Department of Computer Science
Series of Publications A
Report A-2004-8

Computational Tools for a Novel Transcriptional Profiling Method

Teemu Kivioja

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Room 5, University Main Building, On November 5, 2004, at noon.

University of Helsinki
Finland

Contact information

Postal address:
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki
Finland

Email address: postmaster@cs.Helsinki.FI (Internet)

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 952-10-2130-6 (paperback)
ISBN 952-10-2131-4 (PDF)
Computing Reviews (1998) Classification: J.3, F.2.2
Helsinki 2004
Helsinki University Printing House

Computational Tools for a Novel Transcriptional Profiling Method

Teemu Kivioja

Department of Computer Science
P.O. Box 68, FIN-00014 University of Helsinki, Finland
Teemu.Kivioja@cs.Helsinki.FI
http://www.cs.helsinki.fi/u/kivioja/

PhD Thesis, Series of Publications A, Report A-2004-8
Helsinki, October 2004, 98 pages
ISSN 1238-8645
ISBN 952-10-2130-6 (paperback)
ISBN 952-10-2131-4 (PDF)

Abstract

In this thesis we provide computational tools for the planning of VTT-TRAC experiments. VTT-TRAC is a novel method for measuring expression levels of genes. Monitoring gene expression by measuring the amounts of transcribed mRNAs (transcriptional profiling) has become an important experimental method in molecular biology. This has been due to rapid advance in the high-throughput measurement technology. Methods like microarrays are capable of measuring thousands of expression levels in one experiment. However, new methods that are fast and reliable are still needed. In addition to scientific research, such methods have important applications for example in medical diagnostics and bioprocess control.

High-throughput technologies require automatic tools for planning of experiments. In VTT-TRAC a special fragment of DNA called a *probe* has to be selected for each profiled gene. In addition, the method allows *multiplexing* i.e. profiling a large number of genes together as a group called a *pool* as long as the probes in each pool can be separated from each other. Thus, the major computational challenge is to divide the probes into as small a number of pools as possible so that the whole experiment can be done as cost-efficiently as possible.

In this thesis we analyze and solve the key computational problems in the automatic planning of VTT-TRAC experiments. Especially, we show that

dividing the probes into a minimal number of pools is an NP-hard problem and give an efficient approximation algorithm that is also practical. In addition, we develop a flexible method for the selection of probes. We have implemented our algorithms as a fully automatic planning software. We show by planning different kinds of real experiments that our tools allow convenient planning of cost-efficient experiments. We demonstrate computationally that almost the whole yeast genome could be profiled with less than 50 pools.

The planning of a VTT-TRAC experiment requires sequence information that is not available for all organisms. cDNA-AFLP is a method that can be used to study gene expression without complete sequence information. We propose a computational method that can be used to optimize cDNA-AFLP and VTT-TRAC experiments based on partial sequence information or sequence information on a related organism.

Computing Reviews (1998) Categories and Subject Descriptors:
J.3 Life and Medical Sciences—biology and genetics
F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—pattern matching, sequencing and scheduling

General Terms:
Algorithms

Additional Key Words and Phrases:
Bioinformatics, experiment design, transcriptional profiling, hybridization probe, multiplexing

Acknowledgements

I would like to thank my supervisor Professor Esko Ukkonen for his support and encouragement during this work. He has a truly remarkable understanding of computer science and how it can be applied to new problems. I am grateful that I have had this opportunity to learn from him.

The work presented in the thesis has been done in collaboration with VTT Biotechnology. The collaboration has been lead by Professor Ukkonen from the Department of Computer Science and Research Professors Hans Söderlund and Merja Penttilä from VTT. I would like to thank them and the developers of the TRAC method for providing such an excellent topic for a thesis. I hope the thesis is worthy of the tasks it aims to solve. I have enjoyed the collaboration with Kari Kataja, Jari Rautio, Reetta Satokari, and Mikko Arvas. Particularly, I would like to thank Mikko Arvas for patiently explaining to me many of the biological issues.

I wish to thank the Research Professor Hans Söderlund and Professor Inge Jonassen for their comments on the manuscript of the thesis that helped me to improve it and Marina Kurtén for correcting its language.

The Department of Computer Science of the University of Helsinki has provided me an excellent working environment. The computing facilities staff of the department deserve special thanks for their outstanding work. Graduate School in Computational Biology, Bioinformatics, and Biometry (ComBi) and the From Data to Knowledge research unit (FDK) have provided me financial and other support.

I have had the privilege of having incredible colleagues and friends at the department who I wish to thank: Kimmo Fredriksson, Taneli Mielikäinen, Anatoly Verkhovsky, Veli Mäkinen, Ari Rantanen, Matti Kääriäinen, Pasi Rastas, and the whole group of people who used to gather at the corner table of Vallila Unicafe at around 4 p.m. I have learned a lot from them but more importantly they have made these years enjoyable. I wish to especially thank Janne Ravantti for his support. I have always been able to count on his help, whatever the problem was.

Finally, I would like to thank my parents, Kirsti and Aaro, for all the support over the years.

Helsinki, October 2004
Teemu Kivioja

Contents

Chapter 1

Introduction

1.1 Multiplexing of experiments in molecular biology

A revolution in the measurement technology has started a new era in molecular biology. Previously biologists typically studied one entity such as a single gene or protein at a time. Now the new emerging high-throughput technologies enable gathering measurement data about dozens or even thousands of genes, proteins, or interactions in a single experiment.

Such powerful technologies also bring along new kinds of challenges. The new challenges for the analysis of the biological data have been discussed widely in bioinformatics but the other side of the matter, namely the experiment planning, has not raised as much interest. As the measurement technologies become more complex and allow large-scale experiments to be performed, designing the experiment in such a manner that it produces as much biological information as possible also becomes a challenge.

The subject of this thesis is how to use computational tools to organize the measurements as efficiently as possible in a novel method called VTT-TRAC [SIP$^+$03, SKS, KSA$^+$04, RKS$^+$04]. It is a new gene expression monitoring method developed at VTT Biotechnology. Our aim is to organize the measurements in such a way that the whole TRAC experiment can be done using a minimum amount of resources. If a measurement technique is such that one can only measure one variable at a time, such as the expression level of one gene in one condition, the problem is trivial: The cost of the experiment is the cost of measuring one expression level times the number of studied genes and conditions. However, VTT-TRAC and many other new measurement techniques are such that they allow *multiplexing* i.e. measuring several variables simultaneously if some constraints are not

violated. Multiplexing can dramatically lower the cost of experiments and allow large-scale experiments that would not be feasible otherwise.

The multiplexing task in VTT-TRAC and several other methods can be described as follows. We have a set of variables X of some biological entities whose values we want to measure, for example the expression levels of a set of genes. The characteristics of the entities, such as the DNA sequences of the genes, and the measurement technology used determine whether a subset of variables can be measured together. In the VTT-TRAC the constraint posed by the measurement device is simple: The amounts of several different molecules in the sample can be measured simultaneously if the molecules have substantially different sizes. We call a *pool* a subset of variables that can be measured together. Then the computational task is to partition X into a minimal number of pools. Such a partition allows measuring all variables in X with minimal cost.

Other examples of such multiplexing tasks include multiplexing PCR [NS97] and genotyping using mass spectrometry [AMY03] or all k-mer arrays [SBDY04]. Other examples are likely to arise since the technology is developing rapidly. For example, quantification of all proteins (proteomics) or metabolites (metabolomics) in a cell sample are enormous challenges that will require clever techniques for dividing the task into manageable parts.

Optimization problems like the general multiplexing task described above are not new in computer science. On the abstract level they are similar to many planning tasks arising in the fields like warehousing, logistics, and scheduling [AMO93]. These *combinatorial optimization* problems have been one of the major inspirations for algorithm and operations research ever since the dawn of the computer age in the late 1940s. Typically they involve selecting the optimal one from a vast but finite number of different alternatives. Since such a problem can in principle always be solved by enumerating all possible solutions and choosing the best, the time requirement of the algorithm that is used to solve the problem becomes the critical question.

Since VTT-TRAC measures mRNA levels, the planning of the experiment requires sequence information. One needs to take into account not only the sequences of the measured mRNAs but also the sequences of all other mRNAs that might be present in the sample. In some cases the whole genome of the target organism has to be taken into consideration. Thus, the planning of VTT-TRAC experiments involves computations on large sets of sequences. Luckily, the enormous increase in sequence data in biology has inspired computer scientists to create efficient computational methods for handling such data. Since biological sequences can be interpreted as

strings of characters, sequence analysis algorithms are usually algorithms on strings from the point of view of computer science [Gus97].

In this thesis we apply techniques from both combinatorial optimization and string algorithms to create practical automatic tools for the planning of VTT-TRAC experiments. We believe that similar computational approaches could be useful for other new measurement methods as well, especially if the computational aspects are taken into account at the measurement technology development stage, as has been the case with VTT-TRAC.

1.2 Transcriptional profiling

Monitoring the expression of genes by measuring the amounts of the transcribed mRNAs (transcriptional profiling) has become one of the most important methods in functional genomics. This is largely due to microarrays that allow measuring thousands of mRNA levels in parallel. One of the first pioneering articles on the microarray technology published in 1996 [LDB+96] started with the following words:

> The human genome encodes approximately 100,000 different genes, and at least partial sequence information for nearly all will be available soon. Sequence information alone, however, is insufficient for a full understanding of gene function, expression, regulation, and splice-site variation. Because cellular processes are governed by the repertoire of expressed genes, and the levels and timing of expression, it is important to have experimental tools for the direct monitoring of large numbers of mRNAs in parallel.

Now, almost a decade later, the statement is even more true than it was back then: The whole human genome has now been sequenced and to everybody's surprise it might contain as few as about 30,000 protein-coding genes, only about twice as many as the worm or fly genome [L+01]. Thus, knowing when the genes are expressed, as opposed to simply knowing the list of genes and their functions, now seems even more important for understanding complex organisms such as ourselves than could be imagined a decade ago.

Despite the success of microarrays during the last ten years or so, there is still a requirement for new methods that enable robust, sensitive, and cost-efficient transcriptional profiling of a set of genes [RKS+04]. Microarrays are a highly parallel but expensive technique and have relatively

low sensitivity and poor reproducibility. There are other high-throughput methods with their own advantages and disadvantages but none of them has yet gained wide popularity comparable to microarrays. Conventional methods such as Northern blots are inherently serial, suitable for measuring a single mRNA at a time [LDB+96].

1.3 VTT-TRAC

VTT-TRAC (transcriptional profiling with the aid of affinity capture) is a novel method that allows fast gene expression monitoring [SIP+03, RKS+04]. The key techniques used in TRAC are:

Hybridization Like microarrays, TRAC utilizes *hybridization* i.e. forming of a double helix by base pairing of two complementary single-stranded chains of nucleic acids. The amount of an mRNA molecule is measured using a *probe*: a fragment of DNA or RNA that is designed to selectively hybridize with its target mRNA. The probes are fluorescently labeled to allow detection and quantification. The probes used in TRAC can be either short oligonucleotide probes, typically 20..40 bases long, or they can be longer PCR-amplified probes, typically 50..1000 bases long.

PCR PCR (polymerase chain reaction) is a standard method for copying and amplifying a fragment from a larger sequence, for example from the whole genome. PCR is also based on specific hybridization. A pair short fragments of DNA called *primers* are specifically designed for the amplified region so that each of them is complementary to 3' end of one of two strands of the region. Given single stranded DNA molecules, the primers hybridize to their binding sites flanking the target region. An enzyme called DNA polymerase adds nucleotides after the 3' end of each primer using the other strand as template and thus builds a copy of the strand started by the primer. Thus, the 3' ends of the primers are especially important because the polymerase reaction starts only if the 3' ends of the primers bind tightly. The molecules are made single stranded again by heating and the process is repeated many times (20-30) resulting in an exponential blow-up in the number of copies of the target region.

Affinity capture The mRNAs are labeled with a small molecule called biotin so that they can be captured later with streptavidin-coated magnetic beads. Streptavidin is a protein that binds tightly to biotin.

Electrophoresis Electrophoresis allows separating nucleic acids according to their size. Both capillary and gel electrophoresis can be used. Capillary electrophoresis is routinely used for DNA sequencing, so the equipment is available in most laboratories.

The main steps of TRAC analysis are as follows [RKS+04]. The steps are also illustrated in Figure 1.1.

1. The biotin-labeled mRNA sample is hybridized in solution with the fluorescently labeled detection probes.
2. The mRNA molecules are captured and the rest of the material is washed off. Thus, the probes that are not bound to the RNAs are also removed.
3. The hybridized probes are separated from the mRNA molecules.
4. The probes are identified and quantified using electrophoresis. The size of the probe identifies the mRNA. The fluorescence intensity of the probe labels gives the quantification.

In TRAC the hybridization is done in solution which removes some of the problems in the methods [RKS+04] in which the hybridization is done on a solid phase. TRAC has been shown to be highly sensitive and reproducible. TRAC analysis is fast compared to many other methods. The whole analysis can be done in a few hours, which makes TRAC also suitable for bioprocess control or analysis of clinical samples. A large part of the TRAC analysis has already been automated and the aim is to fully automate the process.

From the computational point of view the key feature of TRAC is that it can be multiplexed: Two mRNAs can be analyzed simultaneously as long as their probes can be separated by electrophoresis at the end of the process. An accurate quantification is possible only if different probes are reliably separated from each other. Thus, the *resolution* of the electrophoresis equipment, i.e. the smallest size difference it can reliably detect, has to be taken into account in the multiplexing. The speed of the analysis, automation, and multiplexing together make high-throughput TRAC analysis possible.

1.4 Planning of TRAC experiments

The subject of this thesis is computational tools that allow efficient planning and multiplexing of TRAC experiments. As in other hybridization-based

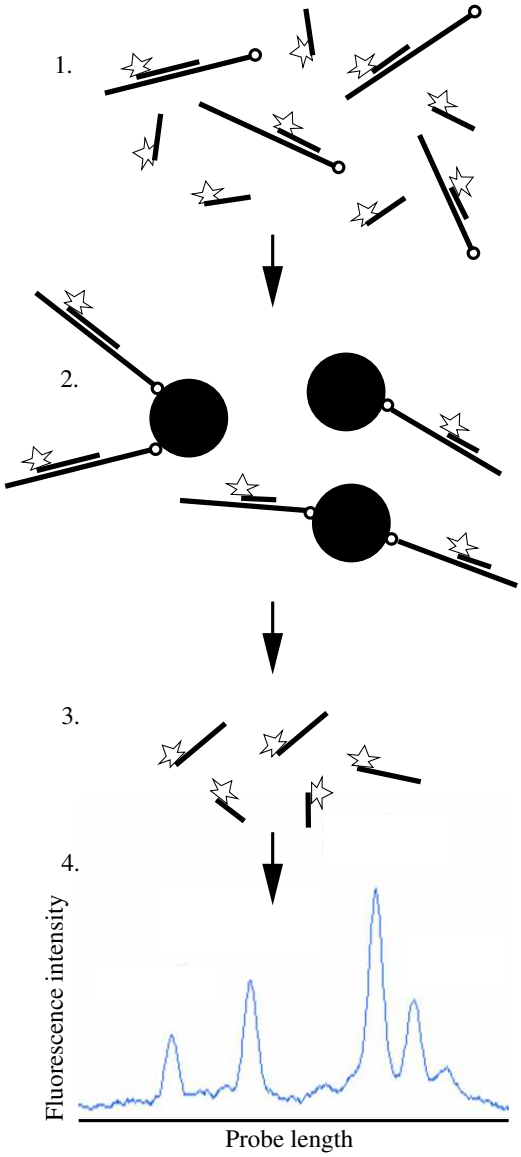

Figure 1.1: VTT-TRAC analysis [RKS+04]. Biotin-labeled mRNAs are hybridized with the fluorescently labeled probes (1). The mRNAs are captured to streptavidin-coated magnetic beads (2). The bound probes are separated from mRNAs (3) and finally detected using electrophoresis (4).

methods, a specific hybridization probe has to be selected for each profiled gene. However, the major new computational challenge in TRAC is the efficient multiplexing of the measurements. The probes must be divided into pools such that the probes in each pool can be distinguished from each other based on their different size. The minimization of the number of pools is important because the cost of the analysis depends heavily on the number of pools.

One of the major advantages of TRAC is that a multiplexed analysis is simple to set up for a any set of genes: for a few dozen genes involved in a certain biological process or for all genes of a genome if enough resources are available. Thus, the computational tools should maximally support that advantage.

The plan of the TRAC experiment should fulfill the following basic requirements:

- Each profiled gene has a specific probe suitable for hybridization (and amplification if PCR-based probes are used). In particular, the sequence of the probe must identify a unique transcribed region in the genome to avoid hybridization to other mRNAs except the target of the probe.

- The probes should be of a size that is within the range of DNA sequencers (or within the size range of any other measurement device used instead of a sequencer). In case of oligonucleotide probes the length of probes is limited by the technology used to produce the oligos.

- The probes should be assigned into a minimal number of pools with the property that the sizes of all the probes in one pool should be distinct so that they are well separated and quantifiable in the electrophoresis.

We divide the computational problem of finding such a plan into two major steps:

1. Generate a large set of potential probes called *candidate probes*. Each candidate probe should fulfill all the requirements of a good hybridization probe. Each profiled gene should have at least one candidate probe but hopefully many candidate probes of different sizes.

2. Select from the candidate probes a representative probe for each gene and assign it into one of the pools such that the total number of pools becomes as small as possible.

The requirements for the first step depend mainly on the sample, hybridization conditions, and how the probes are produced. The requirements for the second step depend mainly on the device used to separate and detect the probes at the end of the experiment. In the software implementation the two steps are combined into one automatic process.

The description of the computational problem given above also contains implicit assumptions. In step 1, the properties of each candidate probe are considered separately. In step 2, the only properties of the probes that are considered in the pooling are their sizes. Thus, we assume that the hybridization between different probes in the same pool is not a significant problem. In addition, we assume that we can select the probes in step 1 so that the same hybridization temperature can be applied for all pools (we relax this assumption in Section 3.6). So far, the laboratory experiments support both assumptions.

1.5 The contributions and the structure of the thesis

We present a complete set of computational tools for efficient planning and multiplexing of VTT-TRAC experiments. The main contributions of the thesis are:

- We identify the key computational problems in the planning of VTT-TRAC experiments and formalize them. The problem of selecting hybridization probes (step 1) is very similar to selecting microarray probes (see e.g. [LS01]). The problem of assigning probes into a minimum number of pools according to the size (probe assignment, step 2) is a new one in bioinformatics. However, we show that the abstract formalization of the problem is similar to certain problems in the scheduling theory (see e.g. [ES03]).

- We give algorithms for solving the key computational problems. We show that the probe assignment problem is NP-hard. The main technical contribution of the thesis is a factor 2 approximation algorithm for the probe assignment problem. Algorithms for the maximization version of the same abstract optimization problem have been given in the scheduling literature. However, our algorithm is the first one for the minimization version.

- Our computational procedure for generating the candidate probes is a novel combination of well-known algorithms and software tools. It

allows strict requirements to be set for the selected probes to ensure that they are suitable for hybridization. The most important requirement is that the sequences of the probes are unique. In addition, the procedure includes tools for the estimation of the thermodynamic properties of probes such as the melting temperature. Similar requirements can be set for the primers of the PCR-amplified probes. The procedure is guaranteed to select only probes that fulfill all the requirements. It is fast enough for selecting a probe for all genes of the yeast genome. Somewhat similar approaches have been used independently in several microarray probe selection tools. However, to our knowledge our system is the only one that can select both oligo and PCR-based probes that fulfill stringent requirements. Our method is practical for both small and large sets of profiled genes.

- We have implemented our algorithms as a software package called Tracfinder. The software allows fully automatic selection of probes and their assignment into pools. We show by using the software for planning of real experiments that our methods are practical. In addition, we demonstrate with the yeast genome that from a computational point of view large-scale VTT-TRAC experiments are feasible. We are able to pack on average about 130 probes into one pool which means that the whole yeast genome could be easily profiled with one 96-well plate.

- As a partly distinct contribution from the above ones, we study the selection of enzymes and selective PCR primers for cDNA-AFLP experiments. cDNA-AFLP [BvdHdB$^+$96] is a transcriptional profiling method that can be used also when complete sequence information is not available. We study planing of cDNA-AFLP experiments because the same computational tools could be used to optimize TRAC experiments for (partly) unknown genomes.

  We are the first to study the problem as a rigorous optimization problem. We show that the general form of the problem is NP-hard. We give a polynomial algorithm for a restricted case of the problem and an efficient heuristic algorithm for the general problem. By simulating cDNA-AFLP for several large data sets, we show that our optimization methods could save a significant amount of resources in cDNA-AFLP experiments.

Some of the results on the planning of TRAC experiments were originally published in [KAK$^+$02] but the topic is covered more extensively in

this thesis. In particular, the probe assignment problem is shown NP-hard and its connections to other combinatorial problems are discussed more.

The structure of the thesis is as follows. In Chapter 2, we describe our method for finding the candidate probes. This Chapter also includes a literature review of hybridization probe selection. In Chapter 3, we formalize the problem of assigning the probes into pools, show that the problem is NP-complete, and give an approximation algorithm for it. In Chapter 4, we introduce our software for TRAC experiment planning and multiplexing. In Chapter 5 we describe how we have used and modified the software to do experiment plans and show the results. In Chapter 6, we propose methods for optimizing cDNA-AFLP experiments. In the last chapter we summarize the current state of the automatic planning of VTT-TRAC experiments.

Chapter 2

The selection of hybridization probes

In this chapter we concentrate on the selection of good hybridization probes. We start by describing the main characteristics of such probes and then do a literature review of the other probe selection methods. After that we describe our own method.

2.1 Probe specificity and sensitivity

The basic demands for a good probe are the same for all hybridization-based measurement techniques. Thus, most of the discussion in this chapter applies to microarray techniques [SSDB95, LDB+96] as well as to TRAC. The most important difference between TRAC and microarrays in terms of individual probes is that in TRAC the hybridization is done in solution while in microarrays the probes are attached on a solid phase. However, there is not enough experimental data yet about differences in solution-based and solid-phase-based hybridization in order to design the probes specifically for either platform. We use the existing microarray literature, many of the basic concepts are from Schena[1] [Sch03].

We call a *probe* a fragment of DNA which is used to hybridize with a nucleic acid, DNA or RNA, called a *target*, in order to measure the amount of the target. Typically, the target is an mRNA molecule whose amount is measured with a DNA probe in order to assess the expression level of the corresponding gene. Hybridization is a chemical reaction where two

[1]Note however that Schena uses the words probe and target in exactly the opposite manner compared to us. In the microarray technique the mRNA molecules are labeled and thus Schena calls them probes.

complementary nucleic acid strands form a double helix by base pairing: guanine (G) with cytosine (C) and adenine (A) with thymine (T) [AJL+02]. We assume that the sequence of the probe is always an exact complement of a substring of the target sequence. We call this substring the *target substring*.

For each gene of length n there are $n - m + 1$ possible candidate probes of length m. Which one of them is the best? First, a probe has to be *sensitive* i.e. it should produce a clear signal by binding to the target molecule. Second, a probe has to be *specific* i.e. it should not bind to any other molecule that might exist in the sample. If a probe binds to such *non-target* molecules, the phenomenon is called *cross-hybridization*. Cross-hybridization is possible because two nucleic acids can hybridize even if they are not exact complements. Continuing the expression profiling example, a probe should not bind to mRNA:s of other genes of the organism. We call the sequences of non-target molecules *non-target sequences*. Only if a probe is both sensitive and specific, it can be used reliably to measure the amount of its target in a complex mixture of molecules.

The sensitivity of the probe depends on the efficiency of hybridization reaction between the probe and the target and the strength of the resulting *duplex* molecule. Hybridization occurs by hydrogen bond formation between the bases of complementary nucleic acids. A base pair of guanine (G) and cytosine (C) contains three hydrogen bonds but a base pair of adenine (A) and thymine (T) contains only two hydrogen bonds. Also the so-called stacking interactions between neighboring base pairs are stronger for sequences rich in C and G [vHJH98]. Thus, the strength of the duplex depends heavily both on the length of the probe and its sequence composition: roughly speaking a long probe which consists mostly of C and G forms a much stronger duplex with the target than a short probe composed mostly of A and T.

