# Accurate method for fast design of diagnostic oligonucleotide probe sets for DNA microarrays

Andreas Krause, Markus Kräutner, Harald Meier*

Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR)
Institut für Informatik, Technische Universität München,
Boltzmannstrasse 3, 85748 Garching, Germany, *email: meierh@in.tum.de
phone: +49 (89) 289-19476, fax: +49 (89) 289-17662

## Abstract

*We present a method for the automatic generation of oligonucleotide probe sets for DNA microarrays. This approach is well suited particularly for specificity evaluation of designed probes in large data sets. Algorithms for probe preselection, hybridization prediction and probe selection are presented. Combinatorial techniques are introduced for the selection of probe sets of high differentiation capability even from sequence databases of homologous conserved genes. These techniques include the automatic generation of group specific probes and the design of exluding probes. A basic prototype was implemented including a shared memory parallelization and distribution. The principal applicability of our method to a database of very conserved sequence data was shown and the run-time performance estimated.*

## 1. Introduction

Oligonucleotide microarrays are currently used in gene expression studies [7] as well as for diagnostic purposes, e.g. the identification of microorganisms in clinical, food and environmental samples [2, 12, 20].

Apart from engineering aspects of microarray technologies, the main problem is to determine suitable sets of oligonucleotide probes. These probe sets should reliably detect and differentiate target sequences of interest (rather the respective genes, gene transcripts or organisms). Large databases have, therefore, to be analyzed efficiently.

Algorithms and their implementations have been published, which solve these problems automatically [10, 15, 16]. The resulting programs work well for data sets that enable the finding of sequence specific probes due to a high variability within the analyzed target sequences. None of these algorithms and their implementations, however, solve the problems that occur when sequence databases of homologous genes need to be analyzed.

This problem occurs for instance, when a microarray is designed for the identification of organisms on the basis of sequence variabilities of conserved genes, e.g. the ribosomal ribonucleic acid sequences. In these cases additional functionalities of the chip designing program are required to fulfill the identification demands of the DNA microarray. If it is impossible to design sequence specific probes due to high similarities, it could be advantageous to determine probes that are specific for groups of closely related sequences. In some cases it could even be helpful to select probes that detect target sequences as well as some non target sequences. These probes have identification value when applied in combination with probes that react specifically with the respective non-target sequences, so called excluding or negative probes. In this paper we describe an algorithm and its parallel implementation that meets the requirements mentioned above and is, therefore, well suited for the fast design of oligonucleotide probe sets from nucleic acid sequence databases of both, variable and highly conserved genes.

## 2. Problem statement

Probe design is a combinatorial optimization problem, where sensitivity and specificity must be optimized while several qualitative criteria, e.g. guanine/cytosine (G/C) content and melting temperature range of the probes, have to be maintained. Maximizing *specificity* and *sensitivity* are often conflicting goals. In practice it is often impossible to design specific probes for each selected sequence, especially when dealing with highly conserved data. To nevertheless obtain

a feasible working solution, one has to make compromises. Thus we extended the typical probe design problem 2.1 by another problem 2.2 which allows to relax criteria if the data is highly conserved.

**Problem 2.1 (Positive probes).** *Given a selected subset $S_1$ in a database $S_0$ of sequences, find for each sequence $s$ in $S_1$ at least one* positive probe *$p$ which hybridizes within $S_1$ only with $s$; it may however cross-hybridize with some sequences $B$, where $B \subseteq S_2 := S_0 \setminus S_1$ if this can't be avoided. High specificity means that the number of non-target matches is minimized, while high sensitivity means that a maximum number of selected target sequences is covered.*

**Problem 2.2 (Negative probes).** *Given the positive probes identified in* Problem 2.1*, determine as few as possible* negative probes *which together hybridize with all sequences in $B$ but with none in $S_1$. High specificity means in this context, that* no *sequences in $S_1$ may cross-hybridize with any negative probe, while high sensitivity means, that a maximum number of sequences in $B$ must be covered.*

### 2.1. Probe design constraints

The following constraints are imposed onto the probe selection process:

- Minimum and maximum length of the probes.
- The melting temperature of the probe-target hybrids must not differ more than a maximum value. This can be accomplished e.g. by specifying a range of percentage of G/C content.
- Probes should not contain self complementary regions that are longer than four sequential nucleotides.
- There must be a minimum difference in the melting temperatures of target and non-target sequences. This can be accomplished by ensuring a minimum number of mismatches (stronger than G-U/T) to all other sequences.

### 2.2. System constraints

For practical reasons, other constraints are imposed onto the development of such a system. The most important are:

- *Execution time*: A probe set for a chip must be computable in at most a few hours.
- *Usability*: The software must be able to read different standardized sequence and alignment data formats; a user interface should allow the selection of sequences and provide methods to visualize the specificity of designed probe sets.

## 3. Algorithmics

Our approach on the generation of oligonucleotide probe sets comprises three steps (as shown in Figure 1). Firstly, a pool of suitable probe candidates is generated (s. 3.1). In the next step the hybridization behaviour of these probes is predicted (s. 3.2). Based on the results the probes are selected (s. 3.3).

