. Strachan and A. P. Read. *Human Molecular Genetics*. Garland Science Publishers, 3rd edition, 2003.

N. Sugimoto, S. ich Nakano, M. Yoneyama, and K. ich Honda. Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes. *Nucleic Acids Research*, 24(22):4501–4505, 1996.

N. Sugimoto, S. Nakano, M. Katoh, A. Matsumura, et al. Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. *Biochemistry*, 34:11211–11216, 1995.

W.-K. Sun and W.-H. Lee. Fast and accurate probe selection algorithm for large genomes. In *Proceedings of the 2nd IEEE Computational Systems Bioinformatics (CSB'03) Conference*, pages 65–74. IEEE, 2003.

J. Theilhaber, S. Bushnell, A. Jackson, and R. Fuchs. Bayesian estimation of fold-changes in the analysis of gene expression: The PFOLD algorithm. *Journal of Computational Biology*, 8:585–614, 2001.

E. Ukkonnen. On-line construction of suffix trees. *Algorithmica*, 14:249–60, 1995.

R. Van Gelder, M. von Zastrow, A. Yool, W. Dement, J. Barchas, and J. Eberwine. Amplified RNA synthesized from limited quantities of heterogenous cDNA. *Proceedings of the National Academy of Sciences of the U.S.A.*, 87:1663–1667, 1990.

V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. Serial analysis of gene expression. *Science*, 270(5235):484–487, 1995.

J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001. Erratum in *Science* 292(5523):1838.

D. Wang, L. Coscoy, M. Zylberberg, P. C. Avila, H. A. Bousheym, D. Ganem, and J. L. DeRisi. Microarray-based detection and genotyping of viral pathogens. *Proceedings of the National Academy of Sciences of the U.S.A.*, 99:15687–15692, 2002.

H. Wang, E. Hubbell, J. shan Hu, G. Mei, M. Cline, G. Lu, T. Clark, M. A. Siani-Rose, M. Ares, D. C. Kulp, and D. Haussler. Gene structure-based splice variant deconvolution using a microarry platform. *Bioinformatics*, 19(Suppl. 1):i315–i322, 2003. Proceedings of ISMB 2003.

X. Wang and B. Seed. Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics*, 19(7):796–802, 2003.

M. S. Waterman. *Introduction to Computational Biology.* Chapman and Hall, London, 1995.

J. D. Watson and F. H. C. Crick. Molecular structure of nucleic acids. A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11. IEEE, 1973.

J. Werner. *Numerische Mathematik 1.* Vieweg, 1992.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences of the U.S.A.*, 98(20):11462–11467, 2001.

Y. H. Yang, S. Dudoit, P. Luu, D. M. Lin, V. Peng, J. Ngai, and T. P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30(4):Article e15, 2002.

R. A. Young. Biomedical discovery with DNA arrays. *Cell*, 102:9–15, 2000.

L. Zhang, M. F. Miles, and K. D. Aldape. A model of molecular interactions on short oligonucleotide arrays. *Nature Biotechnology*, 21(7):818–821, 2003.

M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, 2003.

M. Zuker, D. H. Matthews, and D. H. Turner. *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A practical guide*. NATO ASI Series, Kluwer, Dordrecht, 1999.

Appendix

Appendix A

Software

This appendix summarizes the software that has been developed for probe selection and microarray design. It is available free of charge for academic users from `http://oligos.molgen.mpg.de`. This software is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of **merchantability** or **fitness for a particular purpose**. For the exact license and warranty conditions, see `http://oligos.molgen.mpg.de`.

A.1 Promide

PROMIDE is a collection of command-line tools for PRObe selection and MIcroarray DEsign. PROMIDE is currently in beta stage. When you obtain any version of PROMIDE, you do so entirely at your own risk.

To use PROMIDE, you need a working installation of `PERL`, which can be obtained from `http://www.cpan.org`. At present, you also need a tool to build enhanced suffix arrays, `mkvtree`, written by Stefan Kurtz at University of Hamburg. You have to obtain a separate license for `mkvtree`; like PROMIDE, it is provided at no cost to academic users. Please contact the author Stefan Kurtz at `kurtz@zbh.uni-hamburg.de`.

The following tools are part of PROMIDE.

Sequence manipulation: `ReverseComplement.pl` This tool reads one or several file(s) in multiple FASTA format given on the command line (or from `stdin`) and either replaces each sequence by its reversed complement or adds the reversed complement after each sequence. The output, a new FASTA file, is sent to `stdout`, but can be redirected to an arbitrary file.