Hydrogen bonds can also be formed between different bases of one single-stranded nucleic acid. Such intramolecular base pairing is called the *secondary structure* of the nucleic acid. RNA forms stable secondary structures more easily than DNA. The secondary structure of the target or probe or both can have a significant effect on their interaction. It is assumed that the duplex formation begins by base pairing of a few unpaired bases and continues, one base pair at a time, through a zippering process [SMS99]. Stable secondary structures can inhibit this process.

The interaction of a probe and its target is measured by the *melting temperature* (T_m) which is the temperature at which half of the complementary molecules are in the duplex state. In addition to the characteristics of probe

and target molecules, the melting temperature depends on the conditions of the experiment such as the salt concentration of the solution. All the probes that are hybridized together should have similar melting temperatures. Only then the hybridization temperature can be chosen so that the probes have similar sensitivity in the experiment.

The specificity of a probe depends on its interactions with non-target molecules. The risk of cross-hybridization is real because genomes contain *homologous* genes i.e. genes that have similar sequences because they have evolved from a common ancestral gene. Mismatching base pairs between the probe and the non-target do reduce hybridization efficiency and lower the melting temperature of the duplex. The signal from unspecific hybridization can still be significant for example if the similar non-target molecules are present in the sample in a higher concentration than the true target molecule. There probably are cases where there the differences between homologous genes are so small that they cannot be separated by hybridization techniques no matter which probe is selected.

2.2 Methods proposed for predicting the sensitivity and specificity of a probe from the sequence

In theory, the sensitivity and specificity of a probe can be predicted from the sequence using thermodynamic principles. Omitting all details, we simply state that the interesting quantity is the free energy change (ΔG) when a structure is formed. A large negative value of ΔG indicates that the reaction has a strong tendency to occur [AJL$^+$02]. Thus, an ideal probe for a target would be the one which has low hybridization free energy for the target and high hybridization free energy for all non-targets [LS01].

The so-called nearest-neighbor (NN) model is the basic approach to compute free energy change and melting temperature for hybridization of two complementary nucleic acids [San98]. The nearest-neighbor model assumes that the stability of a given base pair depends only on itself and the two neighboring base pairs. Then the free energy change in helix formation can be approximated for any sequence from the experimentally measured energy parameters for all combinations of two consecutive base pairs.

A lot of work has been done to extend the approach to a prediction of secondary structure formation by measuring energy parameters for other small structures such as mismatching base pairs and loops [ST97, MSZT99]. The assumption is that the energy of a secondary structure can be obtained by summing the energy contributions of its substructures [HFS$^+$94]. Then

the minimum free energy structure can be found by dynamic programming. The algorithm can be generalized in a straightforward manner to predict the free energy change in hybridization of two nucleic acids that are not exact complements [MBF+99].

Unfortunately, sequence-based thermodynamic predictions are not completely reliable at present [MSZT99]. In addition, the dynamic programming algorithm for computing a minimum free energy structure has time complexity $O(n^3)$ where n is the length of the sequence, see for example [SM97] (see [Zuk03] for more complete historical notes). Thus, it is not computationally feasible to use even a crude thermodynamic model to predict all interactions in a complex mixture of nucleic acids. Finally, the interactions depend also on the concentrations of different molecules in the sample but in transcriptional profiling the concentrations of targets are the variables we want to measure!

Several attempts have been made to use the above thermodynamic model to predict which short oligonucleotides hybridize best to long target molecules, see [MBF+99, LBG03, MSN+03] and the references in these articles. Array technologies have made it possible to test thousands of probes [LBG03, MSN+03]. First of all, the experiments show that most short oligos (about 20bp) do not bind efficiently to their RNA targets [LBG03, TMN+02], at least not in array conditions. The efficiency does correlate with thermodynamic properties, especially with free energy change in hybridization and to lesser extent with secondary structure of the probe, but reliable prediction has proven difficult. It seems especially difficult to predict the effect of the secondary structure of the RNA target even though some suggest it to be in reality the principal factor influencing the hybridization efficiency [LBG03].

Since hybridization depends heavily on the details of measurement technology and experiment conditions, we have analyzed measurement data from experiments that are as close as possible to TRAC technology. In particular, in all these experiments hybridization were done in solution. The data consisted of several sets of measurements, one typically evaluating 2-5 different probes for half a dozen different targets. The details such as experiment conditions or how the tested probes were selected differed between sets. The measurements have been done by Jari Rautio at VTT Biotechnology and by several people[2] at the University of Oulu Bioprocess Engineering Laboratory.

The thermodynamic properties that we have computed and compared to experimental data were the free energy change in the hybridization

[2]Daniela Böhm, Christina Falschlehner, Rami Kuivila, and Timo Nieminen

(computed with program Melting [Nov01]), the melting temperature of the duplex (Melting [Nov01]), and the free energy change of the probe (Mfold [Zuk03]) and the target secondary structure (ViennaRNA [HFS+94]). The most complex of these to estimate is the energy change of the target secondary structure. We assumed that the global secondary structure of the RNA molecule is maintained when a short probe binds to it. Thus, we first computed the minimum energy secondary structure of the target RNA without any constraints and secondly computed the energy of the structure that has otherwise the same base pairing but the probe binding site is free. The energy change of the target secondary structure was approximated as the difference of these two energies (this measure is called A_c in [LBG03]). At the beginning, we also experimented with a measure that allows the global structure of RNA to change (A_u in [LBG03]) but since it gave worse predictions, we discarded it. Note that both measures are simplifications, for example an RNA molecule can have several different stable secondary structures.

As one example of the data, the set of measurements done in conditions that are closest to current oligo-TRAC conditions consisted of 4-5 different probes for 4 targets, 19 probes in total. The lengths of the DNA probes were between 25 and 33 bases. The probes had been selected using the program Primer3 [RS98] so that they have similar melting temperatures and do not form hairpin loops. Thus, it was not surprising that the hybridization and probe secondary structure energies did not correlate with the hybridization signal (correlation coefficients -0.04 and -0.13). On the other hand, the target secondary structure properties partly explained the remaining signal differences as shown in Figure 2.1 (correlation coefficient 0.52).

In general, we could not draw any definite conclusions since the number of tested probes and targets was relatively small. In particular, there were only a few really bad probes with very low signal. The results also varied depending on the conditions of the experiment and the probe-selection criteria. As expected, the free energy change in hybridization and melting temperature correlated with signal intensity unless probes had been selected using a program that takes the melting temperature into account. We did not observe any significant correlation between the probe secondary structure and the hybridization signal. In some sets we did observe a clear correlation between the target secondary structure and hybridization signal.

An alternative approach to thermodynamic predictions is trying to learn to predict probe sensitivity and specificity from simple sequence features without any explicit physical model [LDB+96]. One advantage of such

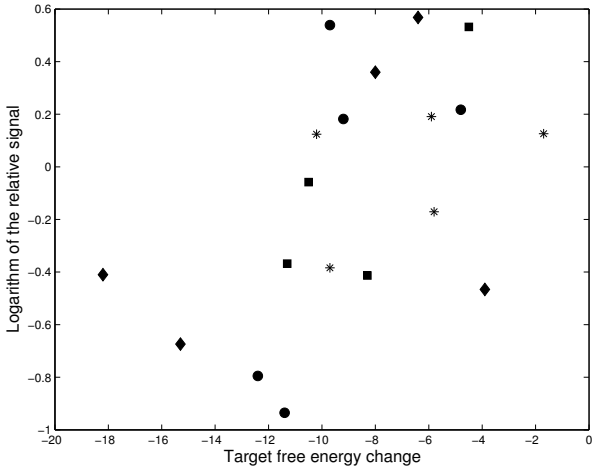

Figure 2.1: The effect of the target RNA secondary structure on a hybridization signal, experimental data from Jari Rautio. The natural logarithm of a relative hybridization signal of the probe is plotted as a function of the predicted target energy change in hybridization (kcal/mol). The relative hybridization signal is defined as the signal of the probe divided by the average signal of all the probes of the same target. Each different marker in the figure corresponds to one target.

an approach is that simple sequence features are usually faster to compute than the thermodynamic properties. Tobler et al. [TMN+02] have evaluated the capability of several machine learning algorithms to predict microarray probe sensitivity. Not surprisingly, the amount of cytosine (C) was found to be the most predictive feature.

In several studies cross-hybridization has been tested experimentally and compared directly with sequence similarity. The concepts used, sequence similarity and identity, are not rigorously defined in the articles. We assume that both refer to the percentage of identical aligned bases in the best global alignment between two sequences (under some unknown scoring scheme).

Kane et al. [KJS+00] have tested 50 bases long and Hughes et al. [HMJ+01] 60 bases long oligonucleotide probes in a microarray setting. Kane et al. suggest that cross-hybridization is small if the sequence similarity between the target substring and non-target sequence is smaller than 75%. In addition, if the sequence similarity is 50%-75% then the sequences should not have a common substring longer than 15 bases. According to Hughes et. al,

if the sequence identity is less than 70%, the signal from cross-hybridization is reduced to the background level. They also note the correlation between hybridization and the free energy of the longest perfectly matching sequence but suggest no threshold for that.

Evertz et al. [EAYR+01] and Xu et al. [XBD+01] have tested cross-hybridization of PCR-based microarray probes of variable length. They both suggest that the overall sequence identity between probe and non-target sequences should be less than 80% to avoid severe, $> 20\%$, cross-hybridization. The cross-hybridization was measured as the ratio of the signal given by the non-target and the signal given by the correct target when there are equal amounts of both molecules in the sample. Girke et al. [GTR+00] also provide some evidence that 80% is a good threshold for overall sequence identity of long PCR probes. Both Evertz et al. and Xu et al. also note that a relatively long common substring can result in cross-hybridization even when the overall similarity is not high. Wren et al. [WKJ+02] have tested the effect of these identical stretches in cross-hybridization of PCR-based probes. Interestingly, they take into account the more stable pairing between bases G and C than between A and T. They use as a measure the maximum number of hydrogen bonds in any helix corresponding to a common substring of the probe and the non-target sequence. They observed significant cross-hybridization when the number of hydrogen bonds is larger than 45 (which corresponds to a ≥ 15 bases long common substring).

The results probably need to be treated as guidelines, not as exact thresholds. Most of the cross-hybridization studies are based on evaluation of a couple of different targets and a few dozen different probes. In addition, this kind of studies are extremely sensitive to experimental conditions such as temperature and salt concentration which makes the comparison of experiments difficult. Nevertheless, it is encouraging that the results reported in different articles are in quite good agreement. It would be interesting to see how the experimental data correlates with other measures for comparing sequences or with thermodynamic properties.

2.3 Fast computation of specificity

There has been an avalanche of articles about hybridization probe selection in the last couple of years, mostly due to the popularity of microarrays. The situation is a bit discouraging because a large number of quite similar methods have been developed simultaneously. Thus, at this point it is hard to notice clear progress towards better methods. Nevertheless, we try to

provide the most comprehensive literature review to date on this subject. Hopefully we are able to show what type of approaches have already been explored so that in the future one could concentrate on either improving the old ones or discovering totally new ones.

Predicting the specificity of probes seems to the task in probe selection that has been solved in several different ways. It is also the most computationally intensive one of the common tasks. There seems to be relatively wide consensus that at least for short oligos the thermodynamic predictions based on the nearest-neighbor model would be the most accurate method to assess the specificity of a probe. However, on a genomic scale it is considered to be a computationally infeasible approach. Thus, all methods at least initially use some simple measure for comparing a target substring to non-target sequences to predict the specificity of the probe. In order to highlight the differences and especially similarities of the methods, we classify them according to the applied measure.

Another critical decision is the threshold value of the sequence comparison measure that is used to separate the specific probes from the rest. A good specificity criterion is such that it never accepts a probe that is not specific and only seldom rejects a probe that is actually specific. In addition, a good criterion is such that it can be evaluated fast on genomic scale using a reasonable amount of space. A significant amount of time can be used to preprocess the sequence data because usually a large number of probe candidates needs to be evaluated. It is difficult to compare different methods because both the chosen measure for sequence comparison and the threshold value have huge impact on the time requirement. On the other hand, there is not enough experimental cross-hybridization data available to judge which measures and thresholds are justified and reliable.

In many of the methods a fast string comparison is performed first to narrow the search to the most similar non-target sequences and then a slower but more accurate method is used to make the final determination. The specificity criterion can also be a combination of several sequence comparison measures.

The discussion will be mostly informal since the methods do not naturally fall into any common framework. A few definitions are still useful. Let P be the target substring and T the set of all non-target sequences. In addition, let G be the set of all gene sequences of a genome. Often the task is to find a specific probe for every gene in a genome. In that case every probe length substring of every sequence S in G is a potential target substring for a probe. If a probe sequence P is a substring of S, then the set of non-target sequences T for P is usually $G \setminus \{S\}$ (in some cases the sample

can also include sequences of other organisms that have to be avoided).

The first two measures, edit distance and Hamming distance, are *distance measures* between two sequences. They measure how many operations, or in a biological context how many mutations, are needed to transform one sequence into the other. When a distance measure is used, specificity prediction becomes equivalent to *approximate string matching*. We define approximate string matching problem as follows: Given the target substring P, the set of all non-target sequences T, an integer k, and a distance measure d, is there an approximate match i.e. a substring X of a sequence S in T such that $d(P, X) \leq k$, that is, the distance between pattern P and the substring X is at most k. If the target substring P has an approximate match in T, its complement is not a specific probe because the probe might hybridize to the corresponding non-target region. Approximate string matching has been studied extensively.

Finding specific probes for all genes of the genome G can be seen as a complement of finding all *approximate repeats*[3] in G [KCO+01]. We define an approximate repeat as a pair of substrings (X_1, X_2) of a text such that their distance is smaller than the given threshold k i.e. $d(X_1, X_2) \leq k$. Assume that the target substring should be of length m and the probe is specific if its distance d to all non-target sequences is $\geq k$. Then discarding all approximate repeats (X_1, X_2) of length m in G such that X_1 and X_2 are substrings of different sequences guarantees that every remaining substring defines a specific probe. The connection to repeat finding is relevant because a lot of algorithmic work has been done to find repeats from sequences [KOS+00].

An alternative way of formalizing the relatedness of two sequences is to measure their *similarity* instead of their distance [Gus97]. Then the basic concept is *alignment*, placing of two sequences one above the other and adding spaces so that each character is opposite a unique character or space in the other sequence. In addition, a *scoring scheme* is needed to attach a score s to any alignment, we assume the scheme to be such that high scores are attached to alignments that have a high number of matching characters. The similarity measures that have been used to predict probe specificity are *local similarity scores*. The maximum local similarity score of two sequences is the the maximum score of any alignment of their substrings. Accordingly, we define the *local similarity problem* as follows: Given the target substring P and the set of all non-target sequences T, a real value c, and a scoring scheme C, is there a pair of substrings (X, Y) such that of X is a substring of P and Y is a substring of a sequence S in T and $s(X, Y) \geq c$. If there is

[3] also called degenerate repeats

such a pair of substrings, the complement of P is not a specific probe.

The basic difference between the similarity- and distance-based approach is that the former allows defining the specificity using only a part of the probe when the latter always takes into account the whole probe. So, one chooses a local similarity score if one wants to approximate the hybridization of a probe and a non-target as the interaction between their most complementary parts ignoring the interactions outside this area.

A common assumption behind both the above approaches is that the specificity of a probe can be predicted by finding the non-target molecule that hybridizes most efficiently with the probe and approximating all cross-hybridization by this interaction. The measures that we call *aggregate scores* discard this assumption and try to approximate the effect of hybridization between the probe and all non-targets. Thus, an aggregate score is not defined between two sequences but between the target substring P and the set of all non-target sequences T.

Finally, the measures we call *energy scores* try to directly approximate some thermodynamic property such as free energy in hybridization or a melting temperature but with some simplification that speeds up the computation. Below we list the methods found in the literature according to the measure used. Many of the articles belong to several categories. Despite the wide range of different technical solutions, there are some recurring themes. One of them is reducing the comparison of two sequences to finding short common substrings. We call the substrings of length q q-grams [JU91]. The common q-grams can be used either as a basis for a comparison measure or as an algorithmic tool to efficiently compute some other measure.

Edit distance. The edit distance between two sequences is the minimum number of character changes, insertions, and deletions required to transform one sequence into the other. In order to separate the basic distance measure from a measure where operations have different costs, edit distance is also called *unit cost edit distance* or *Levenshtein distance*. The approximate string matching problem with d being the edit distance is called *string matching with k differences*.

Myers' bit-parallel dynamic programming algorithm solves string matching with k differences problem directly in time $O(nm/w)$ where n is the total length of text T, m is the length of the pattern P, and w is the size of the machine word [Mye99]. Still, it is too slow for checking all possible probe candidates on the genome level. Speeding up approximate string matching using filters and index structures has been discussed extensively in string matching literature. Usually the idea is that first a fast filter is applied to discard the text regions where no approximate match of the pat-

tern can exist. After that the remaining text is checked using for example Myers' algorithm. Often the filter is based on the observation that a small number of edit operations cannot destroy all substrings of the pattern and thus the threshold k can be used to compute what kind of substrings of the pattern must be preserved by any approximate match. We call such filters *q-gram filters*.

Hyyrö [Hyy01] tests how fast different q-gram filters and index structures presented in the literature can check the specificity of short oligonucleotide probes. He defines the specificity of a 25bp long oligonucleotide probe as edit distance being at least 5 to non-targets (i.e. he solves string matching with 4 differences). Since there are usually a large number of candidate probes that need to be checked, it would be useful if several patterns could be processed simultaneously. This has been studied recently by Fredriksson and Navarro [FN03]. The q-gram filters are very efficient for small distances but their time requirement grows rapidly after some threshold as the area that cannot be filtered out expands.

Kurz et al. [KOS$^+$00, KCO$^+$01] take the approximate repeat-finding approach. Their algorithm first identifies all exact repeats of length $> q$ where q is chosen so that every approximate repeat has to contain an exact repeat of that length. The exact repeats are then extended to see whether there is an approximate repeat. They define the specificity of a 20bp long PCR primer as an edit distance $k = 3$. We have found that the time requirement of their algorithm grows fast when the edit distance is made larger (data not shown).

Li and Stormo [LS01] use edit distance and Myers' bit-parallel algorithm as an intermediate step in their method. They first narrow down the number of candidates by taking those that have the best aggregate score (see below). These candidates are then checked with Myers' bit-parallel algorithm and finally an energy based (see below) score is computed for the remaining candidates.

Hamming distance. The Hamming distance between two sequences of equal length is the number of positions with mismatching characters. The approximate string matching problem with d being the Hamming distance is called *string matching with k mismatches*.

Hamming distance has been used to evaluate the specificity of short oligonucleotide probes [LWY02, ZCJL03]. Both methods are based on identical [LWY02] or almost identical [ZCJL03] (1 mismatch) q-grams of pattern P and a match X. Lipson et al. [LWY02] extend their method to take into account the relative concentrations of non-targets so that a larger distance is required if the non-target is expected to be present in high concentra-

tion in the sample. The algorithm of Zheng et al. [ZCJL03] can be seen as approximate repeat finding. In their test they define a 33bp long oligonucleotide probe to be specific if its Hamming distance to non-targets is at least 5.

Sung and Lee [SL03] use Hamming distance to evaluate specificity of oligonucleotide probes that can be as long as 50-70 bases. Their algorithm appears to be fast and it finds target substrings that have a large Hamming distance ($> 30\%$ of probe length) to all non-target sequences. On the other hand, one may ask whether Hamming distance can be a relevant measure for over 50 bases long probe since a single base deletion or insertion can lead to a large Hamming distance.

All the above methods that are based on distance measures are safe in the sense that they only return probes that are specific according to the chosen distance measure (some of the papers also consider faster variants that do not have this guarantee). Thus, one only needs to consider whether the distance measure is relevant and the algorithm fast enough with realistic parameter values.

Local similarity scores. Blast is a heuristic algorithm for finding high-scoring local alignments fast. A large number of systems either directly use Blast [AGM+90] local alignment scores or compute some other score from Blast local alignments as at least one part of specificity prediction [XLW+02, MSG02, NWK03, RZG03, WS03, TDS+03, HSPB04]. Blast should be configured carefully if one wants to use it for this purpose, especially if the probes are short.

In addition to long similar areas searched with Blast, Wang and Seed [WS03] emphasize avoiding common substrings (in tests over 15 bases long) between a probe and non-target sequences. They do not describe the algorithm in detail and do not give its worst case complexity.

There are two software systems for finding long PCR-based probes [XLW+02, TDS+03] that use local similarity scores. They both use the Primer3 [RS98] program for selecting primers. Xu et al. [XLW+02] first determine the specific areas from the profiled genes, then let the Primer3 [RS98] program select primers for those areas, and finally check the specificity of primers. They define the probe as specific if its global sequence identity with any non-target is lower than a threshold (75%) and it does not have a good local alignment with any non-target (determined by Blast [AGM+90] E-value). The initial list of non-target genes that are similar to the target gene is made with Blast [AGM+90]. These non-target sequences are then aligned with target sequences using the Smith-Waterman [SW81] full dynamic programming algorithm. Thareau et al. [TDS+03] use only Blast for

predicting specificity.

Aggregate scores. Li and Stormo [LS01] compute for each q-gram of the candidate target substring, how many times it occurs in the non-target sequences and the score is the number of these occurrences. They repeat the computation for different lengths q and for each q save the lowest scoring probe candidates. The candidates saved most often are ranked as most likely specific probes for a gene and are given as input for Myers' algorithm.

Rahmann's [Rah03b] score is based on the longest common substrings of the target substring and all non-target sequences. Rahmann computes for different lengths l the number of non-target sequences for which the length of the longest common substring with the target substring is l. The score is based on the number of longest common substrings and weighted by a factor that depends on the length of the common substring.

Chang and Peck [cCP03] mark all q-grams that appear in more than one gene sequence. The rest of the q-grams of a target gene are unique. They define two scores for a target substring based on its content of unique q-grams: one that measures their density and another that measures their aggregation. The probes are chosen so that they have high density and low aggregation of unique q-grams. Somewhat confusingly, they [cCP03] use Blast to test the validity of their specificity scores.

Energy scores. Rouillard et al. [RZG03] use Blast as the first filter and then compute the minimum energy structures formed by the probe and the most similar non-targets given by Blast using the program Mfold [Zuk03].

Li and Stormo [LS01] use heuristic energy computation as the last step of their three-step process (see above). They compute the best alignment of the probe and a similar non-target sequence and compute the melting temperature of duplex only based on the alignment and small modifications of it. The idea is that since the number of differences between the sequences is relatively small, the base pairing in the minimum energy structure is usually close to the best alignment.

Kaderali and Schliep [KS02] compute the melting temperature of the helix formed by the probe and each non-target. In order to speed up the computation, they ignore some complex substructures. Their dynamic programming algorithm is not guaranteed to find the optimal structure.

Krause et al. [KKM03] first identify non-target substrings that can be transformed into the target substring with at most few changes and a single insertion or deletion. Thus, the method is probably suitable only for short oligos. For the identified pairs of probes and non-targets the method computes a heuristic thermodynamic alignment that is similar to the one

used by Kaderali and Schliep [KS02].

Lipson et al. [LWY02] argue that cross-hybridization is most probably the result of hybridization initiated around a perfectly complementary area between the probe and non-target. Thus, they suggest defining the specificity of a probe based on the highest melting temperature of a common substring of the target substring and a non-target sequence.

Rahmann's second paper [Rah03a] takes quite a similar approach as Lipson et al. In addition to common substrings, he allows one mismatch. The specificity of the probe is then defined as the difference of hybridization free energy calculated with the target and minimum hybridization free energy calculated from these perfect or near-perfect matches to non-targets. He uses the method to find 19..21 bases long oligonucleotide probes.

Hornshøja et al. [HSPB04] first use Blast and then compute the melting temperatures of the probe and non-targets identified by the Blast hits. When computing the melting temperature they only consider mismatches but not insertions or deletions. Somewhat surprisingly they claim that before them, "No attempts utilize thermodynamics to estimate a melting temperature to non-targets for cross-hybridization prediction."

2.4 The selection of TRAC probes

When selecting candidate probes for TRAC, our aim is to select for each profiled gene a large number of candidate probes of different length that each fulfill all requirements of a good hybridization probe. Candidates of different length are needed because otherwise it is not possible to divide the probes into a small number of pools in the next step.

Since TRAC can be modified for several different applications, the method for selecting the candidate probes must also be flexible. Even though TRAC can in principle be used to profile all genes of the genome, its strength is the possibility to efficiently profile any subset of interesting genes. So, ideally the probe-selection method should scale from a few dozen to thousands of profiled genes.