### 3.1. Probe preselection

All possible probe candidates are generated. For performance reasons these should be as few as possible, while all optimal candidates must be kept.

#### 3.1.1 Attributizing a suffix tree

To find suitable probe candidates, a generalized suffix tree is constructed from the sequences in set $S_1$ in linear time [8]. The suffix tree is then traversed and extensible filter chains determine if branches of unsuitable probes are cut off. The current implementation utilizes filters for *probe length*, *G-C content* and *self-complementarity*. The latter uses a suffix tree to check, if a probe contains two disjoint substrings of a certain minimal length which are the Watson-Crick complements of each other. The remaining probes are uniquely inserted into hashtables. The number of probes that have to be analyzed in the further probe design process are, thereby, reduced dramatically.

#### 3.1.2 Further preselection by removing unspecific probes

A probe match algorithm is applied on $S_1$ to determine probes which are found in many sequences of different groups - usually those stemming from highly conserved regions. These are not suitable as specific probes and, therefore, removed from the hashtable of possible probes. After this step typically only a small percentage of the original probe pool is left, which helps to keep the hybridization prediction matrix sparse.

### 3.2. Hybridization prediction

This is the most time-consuming part, as all generated probe sequences are searched inexactly in the database containing all sequences. This results in a hybridization matrix which predicts melting temperatures for all probe - target pairings. Our approach uses an *intelligent hashing* strategy to solve this problem efficiently.

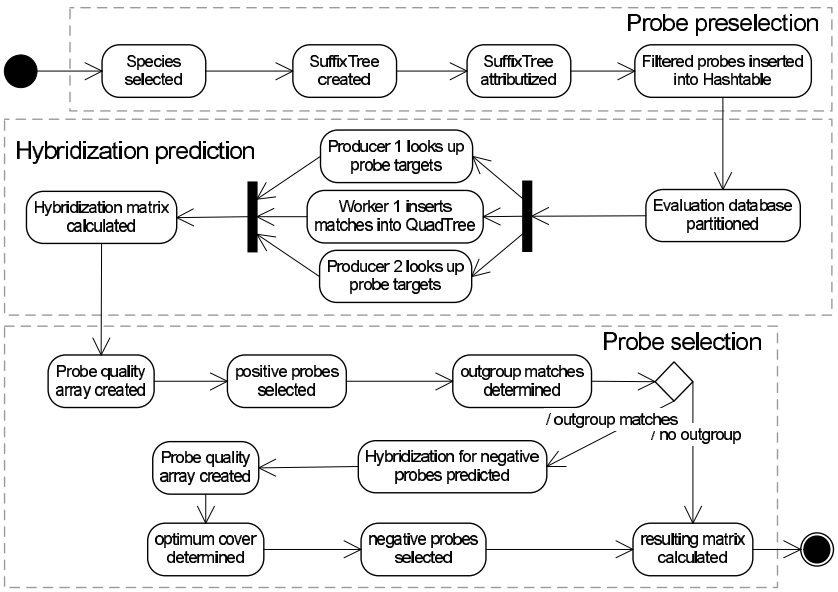


**Figure 1. Stepwise decomposition of the probe design algorithms. The program flow is visualized in UML statechart notation. The hybridization prediction part visualizes the worker-producer architecture for multithreaded computation, described in 5.1.**

#### 3.2.1 Basic idea

All probe candidates of equal length are stored in several hashtables. A target frame of the same length is moved over target and nontarget sequences. At each place, hash values are calculated and looked up in the hashtables. If a hit occurs, the melting temperature is predicted, and the probe - target interaction is stored in the hybridization matrix. For performance reasons, the computation of the hash values should take constant time. The hash functions must be chosen in a way that makes this inexact matching possible (this is meant by the term intelligent hashing), which is accomplished by a coding theoretical approach as described in 4.2.1.

#### 3.2.2 Design of the codes

Currently we use one code to guarantee general mismatches, one to handle weak G-U/T mismatch pairings, and one to guarantee the detection of single insertions and deletions, combined with any number of weak mismatches. Thus, if our algorithm interprets a probe-target pairing as non-matching, then this pairing has (for suitably chosen parameters)

- at least four central mismatches of which at least two are no G-U/T pairings.
- one insertion or deletion and at least one non G-U/T central mismatch.

The codes have been designed to guarantee experimentally identified criteria of unspecificity.

**Empirical simulation** To test the effectiveness of these codes, one million random probe-target pairings have been generated, in which the above criteria are minimally failed (thus only worst-case constellations are examined). Four mismatches were chosen or one insertion or deletion plus one strong central mismatch. Tab. 1 shows the results of the experiment.

| $\Delta T_m$ °C | 0-4 | 5-9 | 10-14 | 15-19 |
|---|---|---|---|---|
| cases out of $10^6$ | 7 | 603 | 16207 | 46576 |

**Tab. 1** *Melting temperature difference (rounded downwards) for worst case hash misses*

This experiment shows that in more than 93 % of the *critical* cases, a melting temperature difference of more than 20 °C can be expected, whereas a difference of below 10 °C is very improbable compared to sequencing and experimental errors.

