Lecture 3: March 18, 2004

Lecturer: Ron Shamir *Scribe: Alon Shalita and Omer Czerniak*[1]

3.1 Spectrum Alignment

This work by Pe'er et al. [5] presents an algorithm for reconstructing a target sequence using the SBH technology integrated with known similar reference sequence. The goal is to get the best reconstruction, given these two sources of information.

3.1.1 Motivation

Some motivation for using the method:

- Checking for specific mutation, e.g. Tay-Sachs, CF, Fragile-X, Gaucher. To date, those tests are performed separately for each mutation with different technologies. Since we know the sequence of the gene (and sometimes even where the mutation might appears) we would like to use this information for finding the correct path of the SBH output.
- Identifying unknown micro-organism can be done by the sequence of a conserved known gene.
- Identifying a specific strain by more divergent sequences.
- Finding the gene's sequence in one species, given this gene's sequence in another species.
- HLA-typing (graft recognition).
- Detecting somatic mutations (cancer).

All those challenges present one common problem: we need to reconstruct a target sequence given a known similar reference sequence. Current technologies are either target-specific, or ignore existing information on target (redo sequencing the target).

[1]based in part on scribe by Tamir Tuller and Kobi Lindzen 2002

3.1.2 The Scheme

The scheme of finding the target sequence that fits best, both to the reference sequence and the hybridization spectrum, includes the following steps:

1. Create a model that presents the reference sequence data.
2. Convert the SBH data into a graph.
3. Using those two presentation, reconstruct the target sequence.

HMM of the Reference Sequence

The reference sequence will be presented as a hidden markov model (HMM), using Position Weight Matrix (PWM). It can be described as chain of states, one state for each nucleotide in the reference sequence, with nucleotide probabilities $v_k(A)$, $v_k(C)$, $v_k(G)$, $v_k(T)$ per each state q_k (see Figure 3.1). We build this model according to the known distribution of the gene among the population, assuming they are independent.

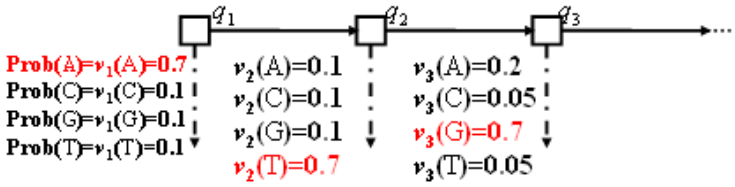

Figure 3.1: A simple example HMM for the reference sequence ATG: The nucleotide T has probability of 0.7 to appear in the second position. In this models there are no deletions or insertions.

A more complicated HMM model can also include deletion and insertion states (see Figures 3.2). Here we will present only the simple model, the interested reader is referred to the paper [5]. More details on HMMs can be found in [10, 11].

Scoring according to the A-priori Distribution

The HMM score of the target sequence $T = t_1t_2t_3...t_L$ is the probability of observing sequence, given the PWM. For the gapless HMM we get (see Figure 3.1):

$$P(T = t_1t_2t_3...t_L|PWM) = \prod j(v_j(t_j)) \tag{3.1}$$

$$PWMScore(T)_L = \log P(T|PWM) = \sum_i log(v_i(t_i)) \tag{3.2}$$

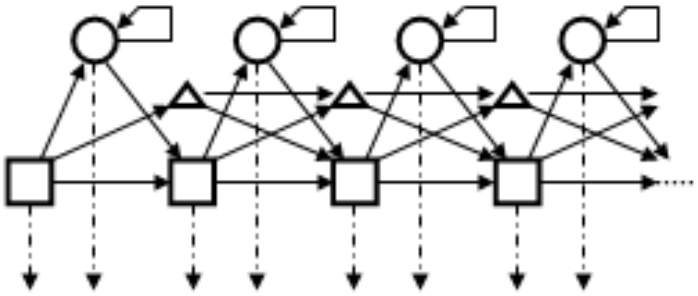

Figure 3.2: An HMM model allowing deletions and insertions. The triangles are deletion states, the circles are insertion states and the squares are match states. The solid arrow connect two states. The dashed arrow specifies emission.

For efficient computation of this score we will use the following identity:

$$PWMScore_L = PWMScore_{L-1} + log(v_L(t_L)) \tag{3.3}$$

The score can be generalized to general HMM. [5]

The Hybridization Graph

A hybridization experiment measures for each k-mer the intensity of its hybridization signal. Due to the randomized nature of hybridization, most signal level cannot be binirized satisfactorily. Let *Signal(x)* be the intensity of each k-mer x upon the chip. For each k-mer x we will calculate two probabilities:

$$P_1(x) = Prob(signal(x)|x \text{ occurs along target}) \tag{3.4}$$

$$P_0(x) = Prob(signal(x)|x \text{ does not occur along target}) \tag{3.5}$$

For the hybridization spectrum we build the following directed graph, $G(V, E)$ (see Figure 3.3):

- The set of vertices V contains all the $(k-1)$-mers in the spectrum.
- The set of edges E contains all k-mers observed in the hybridization. We build edges between max overlap $(k-1)$-mers, each edge is annotated with its $P_1(x)$ and $P_0(x)$ probabilities.

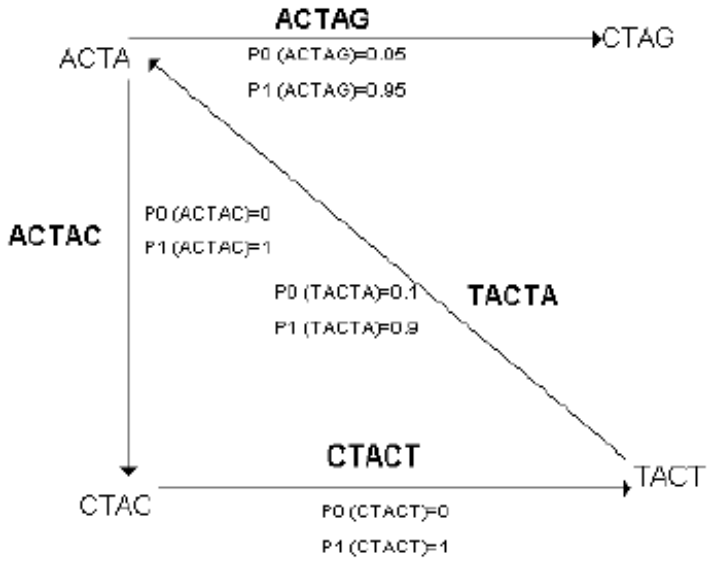

Figure 3.3: Example of the hybridization spectrum graph, the observed k-mer are written on the edge with their probabilities. Each vertex presents $k-1$-mer.