The probes used in TRAC can either be short oligonucleotides which are usually directly synthesized in a factory or the probes can be amplified from a genomic sequence or cDNA library using PCR. How probes are produced affects the candidate selection process. If PCR is used, a probe must contain a good PCR primer pair at the ends so that it can be amplified from the genome. In addition, if the probes are amplified from the genomic sequence, the whole genome has to be taken into account in probe selection instead of just the gene sequences. The reason is that PCR primers may

amplify other regions from the genome than the intended target substring if their specificity is not checked.

Based on the experimental evidence presented in the literature (see Section 2.2), we define the specificity of a probe as a combination of two conditions. We say that the target substring P is specific if

1. there is no non-target sequence S in T such that P and S have a common substring longer than l and
2. P does not have an approximate match with at most k differences in T where $k = \lfloor \beta |P| + 1/2 \rfloor$.

In practice, we change the values of the parameters l and β slightly according to the application but for example with yeast we have used values $l = 15$ and $\beta = 0.2$ both for oligo and PCR probes.

Ideally, we would like to collect for each gene one candidate probe of every distinct length if one that fulfills all the requirements exists. In most applications the number of potential target substrings i.e. the number of probe length fragments of the profiled genes is so large that it is not possible to evaluate them all. Thus we have adopted the following strategy. We first discard the areas where there cannot be any specific probe by filtering out all exact repeats longer than l from the set of gene sequences G. Then we produce a large set of candidate probes from the remaining areas. Finally we check the specificity and sensitivity of the candidates so that the most computationally intensive checks are made last.

We chose the above strategy based on our specificity conditions. Filtering out all exact repeats in the beginning is worthwhile because it guarantees that the first condition is met and it can be done efficiently even for a large set of sequences. In principle, the same approach could be used to guarantee the second condition. However, we want to able to use fairly large edit distance which makes finding all approximate repeats difficult. Therefore, we check the second condition individually for each candidate probe. Myers' algorithm is a good general tool for that purpose because q-gram filters become inefficient as the allowed edit distance grows.

Next we describe the steps of the candidate probe selection in detail. A diagram of the process is shown in Figure 2.2.

1. **Filter out all exact repeats** Find all exact repeats longer than l from the gene sequences G and discard them. This step guarantees that all selected probes fulfill the first specificity requirement. The exact repeats are found using program REPuter[4] [KOS+00, KCO+01]

[4]The author would like to thank Dr. R. Giegerich for pointing out that REPuter can be used to find unique areas from a genome efficiently.

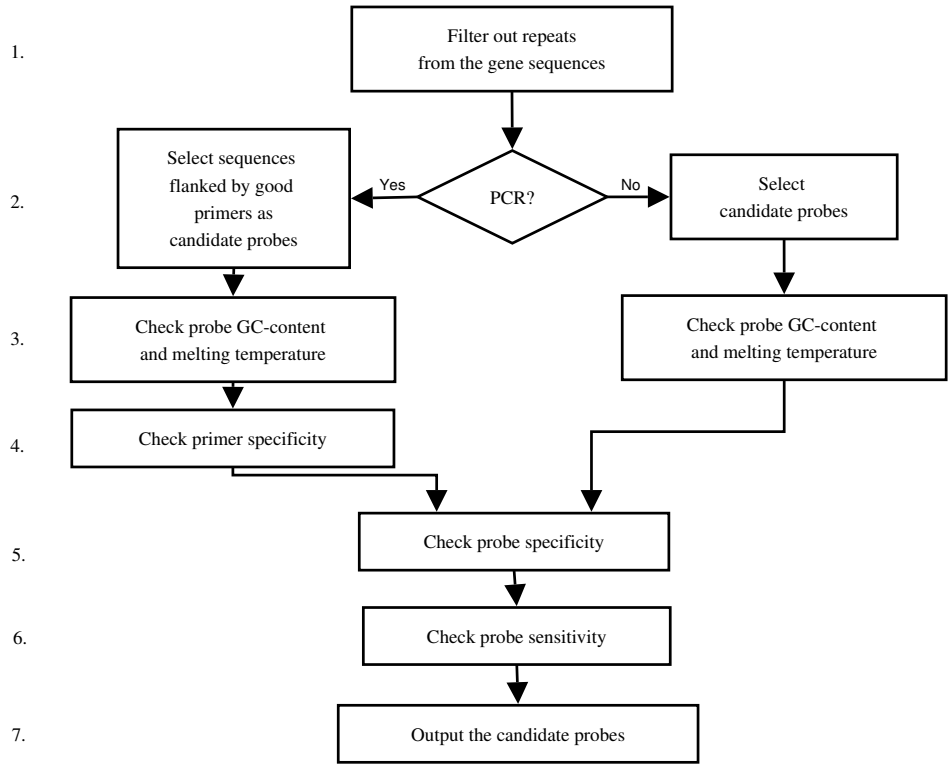

Figure 2.2: Candidate probe selection. The numbering on the left refers to steps described in the text.

that finds all exact repeats in linear time in the total size of the input sequences G and the maximal repeats in it (a repeat is maximal if it is not contained in any other repeat). It uses the suffix tree as its main data structure.

2. **Select the initial set of candidate probes** Select a set of candidate probes from each area not filtered out in the first step. When searching for PCR-based probes, run the Primer3 primer selection program [RS98] for each such area. For each primer pair suggested by Primer3 make the sequence fragment that contains the pair and the sequence between the pair as a candidate probe. When searching for oligo probes, simply select probe length fragments as candidate probes. The maximum number of initial candidate probes per gene can be limited to the user-specified number N to keep the running time of the program reasonable.

3. **Check the melting temperature and GC-content of the probes** Check that the melting temperature and GC-content of a probe are within the limits given by the user. The melting temperature is computed using the program Melting [Nov01]. The GC-content is the fraction G and C in the probe sequence. It can be used for example as a crude estimate of sensitivity for long PCR probes.

4. **Check the specificity of primers** This step is needed only if the probes will be produced by PCR from genomic sequence. For each candidate probe search alternative binding sites of its primers in the whole genome. If the primers have alternative binding sites for which the distance between the left primer and right primer is less than a threshold α, reject the candidate probe. The value of threshold α should be chosen so that PCR cannot amplify longer fragments than α. It can be estimated from the speed of the polymerase reaction and duration of the PCR cycle.

   Since PCR primers also work by hybridization, we define an alternative binding of a primer in the same way as we do for the whole probe. The substring X of a non-target sequence corresponds to an alternative binding site of the primer if the edit distance between X and the primer sequence P is $\leq \lfloor \beta |P| + 1/2 \rfloor$. The corresponding approximate string matching problem is solved using Myers' algorithm. For yeast we have used parameter values $\alpha = 5000$ and $\beta' = 0.2$.

5. **Check the specificity of probes** Check that a target substring fulfills the second specificity requirement. Solve the approximate string matching problem using Myers' bit-parallel algorithm [Mye99]. If n is the total size of the non-target sequences and w is the size of the machine word, Myers' algorithm takes time $O(nm/w)$ for each probe of length m.

6. **Check the sensitivity of probes** Check that the hybridization free energy change (computed using program Melting [Nov01]) and target free energy change (computed using Vienna RNA package [HFS+94]) are within user-given limits. The target free energy change is approximated as explained in Section 2.2. It takes time $O(M^3)$ for each target sequence of length M. This step is done last because the target energy calculations are typically computationally most expensive. Currently, this step is not usually performed for long PCR-based probes because there is not enough evidence that the predictions are accurate for long nucleic acids.

The probe free energy change is not calculated separately. However, if a probe has a stable secondary structure, its binding site in the target also has a stable secondary structure because the sequences are exact complements.

7. **Output the candidate probes**

The above procedure guarantees that all the selected probes fulfill all the requirements. The method is heuristic only in the sense that it may not find a probe for a gene even if there is one that fulfills all the requirements. This may happen if all possible initial candidate probes are not produced in step 2 because of the limit N. The user can choose whether all initial oligo candidate probes are produced or not. The number of potential PCR-based candidate probes is so large that it is not feasible to evaluate them all with our approach. Thus, the user has to give the upper limit N.

Most of the individual steps in our method appear also in one or several of the independently developed microarray probe selection methods listed in Section 2.3. Many of the programs such as Melting [Nov01] and Primer3 [RS98] are also used by others. However, our combination of different tools makes the method quite flexible as a whole:

- Filtering of all exact repeats in the beginning is an efficient way to discard substrings that are not specific. All our initial candidate probes already fulfill the first specificity condition. The candidate probes that fulfill the first specificity condition are more likely to fulfill the second one, as well. Thus, compared to a random selection, less initial candidate probes are needed to produce the same amount of good probes.

- The thresholds of the specificity conditions, the length of the common substring l and the number of differences k, can be varied without an explosion in the time and space requirement. The time and space requirement of the suffix tree algorithm for finding repeats depends linearly on the size of the input and output. They do not directly depend on the length of the repeats as they do in the hashing techniques that are widely used for this purpose [KCO+01]. The time requirement of Myers' algorithm is independent of the number of differences k. Thus, a large edit distance can be used which is not possible with q-gram algorithms.

- The procedure can be modified according to the task. All possible candidate probes can be evaluated if the aim is to select oligo probes

for a small set of genes. On the other hand, probes can be searched for all the genes of a genome by taking only a sample of possible candidate probes.

- The same overall process is used for selection of both PCR-based and oligo probes.

2.5 Conclusion

We are by no means able at this point to predict reliably the sensitivity of oligo-TRAC probes from sequence data. Nevertheless, the literature together with the experimental evidence suggests that we can increase the probability of selecting a sensitive probe by discarding the most questionable ones based on the predicted hybridization free energy change and the energy change of the target secondary structure. Even a modest decrease in the number of bad probes that can be achieved by computational means can save a lot of laboratory work.

The number of different measures and algorithms used for estimating probe specificity is already high. An expert on string matching algorithms could probably come up with quite a few new ones. The problem is that currently there seems to be no way to compare different methods in a disciplined manner.

Many of the current solutions are based on the assumption that thermodynamic calculations are the most accurate computational method to predict cross-hybridization but too slow for large-scale evaluation of probes. Thus, one step forward could be to take the pairs of probes and targets for which there is quantitative experimental data about cross-hybridization and compute their thermodynamic properties using the most accurate but slow algorithms.

If the slow energy calculations turn out to predict experimental data well, then one could set up a large artificial data set to assess the suitability of different faster measures and algorithms. The artificial data set would consist of a large set of pairs of probes and targets and hybridization values attached to each pair. The hybridization values would come from the energy calculations. Then one could evaluate the fast methods against the artificial data set. The best one would be the one that predicts most accurately the hybridization values of the probe-target pairs. This approach would also allow the optimization of the scoring schemes used in the methods that are based on sequence comparison.

Chapter 3

Assigning probes into a small number of pools

Dividing the hybridization probes into a small number of such pools that the probes in each pool can be separated using electrophoresis is a central computational challenge in TRAC. We will call this problem *probe assignment* and it is the focus of this chapter. First, we define the problem formally and put it in context by comparing it to similar problems in computational biology and scheduling theory. We show that probe assignment is NP-hard and derive an efficient optimal algorithm for a relevant special case and then extend the same idea to get a factor 2 approximation algorithm for the general problem. At the end of the chapter we relax one of the assumptions made in the formulation of the basic problem and derive an approximation algorithm for the modified problem.

3.1 Probe assignment problem

Let us assume that we have a set C of candidate probes available. We assume that the only properties of probes needed in the combinatorial formulation of the probe assignment problem are their lengths and the relation between the genes and the corresponding candidates. In Section 3.6 we will also take into account the hybridization temperature but till then we assume that all probes in C can be hybridized in the same temperature. We assume that each profiled gene has at least one candidate probe in C.

We call a *pool* a set of probes such that the probes in the pool can be differentiated by electrophoresis. This requires that the lengths of the probes in the same pool are different enough. To formalize this, we let $|p|$ denote the length of the sequence of a probe p. However, we do not

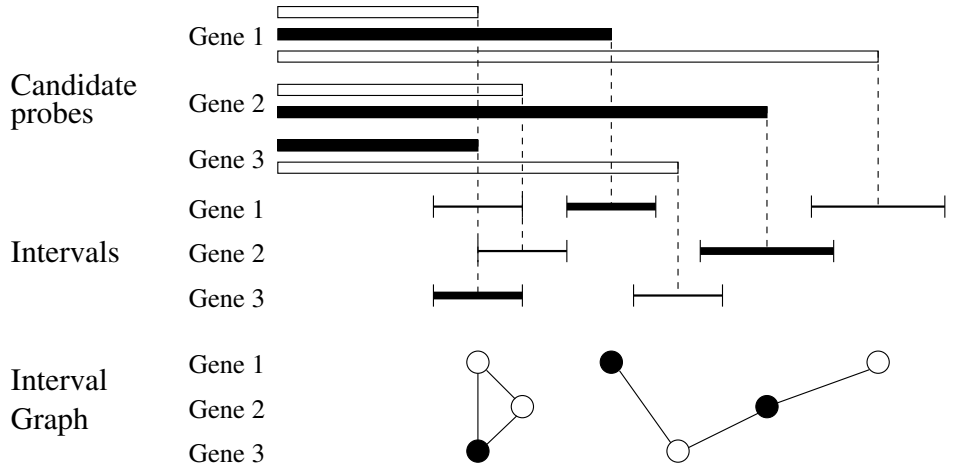

Figure 3.1: An instance of probe assignment. Three probes that can be put into the same pool are marked with black. In this case they provide a solution for $k = 1$.

require $|p|$ to be an integer. So, even though we will speak of probe length, the results also apply to any technique that can measure the size of a fragment of DNA. Moreover, attach to each p an open interval of reals $I(p) = (|p| - d(|p|)/2, |p| + d(|p|)/2)$ where $d(|p|) > 0$ is the required length difference for fragments of length p. Now, if p and q are two different elements of the same pool, then $I(p) \cap I(q)$ must be empty. See also the example in Figure 3.1.

We assume that the "resolution" function d has the following natural property. For any three probes p, q, and t such that $|p| \leq |q| \leq |t|$, the function d must be such that if p and t cannot be in the same pool, then neither can q and t, i.e. if $I(p) \cap I(t) \neq \emptyset$ then $I(q) \cap I(t) \neq \emptyset$.

In this chapter we will concentrate on the following computational problem.

Probe Assignment Problem (PA): Let C be a set of *candidate probes* for a set of n genes such that $C = C_1 \cup C_2 \cup \ldots \cup C_n$ where C_i is the set of candidates for gene i and $C_i \cap C_j = \emptyset$ when $i \neq j$. Moreover, let $k \leq n$ be an integer. The *PA problem* is to select a subset $S \subseteq C$ of size n and partition it into k disjoint sets $S_1, S_2, \ldots, S_k$ such that *i)* $|S \cap C_i| = 1$ for each i (i.e. every gene has exactly one probe in S); *ii)* each S_j is a pool (i.e., $I(p) \cap I(q) = \emptyset$ whenever $p, q \in S_j, p \neq q$).

We are actually interested in the corresponding optimization problem.

Probe Assignment Optimization Problem (PA Optimization): Find the smallest number of the pools k for which the PA problem has a solution.

Note that the trivial solution is to pick one probe from each C_i and put each probe into its own pool thus using n pools.

Next we describe some basic properties of the PA problem. We use two concepts from graph theory: *interval graphs* and *colorings*. Introducing these concepts is not absolutely necessary for our purposes (in fact, our algorithms do not explicitly construct an interval graph). However, these concepts link the PA problem to the vast amount of research done in graph theory. Unless shown otherwise we use Golumbic's book [Gol80] as a reference on graph theory.

An undirected graph $G = (V, E)$ is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals I of the real line such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection [Gol80]. Set I is called an *interval representation* of G. We use open intervals but the resulting graph class is the same regardless of whether open or closed intervals are used.

Graph G is called a *proper interval graph* if there is an interval representation for G such that no interval contains another. Graph G is called a *unit interval graph* if there is an interval representation I for G, called a unit interval representation, such that the intervals in I are of unit length. Every proper interval graph is also a unit interval graph and vice versa.

We formulated the PA problem using the biological concepts, gene and probes, but the results and algorithms in this chapter can be applied to any finite family consisting of disjoint sets of intervals as long as no interval properly contains another. For any proper interval graph, there is also a unit interval representation and such a presentation can be computed efficiently [CKN$^+$95]. A set of unit intervals can always be interpreted as a set of probes by setting $d \equiv 1$.

A k-coloring of graph G is a partition of its vertices into disjoint sets $V = X_1 \cup X_2 \cup \ldots \cup X_k$ such that each X_i is an independent set. When vertices of each X_i are "painted" with color i, adjacent vertices always receive different colors. In general, graph coloring is an NP-hard problem. However, interval graphs can be colored with a minimum number of colors in linear time in the number of intervals if the sorted list of interval endpoints is given (we will describe the algorithm later in this section). Another useful fact is that for interval graphs the size of the minimum coloring is the same as the size of the largest clique.

To come back to the PA problem, notice that it has two components: selecting n probes from the set of all candidates and partitioning them into k pools. Assume for a moment that we have already somehow selected the probes. Then the partition of intervals into a minimum number of pools can be computed efficiently by coloring the interval graph induced by the intervals of the probes. Given the coloring, we can construct the pools by assigning the probes whose vertices have the same color into one pool. Since two vertices could receive the same color only if the corresponding intervals did not intersect, the intervals of the probes in each pool are pairwise disjoint.

The number of colors needed to color the interval graph induced by the chosen probes equals the size of the largest clique of the graph. On the other hand, each clique of the graph corresponds to a set of intervals that intersect a common point on the real line [Gol80]. Thus, the hard part of PA is selecting n intervals of probes so that for any point x on the real line at most k intervals intersecting x are selected. This will be the focus of the next sections. However, it is worthwhile to make note of a special case of PA which avoids the hard part.

Theorem 3.1 *If each of n genes has exactly one candidate probe, the probes can be partitioned into a minimum number of pools in time $O(n \log n)$.*

Proof. The endpoints of the intervals of the probes can be sorted in time $O(n \log n)$. After that the partitioning into pools can be done in time $O(n)$ using interval graph coloring as discussed above and below. □

We briefly describe an optimal coloring procedure for interval graphs because we will make use of it more than once. It is a simple greedy algorithm and appears to have been discovered independently several times [GLL82]. We follow the exposition by Gupta et al. [GLL79] who call the problem channel assignment. The input of the algorithm is a sorted list of interval endpoints and it scans the list in the increasing order. If a left endpoint of an interval is encountered, the interval is colored with the first available color and the color is made unavailable. If a right endpoint is encountered, the color of the interval is made available again. The algorithm runs in linear time in the number of intervals n if the currently available colors are kept in a stack. We assume that the stack has been initialized so that in the beginning there are $\geq n$ available colors. The correctness of the algorithms follows from the fact that it never uses more colors than there are intersecting intervals at any given point.

We end the section by discussing another problem in computational biology that is similar to probe assignment, namely designing *multiplexed*

genotyping assays [Fuc97, YWR00, AMY03]. In genotyping the aim is to measure genetic variation between individuals. More specifically, the task is to determine for each site in a set of sites in the human genome which one of the variants known to occur in that site an individual possesses. The sites where enough variation is known to exist are called *markers* and the different variants of a marker are called *alleles*. An allele is typically the value of a single nucleotide (SNP marker) or the number of short repeats (microsatellite marker).

The genotyping technologies we are interested in have the common feature that some biochemical technique is used to link each allele of a marker to a different fragment of DNA. Then determining the allele present in a sample becomes equivalent to measuring the size of a fragment of DNA using for example electrophoresis [Fuc97] or mass spectrometry [AMY03]. Multiplexing means determining the alleles of several markers in one assay (or pool in our terminology), i.e. one has to be able to not only separate the fragments of different alleles of the same marker but also the fragments from different markers from each other.

The problem of minimizing the number of genotyping assays is the same problem as probe assignment except for one crucial difference. As in probe assignment, a size interval has to be reserved for each fragment to be detected. The difference is that for each site we have to be able to detect all alleles. Thus, for each marker, instead of one interval, a tuple of intervals that has an interval for each allele has to be selected and assigned to an assay. Probe assignment can thus be seen as a special case of this more general multiplexing problem. Only heuristic algorithms without approximation guarantees are known so far for minimizing the number of assays [AMY03] but for the probe assignment we derive an approximation algorithm in Section 3.4.

Both minimizing genotyping assays and probe assignment are similar to problems in scheduling theory on the abstract level. Tuples of intervals are considered by Bar-Yehuda et al. [BYHN$^+$02]. In fact, they give an algorithm that can be used in multiplexing to divide the markers into at most $2t$ times the optimal number of assays if each marker has exactly one possible tuple of intervals and each tuple has at most t intervals. Scheduling theory relevant for assessing the complexity of the probe assignment problem will be discussed in the next section.

3.2 NP-hardness of probe assignment

In this section we show that probe assignment is an NP-hard decision problem. So we do not have much hope of finding a polynomial algorithm that is always guaranteed to find an optimal solution for PA. We have to settle for approximate solutions. At the end of this section we will introduce NP-hard scheduling problems that are similar to PA. In fact, NP-hardness of PA could be derived from them. Nevertheless, we proceed to prove NP-hardness of PA by reducing the 3-dimensional matching problem to it and justify the decision later.

3-Dimensional Matching Problem (3DM): Set $T \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number of q elements. Does T contain a 3-dimensional matching, i.e. a subset $T' \subseteq T$ such that $|T'| = q$ and no two elements of T' agree in any coordinate?

In addition, we assume that no element occurs in more than three triples (bounded 3-dimensional matching where bound $B \equiv 3$). The problem remains NP-complete [GJ79]. We show here that PA is NP-hard even when there is only one pool and two probes can always be in the same pool if their length difference is at least two nucleotides. On the other hand, in the next section we will show that PA can be solved optimally for any number of pools if a length difference of one nucleotide is enough to separate two probes.

Theorem 3.2 *Probe assignment is NP-hard already when all probe lengths are integers, number of pools $k \equiv 1$, and resolution $d \equiv 2$.*

Proof. Construct a PA instance C from a bounded 3DM instance T as follows. See also Figure 3.2.

- For each triple $(w_i, x_j, y_k) \in T$, where $1 \leq i, j, k \leq q$, add a candidate probe w_{ijk}. If x_j occurs in at least two triples, add a probe x'_{ijk} and if x_j occurs in three triples another probe x''_{ijk}. Similarly, if y_k occurs in at least two triples, add a probe y'_{ijk} and if y_k occurs in three triples, a probe y''_{ijk}. Set the lengths of the probes so that the interval $I(w_{ijk})$ given by the resolution function $d \equiv 2$ intersects all other intervals from this triple. In addition, set the lengths so that $I(x'_{ijk})$ intersects $I(x''_{ijk})$ and $I(y'_{ijk})$ intersects $I(y''_{ijk})$ but neither $I(x'_{ijk})$ nor $I(x''_{ijk})$ intersects $I(y'_{ijk})$ or $I(y''_{ijk})$. There should no intersections of intervals except the ones listed above. An example of lengths that satisfy these conditions is $I(w_{ijk}) = (c_{ijk} - 1, c_{ijk} + 1)$, $I(x'_{ijk}) = I(x''_{ijk}) = (c_{ijk} - 2, c_{ijk})$, and $I(y'_{ijk}) = I(y''_{ijk}) = (c_{ijk}, c_{ijk} + 2)$ where $c_{ijk} = 4iq^2 + 4jq + 4k$.

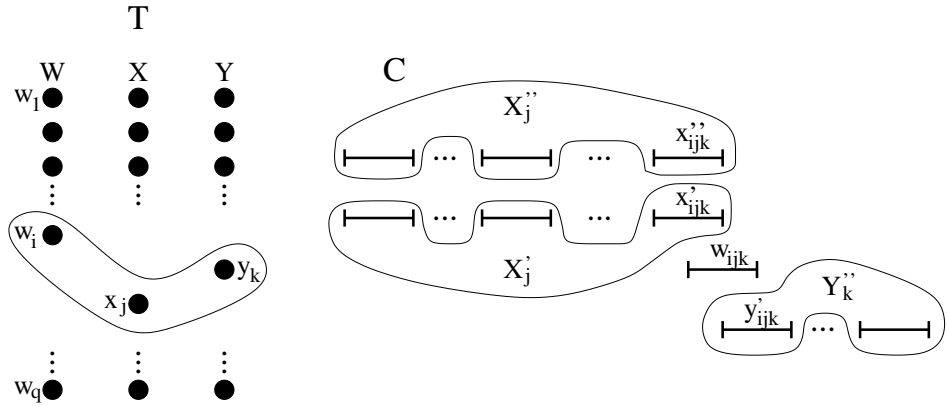

Figure 3.2: The construction of a PA instance from a 3DM instance when x_j occurs in three triples and y_k in two triples.