All calculated checksums utilize only a portion of the original probe text bits. E.g., to ignore border mismatches, one can exclude the probes' border regions from the checksum generation. This is interesting in terms of hybridization thermodynamics. The checksum calculation functions should be chosen to be injective, as the size of the function's range directly determines the number of expected hashtable collisions to which running time is proportional. If $m - 1$

mismatches must be detected, which should not lie on the $b$ border nucleotides of a probe of length $l$, the number $k$ of available key bits can be calculated by

$$k = 2 \left\lfloor \frac{l - 2b}{m} \right\rfloor$$

**Complexity** Since the number of possible probe candidates – entries in the hashtable – is roughly $\|S_1\|$, and each position of the database $S_0$ must be checked, the expected complexity for the hybridization prediction algorithm is

$$ops = C \frac{\alpha \|S_0\| \|S_1\|}{2^k}$$

where $C$ is a small implementation constant and $\alpha \approx 0.1$ is the percentage of probes kept after preselection. If single insertions and deletions should be detected, then $m$ must be equal to $4$.

#### 3.2.3 Hashtable lookup and local thermodynamical alignments

The interesting point in intelligent hashing is to allow "inexact equality". This is accomplished by the most time-consuming subroutine, the probe comparison operator. This routine constructs a local thermodynamical alignment as described in [10], but only the inner three bands of the dynamical programming matrix are computed, since only single insertions or deletions must be detected. The comparison routine computes the expected nearest neighbour melting temperature; for details about precision of this thermodynamical model see [18]. This local alignment can be constructed in time $\mathcal{O}(l)$ for probes of length $l$. While this characterwise comparison and calculation gives another factor - linear with the probe length - the coding theoretical approach does reduce the number of necessary comparisons as far as possible. Another advantage of using hashtables for searching is that unspecific probes can be simply removed if the number of matches surpasses a predefined threshold.

#### 3.2.4 Storage of the sparse hybridization matrix

To store the hybridization matrix, a quadruple tree datastructure is used; it allows efficient enumeration of row or column entries and selected subsections can easily be "sliced" out. Thus when the chip has been designed, only the subregion concerning all involved probe-target interactions can be saved to disk.

### 3.3. Probe selection

Using this matrix, *Probe-selection* generates a solution to the positive probe problem 2.1. Subsequently, negative probes are calculated, which solve problem 2.2.

#### 3.3.1 Selecting positive probes

Once the hybridization matrix has been constructed, it is used to select the probes and solve problem 2.1 (positive probes). Checking each possible subset of probes is too expensive. Therefore, an efficient algorithm is required for the probe selection process. While other approaches, such as [10, 13, 15, 16] leave the user merely with a sorted result list of probes at this point, our approach, in addition to this possibility, utilizes the combinatorial ideas presented in 4.1 to choose an optimal probe covering even for conserved data.

Our algorithm works in the following way: For each probe candidate, the number $g$ of matches in $S_1$ and the number $b$ of matches in $S_2$ is counted and the highest predicted melting temperature $t$ within $S_1$ is identified. Probes for which $g$ or $b$ is too large are removed. The probes are then sorted, with respect to $g$, $b$ and $t$.

Afterwards, a modified Depth First Search is conducted on the selections of possible probe subsets. In a descending order, probe $p_n$ is chosen iff it allows to double the number of potential binary chip experiment results, which means it is independent from the current probe selection (see 4.14). If, for a predefined number of steps no new probe can be chosen, the branch is truncated. After a predefined maximum number of iterations, this selection process stops. This algorithm can not guarantee the optimal selection in terms of coverage, but it performs reasonably well in practice. It also guarantees to choose all specific probes if any were found.

#### 3.3.2 Covering the cross-hybridizing sequences

For the set $B$ of cross-hybridizing sequences a new set of probe candidates is created by the probe preselection procedure described above. Subsequently, hybridization prediction is conducted on $B \cup S_1$. All probes which show hybridization within $S_1$ are removed. Afterwards, the remaining probes are sorted by the number of hits in $B$, descendingly. Successively, probes are chosen, which cover as many yet uncovered elements of $B$ as possible in each step. This procedure guarantees an optimal selection of negative probes, in terms of coverage and probe number usage. Of course, the negative probes selected this way can possibly cross-hybridize with other sequences in $S_2 \setminus B$, but the minimization of this quantity would result in a computationally expensive iterative probe-match process; it is, however, not necessary to optimize the negative probes this way. A microarray hybridization experiment that results in a reaction of both, positive and negative probes has to be interpreted very cautiously, as it does not precisely indicate the presence of the target sequence.

## 4. Mathematical aspects

### 4.1. Combinatorial criteria for probe selection

In this section we present some mathematical theory to formalize the process of probe selection and hybridization prediction. We develop the notion of independent probe sets which maximize the number of binary chip result interpretations. This leads to a probe set for optimal identification and differentiation of selected target sequences.