`ReverseComplement.pl` [mode] [options] [infile(s)]

Valid Modes

`-r` (default)	Replace original sequences by reversed complements
`-a`	Add complements instead of replacing original sequences
`-f`	Don't reverse-complement; reFormat only

Options

`-c`	Convert all sequences to upper case
`-p` 'pattern'	Scan FASTA header for a valid `PERL` pattern and add '`-RC`' after the first occurrence of the pattern in the reverse-complemented sequence. Default is `/^>\S+/`, i.e., the beginning of the header until the first whitespace character.
`-w`	Warn on strange characters, which cannot be complemented. Don't warn on `N` or `X`, which will be left `N`resp. `X`. Also warn on empty sequences and sequences shorter than 20.

When more than one option of `-r`, `-a`, `-f` is given, `-f` overrides the other two, and `-r` overrides `-a`. The `-p` 'pattern' option is only relevant in Replace (`-r`) and add (`-a`) mode. The output will have 70 characters per sequence line, but the headers can be of arbitrary length. The program is designed to be used as a filter.

Index Creation: `PrepareIndex.pl` The index is created from one or several FASTA files, and consists of several tables; each table is stored in its own file.

`PrepareIndex.pl` [options] fastafile(s)

Options

`-p` 'pattern'	`PERL` pattern for collection ID
`-q` 'pattern'	Secondary pattern for collection ID
`-i` indexname	Basename of all index tables
`-b` length	Explicitly specify bucket prefix length

While each specified input FASTA file is read, the sequence headers are scanned for the `-p` pattern (or, if that fails, for the `-q` pattern) to determine the collection ID of a sequence. Sequences with the same collection ID are assumed to belong together logically. Specificity of a probe candidate is checked against each collection, not against each sequence. If no patterns are given, the collection ID is assumed to be the first whitespace-terminated string after the FASTA header symbol (`>`). If no indexname is

given, the name of the first FASTA file is used. If no bucket prefix length is given, a reasonable default, growing logarithmically with the total sequence length, is used.

The index consists of the following tables.

Descriptor Tables. These are text files and contain information about the index and the collections IDs.

- `al1`: Alphabet table
- `des`: Sequence descriptions from FASTA headers
- `cid`: Collection ID for each collection
- `cos`: Internal collection number for each sequence
- `prj`: Human-readable information about the index

Suffix Array Tables. These are binary files and comprise the enhanced suffix array.

- `tis`: Encoded concatenated sequences
- `suf`: The suffix array (`pos` within this thesis)
- `bck`: bucket start and end points
- `cl`: Collection numbers
- `lcp`, `llv`: Longest common prefix lengths

Auxiliary Tables. These tables are built during index construction but are not needed for the further probe selection process.

- `ssp`: Sequence separator positions
- `sds`: Sequence description start positions

The PERL Script `PrepareIndex.pl` calls two external programs to build the index. These are `mkvtree` by Stefan Kurtz to build the enhanced suffix array without the `cl` table, and `build_cl` to add the `cl` table to the suffix array.

Target Preparation: `PrepareTargets.pl` This tool prepares one or several FASTA files for processing as target sequences (sequences from which probes shall be selected). Its main purposes are to

- Extract a sequence-specific ID from each header that is used as a name for the sequence during the probe selection process
- Extract a (possibly different) collection-specific ID from each header that allows to group one or more of the sequences into a collection. During probe selection, each probe candidate from a target is checked for specificity against all transcript collections, except its own collection.
- Remove sequences that are too short for meaningful probe design.
- Replace all runs of wildcard characters by a single "don't know" character `X`.
- Determine the length of the condensed sequences.

The output consists of two files:

1. the *target file* in FASTA format, whose extension is `.targets`.
2. the *target ID file* with extension `.tid`, containing the extracted IDs. The first line contains the keyword `TARGETS`, followed by a space, followed by the number of targets, followed by another space and the length of the longest target. Then there is one line per target. Each line contains three space-separated entries: the sequence ID, the collection-ID, and the sequence length.

The calling syntax is as follows.