Scoring According to the Spectrum

The spectrum score of the target sequence is the probability of observing spectrum, given the target sequence:

$$SpecScore_L(T) = \log P(spectrum|T = t_1t_2t_3...t_L) = \log\{\prod_{x\in T} P_1(x) \prod_{x\notin T} P_0(x)\} =$$

$$\sum_{x\notin T} \log P_0(x) + \sum_{x\in T} \log P_1(x) - \sum_{x\in T} \log P_0(x) + \sum_{x\in T} \log P_0(x) =$$

$$\sum_{x} logP_0(x) + \sum_{x\in T} \log(P_1(x)/P_0(x))$$

Let $W(x) = log(P_1(x)/P_0(x))$. Then:

$$SpecScore_L(T) \approx \sum_{x} logP_0(x) + \sum_{j} W(t_{j-k+1}t_{j-k+2}t_{j-k+3}...t_j) \tag{3.6}$$

This score is weight of the path in G which corresponds to the target sequence. Again efficient computation of the score can be achieved using the following identity:

$$SpecScore_L \approx SpecScore_{L-1} + W(\text{current } k\text{-mer}) \tag{3.7}$$

The Total Likelihood Score

Our goal now is to find the target sequence which corresponds to paths in both graphs. In order to achieve it, the authors developed scoring method which combine the two information sources. The total likelihood score is the sum of the spectrum and the PWM scores :

$$TotScore_L(T) = SpecScore_L(T) + PWMScore_L(T) \tag{3.8}$$

We search for $T = t_1t_2t_3...t_L$ which maximizes the total score. We have seen we can solve this problem using dynamic programming. For $0 < j < L$ and a $(k-1)$-mer x let $S_j(x) \equiv max\{TotScore_j(T)|$ sequence T ending with $x\}$. $S_j(x)$ can be calculated using the recursion:

$$S_j(x = x_1x_2 \dots x_{k-1}) = \max_{x_0}\{S_{j-1}(x_0x_1 \dots x_{k-2}) + W(x_0x_1 \dots x_{k-1})\} + log(v_j(x_{k-1}))$$

$S_{opt} = \max_x(S_L(x))$. S_{opt} gives us the best score for the joint path (with length L) in both graphs. In order to get the corresponds path in each iteration we will keep the character that brought us to this iteration and in the end we will be able to trace back the path (a common technique with all dynamic programming variants).

Complexity

It is easy to see that running time and the space requirement are $O(LK)$, as the size of the table in our dynamic programming algorithm. The space requirement can be reduce by using a divide and conquer approach [4],when using this approach we can get time complexity of $O(LKlogL)$ and space requirement of $O(K + L)$. The typical length of the HMM, L, is $10^3 \leq L \leq 10^4$, the probabilistic spectrum is of length $K = O(4^k)$, typically $8 \leq k \leq 10$.

3.1.3 Simulation Studies

The algorithm was implemented and tested on simulated data. As a reference, a prefixes of real genomic DNA sequences were used. Each simulated run randomly generates:

1. A target sequence according to a prescribed probability $1 : 200bp$ of substitution.
2. A probabilistic 8 - spectrum according to a prescribed hybridization error .

For simplicity, insertions and deletion were not allowed, substitution were equiprobable, and all the probabilistic parameters were constant, i.e., position k-mer independent. For each parameter set, several simulated data were generated and the algorithm was applied to each.

Two figures of merit were checked:

1. Full success rate - The fraction of runs for which the target sequence was perfectly reconstructed.

2. The percentage miss-called bases - The fractions of base-calling errors made by the algorithm. The simulation results can be seen on Figures 3.4, 3.5, 3.6 and 3.7.

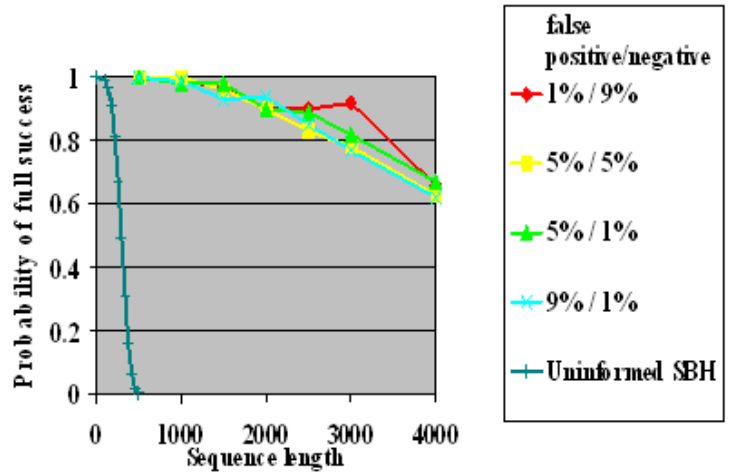

Figure 3.4: Source: [5]. Success percentage versus the hybridization error level : The probability of full success in reconstructing the target sequence, as a function of the sequence length, different graphs for different hybridization error levels.

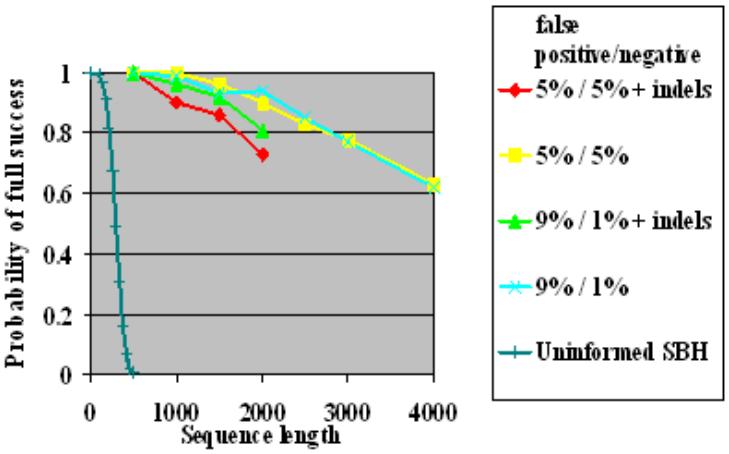

Figure 3.5: Source: [5]. Success percentage versus the insertion/deletion level: The probability of full success in reconstructing the target sequence, as a function of the sequence length, different graphs for different indels and similarity levels between the target and the reference sequences.

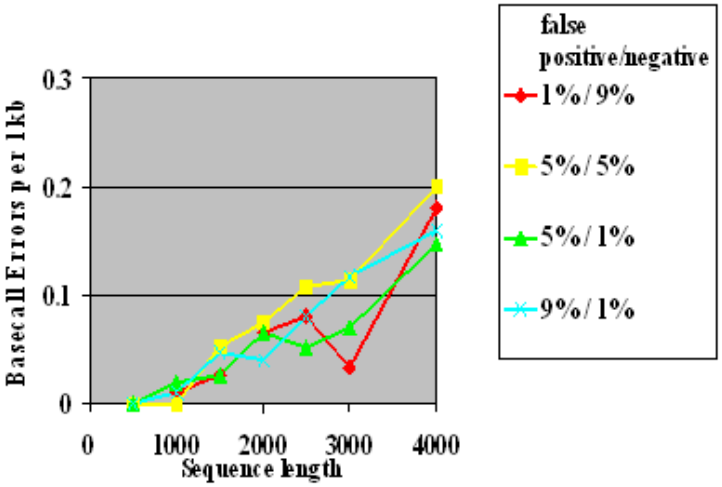

Figure 3.6: Source: [5]. Percentage of miss-called bases as a function of the sequence length versus the hybridization error level.

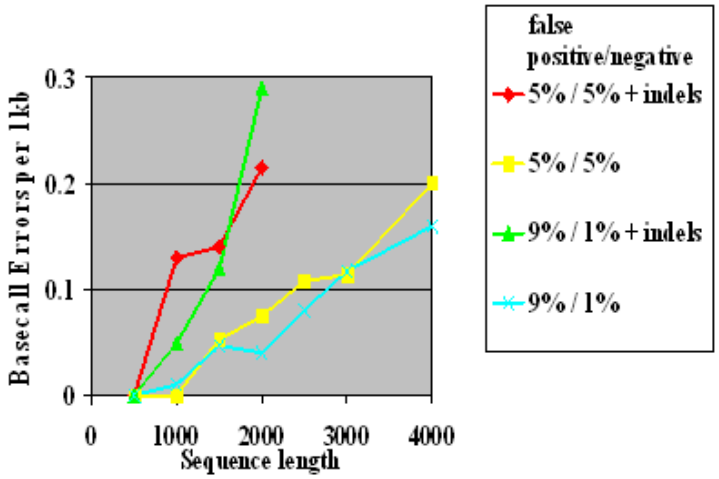

Figure 3.7: Source: [5]. Percentage of miss-called bases as a function of the sequence length versus the insertion/deletion level.