#### 4.1.1 Basic definitions

**Definition 4.1.** We model nucleic acid sequences as strings over a finite *alphabet* $\Sigma := \{A, C, G, U/T\}$. These sequences are taken from a *universe* $\mathcal{U} := \Sigma^+$. The *database* $S_0$ is a finite subset $S_0 \subset \mathcal{U}$, as are the *selected sequences* $S_1 \subset S_0$. The complement of $S_1$ in $S_0$ is $S_2 := S_0 \setminus S_1$. A *target region* of a sequence $s \in \mathcal{U}, s = (s_1, \ldots, s_n)$ is a substring $s_{k,l} = (s_k, s_{k+1}, \ldots, s_{k+l-1})$ of $s$ which starts at position $k$ and has length $l$.

$P^l := \Sigma^l$ denotes probes of length $l$. $P_0$ is the set of all probes. For $p \in P_0, p = (p_1, p_2, \ldots, p_n)$ the *reverse complement* is $\bar{p} := (\bar{p}_n, \bar{p}_{n-1}, \ldots, \bar{p}_1)$ for $\bar{A} := U/T, \bar{C} := G, \bar{G} := C, \bar{U}/\bar{T} := A$.

The *length* of a probe $p \in P_0$ is $\|p\| := l$ iff $p \in P^l$; this definition is extended to finite sets: $\|S\| := \sum_{s \in S} \|s\|$

*Remark* 4.2. For defining the distance $d : P^l \times P^l \to \mathbb{R}_0^+$ between two probes of equal length we utilize a metrics, like the *Hamming distance* or the *Levenshtein distance*. For definitions and efficient calculation see [19], pp. 94.

**Definition 4.3.** The distance function for probes of length $l$ and sequences of arbitrary length is:

$$d^l : P^l \times \mathcal{U} \to \mathbb{R}_0^+$$
$$d^l(p, s) = \min_{i=1..\|s\|-l+1} \{d(p, \bar{s}_{i,l})\}, \|s\| \geq l$$

The canonical extension to sets $S \subset S_0$ of sequences is:

$$d^l(p, S) = \min_{s \in S} \{d^l(p, s)\}$$

**Definition 4.4.** The *range* of a probe $p \in P_0$ in $S \subset S_0$ is

$$S(p, \Delta) := \{s \in S : d^{\|p\|}(p, s) \leq \Delta\}$$
$$\text{for } S \subset \mathcal{U} \text{ finite }, \Delta \in \mathbb{R}_0^+$$

*Remark* 4.5. In the following, the value of $\Delta$ is assumed to be fixed, therefore we write $S(p) := S(p, \Delta)$.

*Remark* 4.6. Sometimes it is impossible to find specific probes for each selected sequence in $S_1$, e.g. if $S_1$ contains families of very similar sequences. In this case we want to select probes only if they maximize the number of potential binary chip experiment results. This notion will be formalized next.

#### 4.1.2 Binary chip experiment results

**Lemma 4.7.** *Let $P \subset P_0$, $|P| = k < \infty$ be the set of probes which are placed on a chip, and*

$$h : \{1..k\} \to \{0, 1\}$$
$$h(j) = \begin{cases} 1 & \text{if probe } p_j \text{ has hybridized,} \\ 0 & \text{otherwise} \end{cases}$$

*be the hybridization result function. Then*

$$C(P, h) = \left( \bigcup_{j \in h^{-1}(1)} S_0(p_j) \right) \setminus \left( \bigcup_{j \in h^{-1}(0)} S_0(p_j) \right)$$

*gives the set of sequences identified by the chip experiment. The number*

$$c(P) := |\{C(P, h) | h : \{1..|P|\} \to \{0, 1\}\}|$$

*counts the number of different experimental results.*

**Proof:** If a probe matches, any subset of sequences of the probe's range can be in the result, this is expressed in the first union. If a probe does not match, none of the sequences in the probe's range can be in the result, which is guaranteed by subtracting the second union. In the definition of $c(P)$ only unique chip results are counted. ■

*Remark* 4.8. Since we want to select probes which maximize the number of potential binary chip experiment results, we need a measure for the expression power of the addition of a new probe to a given probe set.

**Definition 4.9.** Let $P \subset P_0$ finite, $p \in P_0 \setminus P$, define $c(\emptyset) := 1$ and

$$\mu(P, p) := \frac{c(P \cup \{p\})}{c(P)} \in \mathbb{Q}$$

the *expression power of $p$ with respect to $P$*

**Lemma 4.10.** *For $P_n = \{p_1, \ldots, p_n\}$ it holds that*

$$c(P_n) = \prod_{i=1}^{n} \mu \left( \bigcup_{j=1}^{i-1} \{p_j\}, p_i \right)$$

**Proof:** This is a telescope product - only the enumerator of the last and the denominator of the first term remain. ■

**Lemma 4.11.** *Define*

$$C'(P, h) = \bigcup_{j \in h^{-1}(1)} S_0(p_j)$$
$$c'(P) := |\{C'(P, h) | h : \{1..|P|\} \to \{0, 1\}\}|$$