`PrepareTargets.pl` [options] fastafile(s)

Options

`-s` 'pattern'	`PERL` pattern for sequence ID
`-p` 'pattern'	`PERL` pattern for collection ID
`-q` 'pattern'	Secondary pattern for collection ID
`-t` *targetname*	Basename of `.targets` and `.tid` files
`-c` *cidname*	Basename of collection ID (`.cid`) file
`-d` *len*	Discard targets shorter than *len* [200]
`-i`	Case insensitive: Treat `agct` as `ACGT`, default: as `X`
`-n`	NoCondense: Do not condense runs of `X`into a single `X`

This program is *not* intended as a filter. If no sequence or collection pattern is given, the string after the FASTA header symbol (`>`) until the first whitespace character is taken as the sequence resp. collection ID. If no targetname is given, the name of the first input file is taken. By default, the program is case sensitive, and the nucleotides

`acgt` are treated as if they were masked (i.e., as `X`), and no probes containing lowercase characters will be selected. By default, runs of `X`es are condensed into a single `X`, but this behavior may be turned off by the `-n` option.

Target-Index Matching Statistics: `tims` To find specific probes, the matching statistics of each target against each sequence in the index are computed. This is a time consuming part of the probe selection process. The tool `tims` will do the work.

`tims` [options] targetname [indexname]

Options

`-r` $\min_0 : \min_1 : \max$	Range of relevant matching statistics
`-s` statfile	Basename of the `.ms` output file [targetname]
`-0`	Compute exact matching statistics only; no ms1

When called, the program looks for the target files targetname`.targets` and targetname`.tid` and the required index tables. If indexname is omitted, targetname is used. If no range (`-r`) for the matching statistics is given, the lower bound for exact matching statistics (ms^0) is set to the detectable minimum, i.e., the bucket prefix length of the index, the lower bound for the one-mismatch-matching-statistics (ms^1) is set to that value plus three, and the upper bound is set to 32. In any case, the ranges must be subintervals of these default values. The name of the output file can be set with the `-s` option, but its extension is always `.ms` for "matching statistics". By default, the program computes both mismatch-free matching statistics (ms^0) and matching statistics allowing one mismatch (ms^1). The latter computation can take up to 100 times longer than the ms^0 computation. Therefore it can be disabled with the `-0` option (a zero, not a capital O). For a more sensitive specificity computation, one should avoid using this option, however.

The matching statistics are written into the `.ms` file, which has a format that depends on the given options.

Probe Candidate Selection: `probecand` When the target-index matching statistics have been computed, probes can be selected from each target using the program `probecand`. Here, the user can specify a length range for the probes (the length cannot exceed 32), reaction conditions (temperature and salt concentration), and stability properties of the desired probes.

`probecand` [options] targetname

Options

`-t` temperature	Temperature (Celsius) [50]
`-l` Lmin:Lmax	Length range [20:30]
`-N` lnsalt	ln of [Na^+] concentration [0]
`-g` Gmin:Gmax	Desired Gibbs free energy range of probes
`-p` Parameters	Select parameter set for Gibbs free energy [SL0]
`-m` method	Selection method (LCF, LCFS, dG, dGS) [dG]
`-n` nmin:nmax	Desired number of candidates per Target [20:200]
`-c` cmin:cmax	Desired candidate coverage of target [0:1]
`-0`	Ignore matching statistics with 1 mismatch
`-F` free	Number of unpenalized full matches [0]
`-u` nonunique	Allowable degree of probe non-uniqueness [1]
`-L`	Enumerate all collections with perfect matches
`-b` badshift	(Positive numbers accept more probes) [0]
`-s` statfile	Basename of matching statistics file [targetname]
`-o` outputfile	Basename of probe candidate files [targetname]
`-x`	Write additional verbose output file

If no name for the statfile or outfile is explicitly given, then targetname is used.

The format of the output file `.oligos` is as follows. It contains one block of lines for every Target sequence. A block starts with a header line (recognizable by the '`>`'-sign as its initial character), and ends with an empty line.

The header starts with the sequence identifier, a colon (':'), its internal number, and another colon. This is followed by a tabulator and the following tab-separated information: (target length; number of probes that match the reaction conditions; number of high-quality probes [does not necessarily equal the number of unique probes if the `-u` option was specified]; number of unique probes), e.g.,

```
>YAL001C:0:       3483      572      545      545
```

The probe list follows with one line per probe. The number of probes lies between the limits given by the `-n` or `-c` option, unless not enough probes match the reaction conditions in the first place. Naturally, if you request 20 probes when only 10 high-quality probes exist, the remaining 10 probes will be of low quality.

