

Fast and Sensitive Probe Selection for DNA Chips Using Jumps in Matching Statistics

Sven Rahmann

Dept. of Computational Molecular Biology
Max Planck Institute for Molecular Genetics
Innestraße 73, D-14195 Berlin, Germany
Sven.Rahmann@molgen.mpg.de

and Dept. of Mathematics and Computer Science
Freie Universität Berlin

Abstract

The design of large scale DNA microarrays is a challenging problem. So far, probe selection algorithms must trade the ability to cope with large scale problems for a loss of accuracy in the estimation of probe quality. We present an approach based on jumps in matching statistics that combines the best of both worlds.

This article consists of two parts. The first part is theoretical. We introduce the notion of jumps in matching statistics between two strings and derive their properties. We estimate the frequency of jumps for random strings in a non-uniform Bernoulli model and present a new heuristic argument to find the center of the length distribution of the longest substring that two random strings have in common. The results are generalized to near-perfect matches with a small number of mismatches.

In the second part, we use the concept of jumps to improve the accuracy of the longest common factor approach for probe selection by moving from a string-based to an energy-based specificity measure, while only slightly more than doubling the selection time.

1. Introduction

1.1. Motivation

DNA microarrays are a widely used tool to monitor gene expression. An oligonucleotide array (“chip”) is a plastic or glass slide containing many spots, each consisting of many copies of a known oligomer (a DNA 25-mer, say) attached to the chip. In an experiment, an mRNA sample is extracted from several cells, reverse-transcribed into cDNA, and fluorescently labeled. The labeled cDNA is

then allowed bind to (hybridize with) the oligos on the array. Whenever the Watson-Crick complementary sequence of an oligo is present in a cDNA sequence, that cDNA will hybridize to the oligo. Unhybridized cDNA is washed off the chip, and the amount of hybridized cDNA at each spot can be measured via the fluorescent dye intensity. The idea behind this procedure is that each spot represents one gene and that the amount of hybridized cDNA at a spot is a measure of the gene’s transcript abundance in the cell sample, which is often interpreted as the gene expression level or its “activity” in the cell sample.

Oligos should be gene-specific, i.e., only the transcripts of a single gene should hybridize to a given oligo, so the measurement taken at the oligo’s spot can be interpreted as the corresponding gene’s expression level in the sample. Note that a cDNA need not contain the *perfect* Watson-Crick complement of an oligo to hybridize. Unfortunately, the extent of hybridization depends on many parameters, and the hybridization process on oligo chips is not yet fully understood. We focus on microarrays with short oligonucleotide probes (at most 30-mers, say), for which custom design methodology and technology is currently emerging.

Several algorithms for oligo selection have been proposed recently (e.g., [5, 6, 7]); they vary greatly in speed and potential sensitivity. In [7], we proposed a fast oligo selection algorithm that approximates oligo specificity by longest common factor statistics. The *longest common factor* length $\text{lcf}(s, t)$ of two strings s and t is the maximum length of a substring that occurs in both s and t .

For each candidate oligo o of gene g , and for each length $\ell \leq |o|$, we define the *longest common factor statistics* $\text{lcf}(o, \ell)$ as the number of genes $g' \neq g$ containing an oligo o' with $\text{lcf}(o, o') = \ell$. Thus when $\text{lcf}(o, \ell)$ is nonzero for some ℓ close to $|o|$, the oligo is unspecific and should not be used for the chip. The main advantage of this approach

is its speed in large scale applications. For each gene g , we compute matching statistics against all other genes using an enhanced suffix array [2]. In practice, matching statistics less than half of the oligo length are meaningless and can be considered as zero, which allows to speed up the computations considerably. The details are described in [7].

A valid criticism against this longest common factor approach is that it ignores that a GC-binding is more stable than an AT-binding. Here we propose an improved specificity measure based on difference in Gibbs free energy between the intended match and all potential secondary binding sites. The challenge is to estimate this energy term rapidly and accurately at the same time. We present an algorithm that uses the concept of *jumps in matching statistics* (which we describe next), which is only approximately two to three times slower than the naive longest common factor approach. The algorithm is presented in Section 4, after a more theoretical discussion of jumps in matching statistics.