*Then it holds, that $c'(P) = c(P)$*

**Proof:** We have:
$C(P,h) =$
$\left(\bigcup_{j\in h^{-1}(1)} S_0(p_j)\right) \setminus \left(\bigcup_{j\in h^{-1}(0)} S_0(p_j)\right) =$
$\left(\left(\bigcup_{j\in h^{-1}(1)} S_0(p_j)\right) \cup \left(\bigcup_{j\in h^{-1}(0)} S_0(p_j)\right)\right) \setminus$
$\left(\bigcup_{j\in h^{-1}(0)} S_0(p_j)\right) = \left(\bigcup_{p\in P} S_0(p)\right) \setminus$
$\left(\bigcup_{j\in \bar{h}^{-1}(1)} S_0(p_j)\right) = \Omega \setminus C'(P,\bar{h})$
where $\Omega = \left(\bigcup_{p\in P} S_0(p)\right)$ independent from $h$, $\bar{h}(j) := 1 - h(j)$ for every $j = 1..n$. As $\bar{\cdot}$ is a permutation of all $h : \{1..n\} \to \{0,1\}$, and $C'(P,\bar{h}_1) \neq C'(P,\bar{h}_2)$ iff $C(P,h_1) \neq C(P,h_2)$ the claim holds. ■

**Lemma 4.12.** *For finite $P \subset P_0$ and $p \in P_0 \setminus P$ it holds that*

$$1 \leq \mu(P,p) \leq 2$$

**Proof:** This is an immediate consequence of the representation from Lemma 4.11. ■

*Remark* 4.13. Since we are only interested in probes with maximal expression power, we are looking for necessary and sufficient conditions for $\mu(P,p) = 2$. Therefore we need the notion of *independency* of probes.

### 4.1.3 Independent probe selections maximize the expression power

**Definition 4.14.** Let $P \subset P_0$. $P$ is called *independent* (w.r.t. $S$) iff for all $p \in P$, some $s \in S(p)$ exists with $s \notin S(q)$ for all $q \in P \setminus \{p\}$. Otherwise, $P$ is called $dependent$.
$p \in P_0 \setminus P$ is called *independent* of $P$ (w.r.t. $S$) iff $P \cup \{p\}$ is independent.

*Remark* 4.15. The above definition states that the empty set is independent w.r.t. any set $S$.

**Theorem 4.16.** *Let $P$ be independent, finite subset of $P_0$. Then*

$$c(P) = 2^{|P|}$$

**Proof:** Per definition, for each $p_k \in P$ there exists some $s_{p_k} \in \left(S(p_k) \setminus \left(\bigcup_{p\in P\setminus\{p_k\}} S(p)\right)\right)$, thus $s_{p_k}$ in $C'(P,h)$ iff $h(k) = 1$. Since there are $2^{|P|}$ possible $h$, it holds that $c(P) = 2^{|P|}$ ■

**Theorem 4.17.** *Let $P$ be independent, finite subset of $P_0$ and $p \in P_0 \setminus P$. Then the following statements are equivalent:*
*(i) $p$ independent from $P$*
*(ii) $\mu(P,p) = 2$*

**Proof:** $\boxed{(i) \Rightarrow (ii)}$ From the definition it is clear that if $P$ is independent and $p$ is independent from $P$, then $P \cup \{p\}$ is independent. Thus Theorem 4.16 shows that $c(P \cup \{p\}) = 2^{|P|+1}$ and $c(P) = 2^{|P|}$, and (ii) follows.
$\boxed{(ii) \Rightarrow (i)}$ Suppose, $p$ is dependent from $P$. Then for some $p_j \in P \cup \{p\}$ it holds, that $S(p_j) \subset \left(\bigcup_{p_j\in P\setminus\{p_k\}} S(p_j)\right)$ Thus $C'(P \cup \{p_n\}, (1,\ldots,1, \underbrace{0}_{j\text{-th position}}, 1,\ldots,1)) = C'(P \cup \{p_n\}, (1,1,\ldots,1,1))$ and since $P$ was independent, $c(P) = 2^{|P|}$ but $c(P \cup \{p\}) < 2^{|P|+1}$, thus we see that $\mu(P,p) < 2$ and (i) follows. ■

**Corollary 4.18.** *Those sets $P \subset P_0$, for which $c(P) = 2^{|P|}$ holds are exactly the independent probe selections. These can be constructed inductively by adding probes to independent sets while making sure that the resulting sets in each step remain independent.*

**Proof:** Definition 4.14 shows that adding probes to dependent sets will always result in dependent sets. Since the empty set is independent, and any finite set can be inductively constructed by adding one element after another to smaller sets, the claim follows from 4.10, 4.16 and 4.17.
■

## 4.2. Calculating the distance

The calculation of the distance of probes and sequences is a computationally expensive operation. Additionally, since $|P| \approx \|S_1\|$, the amount of distance calculations is $\Omega(\|S_1\| \cdot \|S_0\|)$, which would result in a long computation time. To speed up the process we only calculate the exact distance if this value is small, and therefore the exact value needs to be known to predict the hybridization behaviour. So we have to solve the following problem.