Each probe line starts with the probe sequence and then contains the following tab-separated information:

- Probe start position in the Target sequence
- Gibbs free energy of the perfect probe duplex (is within specified range)
- Estimated energy difference to the most stable secondary binding site (larger difference means higher specificity)
- Longest common factor length of probe candidate and secondary binding sites
- Number of transcript collections containing the full oligo (when designing unique oligos, this is 1)
- Badness value between 10^{-6} and 10^{6}. Values below 1 indicate a relatively specific oligo; oligos with badness values above 1 should be avoided. These badness values are a re-scaled version of the unspecificity values in Eq. (3.5).

A.2 REFORM

REFORM is a language that allows you to specify the topology of evolutionary trees and let sequences evolve along their edges according to a continuous-time Evolutionary Markov Process (EMP). It is written in PERL v5.6.0 and requires the CPAN Parse::RecDescent module by Damian Conway.

REFORM offers the following features:

- Free specification of the tree topology
- Shortcuts (SUBTREE macro) for quickly specifying complete subtrees
- Sequences in the tree nodes consist of several segments. Each segment evolves according to its own EMP. This is useful for modeling coding and non-coding DNA, for example.
- Rate variation over sites (e.g. using Gamma-distributed rate factors)
- Arbitrary Indel length distributions

We expect that REFORM can be useful in the following situations.

- Evaluation of sequence analysis algorithms and methods for phylogenetic tree reconstruction: REFORM is a good generator for controlled test data.
  - Watch how *long branch attraction* makes Maximum Parsimony fail
  - Assess the difficulty of maximum likelihood tree reconstruction with the wrong model.

  - Generate artificial gene families or virus strains, and try to find gene- or strain-specific oligonucleotide probes for these

- Classroom instruction: Generate sequence data for examples or exercises in a controlled fashion, so you know the "true"solution.
- Playing with evolution in a sandbox. No animals were harmed during the production of this software.

Using REFORM is simple, as it consists of a single PERL script `Reform.pl`. Run a `.reform` model such as `Model-a.reform` by typing

```
> Reform.pl  Model-a.reform
```

to obtain several FASTA sequences as output on `stdout`, according to the specifications of the model.

Example. We provide a part of the source code of Model (a) used in Section 6.3.4 to generate an artificial family of related target sequences.

```
Alphabet DNA "AGCT"
Distribution uniform  [1 1 1 1]
RateMatrix QJC        [[0 1 1 1][1 0 1 1][1 1 0 1][1 1 1 0]]
EvolutionaryModel JCI
{
  SubstitutionRates  QJC
  RelativeGapProb [8 1 4 2 1 ...20]
  DeleteRate 0.005
  InsertRate 0.005
  InsertDistribution uniform
}

Root TestSequence1  DNA
{
  Segment   first
  {
    Sequence  200 * uniform
    Model     JCI
    Speeds    constant 0.9
  }
  Segment   second
  {
    Sequence  200 * uniform
    Model     JCI
```

```
    Speeds    constant 0.95
  }
  Segment   third
  {
    Sequence  200 * uniform
    Model     JCI
    Speeds    constant 1
  }
  Segment   fourth
  {
    Sequence  200 * uniform
    Model     JCI
    Speeds    constant 1.05
  }
  Segment   fifth
  {
    Sequence  200 * uniform
    Model     JCI
    Speeds    constant 1.1
  }
}

Subtree TestSequence1 { 1  4  3 }

Subtree TestSequence1_1_1_1 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_1_1_1 }

Subtree TestSequence1_1_1_2 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_1_1_2 }

Subtree TestSequence1_1_1_3 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_1_1_3 }

Subtree TestSequence1_1_1_4 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_1_1_4 }

Subtree TestSequence1_1_2_1 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_1_2_1 }

...

Subtree TestSequence1_4_4_4 { 0.1 4 1 }
Output FASTALEAVES { TestSequence1_4_4_4 }
```

License and Acknowledgments. REFORM is copyright (C) 2003, Sven Rahmann. All rights reserved. Permission to use this software is granted under the same restrictions as for Perl itself.

The matrix routines used in REFORM were adapted from CPAN package Math::Matrix. Copyright (C) 2001, Brian J. Watson [`bjbrew@power.net`]. Copyright (C) 2001, Ulrich Pfeifer [`pfeifer@wait.de`]. Copyright (C) 1995, Universität Dortmund. Copyright (C) 2001, Matthew Brett [`matthew.brettmrc-cbu.cam.ac.uk`]. All rights reserved.