1.2. Matching Statistics

Before suffix trees became common knowledge, it was an open question whether the longest common substring of two strings s and t can be found in linear time $O(|s| + |t|)$. The answer is yes, and in retrospect, the solution is conceptually quite simple.

The key idea is to compute *matching statistics* $ms^{s|t} = (ms_0, \dots, ms_{|s|-1})$, where ms_i denotes the length of the longest string that starts at position i in s and occurs somewhere in t . Then the longest common substring of s and t has length $\max_i ms_i$. As shown by Chang and Lawler [3] and described in Gusfield's book [4], the vector can be computed in $O(|s|)$ time, that is, in constant amortized time per element, when a suffix tree of t with suffix links is available. The suffix tree can be built in linear time $O(|t|)$, as first shown by Weiner [12]. Thus the longest common substring can be found in $O(|s| + |t|)$ total time.

From the definition of matching statistics, it follows that consecutive entries of the vector are not independent: For $i > 0$ we always have $ms_i \geq ms_{i-1} - 1$ (see Section 2.1). When a strict inequality holds for some position, we say that a *jump* occurs at this position. This notion is easily generalized to matches allowed to contain a fixed small number of mismatches. Our first aim is to investigate some properties of jumps (Sections 2.2 and 2.3). Then we study the frequency of jumps in non-uniform random texts (Section 3.1). In Section 3.2, our results are applied to estimate the typical length of the longest common substring of two random sequences. We provide a numerical example in Section 3.3.

2. Properties of Jumps in Matching Statistics

2.1. Basics

Let Σ be a finite alphabet. We write Σ^* for the set of all strings consisting of characters from Σ , and Σ^+ for all non-empty substrings. We use the term *factor* interchangeably with *substring*. A *prefix* of a string is a factor that starts at the beginning of the string, and a *suffix* is a factor that ends at the end of the string. When s is a string, we write $|s|$ for its length and use subscripts to refer to the individual characters: $s = (s_0, \dots, s_{|s|-1})$. We use the notation $s_{i..j}$ for the substring (s_i, \dots, s_j) , and $s_{(i)} := s_{i..|s|-1}$ for the suffix starting at position i .

Throughout the paper, we use a, b, c for variables taking letters of Σ as values, and we use s, t, \dots, x, y, z for (possibly empty) strings over Σ .

Definition 1 (Matching statistics). For strings s and t , the *matching statistics* of s against t are $ms^{s|t} = ms = (ms_0, \dots, ms_{|s|-1})$, where ms_i is the length of the longest prefix of $s_{(i)}$ that occurs somewhere in t .

As an example, consider the strings $s = \text{GAATACT}$ and $t = \text{ATACGACT}$. Here $ms^{s|t} = (2, 1, 4, 3, 2, 1)$. The third value, 4, is obtained by finding the longest prefix of $s_{(2)} = \text{ATACT}$ that matches somewhere in t . It is ATAC , a string of length 4.

From the definition, one property follows immediately.

Lemma 2 (Suffix property of matching statistics). *Consecutive matching statistics decrease by at most one:*

$$ms_i^{s|t} \geq ms_{i-1}^{s|t} - 1 \quad \text{for all } i = 1, \dots, |s| - 1.$$

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

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

Definition 3 (Jumps in matching statistics). We say that a *jump* occurs at position $i > 0$ in $ms^{s|t}$ if and only if $ms_i^{s|t} \neq 0$ and $ms_i^{s|t} > ms_{i-1}^{s|t} - 1$.

At the beginning of s ($i = 0$), there is always a jump unless $ms_0^{s|t} = 0$.

When there is a jump at position i , we call $J_i := ms_i^{s|t}$ the *jump level*.

Formally, a jump is a pair (i, J_i) of the jump's position and level.