- Partition the candidate probes into "genes" according to the corresponding element in T. For each element $w_p \in W$, add a gene $W'_p = \{w_{ijk} \mid i = p\}$. For each element $x_p \in X$ that occurs in at least two triples in T, add a gene $X'_p = \{x'_{ijk} \mid j = p\}$ and for each element x_p that occurs in three triples, add $X''_p = \{x''_{ijk} \mid j = p\}$. Similarly, for each element $y_p \in Y$ that occurs in at least two triples in T, add a gene $Y'_p = \{y'_{ijk} \mid k = p\}$ and for each element that occurs in three triples add $Y''_p = \{y''_{ijk} \mid k = p\}$.

The idea of the reduction is that we need to choose exactly one probe from each q genes of the type W'_i. We add the corresponding triples (w_i, x_j, y_k) to the 3DM solution T'. The other genes in C are designed so that choosing exactly one probe from each ensures that x_j or y_k cannot occur in any other triple in T' thus making T' a 3-dimensional matching. Next we show formally that the 3DM instance T has a 3-dimensional matching if and only if the PA instance C constructed above has a solution with one pool.

First, let $T' = \{A_1, A_2, \ldots, A_q\}$ be a 3-dimensional matching in T. For each triple $A_{ijk} = (w_i, x_j, y_k) \in T'$, choose the corresponding probe w_{ijk} to the single pool S. Thus, we get exactly one probe for each gene W'_i to S. To show that it is possible to choose probes for all other genes as well, assume that the gene X''_j exists. There are three candidate probes in X''_j, and two of them are such that their interval can be intersected by an interval in S: one by $I(w_{ijk})$ and another by an interval of a probe in X'_j. The interval of the third one cannot be intersected by an interval of any

probe in S because x_j can occur only in one triple in T'. Thus, we can add to S a probe from X''_j and in the same way a probe for all other genes in C.

There is exactly one probe in S from each of the genes produced by the construction and no two intervals of probes in S intersect. Thus, S is a PA solution.

Conversely, suppose there is a one-pool solution S for the PA instance C. For every probe of type $w_{ijk} \in S$, add the triple $A_{ijk} = (w_i, x_j, y_k)$ to the 3-dimensional matching T'. Since there are exactly q such probes, we only need to prove that no element x or y occurs twice in T'. For the sake of contradiction, assume that x_j occurs twice in T' and in three triples in T (the case in which x_j occurs in two triples in T can be handled in an analogous manner). Then there would be two intervals of probes in S, denote them $I(w_{ijk})$ and $I(w_{rjs})$, such that each intersects an interval of one probe in X'_j and an interval of one probe in X''_j. The intervals of the remaining two probes in $X'_j \cup X''_j$, one in X'_j and one in X''_j, intersect. For this reason, there cannot be a probe from both X'_j and X''_j in S which is a contradiction. Thus, no x_j occurs twice in T'. One can show in the same way that no y_k occurs twice in T'.

There are q triples in T' and no element occurs twice. Thus, T' is a 3-dimensional matching. □

Results similar to Theorem 3.2 could be derived from scheduling theory[1]. Erlebach and Spieksma [ES03] extensively discuss scheduling problems that are similar to probe assignment. We compare the following scheduling problem to PA.

Job Interval Selection Problem (JISP): Consider an input that consists of n jobs, each of which is given by a set of intervals on a real line, and a number m of available machines. A feasible solution is a subset of the given intervals such that *i)* at most one interval is selected from each job, and *ii)* for any point p on the real line, at most m intervals overlapping x are selected. The goal is to find a feasible solution that maximizes the number of selected intervals.

JISP is an optimization problem but we can also consider a decision version (let us call it JISP decision) in which we ask whether all n jobs can be scheduled to m machines. It is easy to see the similarity between JISP and PA: PA deals with *genes* instead of *jobs* and *pools* instead of *machines* but the combinatorial structures of the problems are identical. In fact, PA could be easily solved using an algorithm that can solve JISP

[1]The author would like to thank Dr. M. Halldórsson for pointing out the relevant references from the vast scheduling literature.

decision. However, the opposite is not necessarily true, since not all sets of intervals correspond to legitimate sets of probes. So in order to show the NP-hardness of PA we would have to use a restricted case of JISP decision such that it is still NP-hard and any instance of it can be mapped to an instance of PA.

Keil [Kei92] proves that JISP decision is NP-complete for one machine and intervals of constant length (improving results by Nakajima and Hakimi [NH82]). However, their proof uses intervals of length three and we want to have a proof for intervals of length two in order to avoid a gap between the polynomially solvable case and the hard case in terms of resolution. Spieksma [Spi98] proves that JISP is a MAX-SNP-hard optimization problem by giving an approximation preserving reduction from MAX-3SAT-B to JISP. The result is valid also for one machine and intervals of length two with integer endpoints. Thus, the reduction by Spieksma could probably be used to prove Theorem 3.2 but the reduction is rather complicated. Therefore we preferred to provide another reduction.

JISP decision is solvable in polynomial time in the special case where there is a single machine and each job has at most two intervals [Kei92, Spi98]. Thus, the PA problem can be solved efficiently when there is only one pool and there are at most two probe candidates for each gene.

To our knowledge, all variants of JISP discussed in the literature are maximization problems. The goal is to maximize the amount of work that can be done with a fixed amount of resources. In contrast, probe assignment is a minimization problem. Every gene needs to be profiled but using as small amount of resources as possible. Thus, scheduling theory does not directly provide approximability results, either positive or negative, for the PA problem. We do not have any inapproximability results either but we will give an approximation algorithm later in this chapter, though we will start by solving an easier special case.

3.3 A polynomial algorithm for a special case of probe assignment

Separating two probes whose lengths differ only by one nucleotide is possible in TRAC for example when capillary electrophoresis is used to measure oligonucleotide probes [RKS+04]. When resolution $d \equiv 1$, the candidate probe set C becomes such that for any two probes $p, q \in C$, either $|p| = |q|$ or $I(p) \cap I(q) = \emptyset$. Even if we do not have a resolution of one nucleotide, we will in some cases deliberately construct the candidate probe set in such a way that the candidates have this nice property (see Section 5.2). We give

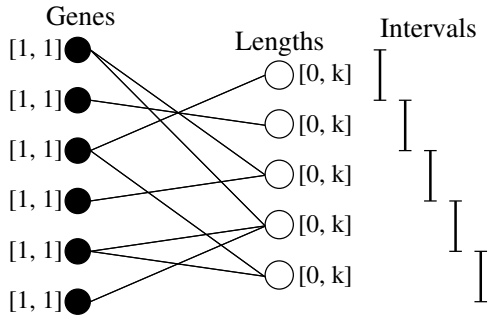

Figure 3.3: Constructing graph $G_{C,k}$. The degree bounds for each vertex are given in brackets.

a polynomial time algorithm for solving PA in such a case by reducing to the following well-known subgraph construction problem [Gab83].

Degree-Constrained Subgraph Construction Problem (DCS): Given a graph $G = (V, E)$ with integer bounds a_i and b_i associated with each vertex $i \in V$, find a subgraph H with the largest possible number of edges, such that each vertex has degree r_i in H with $a_i \leq r_i \leq b_i$.

DCS can be solved in polynomial-time.

Proposition 3.1 (Gabow [Gab83]) *There is an algorithm for DCS that runs in time* $O(\sqrt{\sum_{i \in V} b_i}|E|)$.

DCS is a generalization of matching: Ordinary matching is a DCS problem where $a_i = b_i = 1$ for all i in V. In fact, the following algorithm is based on the observation that the PA problem reduces to ordinary matching in a bipartite graph when $d \equiv 1$ and $k \equiv 1$. Reducing to DCS instead of matching extends the basic idea also into the case where $k > 1$.

For a given PA instance $(C = C_1 \cup C_2 \cup \ldots \cup C_n, k)$, construct a DCS instance $G_{C,k}$ as follows (see also Figure 3.3). Graph $G_{C,k} = (V_1 \cup V_2, E)$ is a bipartite graph where $V_1 = \{1, 2, \ldots, n\}$ (the vertices that correspond to the genes) and $V_2 = \{|p| \mid p \in C\}$ (the vertices that correspond to the lengths of the probes). Moreover, there is an edge $e = (i, l) \in E$ if C_i contains a probe p of length l. Then we say that probe p *supports* edge e. Finally, for each $i \in V_1$ we set the bounds as $a_i = b_i = 1$ and for each $l \in V_2$ as $a_i = 0$, $b_i = k$.

A restricted case of PA optimization can be solved exactly by means of a solution of DCS as follows.

Lemma 3.1 *If a PA instance $(C = C_1 \cup C_2 \cup \ldots \cup C_n, k)$ is such that for any $p, q \in C$ either $|p| = |q|$ or $I(p) \cap I(q) = \emptyset$, then it has a solution if and only if the corresponding DCS instance $G_{C,k}$ has a solution that has n edges.*

Proof. Let first $S = S_1 \cup S_2 \cup \ldots \cup S_k$ be a solution of (C, k). Then the subgraph $H = (V_1 \cup V_2, E_S)$ of $G_{C,k}$ where E_S consists of all edges e such that the underlying probe of e is in S is a solution of the DCS instance: E_S is maximal because $|E_S| = n$ and because from the bipartiteness of $G_{C,k}$ and from the constraints for vertices in V_1 it follows that a DCS solution can not be larger than n edges. Set E_S also satisfies the constraints at the vertices of $G_{C,k}$: there is exactly one probe in S for each gene, hence the constraints of V_1 are satisfied; there can be at most k probes of equal length, namely one in each S_i, and hence the constraints of V_2 are satisfied.

Conversely, a solution $H = (V_1 \cup V_2, E)$ of the DCS instance where $|E| = n$, gives a solution S of the PA problem containing exactly one supporting probe for each edge in E. Then at most k elements are of equal length. This means that k pools suffice as $I(p) \cap I(q) = \emptyset$ whenever $|p| \neq |q|$. □

The algorithm for solving the PA optimization is now easily obtained by using Gabow's algorithm for the DCS as a subroutine in a binary search for the smallest k. More precisely, perform binary search over integers $1, \ldots, n$ to find the smallest k such that Gabow's algorithm finds for the DCS instance $G_{C,k}$ a solution of size n. Let us call this procedure Algorithm 1.

Theorem 3.3 *Algorithm 1 correctly solves the PA optimization problem for any instance (C, k) such that $I(p) \cap I(q) = \emptyset$ whenever $|p| \neq |q|$ for all $p, q \in C$. Its running time is $O((mn)^{3/2} \log n)$ where m is the number of different lengths of probes in C, and n is the number of genes.*

Proof. The correctness should be clear by the construction and Lemma 1.

As regards the running time, we assume that the lengths of the probes in C are given as input. Then it is straightforward to construct $G_{C,k}$ in time $O(mn)$ since the number of probes in C is bounded by mn. Moreover, $G_{C,k}$ needs to be constructed only once during the search as the new $G_{C,k}$ for a different k can be obtained just by updating the bounds $b_i = k$ for vertices $i \in V_2$. Hence, $O(mn)$ covers the time for constructing the $G_{C,k}$s. Gabow's algorithm is performed $O(\log n)$ times, each taking time $\sqrt{\sum b_i}mn$ by Proposition 1. As $\sum_{i \in V_1 \cup V_2} b_i \leq n + mn$, we have $\sqrt{\sum b_i}mn = O((mn)^{3/2})$ which completes the proof. □

3.4 An approximation algorithm for probe assignment

At least when selecting PCR based probes the length difference of two probes in the same pool has to be more than one nucleotide. Then instead of selecting at most k probes of equal length, one is only allowed to select at most k probes from each group of probes whose intervals intersect. Thus, in our reduction of PA to DCS given in the previous section we must pose additional constraints of interval graph nature on the DCS.

We will derive a polynomial-time approximation algorithm for the PA optimization which has a performance ratio of 2. Hence, the algorithm produces a valid assignment into pools that is guaranteed to use at most twice the optimum number of pools.

To generalize the construction of the DCS instance $G_{C,k}$ (see also Figure 3.4), again let $G_{C,k} = (V_1 \cup V_2, E)$ where $V_1 = \{1, 2, \ldots, n\}$ (the genes) but V_2 corresponds to probe groups constructed as follows[2]. First sort the probes p in C into increasing order according to the lengths of the probes $|p|$. Let p_1 be the first probe in this order. Then let g_1 be the set of probes q such that $I(p_1) \cap I(q) \neq \emptyset$. Then g_1 must contain p_1 and possibly some other probes after p_1 in the order. Let p_2 be the first probe not in g_1. Then repeat the above construction to get set g_2 and so on, until the whole ordered set has been assigned to some g_i, $i = 1, \ldots, s$. Now let $V_2 = \{g_1, \ldots, g_s\}$. Finally, there is an edge $e = (i, l) \in E$ if $C_i \cap g_l$ is non-empty. All probes in $C_i \cap g_l$ are said to *support* the edge (i, l). For each $i \in V_1$ the bounds are again $a_i = b_i = 1$ and for each $g \in V_2$, $a_i = 0$ and $b_i = k$.

The approximation algorithm first performs Algorithm 1, that is, finds by binary search the smallest $k = k_{min}$ such that the DCS instance $G_{C,k_{min}}$, as just constructed, has a solution of size n. The next step is to construct a solution of our PA problem from this solution of the DCS. Let S be a probe set constructed by selecting into S exactly one supporting probe for each edge of the DCS solution. The algorithm divides S into a minimum number of pools by coloring the interval graph induced by the intervals of the probes in S as shown in Section 3.1.

Let us call this entire method Algorithm 2.

Theorem 3.4 *Algorithm 2 finds a solution for the PA optimization problem with at most twice as many pools as in the optimal solution. Its run-*

[2]As pointed out by Dr. V. Mäkinen, the same grouping of fragments is used by Li in his proof of the approximation ratio of his shortest common superstring algorithm [Li90]. The proof is also given in [Vaz01, pp. 19–22]

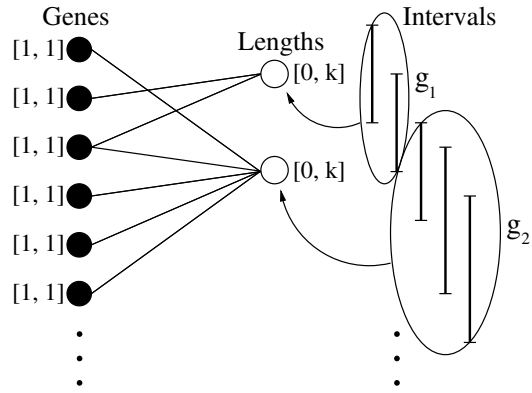

Figure 3.4: Constructing graph $G_{C,k}$ in the general case.

ning time is $O((mn)^{3/2} \log n)$ *where* m *is the number of different lengths of probes in* C*, and* n *is the number of genes.*

Proof. Algorithm 2 produces a feasible solution of PA as it selects a probe for every gene and partitions the selected probes into pools according to a coloring of the corresponding interval graph. In terms of run time, there are two new features as compared to Algorithm 1. First, the two sorting operations can be done in time $O(mn \log mn)$. Second, the construction of the pools at the end can be done in time $O(n)$ after the sorting. Noting also that the present DCS instance is not larger than in the case of Algorithm 1, the time bound of the theorem follows.

The performance ratio of 2 follows from Lemma 2 and Lemma 3 below. □

Lemma 3.2 *Let* k_{min} *be the minimum value of* k *found by Algorithm 2 and let* κ *be the number of non-empty pools* S_i *constructed by Algorithm 2. Then* $\kappa \leq 2k_{min}$*.*

Proof. The procedure for pooling S into S_i's in Algorithm 2 finds a minimum coloring of the interval graph I_S induced by the intervals of the probes in S. For interval graphs the size of the minimum coloring is the same as that of the largest clique [Gol80].

On the other hand, the intervals $I(p)$ and $I(q)$ for some p, q in S can intersect only if $p, q \in g_i \cup g_{i+1}$ for some i. In order to prove that, assume for contradiction that an interval of a probe $p \in g_i$ intersects an interval of a probe $t \in g_{i+2}$. Algorithm 2 constructs the groups in such a way that the interval of the shortest probe of g_{i+1}, let it be q, and the shortest probe of

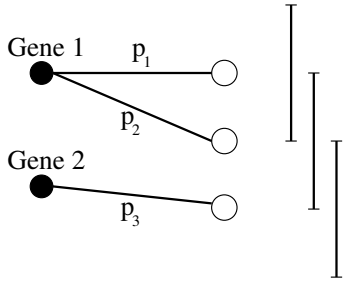

Figure 3.5: An example of using twice the optimal number of pools.

g_{i+2} do not intersect. Thus, $p \leq q \leq t$ and $I(p) \cap I(t) \neq \emptyset$ but $I(q) \cap I(t) = \emptyset$ which contradicts the assumption we made about the intervals of probes in Section 3.1. This means that all cliques of I_s must be of size $\leq 2k_{min}$ because $|g_i \cap S| \leq k_{min}$ for any i. □

Lemma 3.3 *Let κ^* be the optimum (minimum) number of pools. Then $k_{min} \leq \kappa^*$.*

Proof. Let S^* be the set of probes in an optimal solution with κ^* pools. Then the subgraph of G_{C,κ^*} whose edges have supporting probes S^* is a solution for the DCS instance G_{C,κ^*}. Hence, $k_{min} \leq \kappa^*$. □

The performance ratio of 2 is tight. Consider the PA instance shown in Figure 3.5. Clearly, $k_{min} = \kappa^* = 1$ but Algorithm 2 puts p_1 and p_2 in set g_1 and p_3 in set g_2. Algorithm 2 can choose either p_1 or p_2 because they support the same edge in graph $G_{C,k}$. If Algorithm 2 chooses p_2, it uses two pools which is twice the optimal.

The crucial point in the proof of Lemma 3.2 is that only the intervals of the candidate probes in the same or in successive groups can intersect. This was achieved by constructing the groups so that the intervals of the shortest probes of any two successive groups g_i and g_{i+1} do not intersect. Actually, it suffices to construct the groups so that the interval of the longest probe of group g_{i-1} and the shortest probe of g_{i+1} do not intersect. Thus in practice, we construct graph $G_{C,k}$ in both ways and solve DCS in both cases and choose the smaller number of pools and the higher lower bound given by k_{min} (see Lemma 3.3) from the two results. The lower bound for the number of pools allows us to give for any PA instance a nontrivial upper bound $\kappa - k_{min}$ for the number of pools Algorithm 2 wastes compared to an optimal solution.

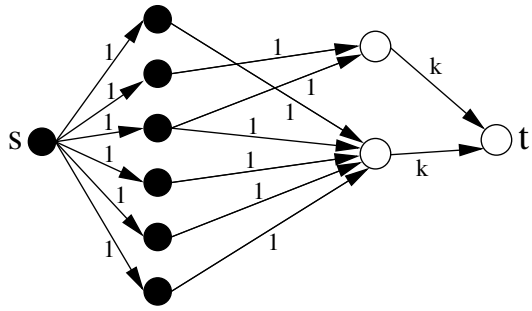

Figure 3.6: A flow network that corresponds to the graph $G_{C,k}$ in Figure 3.4.

3.5 The implementation of the approximation algorithm

Since the approximation algorithm for probe assignment given in the previous section has a central role in the software system for TRAC probe selection and pooling, we here discuss two issues in the implementation of the algorithm. First, we show how we solve the DCS problem using network flow and second a post-processing heuristic that in some cases saves a valuable pool.

We have chosen to solve the DCS problem in Algorithm 2 by reduction to the network flow problem, instead of using the direct solution given by Gabow's algorithm. The reduction to the network flow is simple and fast (see Chapter 5), and good implementations of network flow algorithms are readily available.

In Algorithm 2, the DCS instance $G_{C,k}$ is bipartite. In such a case the reduction to the network flow problem can be done in the same manner as for ordinary matching [CLR90]. We define the corresponding flow network $G'_{C,k}$ as follows (see also Figure 3.6). We add source s and sink t in the node set $V_1 \cup V_2$ and define the directed edges of $G'_{C,k}$ as follows $E' = \{(s,u) \mid u \in V_1\} \cup \{(u,v) \mid u \in V_1, v \in V_2, (u,v) \in E\} \cup \{(v,t) \mid v \in V_2\}$. We assign capacity k to each edge (v,t), $v \in V_2$, and unit capacity to all other edges.

Theorem 3.5 *The DCS instance $G_{C,k} = (V_1 \cup V_2, E)$ has a solution $H = (V_1 \cup V_2, E_H)$ that has n edges if and only if the corresponding flow network $G'_{C,k}$ has an integer-valued maximum flow with value n.*

Proof. Let first f be an integer-valued maximum flow with value n in $G'_{C,k}$. Construct $E_H = \{(u,v) | u \in V_1, v \in V_2, f(u,v) > 0\}$. Since there

is at most 1 unit of flow entering each node $u \in V_1$ and at most k units leaving each node $v \in V_2$ and the flow is an integral-flow, all upper bounds of nodes in H are satisfied. Since from the source s there are no other edges than one edge of unit capacity to each $u \in V_1$, all these edges have to be saturated. Thus, all lower bounds of nodes in H are also satisfied. Since there is 1 unit of flow leaving each node $u \in V_1$ and there are no edges from V_1 to the sink t, $|E_H|$ is equal to the value of the flow n.

To prove the converse, given a solution H we only need to augment the flow for each $(u, v) \in E_H$ along the path $s \to u \to v \to t$ and observe that the net flow across the cut between $V_1 \cup \{s\}$ and $V_2 \cup \{t\}$ is $|E_H| = n$.

One can prove that the maximum flow found by a specific maximum flow algorithm is an integer-valued flow in the same way as in the case of maximum matching in bipartite graphs. □

We have used a push-relabel network flow algorithm implementation by Cherkassky and Goldberg [CG97], the asymptotic running time of which is $O(|V|^2\sqrt{|E|})$. The total running time of this version of Algorithm 2 hence becomes $O((m+n)^2\sqrt{mn}\log n)$.

When an experiment is set to profile a small number of genes such as a few dozen, even one additional pool is a significant expense. In practice we have noticed that when the goal is to put the probes into one or two pools, the approximation algorithm in some cases uses one extra pool compared to what seems possible. Thus, we have designed a post-processing heuristic that tries to improve the solution of the approximation algorithm by doing local search.

The basic idea of the heuristic is to take the set of n probes chosen by the approximation algorithm, find the largest set of probes whose intervals intersect i.e. a maximum clique of the interval graph induced by the intervals of the probes, and try to swap each probe in the set to another candidate of the corresponding gene. Finding a maximum clique and swapping the corresponding probes is iterated until no useful swap can be found. A maximum clique can be easily found by modifying the technique for coloring an interval graph (presented in Section 3.1).

Let ω denote the size of the maximum clique of the interval graph I_S induced by the intervals of the currently selected probes S at any step of the algorithm. In the beginning S is the solution of the approximation algorithm. The algorithm tries to decrease the size of a maximum clique by trying to swap each probe p in the clique for another candidate q of the same gene.

First the heuristic swaps the probe p for another candidate q if the interval of the new probe $I(q)$ intersects fewer intervals of probes in $S \setminus p$

than the old one $I(p)$. In other words, swapping decreases the size of a maximum clique of I_S to $\omega - 1$ without creating another of the size ω. After updating the set S, the algorithm finds a maximum clique of the new interval graph I_S and tries again to find a swap. When no such *single swap* is possible any longer, the heuristic swaps if the swap creates a new clique of size ω and there is a single swap for this new maximum clique. After each such *double swap*, the heuristic checks again whether a single swap can be done in the new set of probes and continues searching for swaps until neither a single swap nor a double swap is possible.