3.2 Genotyping with Partial and Universal Arrays

3.2.1 Polymerase Signaling Assay

The Polymerase Signaling Assay (PSA) is an accurate method for parallel re-sequencing which is effective to sequences up to several thousand base long. It is based on technology developed by Orchid Biosciences Inc. PSA uses special probes that are built from non-specific and specific parts. The non-specific part, the *capture* motif, is identical for all probes, and its role is only to stabilize the target on the probe. The specific part is different between probes, and it contains selected specific k-mer probes that tiles the target sequence and its variants (A part of the target is bond to the non-specific part). If there is a perfect match between the specific part and some part of the target, hybridization occurs which can be detected via the emission of fluorescent from the probe. The fluorescent slide is then scanned to detect the spot that signal a match between their probes and the target sequence.

Orchid plates included 176 different probes, with 5bp specific probes. Those plates used for analysis of Angiotestinoge (AGT) exon 2. The targets are amplicons of homozygote wildtypes/mutants for a single mutation.

3.2.2 Problems of re-sequencing

Computational analysis is based on the spectrum alignment algorithm that was described on the previous section.

Distilled Spectrum

One problem with the original spectrum alignment scheme used is that there are low signal-to-noise regions along the target. The solution to this problem is using enhanced spectrum alignment. The enhanced algorithm applies spectrum alignment iteratively: each iteration identifies reliable regions (regions that have strong support in the spectrum), and then distill them from the spectrum and re-iterate. In heterozygotes, same technique is applied to re-sequence a second allele.

Handling Real signals

As seen in previous section, we used two probabilities, $P_0(x)$ and $P_1(x)$, in order to evaluate the chances that probe x will be matched or unmatched by the target sequence. The problem is that the real signals do not allow a threshold rule for probe matching, as seen in Figure 3.8: Each threshold to divide between true and false signals will incur a high error rate. The solution to this problem is to learn the average matched and unmatched probabilities from the spectrum reference, and use them as threshold.

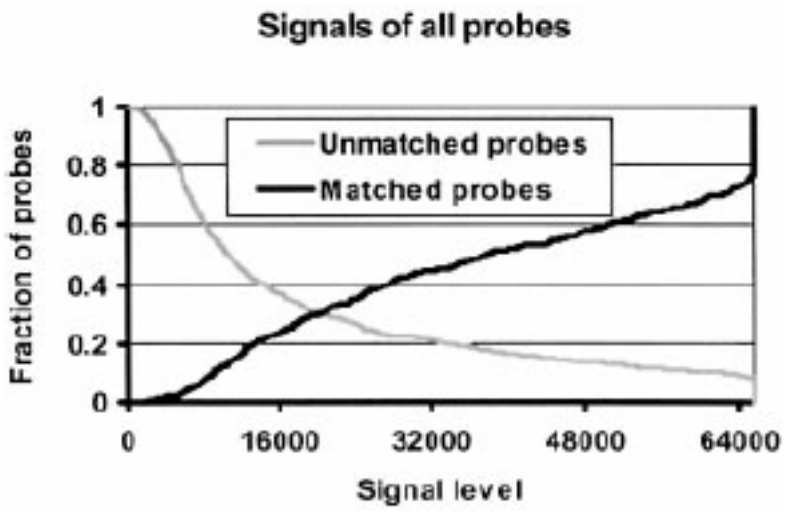

Figure 3.8: Source: [6]. Signal level distributions for matched and unmatched probes.

Another problem is that different probes have different signals. As we can see in figure 3.9, each of the probes might have a threshold to divide between true and false signals, but the average of the thresholds will cause all the the probes of one kind to be matched, and all the probes of the other kind to be unmatched. The solution for this problem is to empirically estimate $P_0(x)$ and $P_1(x)$ on a per-probe basis.

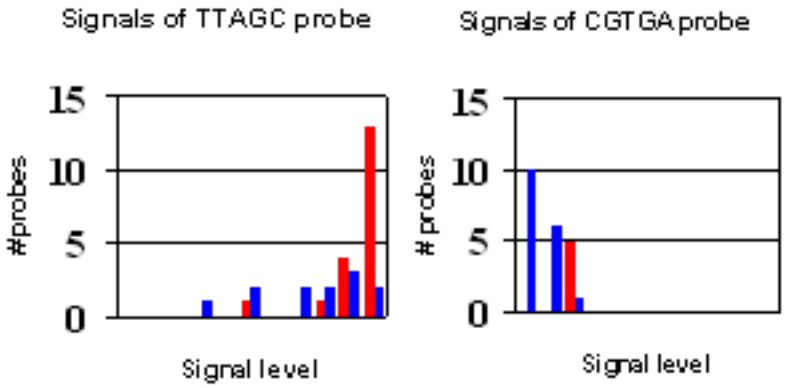

Figure 3.9: Source: [6]. Signal level of two specific probes:$TTAGC$, whose signals are are extremely high, and $CGTGA$ whose signals are extremely low.

Because of those problems, various methods were used to calculate better $P_0(x)$ and $P_1(x)$ values, using prior information from other datasets. For summary of the results see Figure 3.10. More information about those methods can be found at [6].

Re-Sequenced CF Arrays

Partial arrays with 176 probes were used for re-sequence CFTR exon 11. The reference that was used was not only the genomic sequence, but also known mutation from the Human Genom Mutation Database (HGMD). Out of 64 known polymorphism, 60.5 were correctly typed, and only 4 spurious mutations were falsely detected (see Figure 3.11).

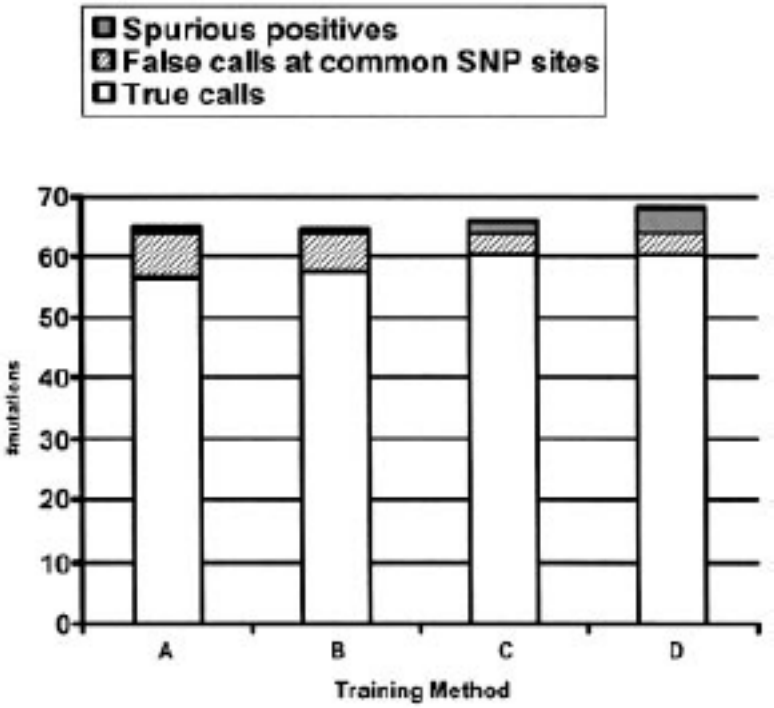

Figure 3.10: Source: [6]. Summary of re-sequencing using different training methods. (A) Probe-independent. (B) Per-probe training using the experiment and with another three matched and unmatched examples. (C) Per-probe training using all data sets. (D) Per-probe training using all the data sets except the one that contains the target.