**Problem 4.19.** *Given $s \in S_1$, $P \subset P^l$, $d_{\max} \in \mathbb{N}$, identify $P' \subset P$ with $p \in P'$ iff $d(p,s) \leq d_{\max}$.*

From the definition of the distance one sees that $d(p,s) > d_{\max}$ iff $d(p,\bar{s}_{k,l}) > d_{\max}$ for all suitable $k$. So our problem is solved, if it is possible to efficiently calculate the sets

$$P(s_{k,l}) := \{p \in P : d(\bar{s}_{k,l}, p) \leq d_{\max}\}$$

in time $\varepsilon\|S_1\|$ for a small $\varepsilon < 1$. Introducing up to $d_{\max}$ arbitrary errors into $s_{k,l}$ and for each malformed copy looking up quickly whether it can be found in the set $P$ or not is still too expensive. But the idea of investigating the neighborhood of one probe $p$ can be modified to utilize the fact that the "neigborhoods" of a probe are typically occupied sparsely. Formally, the set $P^l$ of possible probes can be looked at as a $2l$-dimensional hypercube $Q$, in which vertices are marked if the corresponding probe is in the subset

$P$. The idea is now to construct some "continuous" mapping of $Q$ into some structure in which neighborhoods can be enumerated quickly.

#### 4.2.1 Coding theory

Our approach is to calculate checksums for each probe $p$ and the reverse complements of each target $t$ and make sure that at least one checksum is equal if the distance $d(p,t) \leq d_{\max}$. For $k = d_{\max} + 1$ this can be done by calculating the checksums $c_0, \ldots c_{k-1}$ for a probe/target $p \in P^{kl}$, $p = (p_0, p_1, ...p_{kl-1})$ by $c_j := (p_j, p_{k+j}, p_{2k+j}, \ldots, p_{(l-1)k+j})$. It is also possible to construct codes to check for certain types of mismatches, such as G-U/T mismatches, or for insertions and deletions. These checksums can then be used to store all probes in $P$ in hashtables. For each target, the checksums are calculated and looked up in the hashtable. If a hash miss occurs, it is guaranteed that the target looked up has at least $k$ mismatches. This technique utilizes the fact that the neighborhood of probes in $P$ – which would get similar checksums – is occupied sparsely. The calculation of hash values for the targets can be done in constant time for small probes by utilizing the bit-shift operations implemented in hardware.

## 5. Prototype implementation - preliminary test results

A basic prototype of the algorithm presented above has been implemented in C++ relying on the Qt library [21], keeping our future plans in mind, e.g. to develop a comprehensive, modern graphical user interface. Furthermore, the Qt library enables platform independency. Generic programming using templates was employed for basic data structures. A proprietary memory manager was set up to guarantee effective cache utilization by data locality. Configuration and data exchange currently utilizes the standard data format FASTA (a regular expression can be specified for parsing any FASTA format subtype) as well as the ARB data format for raw and aligned data [13] .

### 5.1. Parallelization and distribution

The prototype has been parallelized on an SMP platform. A worker-producer architecture is used to improve efficiency of the hybridization prediction routine. Several producer threads each search distinct parts of the evaluation database, a worker thread writes the search results to the quadruple tree as displayed in Fig. 1.
The chosen algorithms are well suited for distribution: the calculation of rectangular regions of the quadruple tree matrix can be distributed to several machines within a cluster. Results can then be merged.

### 5.2. Preliminary test results

The software prototype was tested with data from the small subunit ribosomal ribonucleic acid (ssu rRNA) database of the ARB project (`ssujun02.arb, www.arb-home.de`). The original dataset was preprocessed before designing oligonucleotide probe sets. Just almost full length sequences (longer than 1400 bases) were kept, while partial sequences were removed. Furthermore, sequence fragments beyond ssu rRNA alignment regions were truncated. The test-ready dataset contains 20.282 ssu rRNA sequences. 69% of them are prokaryotic. Sequence lengths range between 4.179 and 1.401 bases. However, 97% of the sequences are shorter than 2.000 bases and 70% have a length within a range of 1.400 and 1.600 bases.

**Design of an oligonucleotide probe set.** The program was executed after a subset of the data had been selected as target sequences, in our test all sequences of members of the archaeal kingdom *Crenarchaeota* (68) and of the bacterial genera *Deinococcus* (16), *Thermus* (48) and *Meiothermus* (15). All other sequences within the database were considered to be non-target by the program. The parameter *probe length* was set to 22 bases length and *G/C-content* was restricted to the range between 50 and 60 percent. The algorithm firstly tried to determine a maximum independent probe selection, thereby choosing specific probes for each of the target sequences if available. Additionally, it proposed probes such that every sequence was covered at least by one probe. This led to the generation of group specific probes. Some of the group specific probes hit non target sequences. In this case negative probes that allow the discrimination of these non target sequences were added. The designed set of oligonucleotide probes (22 meres) consisted of 101 probes. 55 sequence specific probes were found. 45 group specific probes were determined. All 147 sequences were covered by one or more probes. Three probes were selected that cover in addition to target sequences also four non-target sequences in total. Those non-target sequences could be covered by two negative probes. The designed probe set of this experiment will be published on `wwwbode.cs.tum.edu/~meierh/`.