2.2. Jumps in Substrings

We discuss a connection between $ms^{s|t}$ and $ms^{u|t}$ when u is a substring of s .

Lemma 4 (Matching statistics of substrings). *Let $u := s_{i..i+L-1}$, so $|u| = L$. Then*

$$ms_k^{u|t} = \min(ms_{i+k}^{s|t}, L - k) \quad (k = 0, \dots, L - 1).$$

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

Lemma 5 (Jumps in matching statistics of substrings). *Jumps in $ms^{u|t}$ can occur at position $k = 0$, and otherwise at most at those positions $0 < k < L$ where a jump occurs in $ms^{s|t}$ at position $i + k$.*

Proof. There is always (unless u_0 does not occur at all in t) a jump at $k = 0$ in u (the start of the substring), even when there is no jump at position i in s . Its jump level $J_0 = ms_0^{u|t}$ is obtained by looking at the closest jump to the left of i in s . Assume it occurs at position $i - d$ for some $d \geq 0$ and its level is j . Because there are no other jumps between $i - d$ and i , we know that $ms_i^{s|t} = j - d$ unless this is below zero. It follows that $J_0 = \min(\max\{j - d, 0\}, L)$.

For $0 < k < L$ note that a jump at position k in $ms^{u|t}$ implies a jump at position $i + k$ in $ms^{s|t}$, because $ms_k^{u|t} > ms_{k-1}^{u|t} - 1$ is equivalent to $\min(ms_{i+k}^{s|t}, L - k) > \min(ms_{i+k-1}^{s|t}, L - k + 1) - 1$, which implies $ms_{i+k}^{s|t} > ms_{i+k-1}^{s|t} - 1$. The converse implication does not hold, as shown by the following example. \square

Example 6. Consider again $s = \text{GAATACT}$ and $t = \text{ATACGACT}$ with $ms^{s|t} = (2, 1, 4, 3, 3, 2, 1)$. We now focus our attention on the substring $u = \text{AATA}$ of s . We obtain the following jumps (non-jumps are marked with a dot).

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

There is an initial jump to level 1 at $k = 0$ (derived from the jump to level 2 at $i = 0$). The jump to level 3 at $k = 1$ is the jump to level 4 at $i = 2$ restricted to the remaining substring length. Finally, the jump to level 3 at $i = 4$ does not become a jump at $k = 3$, because the value of $ms_3^{u|t} = 1$ is already implied by the previous jump.

Definition 7 ($[i, L]$ -jump-set). We define the $[i, L]$ -jump-set of $ms^{s|t}$ as the set consisting of the jumps between positions $i + 1$ and $i + L - 1$ (inclusive), and the closest jump to the left of $i + 1$ at position $i - d$ for the appropriate distance $d \geq 0$.

Corollary 8. *Let u be the substring of length L of s that starts at position i . To obtain the jumps in $ms^{u|t}$, it suffices to look at the $[i, L]$ -jump-set of $ms^{s|t}$.*

2.3. Matching Statistics with Mismatches

Definition 9 (Matching statistics with f mismatches).

For strings s and t , the *matching statistics allowing f mismatches* of s against t are $ms^{s|t;f} = (ms_0^f, \dots, ms_{|s|-1}^f)$, where ms_i^f is the length of the longest prefix of $s_{(i)}$ that occurs somewhere in t with at most f mismatches.

Consider $s = \text{GAATACT}$ and $t = \text{ATACGACT}$. We find $ms^{s|t;0} = (2, 1, 4, 3, 3, 2, 1)$ and $ms^{s|t;1} = (4, 3, 5, 4, 3, 2, 1)$. The first value, 4, results from the match GAAT versus GACT at the end of t . Trivially $ms_k^{s|t;f} \geq ms_k^{s|t;0} + f$, unless the end of either string is hit.

It is easy to see that ms^f also satisfies the suffix property (Lemma 2). So we can define *jumps* in ms^f in the same way as jumps in ms^0 , and all properties still hold, especially Corollary 8. We conclude with a last lemma.