3.3 Gapped Chips for SBH

3.3.1 Motivation

As we saw in Pevzner's algorithm for SBH reconstruction [8], if we use a classical SBH chip with capacity of t probes, the length of random sequence (four independent letters, with

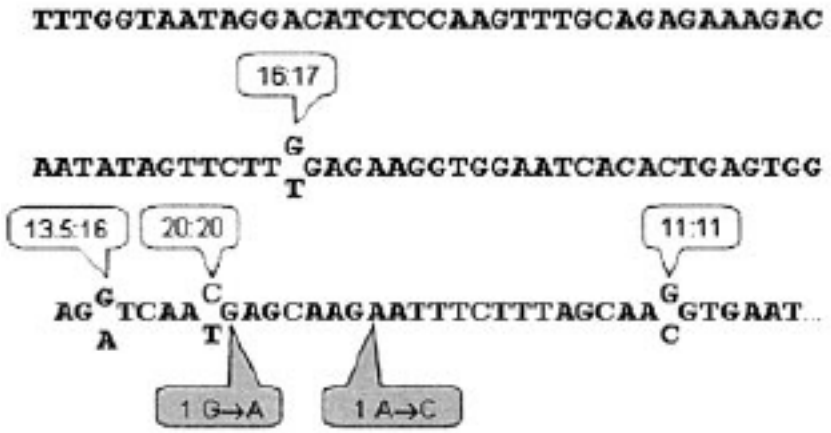

Figure 3.11: Source: [6]. Summary of re-sequencing results for CFTR.

probability of $\frac{1}{4}$ each) that we are able to reconstruct unambiguously is $O(\sqrt{t})$, which is quiet inefficient.

3.3.2 An Upper Bound

In their work Perparate, Frieze and Upfal [1] analyzed (base on information-theoretic argument) the maximal length that can be reconstructed with a general SBH chip, that is, any variant of SBH chip. Their algorithms will be referred as PFU. Unlike the spectrum alignment algorithms this algorithm uses no information about the target. PFU assumes that hybridization is an error-free process, with no false negatives nor false positives.

Theorem 3.1 ([1]) *For any fixed probability $p > 0$, the length of a random string that can be unambiguously reconstructed with probability p by a chip with t probes is $O(t)$.*

Proof: For each probe on the chip we have the information whether it appeared in the target (at least once) or not. Hence, there are 2^t possible spectrum's vectors and 4^m sequences of length m (see Figure 3.12).

In order to have an unambiguous reconstruction, each spectrum's vector can define no more than one possible sequence. Thus,

$$4^m \leq 2^t \Rightarrow m \leq \frac{1}{2}t \Rightarrow m = O(t) \tag{3.9}$$

Since there are vectors that do not match any m-long sequence we get $m < \frac{1}{2}t$. In order to reconstruct unambiguously a fraction p of the 4^m possible sequences we must have

$$p4^m \leq 2^t \Rightarrow m = O(t) \tag{3.10}$$

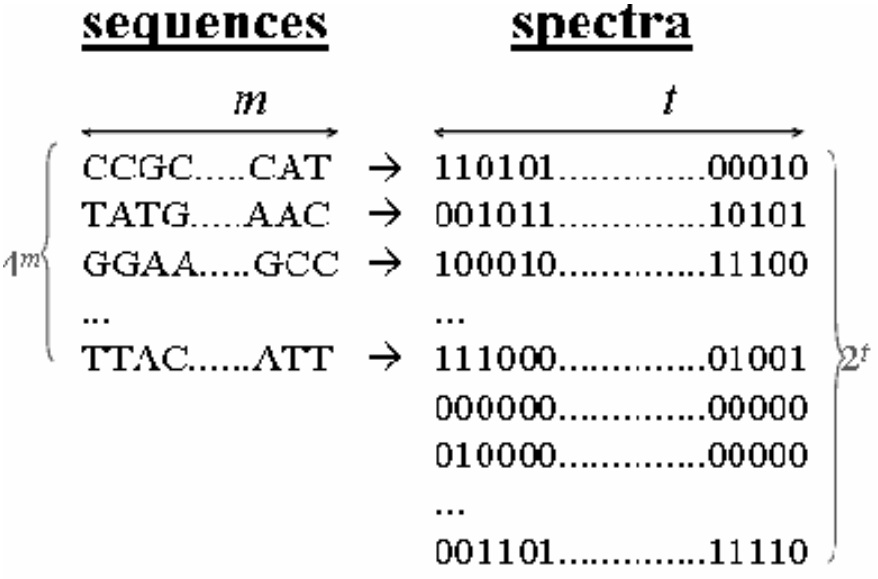

Figure 3.12: Number of possible sequences Versus the number of possible vectors in the spectrum.

■

Pevzner's algorithm uses all $t = 4^k$ probes of length k. The expected length of sequences it can unambiguously reconstruct is only

$$m = O(2^k) = O(\sqrt{t}) = o(t) \tag{3.11}$$

The PFU algorithms achieves $m = \Theta(t)$.

3.3.3 The PFU Algorithm

The key idea is to use *"universal"* (don't care) bases within the probes, that is, bases that bind to any nucleotide (see Figure 3.13). Our probes will include specified bases and unspecified (=universal) bases. For example the probe AC*G*G will be able to hybridize with TGACAA, TGACCA ,TGACGA, TGACTA, TGCCAA, etc. Preparate et al. proposed the following design for the probes (see Figure 3.13). Each probe is of the form $X^s(U^{s-1}X)^r$ where $X \in \{A, C, G, T\}$ is a specified base and U denotes a universal base. For a given r and s, the chip contains all probes of this form. According to the scheme we need to determine $r + s$ specified positions (the X's in Figure 3.14) $\Rightarrow$ The total number of probes is $t = 4^{r+s}$.

Figure 3.13: An example of matching to universal probe, where $s = 3, r = 3$.

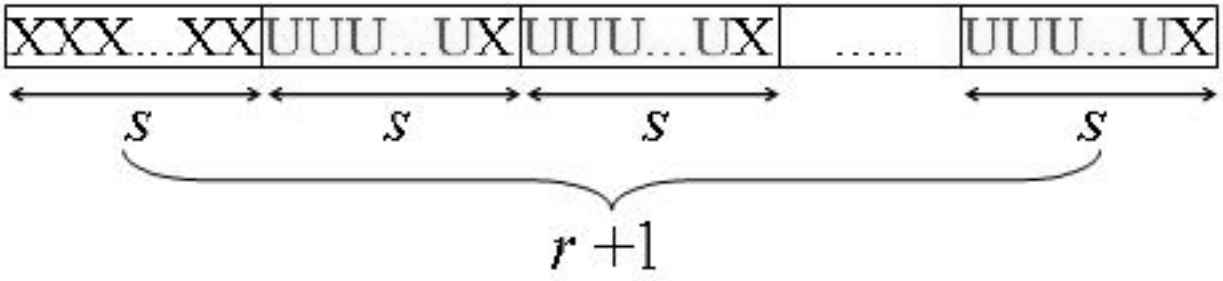

Figure 3.14: The scheme of a probe in PFU's algorithm.