**Estimation of the Runtime Performance.** For this experiment we used the test-ready dataset (20282 sequences) and subsets of it, containing 1000, 2000, 5000, and 10000 sequences. This corresponds to dataset sizes $\|S_0\|$ of 1.7, 3.4, 8, 17, and 35 million bases (Mb). The software prototype was executed on each of the datasets with identical parameter settings and target sequence selection. Runtime performance was estimated on an Intel Dual Pentium III 933 MHz with 1 GByte main memory in single and dual processor mode. Measured values for total running time and the

running time of the parallelized component are displayed in Tab. 2. These results indicate a performance improvement of up to 59% and 65%, respectively.

| Database size in Mb | 1.7 | 3.4 | 8 | 17 | 35 |
|---|---|---|---|---|---|
| Tot. run. time 1 CPU | 62 | 84 | 168 | 309 | 580 |
| Tot. run. time 2 CPUs | 49 | 61 | 113 | 200 | 364 |
| Parall. part 1 CPU | 46 | 70 | 150 | 284 | 544 |
| Parall. part 2 CPUs | 34 | 47 | 96 | 177 | 329 |

**Tab. 2** *Performance measurements for chip design process selection size* $\|S_1\| = 220$ *Kb, running time in* $[s]$

## 6. Discussion

In this document we have presented an approach to the probe generation problem even for datasets of conserved sequences, such as the small subunit ribosomal ribonucleic acid (ssu RNA) database. The advantages of the system are specifically:

***High performance.*** Expected execution time is linear with the size of the evaluation database $S_0$ and even decreases if longer probes are used. First test runs indicate that the expected high performance could actually be achieved in practice. Within a large database a set of 101 oligonucleotide probes specific for the identification and differentiation of 147 sequences was designed on a common personal computer within six minutes only $Tab.2$.

***Low memory consumption.*** Even for very large evaluation databases (several hundred Mbs) 0.5-1 GByte of main memory is sufficient, when the total size of selected target sequences does not exceed one million bases. Memory consumption only depends on the size of the sequence selection, *not* on the evaluation database size.

***Automatic design of group probes and negative probes*** to work with previously unstructured and even conserved data.

***High quality probe design.*** The usage of local thermodynamical alignments fully exploits the nearest neighbour model as an unspecificity measure. Interfaces to software packages as ARB allow the usage of refined and less error-prone data. Proposed probes lay in the same regions of the target sequences – within a few bases – as probes manually designed by expert microbiologists, even the group probe coverage was similar, see 5.2. It was not within the scope of this paper to present comprehensive probe set design results. Further studies will be performed on different data sets and the quality of designed probe sets for custom-made microarrays will be evaluated within the near future.

In a very small percentage of cases, the coding theoretical approach fails to discern sequences with a small difference in the melting temperature as described in 3.2.2. The greedy calculation of the nonlinear nearest neighbour melting temperature can – in very few cases – differ from the optimal alignment. This may also be a problem, as analyzed in [10]. Despite of this fact, our approach works very well in empirical simulations and testcases, as shown in 5.2. One constraint seems to be the quadratic time complexity when probes for all sequences in the database should be generated; but even then, the complexity factor is very small and distribution or batch-processing as described in 6.1 can be fully exploited to handle even very large datasets.

**Comparison with related work** The *ARB* probe design tool allows to design probes for single sequences or a group of sequences. Its alignment is used to refine the data by identifying sequence errors which would otherwise hinder successful probe design. It is however not suited for large scale probe design. Hybridization prediction is done by creating an index search tree containing all substrings of length at most $k$. The time and space complexity of this approach depends on the database size.

The program described in [15] relies on the least common substring (or least common factor, LCF) of each probe / target pairing. It performs well even on large data sets, but there are currently no results analyzing the LCF as an unspecificity measure. This approach only works well with short probes of about 20-25 bases, and it doesn't have any features to compute probe coverings for highly conserved data - it only calculates specificity statistics for each probe and sorts the resulting probes. The memory consumption of this approach is extremely large and depends on the evaluation database size.

There are other systems involving inexact searching, e.g. [14, 16], but most of them cannot deal efficiently with insertions or deletions, or employ simpler thermodynamics. The approach of [10] provides the most exact thermodynamical evaluation, but its execution time is too slow to work with large databases.

### 6.1. Future work

**Comprehensive tests of the algorithm and the evaluation of the quality of designed probe sets** Though the prototype shows promising properties regarding speed and probe quality, these results have to be verified more closely. In detail, this will require tests with additional databases, e.g. genome databases and comparison of the results with that of other available software packages.

**Graphical User Interface** To improve the general usability, the prototype will be extended by a user interface, which will facilitate the import of arbitrary databases and the selection of sequences. This will help to attract more attention

of molecular and computational biologists and accellerate its use in research institutes as well as in companies developing oligonucleotide microarrays.