Lemma 10. *Let u be a substring of s starting at position i . The longest common substring of u and t with at most f mismatches is the maximal level of all jumps in $ms^{u|t;f}$. It can be computed from the $[i, |u|]$ -jump set of $ms^{s|t;f}$.*

Proof. Trivially, $\text{lcf}^f(u, t) = \max_{i=0, \dots, |u|-1} ms_i^{u|t;f}$. The maximum must occur at a jump; therefore it is equal to the maximal jump level in $ms^{u|t;f}$. The last statement now follows from Corollary 8. \square

3. Statistical Analysis of Jumps

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

We consider jumps in “exact” matching statistics and generalize the results to matching statistics with one mismatch in Section 3.1. In Section 3.2, we use our findings to make statements about the longest common substring (exact and with one mismatch) of two strings.

3.1. Frequency of Jumps in Matching Statistics

Consider the number of occurrences K_L of a random string of length L in t . Since there are $n - L + 1$ possi-

ble starting positions, and the probability that the length- L substring at a given position agrees with a random string of length L is just p^L , we have

Lemma 11. $\mathbb{E}[K_L] = (n - L + 1) \cdot p^L$. \square

Exact statements about the distribution of K_L are more difficult, because we need to take possible self-overlaps (also known as autocorrelations or period sets) of all words w into account. This is theoretically possible, and an efficient algorithm to enumerate all autocorrelations for a given word length L is described in [9]. In [8], we proved for the uniform distribution π that the approximation $P(K_L = 0) \approx e^{-\mathbb{E}[K_L]}$ is highly accurate.

Both intuition and simulations (see Section 3.3) suggest that K_L has approximately a Poisson distribution with parameter $\lambda := \mathbb{E}[K_L]$ when π is close to uniform and $\mathbb{E}[K_L] \ll n$. Unless π is close to a Dirac distribution, a typical randomly drawn word W has no or low self-overlap, and therefore its K_L occurrences in t are likely non-overlapping and hence independent. The condition $\mathbb{E}[K_L] \ll n$ ensures that the occurrences are sufficiently rare, so it is unlikely that two occurrences are adjacent. Thus approximately

$$\mathbb{P}(K_L = k) = e^{-\lambda} \cdot \lambda^k / k!, \quad (1)$$

where $\lambda = \mathbb{E}[K_L] = (n - L + 1) \cdot p^L$ by Lemma 11.

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

Lemma 12. *We have*

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

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

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

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

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

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

as claimed. \square

Theorem 13. *An estimate of the expected number E_L of jumps in $ms^{|t|}$ with jump level L is given by*

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

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

Proof. The expectation E_L is

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

As stated in Lemma 12, this probability is ρ_L for $i = 1, \dots, m - L - 1$. For the L -grams at each boundary of s , either the second or third condition in the proof of Lemma 12 is automatically satisfied. Instead of ρ_L we obtain a probability of $\rho'_L = e^{-\lambda p} - e^{-\lambda}$. Summing $(m - L + 1)\rho_L + 2\rho'_L$ yields the stated result. \square

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

Theorem 14. *The expected number E_L of jumps to level L in the matching statistics with one mismatch $ms^{|t|;1}$ is approximately given by*

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

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

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

3.2. Length of the Longest Common Factor

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

Necessary conditions for $E_L^+ \approx 1$ are $\lambda \ll 1$ and that L is small compared to both m and n when m and n are of comparable magnitude. Thus for the case of no mismatches,

we may approximate $\lambda \approx np^L$, and $e^{-\lambda x} \approx 1 - \lambda x$. Therefore $E_L \approx mq^2\lambda \approx mnq^2p^L$. In the case of one mismatch, we obtain the approximations $\lambda \approx nLqp^{L-1}$ and $E_L \approx Lmnq^3p^{L-1}$. By geometric summing, it follows that

$$\begin{aligned} E_L^+ &\approx mnqp^L \quad (\text{exact matches}), \\ E_L^+ &\approx Lmnq^2p^{L-1} \quad (\text{one mismatch}). \end{aligned} \quad (3)$$