3.3.4 The Reconstruction Algorithm

We assume that we are given the $s(r+1)$-prefix of the target string (we will show later how to handle this assumption). Let Z be the putative sequence reconstructed so far (with length greater than $(s(r+1)-1)$). Find set M_0 of all the probes in the spectrum such that, the $(s(r+1)-1)$-prefix of each of the probes matches the $(s(r+1)-1)$-suffix of Z. Let B_0 be set of possible extensions (the base located in position $s(r+1)$ of probes $\in M_0$) - the next symbol in Z. In case where the set B_0 is empty, no extension exists and the algorithm terminate with failure. In case where the set $|B_0|$ contains only one element, a single extension was found and the corresponding symbol is appended to Z. Otherwise, $(4 \geq |B_0| > 1)$, find set M_1 of all the probes such that their $(sr-1)$-prefix matches $(sr-1)$-suffix of Z and their $(s+1)$-suffix "agrees" with any probe $\in M_0$. Let B_1 be set of possible extensions (the bases in position sr of probes $\in M_1$) $\bigcap B_0$. Once again: In case where the set $|B_1|$ is empty declare failure. If $|B_1| = 1$-extend Z and continue, otherwise $(|B_1| > 1)$ continue with M_2 according to the general step. Figure 3.15 demonstrates this process.

In each phase we have $s-1$ new specified bases that have to agree with Z.

The Algorithm

Denote by L the sequence length. A formal description of the PFU algorithm follows.

$M_0, M_1, \ldots, M_r \leftarrow \emptyset$.
$B_0, B_1, \ldots, B_r \leftarrow \emptyset$.
`For` $n = s(r+1)+1$ `to` $L - s(r+1)$ `do`

1. $j \leftarrow 0$.
2. $M_0 \leftarrow \forall$ `probes` $P_x \in Spectrum$ `where` $(s(r+1)-1)$`-prefix of` P_x`=` $(s(r+1)-1)$`-suffix of` Z.
3. $B_0 \leftarrow P_{s(r+1)} \forall P \in M_0$.
4. `If` $|B_0| = 0$
   `Declare "Failure".`
   `End.`
5. `If` $|B_0| = 1$
   $ch \leftarrow B_{0_1}$
   $Z \leftarrow Z + ch$.
   `Next` n.
6. `If` $|B_0| > 1$
   `selectCandidate(`$++j$`).`

`Next` n

`selectCandidate(`j`)`

1. `If` $j > r$
   `Declare "Failure".`
   `End.`
2. $M_j \leftarrow \forall$ `probes` $P_x \in Spectrum$ `where` $(s(r+1-j)-1)$`-prefix of` P_x`=` $(s(r+1-j)-1)$`-suffix of` Z `AND` $\exists$`probe` $Q_y \in M_{j-1}$ `which "agrees" with the` $(sj+1)$`-suffix of` P_x.
3. $B_j \leftarrow P_{s(r+1-j)} \ \forall P \in M_j \bigcap B_{j-1}$.

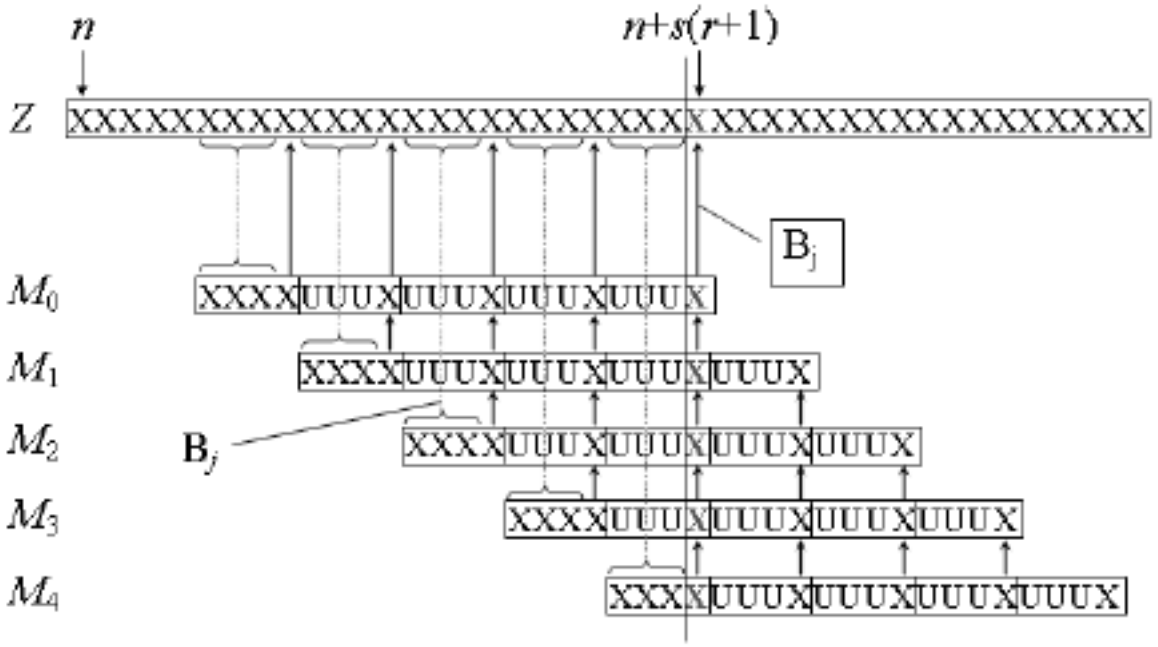

Figure 3.15: An example of the algorithm where $s = 4, r = 4$.

4. If $|B_j| = 0$
   Declare "Failure".
   End.

5. If $|B_j| = 1$
   $ch \leftarrow B_{j_1}$.
   $Z \leftarrow Z + ch$.
   Next n.

6. If $|B_0| > 1$
   $j \leftarrow j + 1$.
   selectCandidate(j).

3.3.5 Advantages of Gapped Probes

After we had reconstructed $(n-1)$-prefix of Z, we have two options for the Z_n symbol. The case where we have only one candidate is trivial, but when prefix of several probes matches

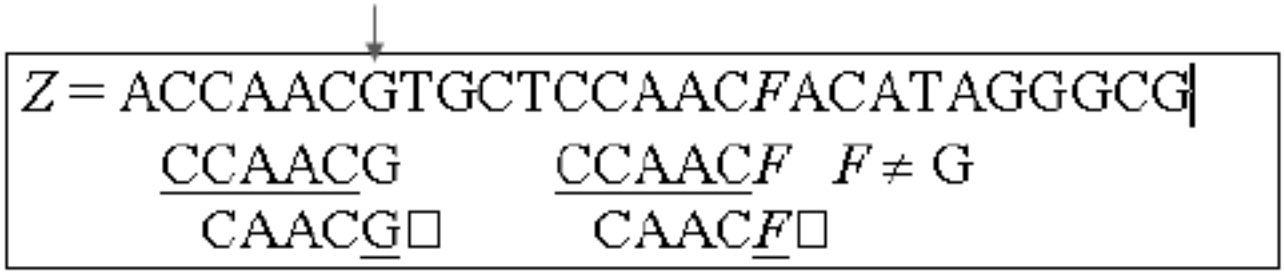

Figure 3.16: An example of the classical probe where $n = 7$. The confirmation does not work because CCAAC**F**_ implies CAAC**F**_ in the spectrum.