The heuristic has polynomial worst-case running time. The number of swaps is bounded by $O(n^2)$ because every swap decreases the size of a maximum clique by one. Thus n swaps must decrease ω by at least one and ω is at most n in the beginning. A single swap takes time $O(n|C|)$ where $|C|$ is the size of the candidate probe set because a maximum clique can be found in $O(n)$ time but when searching a pair of probes for the clique that can be swapped, the number of intersecting intervals has to be computed for $O(|C|)$ candidate probes. A double swap tries to do a single swap for at most $|C| - n$ probes and thus takes time $O(n|C|^2)$. Remembering that $|C| < mn$ gives the total worst-case running time of the heuristic $O(m^2n^5)$. In practice, when the solution of the approximation algorithm is used as starting point, the heuristic takes negligible time compared to the other steps in the experiment design.

As shown in test results the greedy heuristic has helped to save one crucial pool when designing a small experiment with PCR probes. However, it has not improved the solution of the approximation algorithm when the task has been to pool probes for all genes in a genome. Also, we have another trick for oligo probes but since that also involves the selection of candidate probes, we will discuss that later in Section 5.2.

3.6 Partitioning into pools according to size and hybridization temperature

So far in this chapter we have assumed that the probe size is the only feature determining which probes can be in the same pool. The assumption is valid if the candidate probe set has been constructed so that all the probe candidates can be hybridized in the same temperature. However, in some cases it is sensible to use existing probes that others have already been using successfully. For example, when profiling bacterial populations with TRAC one can use known bacterial species-specific hybridization probes from databases. In this case, the suitable hybridization temperatures of

the probes may vary so much that it has to be taken into account when dividing the probes into pools. On the other hand, there is only one probe for each gene, thus selecting the probe is not a problem.

The input of the computational problem is a set of n probes and a length interval $I_l(p)$ for each probe p as defined in Section 3.1. In addition, every probe has an acceptable hybridization temperature interval $I_T(p)$. We assume that the temperature intervals are such that no interval properly contains another. Two probes, p and q can be in the same pool if their sizes are separable i.e. $I_l(p) \cap I_l(q) = \emptyset$ and there is a temperature interval in which both can be hybridized i.e. $I_T(p) \cap I_T(q) \neq \emptyset$. If all pairs of probes in a pool satisfy the pairwise condition on the temperature intervals, there is a temperature point that is acceptable for all probes in the pool. As before, the goal is to partition the probes into a minimal number of pools.

2D Probe Partition (2D PP): Let C be a set n probes. Partition the probes into a minimal number of disjoint sets $S_1, S_2, \ldots, S_k$ and attach a real number T_j to each set S_j so that *i)* each subset is a pool i.e. $I_l(p) \cap I_l(q) = \emptyset$ whenever $p, q \in S_j, p \neq q$ and *ii)* T_j is an acceptable hybridization temperature for all the probes in the pool i.e. $T_j \in I_T(p)$ for all $p \in S_j$.

The problem has an intuitive geometric interpretation shown in Figure 3.7. We borrow the idea from closely related t-union graphs discussed by Bar-Yehuda et al. [BYHN$^+$02]. Think that each probe p corresponds to a box such that its projection on the x-axis is $I_l(p)$ and its projection on the y-axis is $I_T(p)$. Now, two probes p and q can be in the same pool if the projections of their boxes intersect on the y-axis but do not intersect on the x-axis.

Note that 2D PP can be solved efficiently if only length or temperature is considered. The case of only length intervals was shown in Theorem 3.1. If only temperature intervals are considered, the task is equivalent to finding a *minimum clique cover* of the interval graph induced by the intervals $I_T(p)$ (in fact Golumbic has this as an example in his book [Gol80, page 182]). A clique cover of graph $G = (V, E)$ of size k is a partition of vertices V into k disjoint subsets such that each subset is a clique. A minimum clique cover of an interval graph can be computed in time $O(n \log n)$ given an interval representation of the graph with n intervals [GLL82].

We derive a simple greedy algorithm for 2D PP that has an approximation ratio 2. It is based on the same techniques we have used in the previous sections. First, divide the probes into groups as was done in the approximation algorithm for PA (Algorithm 2) but do it based on temperature intervals. Sort the probes p in C into increasing order according to the left endpoints of the temperature intervals $I_T(p)$. Let p_1 be the first probe in

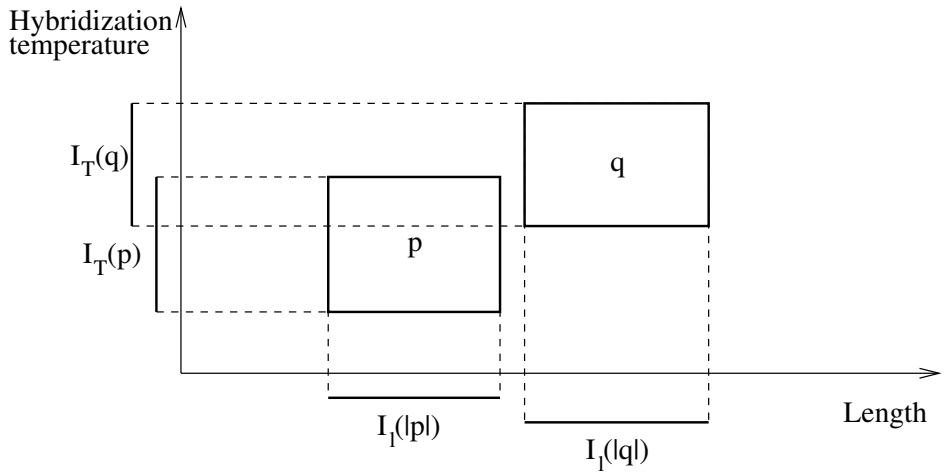

Figure 3.7: The geometric interpretation of 2D probe partition. In this case, probes p and q can be assigned into the same pool.

this order. Then let g_1 be the set of probes q such that $I_T(p_1) \cap I_T(q) \neq \emptyset$. Then repeat the construction to get set g_2 and so on, until the whole ordered set has been assigned to some g_i, $i = 1, \ldots, s$. Second, for each group g_i, partition the probes p in g_i into a minimal number of pools according to the length intervals $I_l(p)$ using interval graph coloring. Finally, attach to every pool constructed from the group g_i a hybridization temperature from interval $I_T(p_1) \cap I_T(p_e)$ where p_1 is the first and p_e the last probe of the group. Take the resulting pools and hybridization temperatures as the solution of 2D PP instance. Let us call this procedure Algorithm 3.

Theorem 3.6 *Algorithm 3 finds a solution for the 2D PP problem with at most twice as many pools as in the optimal solution. The algorithm runs in time* $O(n \log n)$.

Proof. Sorting the endpoints of the temperature intervals and constructing the groups can be done in time $O(n \log n)$. Partitioning a group g_i into pools takes time $O(|g_i| \log |g_i|)$ (Theorem 3.1). Since the groups are disjoint, the total time of the partitioning is bounded by $O(n \log n)$.

The correctness of the algorithm follows from Lemma 3.4 and the approximation ratio from Lemma 3.5 below. □

Lemma 3.4 *Algorithm 3 produces a feasible solution for the 2D PP problem.*

Proof. The length intervals of the probes in the same pool cannot intersect because only probes that belong to the same group can be assigned to the same pool and the probes in each group are assigned to pools according to a coloring of the interval graph induced by their length intervals.

The probes in the same pool have a common acceptable hybridization temperature because they also belong to the same group. Let the group be g_i. The algorithm chooses the hybridization temperature T_j for all pools constructed from g_i from the interval $I_T(p_1) \cap I_T(p_e)$ where $I(p_1)$ has the smallest left endpoint and $I(p_e)$ the largest left endpoint of all temperature intervals of the probes in the group. The temperature interval $I_T(q)$ of a probe $q \in g_i \setminus \{p_1, p_e\}$ must contain T_j because otherwise $I_T(p_1)$ properly contains $I_T(q)$ which was not allowed. □

Lemma 3.5 *If an optimal solution of the 2D PP problem uses c^* pools, Algorithm 2 uses at most $2c^*$ pools.*

Proof. Let P_{odd} be the set of probes in the odd-numbered groups, i.e. $P_{odd} = \{p \mid p \in g_i, i \text{ is odd}\}$,and let P_{even} be the set of probes in even-numbered groups i.e. $P_{even} = C \setminus P_{odd}$. Let c_{odd} (c_{even}) be the number of pools in which the optimal solution assigns probes in P_{odd} (P_{even}). Naturally, the optimal solution uses in total at least $\max(c_{odd}, c_{even})$ pools.

On the other hand, Algorithm 3 assigns the probes in P_{odd} (P_{even}) into at most c_{odd} (c_{even}) pools because the interval graph coloring scheme used for each group is optimal and two probes $p, q \in P_{odd}$ (similarly for P_{even}) can be assigned into the same pool only if they are in the same group. In order to prove the latter statement, assume the contrary: p is in g_i and q in g_j where $j \geq i + 2$ and $I_T(p) \cap I_T(q) \neq \emptyset$. Then $I_T(p)$ must properly contain the first interval of the group g_{i+1} because the groups were constructed so that the first interval of the group g_{i+1} does not intersect $I_T(q)$. But that is a contradiction because we assumed that no temperature interval properly contains another. Thus, Algorithm 3 uses in total at most $c_{odd} + c_{even} \leq 2\max(c_{odd}, c_{even}) \leq 2c^*$ pools. □

Note that Algorithm 3 also gives a nontrivial lower bound for the number of pools $\max(c_{odd}, c_{even})$ to which the size of the solution given by the algorithm can be compared. Another useful property of the algorithm is that it chooses one common hybridization temperature for all the pools constructed from the same group. In fact, using similar arguments as in the proof of Lemma 3.5 one can show that the solution given by Algorithm 3 has at most twice as many different hybridization temperatures as any feasible solution of 2D PP. Having a large number of different hybridization temperatures would complicate performing the experiment in a laboratory.

We have not implemented Algorithm 3. Thus, the real practicality of the approach is yet to be tested.

3.7 Conclusion

There are open combinatorial problems in the topics covered in this chapter that could warrant some additional research. The probe assignment problem can be seen as a restricted minimization version of the JISP problem introduced in Section 3.2. If a practical approximation algorithm could be derived for the general minimization version of JISP, such an algorithm could be useful in multiplexing microsatellite markers.

We do not know whether 2D probe partition is an NP-hard problem. It would also be interesting to know whether 2D PP can be approximated within a constant factor if more than one candidate probe is allowed for a gene. For all of these problems, related work can be found from scheduling literature [ES03, BYHN$^+$02].

Chapter 4

Software for TRAC experiment planning

In this chapter, we introduce the Tracfinder software that we have developed for selection and pooling of TRAC probes. We start by presenting the functionality that the program provides. After that we describe the external tools that we have used in the implementation of the software.

4.1 Functionality

We have tried to implement the Tracfinder program so that it is flexible enough to be used for various TRAC probe selection and pooling tasks. The software can be used to select both PCR based and oligo probes. It can select probes for all genes of the genome or for any given subset of genes. The user can also use the program to divide existing probes into pools or to just list a large set of candidate probes together with their properties like GC-content and melting temperature. The program is quite flexible in the sense that these tasks can be mixed: The user can give her own old good probes for some of the profiled genes and let the program find new probes for the rest of the genes. The program computes properties of old and new probes and pools them together.

The Tracfinder program is used from the command line. The user specifies probe requirements, sequence files containing both target and non-target sequences, and other parameters in a configuration file. FASTA formatted files are used for gene sequences and EMBL genome files for entire genomes. The program then fully automatically searches for probe candidates that meet the requirements as described in Section 2.4 and divides them into pools if pooling was requested by the user. Finally the

program writes the probes (oligo-TRAC) or primers (PCR-TRAC) in each pool into standard output.

The parameters of the program include requirements for probe specificity and sensitivity discussed in Sections 2.2 and 2.4 and electrophoresis length range and resolution. There are also several parameters that allow the user to control the maximum number of candidate probes in different stages. These parameters are needed because for a large number of profiled genes or for large genomes it is currently not feasible to test all possible candidate probes. However, the program can also go through all possible oligo probes if that is requested.

There is one important parameter that has not been discussed yet. Often the probe sequence should be chosen so that it is close to the 3' end of the gene if possible because of the risk of RNA degradation [AJL+02]. Since the biotin label is often attached to the Poly-A tail of the mRNA, the closer the probe binding site is to the 3' end, the smaller is the risk that the mRNA is cut between the label and the probe binding site. If the mRNA is cut between the probe binding site and the biotin label, it is not detected. Tracfinder allows the user to specify how far the probe binding site can be from the 3' end as percentage of the transcript length. In the context of the experiments (Chapter 5), we will discuss many of the parameters in more detail and provide examples of how we have used them in different cases in order to find a good experiment plan.

The source code of the Tracfinder program is available at the following WWW-address:

`http://www.cs.helsinki.fi/research/fdk/programs/tracfinder.`

4.2 Implementation

The Tracfinder program is a Perl script that uses several external programs and libraries to perform various subtasks. We have tried to select tools that are well-documented and also used by others. The external tools used at each step are shown in the flow chart given in Figure 4.1. So far Tracfinder has only been used in Linux but moving to another Unix platform should be straightforward.

All the handling of sequences such as the parsing of sequence files is done with the aid of the BioPerl toolkit [SBB+02]. The description of the probe selection procedure in Section 2.4 includes the programs used at each step but for completeness we also list them here. REPuter [KCO+01] program developed at the University of Bielefeld is used for finding exact repeats. The primers for PCR-based probes are selected by program Primer3 by

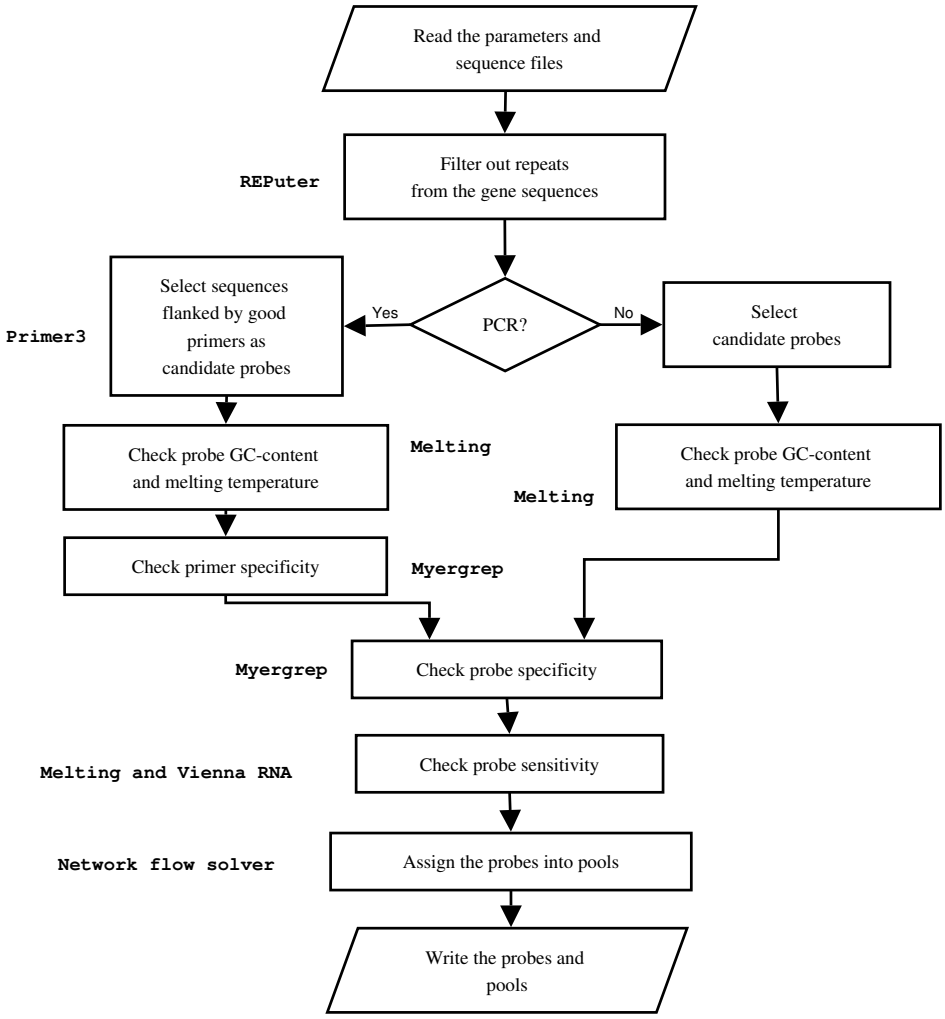

Figure 4.1: The flow chart of the Tracfinder program and the external programs used in each step.

Rozen and Skaletsky [RS98]. The approximate string matching tasks are solved with Myers' own implementation of his algorithm [Mye99] (Myergrep). The hybridization energies and melting temperatures of probes are computed with the aid of the program Melting by Le Novère [Nov01]. The Vienna RNA package [HFS+94] developed at the Institute for Theoretical Chemistry of the University of Vienna computes the secondary structures of RNAs and their energies. Finally, as described in Section 3.5, the probe

assignment algorithm is implemented using a network flow solver [CG97] by Cherkassky and Goldberg.

4.3 Conclusion

As can be seen from the flow chart in Figure 4.1, the current structure of Tracfinder is simple: The initial candidate probes go through a series of checks until only those are left that fulfill the requirements specified by the user. These probes are then assigned into pools. If none of the initial probe candidates of a gene pass all the checks, no probe is returned for the gene. It might be useful to have an iterative structure that allows producing a new set of candidate probes for a gene if the first set fails.

Currently, if the program does not find probes for all profiled genes, the user can run the program again and give the good probes already found as a part of the input. On this second run, the user can relax the requirements or produce more candidate probes for the uncovered genes. For small sets of profiled genes this strategy works quite well (an example is given in Section 5.3). For large-scale applications, it would be useful if the program could automatically cluster the homologous genes for which no specific probe can be found and pick a common probe for each cluster.

Chapter 5

Experiments

In this chapter, we describe four cases in which we have used the program to find probes and to pool them. The cases hopefully illustrate the diversity of the experiment-planning tasks. We will also describe some additions to the computational methods that we have done in order to better adapt to different situations. We aim to show on one hand that the computational tasks are not trivial and on the other hand that the algorithms and the program can solve them. All the experiment plans described in the next sections have been done in co-operation with biologists from VTT biotechnology.

5.1 Profiling yeast genes

The purpose of the first set of tests was to demonstrate that from a computational point of view TRAC is a feasible technique for genome-wide profiling. The aim here is a proof of concept. If a genome-wide experiment would actually be done, some additional tuning of parameters would probably be needed as we shall see with the smaller sets of genes. We used the yeast *Saccharomyces cerevisiae* genome as the target genome. We tried to find PCR-based probes for 6132 genes (all CDS with gene qualifier). The length of the adapters, 32 bases, was added in all probe lengths, resulting in length range 72..478 bases. Additionally, we set Primer3 so that it first tries to find probes whose length is near the upper limit in order to also get long probes into the candidate set. The parameters are summarized in Table 5.1 (the last three specificity parameters are described in detail in Section 2.4). Other Primer3 parameters were kept in their default values. The results are shown in Table 5.2.

We tried two values, 5 and 20, as the maximum number of candidate

Table 5.1: Parameters for candidate probe selection.

Probe size	72..478 bases
Primer size	19..24 bases
Primer melting temperature	62..67°C
Primer optimum temperature	66°C
Maximum number of candidate probes N	5/20
Maximum primer distance α	5000 bases
Maximum length of common substring l	15 bases
Minimum differences β and β'	0.2

probes per gene generated in the primer selection phase. In the larger one of the two searches (20 candidate probes per gene) we found at least one candidate probe for 87% of the genes. About 4% of the genes were already lost in the repeat filtering phase (step 1 in the description of Section 2.4). An additional 3% were lost in the primer selection phase (step 2) and 5% in the final specificity checking phase (steps 4 and 5). The search took 170 hours (about a week) from which over 169 hours was taken by the specificity checking phase (steps 4 and 5). The tests were run on a 1.4GHz AMD Athlon PC with 1.5GB main memory.

We were able to find a probe that fulfilled all requirements for 87% of the yeast genes. Only coding sequences were used. One might be able to find more probes if 3' UTR sequences (untranslated regions) were also available. The rest of the genes could be clustered according to sequence similarity and each homologous cluster of genes could then be probed as one unit. It should be noted that the 87% coverage is computational, not biochemical. The TRAC method described does not biochemically differ from other hybridization-based assays in its capacity to resolve different gene expression products.

Next we ran the probe assignment for these candidate probe sets with two different length resolutions. These results are also given in Table 5.2. We do not know the optimal solutions for these probe assignment instances but the approximation algorithm for probe assignment (Algorithm 2, Section 3.4) gives as a by-product a lower bound k_{min} for the minimum number of pools. Comparing the solutions found by the algorithm and the lower bounds, we see that the approximation algorithm used in all cases less than 1.3 times the optimal number of pools and once even found an optimal solution. The post-processing heuristic (Section 3.5) was not able to improve the solutions given by the approximation algorithm. The probe assignment took in all cases less than 6 minutes. Thus, the assignment takes a negli-

Table 5.2: Results with the yeast genome

N^a	Covered[b]	Probes	Length resolution	
			1% or 2 bases[c]	1.3% or 3 bases
5	4961 (81%)	12283	Pools[d]: 52 Lower bound[e]: 52 Minimum[f]: 32	Pools: 87 Lower bound: 78 Minimum: 46
20	5334 (87%)	53516	Pools: 40 Lower bound: 36 Minimum: 34	Pools: 65 Lower bound: 51 Minimum: 49

[a]Maximum allowed number of candidate probes per gene.
[b]Number of genes for which at least one candidate probe was found by Algorithm 3.
[c]The length difference function was $d(l) = \max(0.01l, 2)$.
[d]Number of pools used by Algorithm 2.
[e]Lower bound k_{min} of the number of pools for the given set of candidate probes.
[f]The (theoretical) minimum number of pools necessary for the given length range and resolution (calculated by dividing the number of covered genes by the maximum number of probes that one could fit into one pool if all probe lengths were available).

gible amount of time compared to the selection of candidate probes. The results also demonstrate the importance of producing enough candidate probes per gene so that assignment to pools can be done as efficiently as possible.

In principle, about 100..150 probes can be packed into one pool (or multiples of that if several labels are used) depending on the range and resolution of the electrophoresis. Hence, theoretically a small bacterial genome could be condensed to 20 or less pools while higher eukaryotes (30,000 genes assumed) would require about 300 pools.

It is difficult to pack all pools full of probes because of the difficulty to find a sufficient number of specific probes whose length is near the upper limit of the electrophoresis. The small number of long probes results from the second part of our specificity definition that prohibited common substrings with other gene sequences (Section 2.4). As shown in Figure 5.1, at least 15 bases long repeats split the yeast genes into short fragments. Since the probes have to be selected from these fragments, this seriously limits selection of long probes. On the positive side, this verifies that masking the repeats first is an efficient way to concentrate the probe selection to the most promising areas. Without repeat filtering, we would waste a lot more time on probes that are not specific according to our definition.

Nevertheless, our results show that it is possible in practice to assign on average more than 80 probes into one pool with quite a moderate electrophoresis resolution (1.3% or 3 bases) or more than 130 probes with a

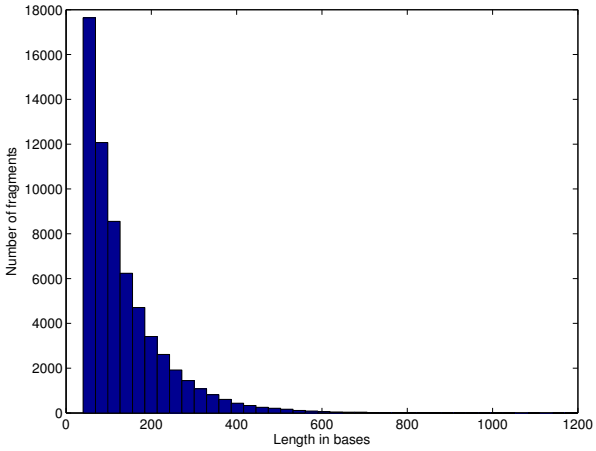

Figure 5.1: The length distribution of unique fragments of yeast genes. A unique fragment of the gene sequence is a fragment that includes no substring longer than $l = 15$ bases that is a common substring with another gene sequence. The average length of unique fragments is 135 bases. Only fragments that are long enough to contain a probe are considered.

somewhat higher resolution (1% or 2 bases). This means that the mRNA levels of all the yeast genes could be profiled using one 96-well plate.