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

Theorem 15. *The center L° of the lcf of two random strings with lengths m and $n = \Theta(m)$ generated with close-to-uniform character distribution π on Σ is*

$$L^\circ \approx \frac{\ln(mn) + \ln q}{\ln(1/p)}. \quad (4)$$

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

For the length of the longest common factor with one mismatch (lcf¹), the center L° is the solution of the equation

$$mnq^2 \cdot L^\circ \cdot p^{L^\circ-1} = 1. \quad (5)$$

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

While this result is approximate, it agrees with more precise results given by Waterman [11] and Abbasi [1] when $m = n$, π is uniform, and the strings are considered as cyclic (so there are no end effects). In this case it has been shown that for any $a \geq 1$, $\mathbb{P}[|\text{lcf}(s, t) - \log_{|\Sigma|} n^2| > a] \leq 18|\Sigma|^{-a}/n^2$; so $\text{lcf}(s, t)$ is indeed tightly concentrated around $\log_{|\Sigma|} n^2$. Even though Equations (4) and (5) do not make a precise statement, they give us an intuitive idea about the longest common factor in the non-uniform Bernoulli model, for unequal string lengths, and allowing mismatches.

3.3. Simulations

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

we found the lcf for each of the 1000 string pairs. Its empirical distribution is shown in the same table.

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

4. Fast and Accurate Probe Selection with Jumps in Matching Statistics

4.1. Measuring Oligo Specificity

The Gibbs free energy $\Delta G(o, T)$ of an oligo o at temperature T measures the oligo's tendency not to hybridize to its perfect match on the target gene. A strongly negative value $\Delta G(o, T) \ll 0$ indicates that the oligo tends to spontaneously hybridize to its perfect match¹ at temperature T . With increasing T , $\Delta G(o, T)$ grows towards zero. Thus $-\Delta G(o, T)$ can be interpreted as a stability measure of the cDNA-oligo duplex. It is common practice to compute it using the Nearest Neighbor (NN) model described in SantaLucia's overview article [10]. Here $\Delta G(o, T)$ is an affine function of T that depends on o only via the frequencies of adjacent nucleotide pairs. As a consequence, the stability of the target cDNA-oligo match can be rapidly evaluated.

We are interested in the stability $-\Delta G(o, g', T)$ of oligo-gene hybridizations (o, g') , where g' does not contain an exact match of o . While some parameters exist in the NN model for this situation, they are less agreed upon than the unified perfect match parameters from [10]. Therefore we propose the following compromise.

We find substrings \bar{o}^0 of o that are also substrings of g' such that $G_0 := \Delta G(\bar{o}^0, T)$ is minimized over all such choices of \bar{o}^0 . Also, we find substrings \bar{o}^1 of o , and o' of g' such that \bar{o}^1 and o' match with at most one mismatch and minimize $G_1 := \Delta G(\bar{o}^1, T)$ over all such pairs (\bar{o}^1, o') .

In other words, we look for the most stable perfect and near-perfect partial match of o and g' and treat it as if they were exact partial matches. We estimate

$$\Delta G(o, g', T) \approx \min\{G_0, G_1 + \gamma\},$$

where $\gamma > 0$ is a penalty parameter, e.g., $\gamma \approx 3000$ cal/mol. In other words, we measure the stability of the hybridization

¹In fact, it hybridizes to its Watson-Crick complement, but the oligo sequence corresponds to the coding mRNA sequence, because the mRNA is first reverse transcribed into cDNA.

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

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

of g' to o by the lowest free energy of either a perfect partial match (G_0), or of a penalized near-perfect partial match ($G_1 + \gamma$).

The specificity of the oligo o can now be quantified by the difference between this estimate of $\Delta G(o, g', T)$ and the known value of $\Delta G(o, T)$.