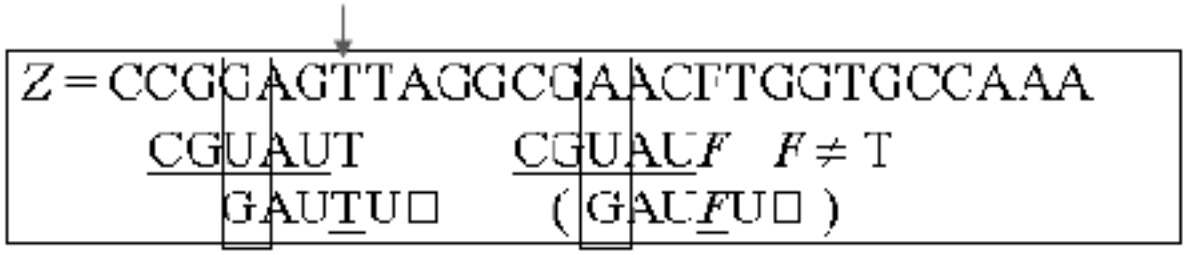

Figure 3.17: An example of an universal probe where $n = 7$. CGUAU**F** does not imply GAU**F**U_. The spectrum contains GAU**T**U_ and not GAU**F**U_, hence, $Z_7 = T$.

the suffix of Z, the question is how to use *other* probes to disambiguate. For every position in the sequence we have several probes (k - in the classical SBH, r - in this case) that can confirm the original symbol. In classical SBH we would like to use the shifted probes to confirm the correct base, but due to the fact that the information within the shifted probes is dependent we cannot use them, as exemplified in Figure 3.16. Using gapped probes, the probability of ambiguous extension in each aligned probe (with respect to a randomly generated sequence), is almost independent of the other probes (see Figure 3.17). This is the feature that gives this method its "power".

3.3.6 Removing the Prefix/Postfix Requirements

The PFU algorithm requires the $(s(r+1)-1)$-prefix of Z as an input. We have two ways to resolve this requirement:

1. Biological solution - Attach a synthesized known string of length $s(r+1)$ as a prefix/postfix to the target DNA (this can be achieved as part of the cloning process of the target DNA).

2. Reveal the needed prefix/postfix by standard sequencing machine or classical SBH chip (for short sequences the results are reliable).

3.3.7 Universal Chips in Practice

There are several works that claim for molecules that can serve as universal bases (for example [7]), but this issue is still vague. In any case, no universal chip has been realized yet.

3.3.8 The Performance of the PFU Algorithm

In order to analyze the PFU algorithm r and s are set as follows: Two constants $\alpha > 1$ and $\beta = o(\log m)$ are chosen such that $r > 0$ and $s > 1$ are integers:

1. $r = \frac{1}{\alpha} \log_4 m + \beta$.

2. $s = \log_4 m + 1 + \alpha - r$.

As a result of those values we get that the length of a probe is $s(r+1) = O(\log^2 m)$.

Denote by F the event that the PFU Algorithm fails to reconstruct a random target string of length m. Perparate et al. prove in their paper [1] the following theorem (we omit the proof here):

Theorem 3.2 *([1]) The algorithm fails to sequence a random string of length m using the above r, s then: $Pr(F) \leq 4^{-\alpha(1+\beta)}$.*

As a result of those values, we conclude the following theorem:

Theorem 3.3 *The length of a random string that the PFU algorithm (using chip with capacity t) succeeds to reconstruct with high constant probability is $\Theta(t)$.*

Proof: As we saw, the number of probes on the chip is 4^{r+s}.

$$t = 4^{r+s} = 4^{\log_4 m + 1 + \alpha} = m4^{1+\alpha} \Rightarrow \tag{3.12}$$

$$m = 4^{-(1+\alpha)}t \Rightarrow \tag{3.13}$$

$$m = \Theta(t) \tag{3.14}$$

■

3.3.9 Results

Given r, s and the desired probability of failure $p = P(F)$, the length of the longest random string that can be unambiguously reconstructed by the PFU algorithm with probability $(1 - p)$ is:

$$m = \sqrt[r+2]{p} \times 4^{(r+s-1)(1-\frac{1}{r+2})}$$

Figure 3.18 demonstrate the universal chip performance for $r + s = 8$. The capacity of this chip (t) is $4^{r+s} = 65,536$ probes (equivalent to a standard chip where $k = 8$). The $s(r + 1)$ column shows the probes' length, $Pr(F)$ shows the requested failure rate of the algorithm while m the sequence's length is being determined according to this value. The $\frac{m}{t}$ column presents the chip's performance. As Preparata et al. proved, its upper bound is 50%. The best design we can achieve for failure rate of 1% is with values $r = 6, s = 2$, where the scheme of the probes will be $XXUXUXUXUXUXUX$. This allows reconstructing up to 2,739 bases. Obviously the results are better than classical SBH.

3.3.10 Summary

In their article Preparate et al. [1] proved:

1. Theoretical upper bound on length of random targets that any chip can unambiguously reconstruct.

2. First time that a periodic gapped probe design was analyzed and a reconstruction algorithm was given to it.

3. Provable performance reaches upper bound (up to a constant) - asymptotically optimal design.

4. Outperforms previous methods also in practice - with real DNA and for practical values of s and r.

5. A year later Preparata and Upfal showed a theoretic design that asymptotically reaches the upper bound (without the constant) [9].

There are two main problems with this work:

1. Handling errors - need to analyze the results when the data on the chip is noisy. A paper regarding this issue was published by Doi and Imai [2].

2. The realization of universal probes.

r	s	$s(r+1)$	$\Pr(F) = 4^{-\alpha(1+\beta)}$	m	$m/t = 4^{-(\alpha+1)}$	α	β
3	5	20	0.01%	373	0.57%	2.73	1.43
			0.10%	591	0.90%	2.40	1.08
			1.00%	937	1.43%	2.06	0.61
4	4	20	0.01%	700	1.07%	2.27	1.92
			0.10%	1028	1.57%	2.00	1.50
			1.00%	1509	2.30%	1.72	0.93
5	3	18	0.01%	1099	1.68%	1.95	2.41
			0.10%	1527	2.33%	1.71	1.91
			1.00%	2122	3.24%	1.47	1.25
6	2	14	0.01%	1540	2.35%	1.71	2.90
			0.10%	2054	3.13%	1.50	2.33
			1.00%	2739	4.18%	1.29	1.57

Figure 3.18: The chip performance for all the possible values for $r + s = 8$. The number of probes is $t = 4^8 = 65,536$.

r	s	$s(r+1)$	$\Pr(F) = 4^{-\alpha(1+\beta)}$	m	$m/t = 4^{-(\alpha+1)}$	α	β
3	6	24	0.01%	1130	0.43%	2.93	1.27
			0.10%	1791	0.68%	2.60	0.92
			1.00%	2839	1.08%	2.26	0.47
4	5	25	0.01%	2224	0.85%	2.44	1.72
			0.10%	3264	1.25%	2.16	1.30
			1.00%	4791	1.83%	1.89	0.76
5	4	24	0.01%	3606	1.38%	2.09	2.18
			0.10%	5010	1.91%	1.85	1.69
			1.00%	6961	2.66%	1.62	1.05
6	3	21	0.01%	5181	1.98%	1.83	2.63
			0.10%	6909	2.64%	1.62	2.07
			1.00%	9213	3.51%	1.42	1.35
7	2	16	0.01%	6869	2.62%	1.63	3.08
			0.10%	8871	3.38%	1.44	2.45
			1.00%	11458	4.37%	1.26	1.64

Figure 3.19: The chip performance for all the possible values for $r + s = 9$. The number of probes is $t = 4^9 = 262,144$.