**Scalability - Genome databases** The current implementation has been designed to work even with the highly conserved ssu-rRNA sequences. In contrast, within one genome database it is more likely that for each selected gene an unique probe can be found. This reduces the problem of finding negative probes.

The selection of unique probes for every gene within one genome can be accomplished, even in the current implementation. To minimize memory usage and hashtable collisions, at most a few hundred or thousand genes should be selected in one run after which the program is restarted with another selection. The designed probes for these subsets of the data can then be merged. The whole processes will be automatized both for distribution and for simple batch processing. This is possible, because the probe match code ensures that all generated probes are specific within the *whole* database.

## 7. Acknowledgement

The authors thank Karl Fürlinger and Asa MacWilliams very much for carefully reading and improving the paper, Dr. Wolfgang Ludwig for providing the ARB-database, the BMBF (BIOLOG 03F0279F) and the Bayerische Forschungsstiftung (Entwicklung von Oligonukleotid-DNA-Chips) for funding this work.

## References

[1] E. Alm, D. Oerther, N. Larsen, D. Stahl, and L. Raskin. The oligonucleotide probe database. *Appl and Environ Microbiol*, 62(10):3557–3559, 1996.

[2] R. Anthony, T. Brown, and G. French. Rapid diagnosis of bacteremia by universal amplification of 23s ribosomal dna followed by hybridization to an oligonucleotide array. *J Clin Microbiol*, 38:781–788, 2000.

[3] K. Ashelford, A. Weightman, and J. Fry. Primrose: a computer program for generating and estimating the phylogenetic range of 16s rrna oligonucleotide probes and primers in conjunction with the rdp-ii database. *Nucl Acids Res*, 30(15):3481–3489, 2002.

[4] D. Bassett, M. Eisen, and M. Boguski. Gene express informatics – it's all in your mine. *Nature Genetics*, 21:51–55, 1999.

[5] J. Bornemann, M. Chrobak, G. Vedova, A. Figueroa, and T. Jiang. Probe selection algorithms with application in the analysis of microbial communities. *Bioinform*, 17:39–48, 2001.

[6] D. Bowtell. Options available – from start to finish – for obtaining expression data by microarray. *Nature genetics*, 21:25–32, 1999. Supplement.

[7] A. de Saizieu, U. Certa, J. Warrington, C. Gray, W. Keck, and J. Mous. Bacterial transcript imaging by hybridization of total rna to oligonucleotide arrays. *Nature Biotechnology*, 16(1):45–48, 1998.

[8] R. Giegerich and S. Kurtz. From ukkonen to mccreight and weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

[9] K. Hwang. *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, chapter 3. McGraw-Hill, New York, USA, 1993.

[10] L. Kaderali and A. Schliep. Selecting signature oligonucleotides to identify organisms using dna arrays. *Bioinform*, 18(4):1340–1349, 2002.

[11] A. Loy, M. Horn, and M. Wagner. probebase – an online resource for rrna-targeted oligonucleitide probes. *Nucl Acids Res*, 31(1):514–516, 2003.

[12] A. Loy, A. Lehner, N. Lee, J. Adamczyk, H. Meier, J. Ernst, K.-H. Schleifer, and M. Wagner. An oligonucleotide microarray for 16s rrna gene-based detection of all recognized lineages of sulfate-reducing prokaryotes in the environment. *Appl and Environ Microbiol*, 68:5064–5081, 2002.

[13] W. Ludwig, O. Strunk, R. Westram, L. Richter, H. Meier, et al. Arb, a software environment for sequence data. *Nucl Acids Res*. accepted Jan 2003.

[14] A. Pozhitkov and D. Tautz. An algorithm and program for finding sequence specific oligo-nucleitide probes for species identification. *BMC Bioinform*, 3(9):1–7, 2002.

[15] S. Rahmann. Rapid large-scale oligonucleotide selection for microarrays. In *Proceedings of the CSB2002, IEEE Computer Society Bioinformatics Conference*, pages 54–63, Los Alamitos, California, 2002. IEEE Computer Society.

[16] J. Rouillard, C. Herbert, and M. Zuker. Oligoarray: genome-scale oligonucleitide design for microarrays. *Bioinformatics Applications Note*, 18(3):486–487, 2002.

[17] S. Russell and P. Norwig. *Artificial Intelligence: a modern approach*, pages 55–121. Prentice-Hall, New York, USA, 1995.

[18] J. SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide dna nearest-neighbor thermodynamics. *Proc Natl Sci*, 95:1460–1465, 1997.

[19] J. Setubal and J. Meidanis. *Sequence Comparison and Database Search*, pages 47–103. PWS Publishing Company, 1997.

[20] J. Small, D. Call, F. Brockman, T. Straub, and D. Chandler. Direct detection of 16s rrna in soil extracts by using oligonucleotide microarrays. *Appl Environ Microbiol*, 67:4708–4716, 2001.

[21] Trolltech. Qt whitepaper. Technical report, Trolltech, 2002.

[22] M. Zuker, D. Mathews, and D. Turner. Algorithms and thermodynamics for rna secondary structure prediction: A practical guide. *Webpage*, 1999.