4.2. Efficient Estimation of $\Delta G(o, g', T)$

Recall that the goal is to find oligos for gene g that are specific in the sense that the difference between the above mentioned estimate of $\Delta G(o, g', T)$ and the known value of $\Delta G(o, T)$ is large.

In a preprocessing step, we prepare the lists of jumps in both $ms^{g|g';0}$ and $ms^{g|g';1}$ for all other genes $g' \neq g$. The lists can be restricted to jumps (i, J_i) with jump level J_i above a given threshold. This threshold should be as large as possible to minimize the size of the list (from Table 1, we see that the number of jumps to level L decreases rapidly with increasing L), but without cutting off potentially stable partial matches. The preparation of the jump lists may be time-consuming (depending on the threshold), but they only need to be computed and stored once.

To find gene-specific oligos in gene g , we do the following each other gene g' , for exact matching statistics $ms^{g|g';0}$ and one-mismatch matching statistics $ms^{g|g';1}$, and for each candidate oligo o from left to right.

- Let i be the starting position of the current oligo o . For each gene $g' \neq g$, we do the following.

1. Find $\ell := ms_0^{o|g'}$ from the first element of the

$[i, |o|]$ -jump-set of $ms^{g|g'}$ as described in the proof of Lemma 5. As we move through g from left to right, this value can be updated in constant time from the previous oligo. Evaluate the stability $-\Delta G(\bar{o}, T)$ of the partial match $\bar{o} := o_{0..\ell-1}$ (penalize it by $\gamma > 0$ in the one-mismatch case).

2. For each remaining jump $(i + k, J_{i+k})$ in the $[i, |o|]$ -jump-set, determine whether there is also a jump at position k in the substring matching statistics $ms^{o|g'}$. If yes, let $\ell := \min\{J_{i+k}, |o| - k\}$ denote the jump level and evaluate the stability $-\Delta G(\bar{o}, T)$ of $\bar{o} := o_{k..k+\ell-1}$ (penalize the stability by $\gamma > 0$ in the one-mismatch case).
3. The most stable match found in this way is compared to the perfect match stability $-\Delta G(o, T)$, and the difference is recorded as the specificity of o with respect to g' .

- Oligo o is rejected when the specificity with respect to too many genes g' is low.

This procedure is fast because most jump sets are small and few ΔG -evaluations are necessary. It suffices to evaluate ΔG at jump positions because it is safe to assume that the stability of non-maximal partial matches between o and g' is lower than the stability of the corresponding maximal match.

4.3. Results: Probes for *Saccharomyces cerevisiae*

We applied probe selection methods based on longest common factor statistics and on stability (ΔG) estimates to

all 6342 open reading frames of *Saccharomyces cerevisiae* (baker's yeast) and their reversed complements (12684 sequences, about 18 MB). All computations were carried out on a Compaq AlphaServer ES45 with 64-bit 1 GHz processors and 27 GB of main memory (only 250 MB were used). Programs were compiled with the native C compiler with all speed optimizations.

The preprocessing took 38 minutes (35 seconds for the creation of the enhanced suffix array, and 37 minutes for the jump lists of all matching statistics). The values of $18 \cdot 10^6 \cdot 12684 \approx 240 \cdot 10^9$ matching statistics are evaluated during this time. Storing these in the straightforward way using 1 byte per value would take $2 \cdot 240$ GB for ms^0 and ms^1 . Our jump list format uses 3 bytes per jump, and we only store jumps to levels of at least 10 for ms^0 and to at least 13 for ms^1 . The resulting file requires only 2.4 GB, only 0.5% compared to the naive approach. This corresponds to about $800 \cdot 10^8$ jumps overall. The estimate E_L^+ from Eq. (3) predicts $2.3 \cdot 10^8$ jumps to levels 10 or above for exact matching statistics ms^0 , and $1.4 \cdot 10^8$ jumps to level 13 or above for ms^1 . The observed number of jumps exceeds the predicted number by a factor of more than 200, because the yeast genome is not a random text: Local similarities are frequent, and jumps to medium levels occur more often than in random texts. Nevertheless, the use of jump lists saves 99.5% of the required space.