REFORM would not have been possible without the Parse::RecDescent module written by Damian Conway, which is available from CPAN. Incidentally, Damian called the tutorial for that module "The man(1) of Descent", and it seems that finally here we've put the module to its intended use...

Appendix B

Probe Selection Projects

The performance of selected probe sets in biological experiments cannot be evaluated *in silico*. It would be best to run controlled spike-in experiments for each target and measure the response of each probe; this would immediately show the extent of cross-hybridization and allow to measure all affinity coefficients (cf. Section 2.4). These experiments would require considerable financial resources and time, however, because many chips and test samples would have be produced and analyzed.

A more practical solution is to run several "real" chip experiments with the generated probes. Then one can use data analysis methods that infer both the affinity coefficients and the expression values, such as the one described in Section 2.4, to evaluate the quality of the probe sets. Comparison of probe sets generated by different methods could also be incorporated in such a study.

We have therefore formed collaborations with other research groups and generated probe sets for their projects. Preliminary experiments have shown that the probes work reasonably well, i.e., in the majority of cases, they show a signal where they should, and signal strength is relatively consistent across different probes for the same target. We give a short list of projects in which PROMIDE is being used.

Neurospora crassa: The orange bread mold *Neurospora crassa* has been used as a model organism for genetic and biomedical studies for more than 50 years. Like yeast, *N. crassa* is a filamentous ascomycete. Its genomic sequence and the set of predicted genes were obtained from the Neurospora sequencing project (Neurospora Sequencing Project Version 3). From the 10082 known and predicted genes, those 9707 of at least 150 bp were used as target sequences (total length 15 Mb). Specificity was checked against all predicted genes and their reversed complements (10082 collections) and, as an additional safety measure, against the whole genomic sequence and its reversed complement (divided into 821 contigs), for a total of 10903 collections (105 Mb). Probe selection with bucket prefix length $q = 10$, $R^0_{\min} = 11$, $R^1_{\min} = 14$, $R_{\max} = 30$ took less than 4 hours, including construction of the index. In contrast, probe selection with the method of Li and Stormo (2001) took several days. Verena Beier at the Functional Genome Analysis department at the German Cancer Research Center (DKFZ) created a chip with probes for selected genes using febit's geniom® one system and

compared the results obtained with our probes to the results obtained with probes from febit AG and from Li and Stormo's software. Preliminary results indicate that signal intensities of probes from the same gene vary considerably in the Li and Stormo probe set, whereas our set and the proprietary febit set show more consistent signals.

Alternatively spliced human transcripts: In collaboration with febit AG, Mannheim, Stefan Haas used PROMIDE and the SpliceNest database (Coward et al., 2002) to select probes for alternative splice isoforms of several human genes. Preliminary results indicate that in comparison to probe sets selected by febit AG and Deutsches Ressourcenzentrum für Genomforschung GmbH, Berlin, the PROMIDE probes often show a slightly lower but more constant signal than the other probe sets.

Candida albicans is a commonly encountered human pathogen fungus that causes a variety of infections ranging from mucosal infections in generally healthy persons to life-threatening systemic infections in individuals with impaired immunity. Few classes of drugs are effective against these fungal infections, and all of them have limitations with regard to efficacy and side-effects. Nicole Hauser's group (Fraunhofer-Institut für Grenzflächen- und Bioverfahrenstechnik (IGB), Stuttgart) has created a cDNA chip for all of the approximately 7000 *C. albicans* genes, but found three gene families (ALS, CDR/MDR, and SAP) whose members were difficult to distinguish between, because of high sequence homology. Therefore we are designing non-unique probes with PROMIDEfor these gene families.

Human transcriptome project: This project is a part of project ENCODE (ENCyclopedia Of Dna Elements) that aims to identify all functional elements within the human genomic DNA. To support probe selection for transcript detection, we have computed the lcf-values of each 25-mer within the non-repeat-masked part of the human genome against all other non-overlapping 25-mers individually. These results can be used in two ways: (1) If we allow some freedom in probe positions for genome tiling chips, probe locations can be moved for a few bp to obtain more specific probes than if a strictly regular probe spacing is enforced. The LCF statistics of each 25-mer have been made available to Affymetrix, Inc., who will decide on the chip design. (2) Individual high lcf-values are crucial in decoding the transfrag detection experiments described in Chapter 7.