Figure 3.20: CGACTAAC - the unknown sequence. The prefix CG is known as well as the probes ($k = 3$).

3.4 Handling Long Targets and Error in Sequencing by Hybridization

This section summarizes the work of Halperin et al. [3], which includes:

1. A polynomial algorithm that handles errors in classical SBH and reconstructs the correct sequence with probability of $1 - \epsilon$.

2. A novel design of universal probes which gives improved results in the aspect of noise.

3. A polynomial algorithm for an error-prone SBH, using such arrays, with success probability of $1 - \epsilon$ and within $\log^2 m$ factor of the upper bound (m denotes the sequence length).

3.4.1 Classical SBH with Noise

A Trivial Algorithm

Extend by one base each time according to *one* of the positive probes that were not used till now and agrees with the known prefix. For example, in Figure 3.20 the only way to extend CG is by A, supported by the probe CGA.

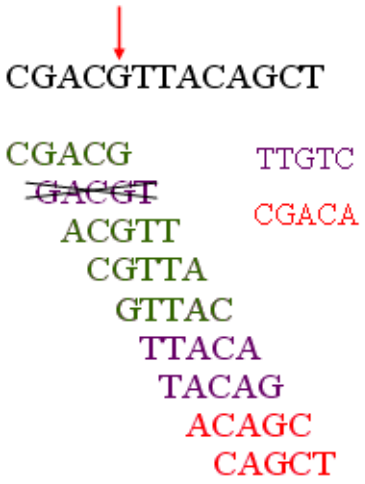

Figure 3.21: CGAC - known prefix. GACGT - false negetive, TTGTC,CGACA - false positive, ACAGC,CAGCT - fooling probes.

Problems in Practice

1. False negative errors - probes that match the sequence but are missing from the spectrum.

2. False positive errors - probes in the experimental spectrum that do not match any position in the sequence.

3. Fooling probes - probes that match other locations in the sequence and imply wrong extensions in the current position.

Figure 3.21 exemplified the errors types. In order to handle those problems the idea is to look ahead in the sequence and to check *all other* probes which can confirm or deny our candidate. We will check *all* the k-length possible extensions and count how many probes support each extension. In the end we will choose the "winning" extension and extend our sequence with the first symbol of that sequence.

Definition A *bad path* is a possible extension whose first base is false.

As we can see in Figure 3.21 we have four supporters to the correct path - CGACGTTAC and three supporters to the bad path - CGACAGCTG (two fooling probes and one false positive). In spite the fact that we do have errors we shall prove that the probability that a bad path wins is small. The intuition for that is that "voting process" is less sensitive to errors.

Lookahead Algorithm

Given a prefix $s_1, \ldots, s_i$:

1. Enumerate all 4^k potential extensions $a = a_1, \ldots, a_k$.
2. Pick an extension a' such that the number of supporting probes $s_{i-k+2}, \ldots, s_i, a'_1, \ldots, a'_k$ is maximal.
3. Set $s_{i+1} = a'_1$.

For each reconstructed position (m) we need to enumerate 4^k possible extensions and check them with a k-size window around the current position. Hence, the running time is $O(m4^k k)$.

The Main Theorem

For analyzing the algorithm we make the following assumptions:

- False negative rate is q.
- False positive rate is $p = O(2^{-k})$. In a classical k-chip we have 4^k points and we showed already that we can unambiguously reconstruct sequences of length $O(2^k)$. Such p implies $p4^k = O(2^k)$ false positive probes (the same order as our reconstructed sequence). If we allow $p > 2^{-k}$ most of the signal we get will be false positive, thus the requirement of such p is reasonable.
- The length of the sequence is $m = O(2^{(1-3q)k})$.
- The probability of errors in each probe are equal and independent.

Theorem 3.4 *The lookahead algorithm fails to reconstruct the sequence with probability bounded by an arbitrary small constant* $(1 - \epsilon)$.

Proof sketch

- We expect to see in the spectrum k supporters to the correct symbol. Due to false negative error each one of them has probability of $1-q$ to be seen. Hence, the number of supporting probes for the correct path $X \sim B(k, 1-q)$.
- For a bad path with j fooling probes, the number of supporting probes is $Y = Y1+Y2$, where:

  The number of fooling probes is $Y1 \sim B(j, 1-q)$ - Their probability to appear is also $1-q$ (since they are also "exposed" to the false negative error). The number of false positives is $Y2 \sim B(k-j, p)$.

- The algorithm fails if at some point $X < Y$.
- We can sum over all possible values for j

$$P_{bad} = sum_{j=0}^{k} P[X < Y | jfooling] \cdot P[jfooling].$$

- Eventually we get two cases for values of j:

  Large j: many fooling probes in the bad path - small probability, since it requires resemblance to the current location substring.

  Small j: means that X is even smaller and the reason for this is many false negatives for the correct path probes. This event also has a low probability since we have a geometric distribution for it.

3.4.2 Gapped Probes Design

Following PFU's algorithm, Halperin et al. suggest to use a universal bases within the probes, but while Preparta et al. suggested a periodic chip this work presents a random design, means, the location of the *don't care* on the probe will be set randomly. Each such setting will be a probe family (the common feature of the family member will be their unspecified bases location), all the families will have the same number of specified (or un specified) bases (see Figure 3.22). The probes are designed with the following features:

- The length of each probe is $ck + 1$ (where $c > 1$).
- Each probe has $k + 1$ specified bases, the rest are universal.
- The last symbol is always a specified base.
- There are βk ($\beta > 1$) families of probes.
- For each family: pick randomly set of k positions of specified bases, and implement all 4^{k+1} possibilities (the last position is also specified).
- The total number of probes (capacity of the chip) is $t = \beta k 4^{k+1}$.

3.4.3 The Reconstruction Algorithm

Again we will count how many probes 'vote' for a possible extension:

- For each specified base x, count how many probes support $s_1, \ldots, s_{ck}, x$.
- Set s_{ck+1} to be the base with maximal support.

Figure 3.22: Two random families. Each family has $4^4 = 256$ members.

Complexity

For each position (m) and for each family ($O(k)$), we need to This technique increases the chance for finding the check for positive signals only for four members (which equal in the ck-prefix and differ in their last position) $\Rightarrow O(km)$. Each family chooses different specific locations and this adds disqualification elements. Due to the random selection, the families are independent and that enabled the authors to analyze the algorithm's performance.

Gapped SBH - Error-Free Case

The authors prove in their work [3], the following theorem:

Theorem 3.5 ([3]) *If there are no hybridization errors, the sequence length is* $m = O(4^k/k)$ *and the number of probes is* $n = \Theta(k4^k)$*, then the algorithm fails with probability bounded by an arbitrary small constant.*

Since the number of probes is $O(k4^k)$, the length of the reconstructed sequence is optimal up to $\log^2$ factor (which is worse than PFU's algorithm).

Gapped SBH - Error-Prone Case

The main result of the authors is:

Theorem 3.6 ([3]) *In the error-prone case, where the length of the sequence is* $m < (\frac{(1-q)4^k}{5k})$*, the number of probes is* $n > \frac{5k4^k}{1-q}$ *and the false positive rate is* $p < \frac{(1-q)}{430}$*, then the algorithm fails with probability bounded by an arbitrary small constant.*

Here we also have $\log^2$ factor distance from the theoretical upper bound with no noise (which holds also in the case of noise).