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

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

It is interesting to compare the predicted ΔG -difference δ and the distance d of the longest common factor of typical oligo candidates. More precisely, if ℓ is the oligo length and lcf^0 and lcf^1 are the lengths of the exact respectively one-mismatch longest common factor, we set $d := \ell - \max\{lcf^0, lcf^1 - 3\}$. As a rule of thumb, a change of 1 bp in d corresponds to a change of 1.1 kcal/mol in δ under the mentioned conditions.

5. Discussion and Conclusion

We have defined jumps in matching statistics and derived some of their statistical properties in random sequences.

We can use lists of jumps in matching statistics to *reduce* the space required to store vectors of matching statistics without explicit compression. The reduction factor can be estimated using Eq. (3) for typical random data. The reduction is not as good for typical DNA sequence data, but still a remarkable 99.5% for the yeast ORFs.

At the same time, the use of jump lists *speeds up* the computations of $lcf^f(u, t)$ for substrings u of s when $ms^{s|t;f}$ is available. The performance gain is large enough to evaluate more realistic functions than the longest common factor in a large scale oligo design setting. We have demonstrated that with jump lists, the much more accurate oligo selection based on Gibbs Free Energy difference takes only 2 to 2.5 times as long as longest common factor based oligo selection.

A natural question is, how much more accurately does the energy-based model predict the true stability of the hybridizations, and does it pay to take into wait twice to three times as long during the selection phase? We think that the second question can be clearly answered positively: During chip design, time is not that critical. While a speed factor of 10 to 100 or more would be painful, a factor between 2 and 3 is easily accommodated. The first question is harder to answer from today's perspective. As noted above, on average the lcf-based measure and the energy-based measure are approximately linearly related (a change of 1 bp corresponding to 1.1 kcal/mol under typical conditions), but the energy measure varies with sequence composition. To determine to what extent this improves real microarray experiments, we need large scale experimental data with many custom probes, which we are currently not in a position to produce.

Software. The probe selection software PROMIDE is licensed free of charge to individual academic users. PROMIDE and sample oligo sets can be obtained at <http://oligos.molgen.mpg.de>.

Acknowledgments. I thank Hannes Luz for very helpful discussions about Section 3, Stefan Haas for sharing his knowledge about primer design principles, Thomas Meinel and Antje Krause for motivation, and Martin Vingron for support.

References

- [1] S. Abbasi. Longest common consecutive substring in two random strings. DIMACS Technical Report, DIMACS, October 1997.

- [2] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *Proceedings of the Second International Workshop on Algorithms in Bioinformatics (WABI 2002)*, volume 2452 of *LNCS*, pages 449–463, 2002.
- [3] W. Chang and E. L. Lawler. Sublinear expected time approximate string matching and biological applications. *Algorithmica*, 12:327–344, 1994.
- [4] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, 1997.
- [5] L. Kaderali and A. Schliep. Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics*, 18(10):1340–1349, 2002.
- [6] F. Li and G. Stormo. Selection of optimal DNA oligos for gene expression analysis. *Bioinformatics*, 17(11):1067–1076, 2001.
- [7] S. Rahmann. Rapid large-scale oligonucleotide selection for microarrays. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference (CSB)*, pages 54–63. IEEE, 2002.
- [8] S. Rahmann and E. Rivals. On the distribution of the number of missing words in random texts. *Combinatorics, Probability, and Computing*, 12:73–87, 2003.
- [9] E. Rivals and S. Rahmann. Combinatorics of periods in strings. In P. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP 2001)*, number 2076 in *Lecture Notes in Computer Science*, pages 615–626, Berlin, 2001. Springer-Verlag.
- [10] J. SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA*, 95:1460–1465, 1998.
- [11] M. S. Waterman. *Introduction to Computational Biology*. Chapman and Hall, London, 1995.
- [12] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11. IEEE, 1973.