Besides these projects, the PROMIDE software has been licensed to other research groups, including

- Dr. Volker Brendel's group at Iowa State University
- Dr. Alex Zelikovsky, CS Department, Georgia State University
- Dr. Amar Mukherjee's lab, University of Central Florida
- Benjamin Horsman, Brinkman Laboratory, Department of Molecular Biology and Biochemistry, Simon Fraser University, Burnaby, B.C., Canada

Appendix C

Anhang laut Promotionsordnung

C.1 Zusammenfassung

DNA Microarrays oder DNA-Chips haben sich in den letzten Jahren zu einem wichtigen Werkzeug in der funktionalen Genomanalyse entwickelt. Verschiedene Technologie-Plattformen existieren; in dieser Arbeit betrachten wir ausschließlich Oligonukleotid-Arrays hoher Dichte.

Diese bestehen aus einer Glas- oder Quartzplatte, die an mehreren Tausend bis zu einer Million wohldefinierten und regelmäßig angeordneten Stellen (sog. *Spots*) mit einsträngigen DNA-Oligonukleotiden aus 8 bis 30 Nukleotiden beladen ist; auf einem Spot befinden sich bis zu 10^9 Moleküle derselben DNA-Sequenz. Die Sequenz der Oligonukleotide ist so gewählt, dass sie als Signatur eines Gen-Transkripts dient, d.h. jeder Spot auf dem Chip repräsentiert durch seine Sequenz ein bestimmtes Transkript des zu untersuchenden Organismus oder mehrerer zu untersuchender Organismen.

In einem typischen Experiment werden aus einer Zell- oder Gewebeprobe die mRNA-Transkripte der exprimierten Gene extrahiert und gegebenenfalls amplifiziert. Die zu den mRNA-Transkripten komplementären cDNAs oder cRNAs, die mit Fluoreszenz-Farbsoffen markiert sind, werden mit den Oligonukleotiden auf dem Chip hybridisiert; hierbei wird ausgenutzt, dass sich komplementäre DNA- und RNA-Basen spezifisch stabil aneinander binden. Nicht hybridisierte Moleküle werden von der Chipoberfläche heruntergewaschen. Anschließend kann an jedem Spot die Fluoreszenz-Intensität gemessen werden, die um so größer ist, je mehr mRNA von dem ensprechenden Gen vorlag, d.h., je stärker das Gen in der Zellprobe exprimiert war.

Auf diese Weise lassen sich relativ schnell Genexpressionsprofile von verschiedenen Zell- oder Gewebetypen erstellen und vergleichen. Beispielsweise erhofft man sich durch Betrachten der Unterschiede von gesunden Zellen und Tumorzellen, verschiedene Krebserkrankungen besser zu verstehen und gegebenenfalls den Therapieerfolg zu verbessern.

Die Genexpressionsanalyse mit DNA-Chips ist eine Technik, die Daten in großer Menge und mit hohem Durchsatz liefert; sie ist allerdings nicht frei von Fehlerquellen: Jeder Schritt muss sorgfältig durchgeführt werden, damit die gemessenen Daten tatsächlich mit der Genaktivität korrelieren. Insbesondere muss der DNA-Chip sorgfältig entworfen und produziert werden. Mit den dabei auftretenden algorithmischen Problemen befasst sich diese Arbeit. Im einzelnen werden folgende Aspekte untersucht (genauere Problemstellungen sind in Kapitel 1 ausformuliert).

- Damit ein Spot genau ein bestimmtes Gen repräsentiert, müssen die dort aufgebrachten Oligonukleotide genspezifisch sein, d.h., unter den Hybridisierungsbedingungen ausschließlich an das Zieltranskript binden. Das Hauptproblem besteht darin, eine geeignete Menge von Oligonukleotiden für jedes Transkript auszuwählen.

- Ein grundsätzliches Problem dabei ist, den Zusammenhang zwischen dem Expressionsniveau eines Gens und der gemessenen Signalintensität an den Spots zu verstehen, insbesondere zunächst für ideal-spezifische Oligonukleotide. Von Sättigungseffekten abgesehen ist dieser Zusammenhang annähernd linear; die Proportionalitätskonstante, auch *Affinitätskoeffizient* genannt, hängt von vielen Faktoren ab, die in Kapitel 2 detailliert beschrieben werden. Ein wichtiger Faktor ist dabei die Bindungsstärke der Hybdridisierung, die mit Hilfe sogenannter *nearest neighbor* Modelle geschätzt wird. Weiterhin geben wir eine Übersicht über Methoden zur Schätzung der Expressionsniveaus und der Affinitätskoeffizienten.