We have also designed a smaller set of yeast probes for laboratory testing. The set consisted of 80 target genes. The requirements for probes were identical to the ones given above except for two modifications. The first laboratory tests hinted that some of the primers did not bind well enough. Thus, we added the Primer3 parameter GC-clamp that allows the user to require the specified number of consecutive Gs and Cs at the 3' end of primers and set its value to two. In addition, we produced up to 200 candidate probes per gene.

We were not able to find a probe for 5 of the 80 genes. All 5 were such that we did not find even initial candidates for them. The candidate probe set for the rest of the genes (75) was successfully assigned into one pool with resolution $d = \max(0.013l, 3)$ for probes of length l. The post-processing heuristic (Section 3.5) was needed to reduce the number of pools from two to one. In fact, this application was the one that indicated the need for such a heuristic.

From the 75 probes 20 have been tested in the laboratory so far (Kari Kataja, personal communication). After the GC-clamp modification, primers amplified the probe sequences well. Unfortunately, 4 of the 20 primer pairs

also amplified an additional PCR product, two pairs to such extent that they have been rejected. It is not clear yet whether these additional products are the result of a failure in the experiment design. One problem has been that the factory-made primers have not been pure i.e. many of the primers have incorrect sequences.

The amplified probes themselves produced a linear signal response to the amount of the target molecule in the sample. The experiment design is not responsible for the sensitivity of the probes since in this case it did not include any sensitivity-related requirements. The result merely shows that long probes are sensitive in TRAC and the experiment design does not lack vital sensitivity requirements. This is good news because predicting thermodynamic properties of several hundred base pairs long nucleic acids would be a daunting task.

So far the specificity of these probes has not been tested. The linear signal response has been shown by diluting a complex sample in which the original amounts of different molecules are unknown. So the experiment does not give direct evidence of the specificity of the hybridization.

5.2 Oligo probes for a set of fungus genes

Tracfinder has been used to design several sets of probes for the filamentous fungus *Trichoderma reesei* which is widely used in industrial enzyme production. As an example, we describe a design for a set of 17 genes. For 6 of them there was already a probe that had been tested and was known to work well. Thus, the task was to find new probes for the 11 new genes and pool all the probes together. The experiments would be done using 23..40 bases long oligonucleotide probes. A length difference of at least 2 bases was needed to separate two probes. The aim was to fit the experiment into two pools. As at most 9 probes can be put to one pool with this length range and resolution, all slots except two had to be filled. In addition, the probes were expected to fulfill fairly stringent requirements concerning sensitivity, specificity, and melting temperature.

The sequence data that we used consisted of 8623 predicted transcripts of the newly sequenced *T. reesei* genome. Unfortunately this data set does not yet cover all genes of the organism since *T. reesei* is expected to have $11,000..14,000$ genes. It took a few iterations to find probe requirements that were stringent but such that all profiled genes had at least one probe. The final parameter values used in the experiment design are summarized in Table 5.3. The thermodynamic parameters are described in Section 2.2. The values of these parameters have been chosen according to the tests

Table 5.3: Parameters of the *T. reesei* design

Probe size	24..40 bases
Probe from 3' end	0.6
Electrophoresis resolution d	2 bases
Sodium concentration	0.75 mol/l
Nucleic acid concentration	1.0×10^{-8}mol/l
Minimum melting temperature T_m	55°C
Maximum melting temperature T_m	70°C
Hybridization temperature	60°C
Maximum hybridization free energy change	-15 kcal/mol
Minimum target free energy change	-10 kcal/mol
GC-content	38..62%
Maximum length of common substring l	15 bases
Minimum difference β	0.2

that are also described in Section 2.2.

Unfortunately the approximation algorithm and the post-processing heuristic were not able to consistently assign the candidate probe sets into two pools but often used three. The solution was simple. Since the pools have to be almost full, the length difference of two consecutive probes has to be two nearly always. Thus we produced all possible candidate probes of odd length and no candidate probes of even length. This scheme has the benefit that the probe assignment algorithm can assign the resulting candidate probe set into an optimal number of pools because two probes either are of the same length or can be put into the same pool (see Section 3.3).

In Figure 5.2 we have shown all the candidate probes of odd length that fulfilled the requirements classified according to the gene and length. Even though the total number of candidate probes is high, they are not evenly distributed between different genes and lengths. For example, in addition to the six genes that have an old probe, there is one other gene that has only one candidate probe. There are very few candidate probes of length 23 or 25. In fact, over 80% of the 23..31 bases long candidate probes failed the second specificity requirement (edit distance) which highlights the difficulty of finding specific probes of this length. Nevertheless, this candidate probe set is such that it fits into two pools which was the original goal of the experiment design. In addition, all probes in the set have sufficiently similar melting temperatures even though their lengths differ considerably. The fact that such probes can be found is crucial for the oligo-TRAC analysis.

The computation of candidate probes took about 13.5 hours on a 2.6GHz

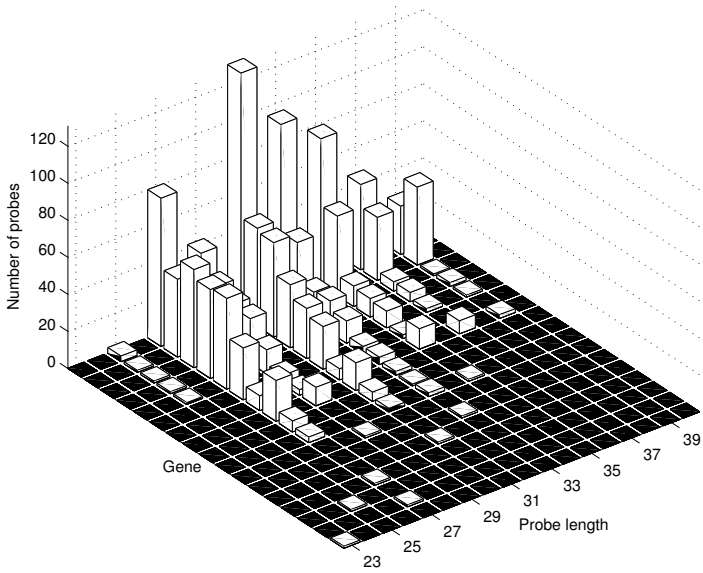

Figure 5.2: *T. reesei* candidate probes classified according to the gene and length. Genes are ordered according to the total number of probes. Black squares mean that the gene does not have any candidate probes of that length (if there were no black squares, probe assignment would be trivial). Only odd-length probes were searched. The first six genes have a probe specified by the user.

Intel Pentium 4 PC with 1GB main memory. About 85% of the processing time was consumed by the computation of target secondary structure energies. Assigning the probes into pools took about a second. This demonstrates that when designing a small experiment, it is feasible to go through all possible candidate probes (as we checked half of them) though taking the target secondary structure into account can make the computation fairly slow.

The set of profiled genes in the experiment was so small that even manual assignment to pools would be possible. However, the above automatic scheme has at least one advantage. The candidate probe set was produced in such a manner that the probe assignment algorithm guarantees the minimal number of pools for the set. This is convenient in practice because one always knows that the only way to reduce the number of pools is to

change the candidate probe set.

The three *T. reesei* pools that have been tested in the laboratory are described in Table 5.4 (data from Jari Rautio). Neither of the pools described above has been used directly but some additional modifications have been done before the tests. In the tests, *T. reesei* samples from different conditions were analyzed with TRAC and the results were compared to Northern blots and data available from other sources. The results of the comparisons are also given in Table 5.4. There are a few genes that have a low signal compared to the background (average signal/noise ratio). For these genes it is not known whether the gene should be expressed in these conditions. So, it is not clear whether the probes have failed.

The comparison to the conventional Northern blot analysis [AJL+02] is notable because in Northern blot analysis cross-hybridization should not be a significant problem. Thus, the fact that the TRAC analysis is in good agreement with Northern blots gives evidence that the probes designed with Tracfinder are specific. As an example, one of the probes that were designed manually before the genomic data was available showed clear cross-hybridization when compared to the Northern blot. This problem disappeared when the probe was replaced with a probe that was designed using the genomic data and Tracfinder.

Finally, the tests with pools that have several probes have indicated that the length of the probe does not always predict its migration in capillary electrophoresis with sufficient accuracy. Thus, a more complex measure might be needed for the size of the probe. Fortunately, our probe assignment algorithms can be used also with other measures of probe size.

5.3 PCR probes for a set of human genes

We have recently started the experiment design for 32 human genes. The goal is to design an assay for cancer diagnostics. The probe pool has not yet been tested in the laboratory.

One of the requirements is that the probes can be amplified by PCR from genomic DNA. Most human genes are available as cDNAs which simplifies the production of probes. A cDNA is a DNA copy of the mRNA molecule and thus lacks the introns. Unfortunately, for some of the genes in this set there is no cDNA easily available. The length of the probes should be 150..1000 bases (that includes adapters at both ends, so the specific sequence fragments should be 114..964 bases) and the differences in probe lengths fairly large: we have used the resolution $d = \max(0.02l, 7)$ for probes of length l i.e. 2% of the length of the probes but at least 7 bases.

Table 5.4: *T. reesei* pools, data from Jari Rautio

	Gene	S[a]	N[b]	M[c]	Length	A_c	dG	T_m	P[d]
Pool I									
1	act	13	x	x	23	-7.8	-15.3	63.7	
2	mca1	4			25	-5.0	-16.6	65.7	x
3	bip1	10	x	x	27	0.0	-13.1	59.9	x
4	hem6	4		x	29	-9.2	-16.4	64.5	x
5	trr1	3			31	-2.0	-16.4	64.0	x
6	hsp70	6			33	-9.3	-16.4	63.9	x
7	rps16b	7		x	35	-9.9	-22.9	70.0	x
8	gsh1	1			37	-3.8	-19.6	66.8	x
9	acs1	3		x	39	-7.9	-22.0	69.0	x
10	AP1	N/A		x	41	-8.0	-21.9	68.0	x
Pool II									
1	cbh1	268	x	x	25	-11.6	-12.3	58.7	
2	egl1	27	x	x	27	-3.4	-14.0	61.4	
3	sod2	3			29	-8.3	-20.4	69.7	x
4	hac1	N/A	x	x	31	-1.7	-23.7	75.2	x
5	pdi1	13	x	x	33	-4.5	-18.6	66.7	x
6	MaL	108		x	35	-2.4	-17.1	64.1	x
7	trx2	6			39	-10.0	-22.2	68.8	x
8	bgl2	5			41	-9.8	-21.9	68.4	x
Pool III									
1	eno	2		x	25	-9.2	-16.1	64.7	x
2	rpl16A	6		x	27	-4.9	-19.5	69.3	x
3	gpd1	8		x	29	-0.5	-19.9	69.4	x
4	hsp30	77			31	0.0	-20.4	69.3	x
5	gap1	8			33	-3.2	-21.6	69.9	x
6	orp1	3		x	37	-3.1	-19.6	67.0	x
7	cpa2	2			39	0.0	-20.9	68.5	x
8	nsf1	14		x	41	-5.4	-22.6	69.0	x
9	arg1	2			43	-1.9	-21.8	67.6	x

[a] Average signal/noise ratio in steady state conditions.

[b] Measured expression level changes between different conditions are consistent with Northern blot analysis of the same conditions (data not shown). For those not marked with x no data of Northern blot comparison is available.

[c] Prediction based on yeast microarray data or published data on expression responses of respective genes in *T. reesei* or other organisms. For those not marked with x no comparable prior data is available.

[d] The probes marked with X were selected using Tracfinder.

Naturally, only one pool should be used. In theory one pool could include 78 probes with this range and resolution. However, this kind of packing is hard to achieve because of the lack of long probes. Long fragments often include common substrings with other genes and human genes usually consist of several exons. Since the probes will be amplified from genomic DNA, we have to be careful not to choose probes that cross exon/intron boundaries.

Our sequence data set consisted of 27,176 human transcripts (all mRNA:s with gene tag parsed from NCBI build 34, version 3). We did not aim to separate different transcripts of the same gene from each other. So, the other transcripts of the same gene as the target transcript were not included in the non-target set. Additionally, we have not checked the specificity of the PCR primers. In this case there are laboratory techniques that allow removal of additional amplified fragments if some of the primers are not specific. The large size of the human genome would have made primer checking very time consuming. However, we tried to keep the size of the largest allowed common substring smaller than the minimum primer size. This guarantees that no non-target gene contains an exact match of both primers.

Based on some test runs, we concluded that the same set of parameters cannot be applied to all genes. The parameters used to select probes for the majority of genes (24) are summarized in Table 5.5. For six genes we allowed the probe to be selected closer to 5' primer end (80% of length from 3' end allowed instead of half) in order to find probes that fulfilled the other requirements. The last two genes were also different: one had so much A and T that we could only find a probe whose GC-content was 35%, the other was so similar to the other genes of its family that we could not find a gene-specific probe.

Naturally, it would be better if all probes would fulfill exactly the same requirements. Still, it is positive that we could apply the same quite strict specificity requirements to all genes except one. The maximum length of common substring was 3 bases longer than we have used previously but the probes are also significantly longer. On the other hand, we have made the other specificity criterion (edit distance) more stringent than before. The described set of probes was successfully assigned into one pool.

5.4 Conclusion

Hopefully we have been able to demonstrate that we are able to produce efficient TRAC experiment designs that do work in the laboratory. In our

Table 5.5: Parameters of the experiment design for the human genes

Probe size	150..1000 bases
Probe from 3' end	0.5
Electrophoresis resolution d	$\max(0.02n, 7)$
Primer size	19..24 bases
Primer melting temperature	62..67℃
Primer optimum temperature	66℃
Primer GC-clamp	2
Maximum primer distance	5000
Maximum number of candidate probes N	100
GC-content	40..60%
Maximum length of common substring l	18 bases
Minimum difference β	0.25
Maximum number of probes	200

opinion, Tracfinder software is comparable to any microarray probe selection system listed in Section 2.3 in terms of finding specific and sensitive probes. In addition, the program is able to divide the probes into small number pools.

Even though the program has been flexible enough to cope with different kinds of designs with small modifications, there is also need for further improvement. The Tracfinder program was originally designed to select probes for all the genes of fairly small genomes such as yeast. Now the main application seems to be designing small probe sets for all kinds of genomes. The program should be modified to support this kind of use better. In addition, the emphasis has so far been on implementing the necessary functionality and not on the speed of the program which could be improved significantly.

At the moment the program checks the specificity of primers using edit distance as it does for the whole probe. This probably is not an ideal solution. First, running Myers' algorithm takes lots of time when there is a large amount of non-coding DNA such as in the human genome. Second, in PCR the binding of the 3' end of the primer is especially important because a primer can only start the polymerase reaction if its 3' end binds. On the other hand, the 5' end of a primer can dangle freely and the primer can still work. Thus, it might be better to use another measure for the specificity of primers. For example one could check only the 3' end of the primer using either exact match or a small Hamming distance.

Naturally, the Tracfinder program has to be modified accordingly when

we have more quantitative information of the sensitivity and specificity of the probes designed.

Chapter 6

Transcriptional profiling when complete sequence data is not available

6.1 TRAC and cDNA-AFLP

Until now the aim has been to design experiments for profiling the transcription of a set of known genes. In this chapter, we concentrate on the TRAC procedure that also allows identification of previously unknown genes based on their expression patterns. Despite the huge advances in sequencing efforts, a vast majority of the organisms are not and will not be sequenced. Even when a genome of an organism has been sequenced, annotating all its genes is not straightforward. Thus, there is need for techniques that can identify novel genes that play a role in the biological process studied. The computational task is to make such experiments as cost-efficient as possible by using the available incomplete sequence information.

Since the aim is to also identify novel genes, gene-specific probes cannot be selected in advance. Instead, the basic idea of the technique is as follows. The starting material is genomic DNA or a cDNA library of the target organism. Some biochemical techniques are then applied to cut the sequences into fragments and to amplify and divide the fragments into small subsets. The fragments and subsets are then used as TRAC probes and pools. As before, the probes are hybridized with an mRNA sample and hybridized probes are isolated and separated using gel or capillary electrophoresis. Because of the missing sequence information, the correspondence between resulting gel bands and the genes of the organism is largely unknown. However, if a band has an interesting pattern across the experiments (different conditions or cell types), the band can be cut out from the gel and the probe

can be sequenced. The probe sequence can then be used as a fingerprint when the interesting gene is isolated in a laboratory.

The crucial question is how to produce the probes and pools without prior sequence knowledge. The technique should be such that as many genes as possible will have a probe that can be separated from other probes. On the other hand, the number of pools and probes cannot be too high or profiling becomes too expensive. One possibility is to use the same technique as in another transcription profiling method called cDNA-AFLP[1]. Differential display methods [VHB+95, BvdHdB+96] such as cDNA-AFLP [BDV+03, BDC+03, FTS+03] are among the few existing methods that can be used on a genome-wide level to identify previously unknown genes that have interesting expression patterns [BZ01]. Thus, in this chapter we will provide computational tools for optimization of cDNA-AFLP but our original motivation is to optimize TRAC pools when the complete genome of the target organism is not known.

In cDNA-AFLP (cDNA Amplified Fragment Length Polymorphism) the cutting into fragments and division into pools is done using restriction enzymes and selective PCR: cDNAs are digested with two different restriction enzymes, adapters are attached to the specific ends of the resulting fragments, and the fragments are amplified using primers extended with additional selective nucleotides. Thus, for each selective primer pair only the fragments whose ends match the primer extensions get amplified and these fragments end up in the same pool. Finally, the fragments in each pool are separated by electrophoresis.

In cDNA-AFLP, the whole procedure described above is repeated for cDNAs produced from different mRNA samples and the resulting pools are separated using gel electrophoresis. In TRAC however, the procedure is only done once for the target organism and for as complete cDNA library as possible to produce the fragments and their division into pools. The fragments are then treated as TRAC probes that can be hybridized with many mRNA samples of the same organism. Finally, the hybridized fragments are run in gel electrophoresis. If the sequences of some transcripts are available for the organism, their location on the gels can be predicted and thus some bands can be readily identified and their expression patterns recorded. On the other hand, an interesting yet unidentified band can be sequenced and this way novel genes relevant for the biological process studied can be found.

TRAC has two potential advantages over cDNA-AFLP. A large part of

[1]Initially another method was considered but it was rejected partly because our computer simulations showed it to be inefficient (data not shown).

the work such as selective PCR has to be done only once and many parts of the TRAC procedure can be automatized. Additionally, even though cDNA-AFLP is a very sensitive method i.e. it can detect low abundance transcripts [FTS+03], it is difficult to get accurate quantitative information from it because the PCR amplification can distort the relative abundances of fragments. The TRAC technique on the other hand is both sensitive and accurate.

Unfortunately, the above procedure does not automatically lead to efficient profiling of a large number of genes. Only one pair of enzymes does not in practice produce a fragment for every cDNA molecule in the collection that could be amplified and detected by electrophoresis. All fragments from a given cDNA can be too long or too short or, if either of the enzymes does not have a restriction site in a cDNA molecule, all fragments lack one of the two specific ends needed for selective PCR. Additionally, several cDNAs having the same nucleotides next to the restriction sites can produce fragments with the same length in which case these fragments end up in the same gel band and consequently do not give any useful information.

Recently the cDNA-AFLP protocol has been improved to get at most one fragment from each transcript [BZ01, BDC+03]. After the digestion with the first enzyme, the 3' ends of the cDNAs are captured and only they are digested with the second enzyme leading to at most one fragment per cDNA which has the specific ends of both enzymes. Obtaining at most one fragment from each transcript reduces the redundancy of the fragment pools and by reducing the total number of fragments it also reduces the number of selected nucleotides needed for reasonable separation of bands on the gels. Thus, we will concentrate on this variant of cDNA-AFLP which we will call the *3'-variant*, illustrated in Figure 6.1.

For the reasons given above the number of transcripts that can be profiled with a given number of restriction enzymes and selective PCR amplifications depends crucially on the choice of restriction enzymes and selective PCR primers. Although sequence information is not absolutely necessary for cDNA-AFLP experiment design, it is extremely useful for designing the experiment in such a way that the expression of as many genes as possible can be measured with reasonable experimental effort. Even when the complete transcriptome of the target organism is not known, there is often lots of sequence information available that can be utilized in experiment design such as sequences of genes cloned from the organism, EST collections, or a genome of a closely related organism. One reason for large amounts of incomplete or inaccurate sequence data is that careful manual genome annotation has trouble keeping up with the fast increase in raw sequence

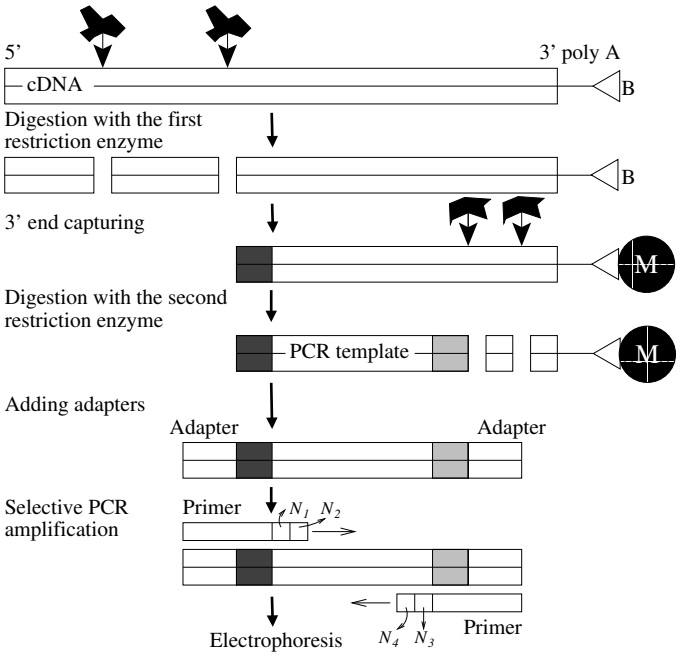

Figure 6.1: cDNA-AFLP with 3' end capture [BZ01, FTS$^+$03]. After the digestion with first restriction enzyme, the biotin (B) labeled 3' ends are captured with the aid of streptavidin coated magnetic beads (M) followed by the digestion with the second restriction enzyme and removal of the 3' ends. Adapters have to be attached to the fragments to facilitate selective PCR (the selective nucleotides are marked as N_1, N_2, N_3 and N_4).

data.

A computer simulation that uses the available sequence data provides a simple and inexpensive possibility to explore different options beforehand. Our method is based on simulating a cDNA-AFLP experiment *in silico* for the known genome of a related organism or for a known part of the target genome. The underlying assumption is that the real target genome has roughly the same characteristics as the sequence data available. Thus, if we can find efficient enzymes and selective primers for the given data, they are also likely to work well on the real target.

At least two programs GenEST [QPJ$^+$01] and AFLPinSilico (also a web service) [RdPR03]) simulate cDNA-AFLP for one enzyme pair and a set of sequences given as input. A web service [BMRG04] provides similar AFLP simulations for a large number of bacterial genomes. *In silico* simulations

have also been used to test the efficiency of potential enzyme pairs and their combinations [BDC+03, FTS+03] before the actual experiment.

In the above methods the coverage of cDNA-AFLP is estimated simply by counting for each enzyme pair the coverage of the fragments which are in the length range of electrophoresis. The total coverage of several enzyme pairs is estimated simply as the union of the coverages of the best individual pairs. The question of the number of selective nucleotides needed for reasonable band separation is treated as a separate issue even though the idea of choosing the most informative combinations of selective nucleotides has been mentioned [RdPR03].

Our main contribution is to treat choosing enzyme pairs and selective primers together as one optimization problem. Our model explicitly takes into account the selective PCR and the electrophoresis so that in an *in silico* experiment only the bands that are sufficiently far apart from other bands in the electrophoresis give information. That allows us to consider all combinations of enzyme pairs and subsets of selective primers as possible experiment designs. Unfortunately finding the most efficient experiment design (the one that enables profiling the largest set of transcripts) from the vast space of possible designs turns out to be an NP-hard optimization problem. We device an optimal algorithm for the special case where only one restriction enzyme pair is used and a heuristic algorithm for the general case that carefully exploits the combinatorial structure of the problem. Finally, we test the heuristic algorithm with several data sets. The tests suggest that optimized selection of enzymes and selective primers could save a significant amount of resources.

6.2 Pool selection problem

The goal of the optimization is to find an experiment design that enables profiling as many different transcripts i.e. RNA products of genes as possible with the available resources. The most costly parts of the cDNA-AFLP procedure are the running of electrophoresis gels and the manual extraction and purification of fragments from them for sequencing. Also, using more than a couple of different restriction enzyme pairs in an experiment would be unpractical. For example, a specific primer has to be designed for each enzyme.

In our computational model the cost of the experiment is the number of pools. In this chapter a pool p is a subset of transcripts determined by a triple (e_1, e_2, s) where e_1 and e_2 are restriction enzymes with specified restriction sites and $s = N_1 N_2 \cdots N_l$ is a sequence of selective nucleotides

$N_i \in \Sigma$, where $\Sigma = \{A, T, G, C\}$. The selective nucleotides can be attached to either or both primers as long as the positions are fixed i.e. the sequence s is a concatenation of two sequences: the selective nucleotides attached to the left primer and the selective nucleotides attached to the right primer (see for example Figure 6.1).

Including the pool p into an experiment means doing the following laboratory work: the cDNA sample digested with the enzyme pair (e_1, e_2) is amplified using primers matching the specific ends of the enzyme pair (e_1, e_2) and extended with the selective nucleotides s. The pool p *covers* a transcript t or $t \in p$ if p can be used to profile t. The profiling is possible if three conditions are met: *i)* the enzyme pair (e_1, e_2) produces a fragment from t that is in the length range of electrophoresis, *ii)* the fragment has the Watson-Crick pairs of nucleotides s in the selective positions, and *iii)* p produces no other fragments of the same length so that the fragment of t can be identified from the gel.

The number of the pools approximates well the amount of resources needed in the experiment as long as we do not use too many different enzyme pairs. Thus, we want to maximize the total coverage of the experiment with a given number of pools and enzyme pairs. We get the following optimization problem (we denote the set of all sequences from the alphabet Σ by Σ^*).

Pool selection: Let T be a collection of transcripts, E a set of restriction enzymes with specified restriction sites, $[d_{min}, d_{max}]$ an electrophoresis length range, and P the set of all pools each determined by a triple from the set $E \times E \times \Sigma^*$. Additionally, let k and n be positive integers. Select from the set of all pools P at most n pools that use at most k different enzyme pairs such that the pools cover as many transcripts as possible.

In practice we also restrict the maximum number of selective nucleotides in a primer pair to some small value l. Then each individual pool can be easily computed as a preprocessing step. However, finding a set of pools maximizing the total coverage is not as easy. We say that an algorithm approximates pool selection within a ratio of $\alpha < 1$ if the number of transcripts covered by the algorithm is at least α-fraction of the number of transcripts covered by the optimal solution. We prove the following theorem. In the theorem, $e = 2.71828\ldots$ is the base of the natural logarithm.

Theorem 6.1 *For any $\epsilon > 0$, pool selection cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \epsilon)$ unless $P = NP$.*

Proof. Theorem 6.1 follows from the strong inapproximability result for max k-cover that Feige gave in his seminal paper on approximating set cover [Fei98] and from Lemma 6.1 given below. □

Max k-cover: Let U be a set of m points, $C = \{C_1, C_2, \ldots C_z\}$ a collection of subsets of U, and k a positive integer. Select k subsets from C such that their union contains as many points as possible.

Proposition 6.1 (Feige) *For any $\epsilon > 0$, max k-cover cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \epsilon)$ unless $P = NP$.*

Lemma 6.1 *If pool selection can be approximated in polynomial time within some ratio of α, where $0 < \alpha \leq 1$, max k-cover can also be approximated within the same ratio.*

Proof. Let us consider the special case where no selective PCR is done but we assume that two fragments are separable from each other if and only if they are of different length. Thus, the pool selection reduces to selecting k pools with k different enzyme pairs. Denote the sequence of nucleotides that starts from position i and ends at position j in transcript t by $t[i..j]$. Also, denote the sequence of nucleotides that specifies the restriction site of an enzyme e by $r(e)$.

We aim at mapping any max k-cover instance to a pool selection instance in such a way that a solution for the latter can be used to provide a solution with at least the same approximation ratio for the former. Set the electrophoresis length range to $[m, 2m]$. For each subset $C_i \in C$, add two new enzymes e_{i1} and e_{i2} with no nucleotide A in their restriction sites. Choose the length of the restriction site to be $m + 1$ because that enables constructing the required number of $\leq 2z$ unique restriction sites since $3^{m+1} > 2^{m+1} \geq 2z$.

For each point $u_j \in U$, $j = 1, \ldots, m$, add a transcript t_j. Construct the sequence of the transcript t_j as follows (see also Figure 6.2). For each subset $C_i \in C$ such that $u_j \in C_i$, set $t_j[5mi..5mi + m] = r(e_{i1})$ and $t_j[5mi + m + j..5mi + 2m + j] = r(e_{i2})$. Fill the rest of the sequence with nucleotide A. The total length of the sequences is $< 5m^2(z+1)$ and the whole construction can be done in polynomial time.

It is easy to verify that the sequences and enzymes constructed have the following desired properties. The enzyme pair (e_{i1}, e_{i2}) produces a fragment from the transcript t_j if and only if $u_j \in C_i$. The fragments produced by the pair (e_{i1}, e_{i2}) have lengths within the length range and if the fragments are from two different transcripts they have different lengths. So, the pool

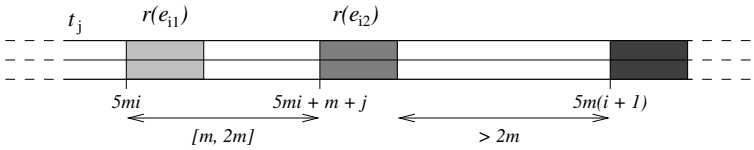

Figure 6.2: Construction of a transcript of the pool selection instance.

with the enzyme pair (e_{i1}, e_{i2}) covers the transcript t_j if and only if $u_j \in C_i$. On the other hand, no enzyme pair of type (e_{p1}, e_{q2}) or of type (e_{p2}, e_{q1}), where $p \neq q$ produces a fragment whose length is within the length range.

Assume that a polynomial time algorithm B approximates pool selection within a ratio of α. Approximate max k-cover using B in the following manner. Run B for the pool selection instance constructed as shown above. For each k pools in the solution given by B, add the corresponding subset to the solution of max coverage instance. The obtained solution for the max coverage instance is of the same size as the solution for pool selection instance because a point is included in the max k-cover solution if and only if the corresponding transcript is covered by the pool selection solution.

On the other hand, the optimal solution of the max k-cover instance cannot be larger than the optimal solution of the pool selection instance because, given the optimal solution of the former, we can use the above construction to get a solution of the same size for the latter. In summary, using algorithm B we could approximate max k-cover within the ratio of α. □

Note that our formulation of the pool selection problem includes several input parameters such as the number of different restriction enzymes available and the length range of the electrophoresis equipment that in reality get only values from some limited range. So the inapproximability result merely shows that one is unlikely to find a polynomial time algorithm that solves the pool selection optimally without restricting the values of some of the parameters. In the next section we give an efficient algorithm for the special case where the enzyme pair is given and then in Section 6.4 we use it to build a heuristic algorithm for the general pool selection problem.

6.3 An algorithm for selecting primers for an enzyme pair

In this section we assume that the enzyme pair of the pools is given and develop an efficient algorithm for selecting the selective nucleotides of the

primers so that the coverage of the pools is maximized. We will call a sequence of selective nucleotides s a *selective pattern*, and since the enzyme pair is fixed we say that a selective pattern s covers a transcript t if t is in the pool determined by s and the fixed enzyme pair. The algorithm can be used to optimize an experiment that uses only one enzyme pair but our main motivation is to get a building block for our general algorithm for pool selection.

The 3'-prime variant of cDNA-AFLP allows efficient selection of selective patterns if we make the following restriction which we will call the *prefix property*: If a selective pattern s is chosen, no prefix of s can be chosen. By a prefix we mean a sequence consisting of the first i contiguous nucleotides of s for some $i < |s|$, for example if we choose pattern ATG we cannot choose its prefix AT. The restriction does not strongly limit the space of possible solutions because a pattern and its prefixes cover largely overlapping sets of transcripts.

The prefix property helps designing an optimization algorithm based on dynamic programming because in this case the pools of an enzyme pair are disjoint. The digestion with an enzyme pair produces at most one fragment from each transcript and the selective pattern unambiguously determines the pool of that fragment. Therefore, if two selective patterns (neither is a prefix of another) cover transcripts so that the first covers c_1 transcripts and the second one c_2 transcripts, together they cover exactly $c_1 + c_2$ transcripts because no transcript can be covered by both.

Generally, let c_s be the number of transcripts covered by the selective pattern s. We assume that the coverage of each individual selective pattern has been computed as a preprocessing step. Let $c_s[m]$ be the maximum number of transcripts covered by m selective patterns with the common prefix s. Similarly, let $c_S[m]$ denote the maximum number of transcripts that can be covered by m such patterns that each of them has a prefix in the set of prefixes S. Additionally, let sN denote the pattern which is a concatenation of s and a nucleotide N. If we allow at most l selective nucleotides, we can compute $c_s[m]$ from the recurrence

$$c_s[m] = \begin{cases} 0 & \text{if } m = 0, \\ c_s & \text{if } m = 1 \text{ and } |s| = l, \\ \max\{c_s, c_{sA}[1], c_{sT}[1], c_{sG}[1], c_{sC}[1]\} & \text{if } m = 1 \text{ and } |s| < l, \\ \max_{0 \le j \le m}(c_{\{sA,sT,sG\}}[j] + c_{sC}[m-j]) & \text{if } m > 1 \text{ and } |s| < l \end{cases} \tag{6.1}$$

where $c_{\{sA,sT,sG\}}$ can be computed in the same manner

$$c_{\{sA,sT,sG\}}[m] = \max_{0 \le j \le m}(c_{\{sA,sT\}}[j] + c_{sG}[m-j]) \tag{6.2}$$

and finally $c_{\{sA,sT\}}$ can be computed directly from the coverages of longer patterns

$$c_{\{sA,sT\}}[m] = \max_{0 \leq j \leq m} (c_{sA}[j] + c_{sT}[m-j]). \tag{6.3}$$

The choice of the order in which the nucleotides are processed is arbitrary. The correctness of the recurrences (1-3) follows from the fact that because of the prefix property, the only way to choose more than one pattern with a common prefix s is to only take patterns longer than s and they differ by at least one nucleotide. Thus, the corresponding subsets are disjoint and the size of their union is the sum of sizes of individual subsets.

Our goal is to compute the maximum coverage possible using at most n selective patterns $c[n] = \max_{0 \leq m \leq n}(c_\epsilon[m])$ where ϵ denotes a sequence of length zero. We cannot simply take the value of $c_\epsilon[n]$ since the coverage is not guaranteed to be an increasing function of the number of patterns m because of the prefix property (for example pattern AT can cover more than ATT and ATG together but because of our restriction we cannot choose the coverage of AT and ATT as value of $c_{\{sA,sT\}}[2]$). A set of at most n patterns giving the optimal coverage can be collected afterwards using traceback if we store the index values giving the maximum coverages while evaluating the recurrences.

The formulas (1-3) can be evaluated in time $O(n^2)$ for a selective pattern s if the values $c_{sN}[m]$ for all $N \in \Sigma$ have already been computed. There are $O(4^l)$ patterns of length $\leq l$. Thus, the result $c[n]$ can be computed in time $O(4^l n^2)$ which is reasonable because l is in practice quite small, for example 4. Additionally, the preprocessing step requires following time. Let M be the total length all the transcripts in T. The fragments produced by an enzyme pair can be computed in time $O(M)$. Since there are at most one fragment per transcript that belongs to at most l pools determined by at most l nucleotides, all the fragments can be divided into pools in time $O(l|T|)$.

The dynamic programming algorithm approach generalizes to the case in which a pair of enzymes is used in both orders. Namely, the sample is first digested using the enzymes in the order (e_1, e_2) and selectively amplified and then separately the same is done in the order (e_2, e_1). Since the 3'-variant produces an amplified fragment from a gene sequence only if the second enzyme has a restriction site closer to the 3' end of the sequence than the first enzyme, the two enzyme orders can never both produce a fragment from a transcript. So, two pools, one produced by (e_1, e_2), another by (e_2, e_1), are always distinct i.e. they never contain the same transcripts.

To extend the dynamic programming algorithm, let $c^1[m]$ be the already computed maximum coverage of at most m pools, where $m = 1, \ldots, n$, for

the ordered pair (e_1, e_2) and let $c^2[m]$ be the corresponding coverage for the pair (e_2, e_1). Then, the maximum number of transcripts covered by the usage of both ordered pairs denoted by $c^{1,2}[n]$ can be obtained from the recurrence

$$c^{1,2}[n] = \max_{0 \leq m \leq n} (c^1[m] + c^2[n-m]).$$

The following theorem summarizes the results for selecting primers for a fixed enzyme pair.

Theorem 6.2 *The optimal set of at most n selective patterns of length $\leq l$ for an enzyme pair can be chosen in time $O(M + l|T| + 4^l n^2)$ if no chosen selective pattern is allowed to be a prefix of another chosen pattern.*

6.4 An algorithm for pool selection

Our heuristic algorithm is a modification of the greedy algorithm for max k-cover [Fei98]. As in the greedy max k-cover algorithm, we iteratively select the enzyme pairs that cover the largest number of yet uncovered transcripts until we have used all k different enzyme pairs. However, compared to max k-cover we have the additional freedom to choose how many of the n pools to allocate for each of the k different enzyme pairs. Thus, when adding the i:th enzyme pair, we also consider different possibilities to divide the pools between the $i-1$ pairs already chosen and the new one. We can do it efficiently if we have stored the largest set of transcripts we have been able to cover using $i-1$ pairs and j pools for each $j = 1, \ldots, n$. Then the dynamic programming algorithm of Section 6.3 can be used to compute the optimal coverage of the new enzyme pair in the set of yet uncovered transcripts simultaneously for all numbers of pools m where $1 \leq m \leq n-j$. We have to modify the algorithm slightly: instead of directly taking c_s, the number of transcripts covered by the selective pattern s, we only count transcripts that have not been covered by the j pools already chosen.

The pseudocode of the greedy algorithm is given in Figure 6.3. We assume that the pools have been computed as a preprocessing step. The algorithm returns the largest subset of transcripts found C_{max} that can be covered using at most k enzyme pairs and n pools. It stores the intermediate results in a table MAXCOVER whose element MAXCOVER$[i, j]$ contains the largest set of transcripts found so far that can be covered with i different enzyme pairs and j pools. The procedure DYNAMIC implements the modified dynamic programming algorithm and its traceback: DYNAMIC(A,

Algorithm Greedy
Input: the set of transcripts T, the set of enzymes E,
the set of all pools,
maximum number of enzyme pairs used k,
maximum number of primers used n
for $i \leftarrow 1$ **to** k **do**
  for $j \leftarrow 1$ **to** n **do**
    MAXCOVER$[i, j] \leftarrow 0$
  for $j \leftarrow 1$ **to** n **do**
    if $i = 1$ **then**
      $C \leftarrow \emptyset$
    else
      $C \leftarrow$ MAXCOVER$[i - 1, j]$
    for each enzyme pair (e_1, e_2), where $e_1, e_2 \in E$ **do**
      NEWCOVER $\leftarrow$ DYNAMIC$(T \setminus C,\ n - j,\ e_1,\ e_2)$
      for $m \leftarrow 1$ **to** $n - j$ **do**
        $C_{old} \leftarrow$ MAXCOVER$[i, j + m]$
        $C_{new} \leftarrow$ NEWCOVER$[m]$
        if $(|C_{new}| + |C| > |C_{old}|)$ **then**
          MAXCOVER$[i, j + m] \leftarrow C \cup C_{new}$
return C_{max} such that $|C_{max}| = \max_{1 \le j \le n}\{|\text{MAXCOVER}[k][j]|\}$

Figure 6.3: The pseudocode of the greedy pool selection algorithm.

m, e_1, e_2) returns a table with m elements such that the j:th element contains the largest subset that can be covered from the set of transcripts A using j pools with the enzyme pair (e_1, e_2). Again, the at most n pools covering the set of transcripts C_{max} can be collected using traceback if some additional bookkeeping is done during the greedy algorithm.

The running time of the pool optimization consists of the following parts. The preprocessing takes time $O((M + l|T|)|E|^2)$ since there are $O(|E|^2)$ enzyme pairs for which the pools have to be computed. The most intensive part the greedy algorithm is calling procedure DYNAMIC $O(k|E|^2 n)$ times. In each call computing the recurrences takes the time $O(4^l n^2)$ as before. In addition, we need to check for each transcript in a pool whether it has already been covered. Because a transcript can be in at most l pools with the same enzyme pair, checking takes time $O(l|T|)$ in one call of DYNAMIC. Finally, we need to collect the transcripts covered by m pools for each m which takes time $O(n|T|)$. Thus, the total running time of the greedy algorithm is $O(k|E|^2 n^2(4^l n + |T|))$ assuming that $n > l$.

6.5 Simulation results

We have implemented the simulation model described in Section 6.2 and the greedy algorithm for pool selection described in Section 6.4 and tested them with yeast (*Saccharomyces cerevisiae*), plant (*Arabidopsis thaliana*), and human (*Homo sapiens*) sequence data sets. In both large scale cDNA-AFLP studies reported recently only a single enzyme pair has been used but in both orders. Breyne et al. [BDC$^+$03, BDV$^+$03] digested a tobacco sample (*Nicotiana tabacum*) with *MseI* and *BstYI* enzymes and first divided it into two parts by preamplifying with a *BstYI* primer with either T or C nucleotide at the 3' end of *BstYI* restriction site PuGATCPy. After that they made selective amplification with all combinations of two and three additional selective nucleotides. Fukumura et al. [FTS$^+$03] tested the coverage of cDNA-AFLP (they call their variant of the technique HiCEP) in yeast (*S. cerevisiae*) by using the enzyme pair (*MspI*, *MseI*) with all combinations of four selective nucleotides.

We simulated experiments in which enzyme pairs were always used in both orders and the number of pools was between 128 and 512 as in the reported studies but we allowed more than one enzyme pair to be used. We compared the performance of our greedy algorithm to a simple strategy that we call fixed. As explained in Section 6.4, the greedy algorithm tries to select such a combination of enzyme pairs and selective nucleotides that it covers as many transcripts as possible. Thus, in a greedy design each restriction enzyme pair may be followed by a different number of PCR amplifications and selective primers may have different numbers of selective nucleotides. In contrast, the fixed design always contains all selective primers for the given number of selective nucleotides, as in the reported studies. For the fixed design we chose those enzyme pairs that individually cover the largest number of transcripts with the given number of selective nucleotides.

We considered all pairs of 10 restriction enzymes with different restriction sites of length four (four-cutters). The used enzymes are listed in Table 6.1. We excluded enzymes that have ambiguous recognition sites or produce blunt-ended fragments. We concentrated on four-cutters because they gave better coverages than other enzymes in preliminary simulations (data not shown). When simulating electrophoresis we used parameters similar to ones used by [FTS$^+$03]. We accepted a fragment if its length was between 40bp and 700bp and assumed that two fragments can be separated if their length difference was more than 0.5% of their total length. The maximum number of selective nucleotides in a primer was four.

The test results are summarized in Table 6.2. The reported coverage of

Table 6.1: The restriction enzymes used in the simulations and their restriction sites.

MaeI	CTAG
Tru1I	TTAA
DpnII	GATC
MspI	CCGG
NlaIII	CATG
Tsp509	AATT
HinP1I	GCGC
Csp6I	GTAC
TaqI	TCGA
MaeII	ACGT

a greedy design is the percentage of transcripts it covers from the data set that was used to optimize the design. The results with the human data set are shown in more detail in Figure 6.4. The greedy designs with 4 enzyme pairs and 512 pools are shown in Table 6.3. Interestingly, when using 512 pools the coverage was higher in the human data set than in the much smaller yeast data set, probably due to the higher average length of the transcripts in the human data set (2692bp vs. 1412bp). The coverages in yeast were lower than reported by [FTS+03], possibly because our data set did not include 3' UTRs (untranslated regions).

In reality, we cannot expect the target sequences to be available for the optimization of the experiment design. Therefore, we also tested the *in silico* coverage of designs that were optimized using sequences from another organism. We used two closely related, recently sequenced filamentous fungi *Neurospora crassa* [G+03] (10082 transcripts) and *Aspergillus nidulans* [Asp03] (9541 transcripts) as the target organisms. We used the greedy algorithm and sequence data from different organisms to choose a design with 3 enzymes and 256 pools (yeast and human data sets were as described above). Table 6.4 shows how many percents of *N. crassa* and *A. nidulans* transcripts were covered by the designs.

The longest simulation took about 2 days with a 3Ghz PC with 2GB of main memory running Linux. The implementation is currently written using Perl and BioPerl [SBB+02]. The running time could be significantly reduced by writing the most intensive parts in C.

The test results are encouraging even though with a particular number of enzyme pairs and pools the improvements in *in silico* coverage achieved by the optimized designs are fairly modest. Considering a large space of ex-

Table 6.2: *In silico* coverages of different experiment designs.

Organism	mRNAs	Pools[a]	Design	Coverage (%)			
				1	2	3	4
S. cerevisiae	6355[b]	128	Fixed[c]	51	-	-	-
		(112)	Greedy[d]	53	68	75	78
		256	Fixed	-	72	-	-
		(217)	Greedy	56	73	81	84
		384	Fixed	-	-	81	-
		(297)	Greedy	57	75	83	86
		512	Fixed	57	-	-	84
		(346, 267)	Greedy	57	76	84	88
A. thaliana	18590[e]	128	Fixed	47	-	-	-
		(116)	Greedy	49	59	63	64
		256	Fixed	-	70	-	-
		(226)	Greedy	57	73	78	81
		384	Fixed	-	-	82	-
		(313)	Greedy	60	78	84	88
		512	Fixed	62	-	-	87
		(512, 374)	Greedy	62	81	88	91
H. sapiens	16138[f]	128	Fixed	51	-	-	-
		(116)	Greedy	53	65	70	71
		256	Fixed	-	74	-	-
		(209)	Greedy	62	77	83	86
		384	Fixed	-	-	84	-
		(268)	Greedy	65	82	88	91
		512	Fixed	65	-	-	89
		(433, 329)	Greedy	65	84	90	93

[a] The number of pools used. The number of pools in the greedy design that covers as many genes as the fixed design is given in the parenthesis (with 512 pools the first figure corresponds to 1 enzyme pair and the second one to 4 enzyme pairs).

[b] All ORF coding sequences from SGD version dated Jun 23, 2003, no UTRs.

[c] *In silico* coverage (%) using all primers with a fixed number of selective nucleotides (3 when 128..384 pools, 3 or 4 when 512 pools), enzyme pairs used in both orders.

[d] *In silico* coverage (%) when using the pools chosen by the greedy algorithm, enzyme pairs used in both orders.

[e] All transcripts from UniGene build 42 which have a RefSeq identifier.

[f] All transcripts from UniGene build 166 which have a RefSeq identifier.

Table 6.3: The greedy designs using 4 enzyme pairs and 512 pools.

Organism	Enzyme pair	Number of primers	
		1st order	2nd order
S. cerevisiae	(DpnII GATC, Csp6I GTAC)	64	64
	(TaqI TCGA, NlaIII CATG)	62	65
	(Tru1I TTAA, MaeII ACGT)	27	91
	(Tru1I TTAA, Tsp509 AATT)	68	70
A. thaliana	(Tru1I TTAA, MspI CCGG)	18	115
	(MaeI CTAG, TaqI TCGA)	64	54
	(MaeII ACGT, NlaIII CATG)	73	47
	(TaqI TCGA, Csp6I GTAC)	71	70
H. sapiens	(MaeI CTAG, DpnII GATC)	45	49
	(Tru1I TTAA, MspI CCGG)	16	108
	(DpnII GATC, Csp6I GTAC)	69	85
	(MaeII ACGT, NlaIII CATG)	103	37

Table 6.4: The *in silico* coverages of the designs optimized with an other organism (3 enzyme pairs and 256 pools)

Target organism	Design organism	Coverage (%)[a]
N. crassa[b]	*N. crassa*	76
	A. nidulans	74
	S. cerevisiae	68
	H. sapiens	65
A. nidulans[c]	*A. nidulans*	83
	N. crassa	81
	S. cerevisiae	76
	H. sapiens	74

[a]The percentage of the target organism transcripts that are covered by the design optimized using the design organism data set.

[b]Assembly 3 [G$^+$03].