- Während der Chipentwurfsphase sind die Affinitätskoeffizienten in der Regel unbekannt, da keine experimentellen Daten vorliegen, aus denen sie geschätzt werden könnten. Man muss daher auf Ersatzmaße zurückgreifen, die sich schnell durch Vergleich der Oligonukleotidsequenz mit den Transkriptsequenzen gewinnen lassen. Letzten Endes interessiert dabei nur die Entscheidung, ob ein Transkript an ein Oligonukleotid bindet oder nicht. Wir treffen diese Entscheidung anhand des längsten gemeinsamen Substrings (longest common factor; LCF) von Oligonukleotid und Transkript. Diese Wahl wird in Kapitel 3 motiviert und mit anderen Ansätzen verglichen. Durch den Vektor der LCF-Längen eines Oligonukleotids mit allen Transkripten lässt sich das Kreuzhybridisierungs-Risiko approximieren.

- Kapitel 4 bildet den algortihmischen Kern dieser Arbeit. Hier stellen wir einen effizienten Algorithmus zur Berechnung sogenannter *matching statistics* von allen Transkriptpaaren vor, der auf erweiterten Suffixarrays beruht. Aus diesen *matching statistics* lassen sich die gewünschten LCF-Vektoren für beliebige Oligonukleotide bestimmen. Da die Anzahl der zu berechnenden *matching statistics* in der Regel sehr groß ist, stellt sich das Problem der effizienten Repräsentierung dieser Werte. Wir nutzen die Struktur der Werte aus und führen den Begriff des Sprungs (*jump*) in *matching statistics* ein. Es genügt dann, die Position und

Höhe der Sprünge zu betrachten. Wir untersuchen die statistische Häufigkeit von Sprüngen und geben ein Verfahren an, um aus den Sprüngen das Kreuzhybridisierungsrisiko von Oligonukleotiden zu schätzen. Weiterhin stellen wir Varianten der Algorithmen vor, die es erlauben, unterschiedliche Gewichting auf schnellere Laufzeit oder genauere Ergebnisse zu legen. Wir illustrieren die Leistungsfähigkeit der entwickelten Verfahren am Beispiel der Oligonukleotidauswahl für *Saccharomyces cerevisiae* (Backhefe).

- Das LCF-Konzept aus Kapitel 3 und die Algorithmen aus Kapitel 4 erlauben es, einzelne Oligonukleotide effizient gemäß ihrer Spezifität zu klassifizieren. In Kapitel 5 kommen wir zurück auf das eingangs gestellte Problem, eine passende *Menge* an Oligonukleotiden für jedes Transkript zu wählen. Dabei sind zusätzliche Bedingungen einzuhalten. Beispielsweise möchte man ein Transkript gleichmäßig mit Oligonukleotiden abdecken; jedoch treten hochspezifische Oligonukleotide oft in Clustern auf, so dass man hier eine Wahl treffen muss. Das Kapitel schließt mit einer Übersicht über den gesamten Auswahlprozess.

- Betrachtet man Gruppen von Transkripten (oder auch ganze Genome) mit hoher Sequenzähnlichkeit (z.B. alternative Spleißvarianten von Genen oder Subtypen desselben Virustyps), so stellt man fest, dass sich oft nicht genügend sequenzspezifische Oligonukleotide finden lassen. In diesem Fall lässt man zu, dass die Oligonukleotide mit mehreren der Sequenzen hybridisieren. Man beobachtet dann bei quantitativer Analyse die Summe der einzelnen Expressionswerte, oder bei qualitativer Anaylse mit binären Signalen das logische Oder der Einzelsignale. Die beobachteten Signale müssen also zunächst "dekodiert" werden. In Kapitel 6 werden Dekodierungsmethoden für sowohl das quantitative als auch das qualitative Szenario vorgestellt. Auch das Oligonukleotid-Auswahlverfahren wird sehr viel komplexer: Die Auswahl muss so erfolgen, dass robustes und effizientes Dekodieren möglich wird; andererseits sollte die Anzahl an Oligonukleotiden aus Wirtschaftilchkeitsgründen möglichst gering sein. Diese Forderungen werden im quantitativen Szenario als Konditionsminimierungsproblem einer (Sub-)Matrix formuliert, und im qualitativen Szenario als ein ganzzahliges lineares Programm. Zusätzlich werden schnelle Lösungsheuristiken angegeben.