[c]Data Version 3/7/2003 [Asp03].

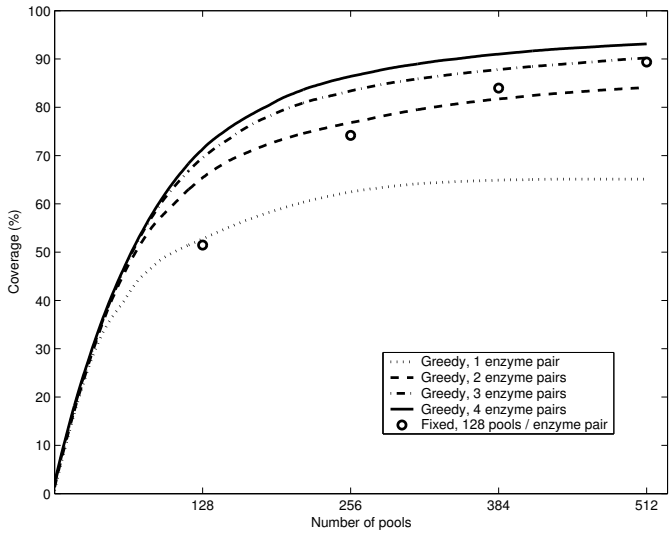

Figure 6.4: The *in silico* coverages in *H. sapiens* data set of greedy designs with 1..512 pools and 1..4 different restriction enzyme pairs and fixed designs with 128 (1 enzyme pair), 256 (2), 384 (3), and 512 (4) pools.

periment designs is still worthwhile. Using only the pools giving good gene coverage (greedy design) allows covering the same amount of genes with a smaller work load than simply using all pools with a certain restriction enzyme combination and number of selective nucleotides (fixed design). For example, consider the fixed designs with 512 pools and four enzyme pairs (Table 6.2). The greedy algorithm is able to cover the same amount of transcripts using also four enzyme pairs but considerably fewer pools in all data sets: yeast 267 pools (48% reduction), plant 374 pools (27% reduction), and human 329 pools (36% reduction). On the other hand, in the human data set the greedy design covers more transcripts with three enzyme pairs and 512 pools than the fixed design with four enzyme pairs and the same number of pools (Figure 6.4).

Table 6.4 shows that if the design has been optimized using sequences from a closely related organism (*A. nidulans* and *N. crassa*), the coverage of the design is almost as good as using sequences from the organism itself. As expected, the designs optimized using a distant sequence set of different size (*H. sapiens*) perform worse.

6.6 Conclusion

A computer simulation alone can never precisely predict the coverage of an cDNA-AFLP experiment or corresponding TRAC procedure. Our knowledge of the transcriptomes is incomplete which is the very reason to use these methods. Especially the 3' UTRs (untranslated regions) that are important when simulating 3' variant are often poorly characterized. Even if we would know the whole transcriptome, the coverage also depends on the number of different expressed transcripts in the experiment, one of the things the experiment is supposed to measure. For example, the human sequence data set we used in simulations is certainly far from being the set of all transcripts but on the other hand it seems unlikely that the whole human transcriptome would be expressed in a single experiment.

Despite the uncertainties, we feel that using the suggested experiment optimization together with expert knowledge and experience could result in considerably more efficient use of resources in cDNA-AFLP experiments. The described methods give flexible means to explore different ways to allocate resources during the experiment design process. In addition, our simulations show that using TRAC to find novel genes is realistic in terms of required resources.

Chapter 7

Conclusion

The aim of this thesis was to provide practical tools for the automatic planning of VTT-TRAC experiments. The Tracfinder software that implements the methods developed has now been used in the planning of real experiments. In these cases we have been able to meet the goals that were set for the experiment plans. The selected probes fulfill strict specificity and sensitivity requirements and they fit into the smallest possible number of pools. Most importantly, the probes that have been tested have worked well. These experiments have been quite small, a few dozen genes and 1 or 2 pools. Thus, even manual planning would have been possible. However, the biologists have recognized that our automatic tool saves a lot of work and most likely improves the quality of the plans. Therefore, we expect that the automatic planning of experiments will become an integral part of VTT-TRAC and increase the attractiveness of the technique.

We wanted to demonstrate *in silico* that the VTT-TRAC procedure can be multiplexed so efficiently that even a whole genome could be profiled with a reasonable amount of resources. We used the yeast genome as a test case. We found at least one PCR-based probe that fulfilled our specificity requirements for 87% of the genes. We were able to assign the probes into 40 pools which means packing on average more than 130 probes into one pool. Even though the achieved packing density is somewhat less than the technical upper limit, about 150 in this case, it clearly shows that our methods allow efficient multiplexing on the genomic scale.

So far, the program has been used to select probes for all genes of relatively small genome (yeast) and for small sets of genes of larger genomes (fungus, human). The current program is too slow for selecting a probe for all genes if the probes are PCR-amplified from the genomic DNA of human or other organism that has large amount of non-coding DNA. Currently, this type of application is not the primary goal.

We proposed a computational method for the optimization of cDNA-AFLP experiments. Our *in silico* tests suggest that the optimization could save a substantial amount of resources but the real practical value of the method has not yet been tested.

The principle behind this thesis has been to treat the experiment-planning tasks as rigorous optimization problems and to provide solutions that are both theoretically justified and practical. Hopefully, we have been able to show that such an approach is fruitful and worth applying to other experimental techniques as well.

References

[AGM+90] Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lippman, D. J., Basic local alignment search tool. *Journal of Molecular Biology*, 215,3(1990), pages 403–410.

[AJL+02] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P., *Molecular Biology of the Cell*. Garland Science, fourth edition, 2002.

[AMO93] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B., *Network flows*. Prentice Hall, 1993.

[AMY03] Aumann, Y., Manisterski, E. and Yakhini, Z., Designing optimally multiplexed SNP genotyping assays. *Proceedings of WABI 2003, LNBI 2812*, 2003, pages 320–338.

[Asp03] Aspergillus Sequencing Project. Center for Genome Research, `http://www.broad.mit.edu`, 2003.

[BDC+03] Breyne, P., Dreesen, R., Cannoot, B., Rombaut, D., Vandepoele, K., Rombauts, S., Vanderhaeghen, R., Inze, D. and Zabeau, M., Quantitative cDNA-AFLP analysis for genome-wide expression studies. *Molecular Genetics and Genomics*, 269,2(2003), pages 173–179.

[BDV+03] Breyne, P., Dreesen, R., Vandepoele, K., Veylder, L. D., Breusegem, F. V., Callewaert, L., Rombauts, S., Raes, J., Cannoot, B., Engler, G., Inzé, D. and Zabeau, M., Transcriptome analysis during cell division in plants. *Proceedings of National Academy of Sciences*, 99,23(2003), pages 14825–14830.

[BMRG04] Bikandi, J., Millán, R. S., Rementeria, A. and Garaizar, J., In silico analysis of complete bacterial genomes: PCR,

AFLP-PCR, and endonuclease restriction. *Bioinformatics*, in press.

[BvdHdB+96] Bachem, C. W., van der Hoeven, R. S., de Bruijn, S., Vreugdenhil, D., Zabeau, M. and Visser, R. G., Visualization of differential gene expression using a novel method of RNA fingerprinting based on AFLP: analysis of gene expression during potato tuber development. *Plant Journal*, 9,5(1996), pages 745–753.

[BYHN+02] Bar-Yehuda, R., Halldórsson, M. M., Naor, J., Shachnai, H. and Shapira, I., Scheduling split intervals. *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2002)*, 2002, pages 732–741.

[BZ01] Breyne, P. and Zabeau, M., Genome-wide expression analysis of plant cell cycle modulated genes. *Current opinion in Plant Biology*, 4, pages 136–142.

[cCP03] chin Chang, P. and Peck, K., Design and assessment of a fast algorithm for identifying specific probes for human and mouse genes. *Bioinformatics*, 19,11(2003), pages 1311–1317.

[CG97] Cherkassky, B. V. and Goldberg, A. V., On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19, pages 390–410.

[CKN+95] Corneil, D. G., Kim, H., Natarajan, S., Olariu, S. and Sprague, A. P., Simple linear time recognition of unit interval graphs. *Information processing letters*, 55, pages 99–104.

[CLR90] Cormen, T. H., Leiserson, C. E. and Rivest, R. L., *Introduction to Algorithms*. MIT Press, 1990.

[EAYR+01] Evertsz, E. M., An-Young, J., Ruvolo, M. V., Lim, A. C. and Reynolds, M. A., Hybridization cross-reactivity with homologous gene families on glass cDNA microarrays. *BioTechniques*, 31, pages 1182–1192.

[ES03] Erlebach, T. and Spieksma, F. C., Interval selection: Applications, algorithms, and lower bounds. *Journal of Algorithms*, 46, pages 27–53.

[Fei98] Feige, U., A threshold $\ln n$ for approximating set cover. *Journal of the ACM*, 45,4(1998), pages 634–652.

[FN03] Fredriksson, K. and Navarro, G., Average-optimal multiple approximate string matching. *Proceedings of CPM'2003, Lecture Notes in Computer Science 2676*, 2003, pages 109–128.

[FTS+03] Fukumura, R., Takahashi, H., Saito, T., Tsutsumi, Y., Fujimori, A., Sato, S., Tatsumi, K., Araki, R. and Abe, M., A sensitive transcriptome analysis method that can detect unknown transcripts. *Nucleic Acid Reseach*, 31,16(2003), page e94.

[Fuc97] Fuchs, R., Grouper - creation of marker sets for multiplexed genotyping. *Bioinformatics*, 13, pages 239–241.

[G+03] Galagan, J. E. et al., The genome sequence of the filamentous fungus *Neurospora crassa*. *Nature*, 422, pages 859–868.

[Gab83] Gabow, H. N., An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. *Proceedings of the 15th ACM Symposium on the Theory of Computing (STOC'83)*, 1983, pages 448–456.

[GJ79] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[GLL79] Gupta, U. I., Lee, D. T. and Leung, J. Y.-T., An optimal solution for the channel-assigment problem. *IEEE Transactions on Computers*, c-28,11(1979), pages 807–810.

[GLL82] Gupta, U., Lee, D. and Leung, J. Y.-T., Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12, pages 459–467.

[Gol80] Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[GTR+00] Girke, T., Todd, J., Ruuska, S., White, J., Benning, C. and Ohlrogge, J., Microarray analysis of developing arabidopsis seeds. *Plant Physiology*, 124, pages 1570–1581.

[Gus97] Gusfield, D., *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[HFS+94] Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M. and Schuster, P., Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, 125, pages 167–188.

[HMJ+01] Hughes, T. R., Mao, M., Jones, A. R., Burchard, J., Marton, M. J., Shannon, K. W., Lefkowitz, S. M., Ziman, M., Schelter, J. M., Meyer, M. R., Kobayashi, S., Davis, C., Dai, H., He, Y. D., Stephaniants1, S. B., Cavet, G., L.Walker, W., West, A., Coffey, E., Shoemaker, D. D., Stoughton, R., Blanchard, A. P., Friend, S. H. and Linsley, P. S., Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nature Biotechnology*, 19, pages 342–347.

[HSPB04] Hornshøja, H., Stengaard, H., Panitz, F. and Bendixen, C., SEPON, a selection and evaluation pipeline for OligoNucleotides based on ESTs with a non-target T_m algorithm for reducing cross-hybridization in microarray gene expression experiments. *Bioinformatics*, 20,3(2004), pages 428–429.

[Hyy01] Hyyrö, H., On using two-phase filtering in indexed approximate string matching with application to searching unique oligonucleotides. *8th International Symposium on String Processing and information Retrieval (SPIRE 2001)*, 2001, pages 84–95.

[JU91] Jokinen, P. and Ukkonen, E., Two algorithms for approximate matching in static strings. *Proceedings of MFCS'91, Lecture Notes in Computer Science 520*, 1991, pages 240–248.

[KAK+02] Kivioja, T., Arvas, M., Kataja, K., Penttilä, M., Söderlund, H. and Ukkonen, E., Assigning probes into a small number of pools separable by electrophoresis. *Bioinformatics*, 18,Suppl. 1, ISMB 2002 special issue(2002), pages S199–206.

[KCO+01] Kurz, S., Choudhuri, J. V., Ohlebusch, E., Schleiermacher, C., Stoye, J. and Giegerich, R., Reputer: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29,22(2001), pages 4633–4642.

[Kei92] Keil, J. M., On the complexity of scheduling tasks with discrete starting times. *Operations Research Letters*, 12, pages 293–295.

[KJS+00] Kane, M. D., Jatkoe, T. A., Stumph, G. R., Lu, J., Thomas, J. D. and Madore, S. J., Assessment of sensitivity and specificity of oligonucleotide (50mer) microarrays. *Nucleic Acids Research*, 28,22(2000), pages 4552–4557.

[KKM03] Krause, A., Kräuner, M. and Meier, H., Accurate method for fast design of diagnostic oligonucleotide probe sets for DNA microarrays. *Proceedings of International Parallel and Distributed Processing Symposium (HiCOMB 2003 workshop, `http://www.hicomb.org/proceedings.html`)*, 2003.

[KOS+00] Kurz, S., Ohlebusch, E., Schleiermacher, C., Stoye, J. and Giegerich, R., Computation and visualization of degenerate repeats in complete genomes. *Proceedings of 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000)*, 2000, pages 228–238.

[KS02] Kaderali, L. and Schliep, A., Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics*, 18,10(2002), pages 1340–1349.

[KSA+04] Kataja, K., Satokari, R. M., Arvas, M., Takkinen, K. and Söderlund, H., A highly sensitive and multiplexed transcript analysis using solution hybridization and affinity capture. Manuscript in preparation, 2004.

[L+01] Lander, E. S. et al., Initial sequencing and analysis of the human genome. *Nature*, 409, pages 860–921.

[LBG03] Luebke, K. J., Balog, R. P. and Garner, H. R., Prioritized selection of oligodeoxyribonucleotide probes for efficient hybridization to RNA transcripts. *Nucleic Acids Research*, 31,2(2003), pages 750–758.

[LDB+96] Lockhart, D. J., Dong, H., Byrne, M. C., Follettie, M. T., Gallo, M. V., Chee, M. S., Mittmann, M., Wang, C., Kobayashi, M., Horton, H. and Brown, E. L., Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14, pages 1675–1680.

[Li90] Li, M., Towards a DNA sequencing theory. *Proceedings of 31st IEEE Annual Symposium on Foundations of Computer Science (FOCS 1990)*, 1990, pages 125–134.

[LS01] Li, F. and Stormo, G. D., Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics*, 17,11(2001), pages 1067–1076.

[LWY02] Lipson, D., Webb, P. and Yakhini, Z., Designing specific oligonucleotide probes for the entire S. cerevisiae transcriptome. *Proceedings of WABI'2002, Lecture Notes in Computer Science 2452*, 2002, pages 491–505.

[MBF+99] Mathews, D. H., Burkard, M. E., Freier, S. M., Wyatt, J. R. and Turner, D. H., Predicting oligonucleotide affinity to nucleic acid targets. *RNA*, 5,11(1999), pages 1458–1469.

[MSG02] Mrowka, R., Schuchhardt, J. and Gille, C., Oligodb - interactive design of oligo DNA for transcription profiling of human genes. *Bioinformatics*, 18,12(2002), pages 1686–1687.

[MSN+03] Matveeva, O. V., Shabalina, S. A., Nemtsov, V. A., Tsodikov, A. D., Gesteland, R. F. and Atkins, J. F., Thermodynamic calculations and statistical correlations for oligo-probes design. *Nucleic Acids Research*, 31,14(2003), pages 4211–4217.

[MSZT99] Mathews, D. H., Sabina, J., Zuker, M. and Turner, D. H., Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288,5(1999), pages 911–940.

[Mye99] Myers, G., A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM*, 46,3(1999), pages 395–415.

[NH82] Nakajima, K. and Hakimi, S. L., Complexity results for scheduling tasks with discrete starting times. *Journal of algorithms*, 3, pages 344–361.

[Nov01] Novère, N. L., MELTING, computing the melting temperature of nucleic acid duplex. *Bioinformatics*, 17,12(2001), pages 1226–1227.

[NS97] Nicodème, P. and Steyaert, J.-M., Selecting optimal oligonucleotide primers for multiplex PCR. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology (ISMB'97)*, 1997, pages 210–213.

[NWK03] Nielsen, H. B., Wernersson, R. and Knudsen, S., Design of oligonucleotides for microarrays and perspectives for design of multi-transcriptome arrays. *Nucleic Acids Research*, 31,13(2003), pages 3497–3500.

[QPJ+01] Qin, L., Prins, P., Jones, J. T., Popeijus, H., Smant, G., Bakker, J. and Helder, J., GenEST, a powerful bidirectional link between cDNA sequence data and gene expression profiles generated by cDNA-AFLP. *Nucleic Acids Research*, 29,7(2001), pages 1616–1622.

[Rah03a] Rahmann, S., Fast and sensitive probe selection for DNA chips using jumps in matching statistics. *Proceedings of the 2003 IEEE Bioinformatics Conference (CSB 2003)*, 2003, pages 57–64.

[Rah03b] Rahmann, S., Fast large scale oligonucleotide selection using the longest common factor approach. *Journal of Bioinformatics and Computational Biology*, 1,2(2003), pages 343–361.

[RdPR03] Rombauts, S., de Peer, Y. V. and Rouzé, P., AFLPinSilico, simulating AFLP fingerprints. *Bioinformatics*, 19,6(2003), pages 776–777.

[RKS+04] Rautio, J. J., Kataja, K., Satokari, R., Penttilä, M., Söderlund, H. and Saloheimo, M., Transcriptional analysis using solution hybridization with capillary electrophoresis-detectable oligonucleotide pools. Manuscript in preparation, 2004.

[RS98] Rozen, S. and Skaletsky, H. J., Primer3, Code available at `http://www-genome.wi.mit.edu/genome_software/other/primer3.html`, 1996, 1997, 1998.

[RZG03] Rouillard, J.-M., Zuker, M. and Gulari, E., OligoArray: design of oligonucleotide probes for DNA probes using a thermodynamic approach. *Nucleic Acids Research*, 31,12(2003), pages 3057–3062.

[San98] SantaLucia, J., A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Sciences*, 95, pages 1460–1465.

[SBB+02] Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D. and Birney, E., The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12,10(2002), pages 1611–1618.

[SBDY04] Sharan, R., Ben-Dor, A. and Yakhini, Z., Multiplexing schemes for generic SNP genotyping assays. *Proceedings of 2004 Pacific Symposium on Biocomputing (PSB 2004)*, 2004, pages 140–151.

[Sch03] Schena, M., *Microarray Analysis*. John Wiley & Sons, 2003.

[SIP+03] Söderlund, H., Ilmén, M., Paloheimo, M., Kataja, K. and Takkinen, K., A method and test kit for quantitative and/or comparative assessment of variations in polynucleotide amounts in cell or tissue samples, Finnish pat. FI20010041, 2003.

[SKS] Satokari, R. M., Kataja, K. and Söderlund, H., Multiplexed quantification of bacterial 16s rRNA by solution hybridization with oligonucleotide probes and affinity capture. in press.

[SL03] Sung, W.-K. and Lee, W.-H., Fast and accurate probe selection algorithm for large genomes. *Proceedings of the 2003 IEEE Bioinformatics Conference (CSB 2003)*, 2003, pages 65–74.

[SM97] Setubal, J. and Meinadis, J., *Introduction to Computational Biology*. PWS Publishing Company, 1997.

[SMS99] Southern, E., Mir, K. and Shchepinov, M., Molecular interactions on microarrays. *Nature Genetics*, 21,Suppl. 1(1999), pages 5–9.

[Spi98] Spieksma, F. C. R., Approximating an interval scheduling problem. *Proceedings of APPROX'98, Lecture Notes in Computer Science 1444*, 1998, pages 169–180.

[SSDB95] Schena, M., Shalon, D., Davis, R. W. and Brown, P. O., Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270,5235(1995), pages 467–470.

[ST97] SantaLucia, J. and Turner, D. H., Measuring the thermodynamics of RNA secondary structure formation. *Biopolymers*, 44,3(1997), pages 309–319.

[SW81] Smith, T. F. and Waterman, M. S., Identification of common molecular subsequences. *Journal of Molecular Biology*, 147,1(1981), pages 195–197.

[TDS+03] Thareau, V., Déhais, P., Serizet, C., Hilson, P., Rouzé, P. and Aubourg, S., Automatic design of gene-specific sequence tags for genome-wide functional studies. *Bioinformatics*, 19, pages 2191–2198.

[TMN+02] Tobler, J. B., Molla, M. N., Nuwaysir, E. F., Green, R. D. and Shavlik, J. W., Evaluating machine learning approaches for aiding probe selection for gene-expression arrays. *Bioinformatics*, 18,Suppl. 1, ISMB 2002 special issue(2002), pages S164–S171.

[Vaz01] Vazirani, V. V., *Approximation Algorithms*. Springer, 2001.

[VHB+95] Vos, P., Hogers, R., Bleeker, M., Reijans, M., van de Lee, T., Hornes, M., Frijters, A., Pot, J., Peleman, J., Kuiper, M. and et al., AFLP: a new technique for DNA fingerprinting. *Nucleic Acids Research*, 23,21(1995), pages 4407–4414.

[vHJH98] van Holde, K. E., Johnson, W. C. and Ho, P. S., *Principles of Physical Biochemistry*. Prentice Hall, 1998.

[WKJ+02] Wren, J. D., Kulkarni, A., Joslin, J., Butow, R. A. and Garner, H. R., Cross-hybridization on PCR-spotted microarrays. *IEEE Engineering in Medicine and Biology*, 21, pages 71–75.

[WS03] Wang, X. and Seed, B., Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics*, 19,7(2003), pages 796–802.

[XBD+01] Xu, W., Bak, S., Decker, A., Paquette, S. M., Feyereisen, R. and Galbraith, D. W., Microarray-based analysis of gene expression in very large gene families: the cytochrome P450 gene superfamily of Arabidopsis thaliana. *Gene*, 272, pages 61–74.

[XLW+02] Xu, D., Li, G., Wu, L., Zhou, J. and Xu, Y., PRIMEGENS: robust and efficient design of gene-specific probes for microarray analysis. *Bioinformatics*, 18,11(2002), pages 1432–1437.

[YWR00] Yakhini, Z., Webb, P. G. and Roth, R. M., Partitioning of polymorphic DNAs, US Patent 6,074,831, 2000.

[ZCJL03] Zheng, J., Close, T., Jiang, T. and Lonardi, S., Efficient selection of unique and popular oligos for large EST databases. *Proceeding of 14th Annual Symposium on Combinatorial Pattern Matching (CPM 2003), Springer-Verlag Lecture Notes in Computer Science 2676*, 2003, pages 384–401.

[Zuk03] Zuker, M., Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31,13(2003), pages 3406–3415.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1995-1 P. Myllymäki: Mapping Bayesian networks to stochastic neural networks: a foundation for hybrid Bayesian-neural systems. 93 pp. (Ph.D. thesis).

A-1996-1 R. Kaivola: Equivalences, preorders and compositional verification for linear time temporal logic and concurrent systems. 185 pp. (Ph.D. thesis).

A-1996-2 T. Elomaa: Tools and techniques for decision tree learning. 140 pp. (Ph.D. thesis).

A-1996-3 J. Tarhio & M. Tienari (eds.): Computer Science at the University of Helsinki 1996. 89 pp.

A-1996-4 H. Ahonen: Generating grammars for structured documents using grammatical inference methods. 107 pp. (Ph.D. thesis).

A-1996-5 H. Toivonen: Discovery of frequent patterns in large data collections. 116 pp. (Ph.D. thesis).

A-1997-1 H. Tirri: Plausible prediction by Bayesian inference. 158 pp. (Ph.D. thesis).

A-1997-2 G. Lindén: Structured document transformations. 122 pp. (Ph.D. thesis).

A-1997-3 M. Nykänen: Querying string databases with modal logic. 150 pp. (Ph.D. thesis).

A-1997-4 E. Sutinen, J. Tarhio, S.-P. Lahtinen, A.-P. Tuovinen, E. Rautama & V. Meisalo: Eliot – an algorithm animation environment. 49 pp.

A-1998-1 G. Lindén & M. Tienari (eds.): Computer Science at the University of Helsinki 1998. 112 pp.

A-1998-2 L. Kutvonen: Trading services in open distributed environments. 231 + 6 pp. (Ph.D. thesis).

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)