- Die diskutierten Oligonukleotid-Auswahlmethoden für die Genexpressionsanalyse setzen voraus, dass man das Transkriptom des zu untersuchenden Organismus kennt. Dieses ist im allgemeinen jedoch schwieriger zu bestimmen als das komplette Genom zu sequenzieren. Im Rahmen des Projekts ENCODE (ENCyclopedia Of Dna Elements[1]) wird versucht, die noch unbekannten Transkripte des Humangenoms innerhalb der genomischen Sequenz zu lokalisieren. Dazu wird das gesamte Genom mit Oligonukleotiden in geringem Abstand zueinander "gekachelt". Ein neues Transkript-Fragment ist entdeckt, wenn man für eine durchgehende Folge von Oligonukleotiden ein Signal sieht und dieses nicht durch

[1] `http://www.genome.gov/10005107`

bekannte Transkripte oder Kreuzhybridisierungen erklärt werden kann. Kapitel 7 zeigt, wie die Methoden aus den vorangehenden Kapiteln effizient für diese Aufgabe eingesetzt werden können. Dabei ist insbesondere zu beachten, dass man die Oligonukleotide im allgemeinen nicht spezifisch ausgewählen kann.

- In Kapitel 8 betrachten wir ein Problem aus der Chipproduktion. Sind die Oligonukleotidsequenzen ausgewählt, so müssen sie auf dem Chip synthetisiert werden. Dies geschieht mit Hilfe photolithographischer Techniken und kombinatorischer Chemie. In einem Syntheseschritt werden ausgewählte Oligonukleotide um dasselbe eine Nukleotid verlängert. Aus Wirtschaftlichkeitsgründen und um die Fehlerzahl zu minimieren, soll die Anzahl der Syntheseschritte minimiert werden. Dies führt auf das Problem, die *kürzeste gemeinsame Supersequenz* von Tausenden sehr kurzer Sequenzen zu bestimmen. Wir stellen eine Heuristik für dieses Problem vor und berechnen die Verteilung von oberen und unteren Schranken für die Supersequenzlänge zufällig gezogener Oligonukleotide.

Die abschließende Diskussion in Kapitel 9 fasst die Ergebnisse der Arbeit zusammen und wagt einen Ausblick in die Zukunft dieses sich sehr schnell entwickelnden Forschungsgebietes.

C.2 Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass diese Dissertation bei keinem anderen Fachbereich zur Begutachtung eingereicht wurde.

Sven Rahmann Berlin, im Februar 2004

C.3 Lebenslauf

Dipl.-Math. Sven Rahmann
Schlossstraße 31
12163 Berlin
Tel.: (030) 79 70 95 17
E-mail: `Sven.Rahmann @ molgen.mpg.de`

Geburtsdatum: 1. August 1974
Geburtsort: Hamburg
Staatsangehörigkeit: Deutsch
Familienstand: ledig

Akademischer Grad

Diplom-Mathematiker 13. Dezember 2000

- Diplomarbeit: "Word Statistics in Random Texts and Applications to Computational Molecular Biology"
- Betreuer: Prof. Dr. Enno Mammen und Prof. Dr. D.W. Müller (Universität Heidelberg), Dr. Eric Rivals (DKFZ; jetzt LIRMM Montpellier, France), und Prof. Dr. Martin Vingron (DKFZ; jetzt MPI für molekulare Genetik, Berlin).

Ausbildung

Seit April 2002 Wissenschaftlicher Mitarbeiter am Fachbereich Mathematik und Informatik, speziell im Studiengang Bioinformatik, der Freien Universität Berlin

Seit Januar 2001 Doktorand in der Abteilung Bioinformatik am Max-Planck-Institut für Moleulare Genetik, Berlin

September 1994 – Dezember 2000 Universitätsstudium der Mathematik und Informatik an folgenden Orten:

- Georg-August-Universität, Göttingen (09/1994 – 09/1997)
- University of California Santa Cruz (09/1997 – 08/1998)
- Ruprecht-Karls-Universität Heidelberg (09/1998 – 12/2000)
- Diplomarbeit in der Arbeitsgruppe Theoretische Bioinformatik am Deutschen Krebsforschungszentrum (DKFZ), Heidelberg (10/1998 – 12/2000)

August 1981 – Juni 1994 Schulausbildung:

- Grundschule in Rellingen
- Wolfgang-Borchert-Gymnasium in Halstenbek
- Deutsche Schule Toulouse in Colomiers bei Toulouse, Frankreich
- Abschluss: Reifeprüfung an der Deutschen Schule Toulouse

Sven Rahmann Berlin, im Februar 2004