3.4.4 Results

The authors compared their algorithm results on *real DNA* with the results of heuristic implementation of the PFU algorithm with noise by Doi and Imai [2]. Figure 3.23 demonstrates the success rate in reconstructing the sequence of the algorithms as function of the sequence length. A sequence with 1500 bases can be reconstructed perfectly with the algorithm of Halperin et al., while with Preparata et al. algorithm, it can be reconstructed only with 80% success rate.

In Figure 3.24 the chip capacity is 4^{10} probes and it presents the possible length of the reconstructed sequence as a function of the error rate when we demand 90% reconstruction success rate. In both figures $p = q$.

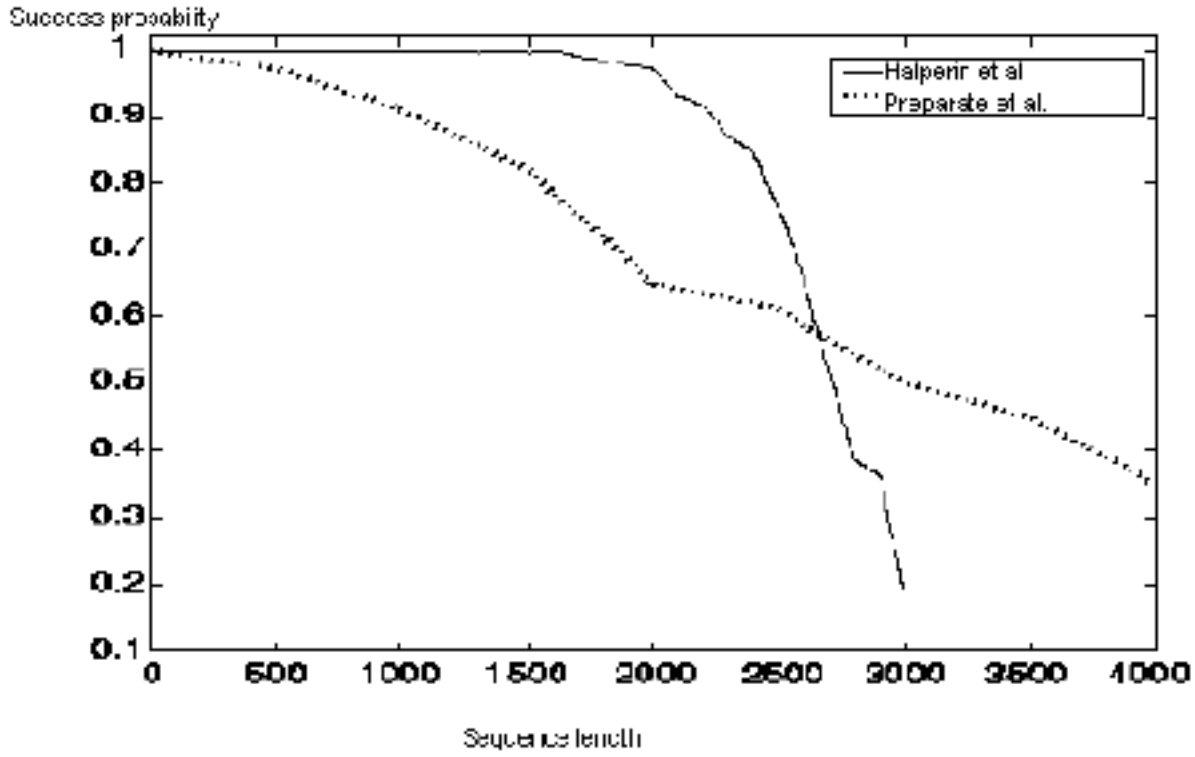

Figure 3.23: Source: [3]. Comparing the gapped design. Chip capacity = 4^8 probes. In Halperin et al. $p = q = 0.005$, in Preparata et al. $p = q = 0.001$.

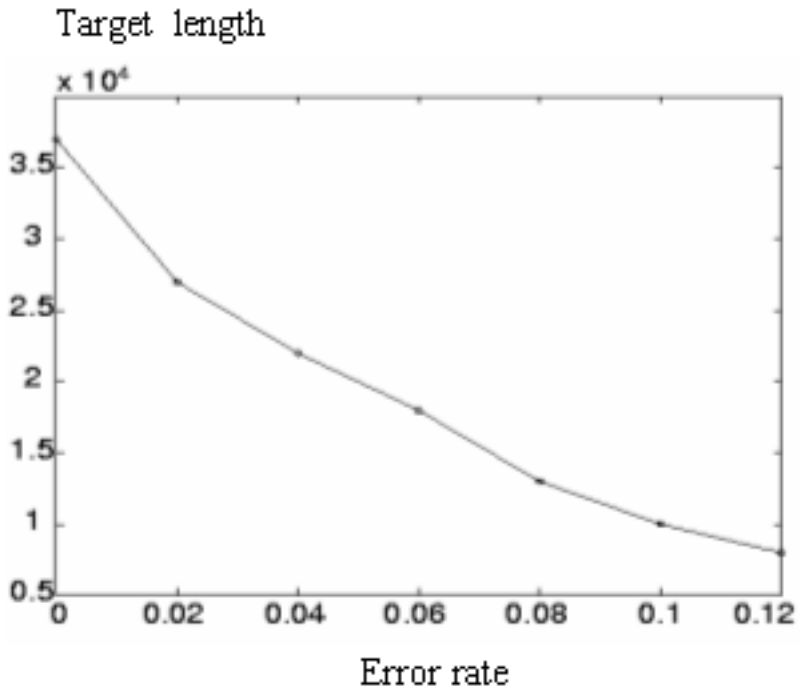

Figure 3.24: Source: [3]. Effect of both False positives and negatives. Chip capacity = 4^{10} probes. Assume $p = q$. The results are for 90% reconstruction success.

Bibliography

[1] F. P. Preparata A. M. Frieze and E. Upfal. On the power of universal bases in sequencing by hybridization. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB'99)*, pages 295–301, 1999.

[2] K. Doi and H. Imai. Sequencing by hybridization in the presence of hybridization errors. *Proceedings of the Workshop on Genome Informatics (GIW '00)*, 11:53–62, 2000.

[3] T. Hrtman E. Halperin, S. Halperin and R. Shamir. Handling long targets and errors in sequencing by hybridization. 2002.

[4] D.S. Hirschberg. Algorithms for the longest common subsequence problem. *J.ACM*, 24:664–675, 1977.

[5] N. Arbili I. Pe'er and R.Shamir. A computational method for resequencing by hybridization. *Proc. Natl Acad Sci.*, 99:15492–15496, 2002.

[6] N. Arbili I. Pe'er and R.Shamir. Advanced computational techniques for resequencing dna with polymerase signaling assay arrays. *Nucleic Acids Research*, 31(19):5667 – 75, 2003.

[7] D. Loakes and D.M. Brown. 5-nitroindole as a universal base analogue. *Nucleic Acid Research*, 22:4039–4043, 1994.

[8] P.A. Pevzner. l-tuple dna sequencing: Computer analysis. *Journal of Biomolcular Structure and Dynamics*, 7:63–83, 1989.

[9] F. P. Preparata and E. Upfal. Sequencing-by-hybridization at the information-theory bound: an optimal algorithm. *Journal of Computational Biology.*, 7:611–630, 2000.

[10] A. Krough R. Durbin, S. Eddy and G. Mitchison. *Biological sequence analysis : Probabilistic Models of Proteins and Nucleic Acids .* Cambridge University press, 1998.

[11] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceeding of the IEEE*, 77(2):257 – 286, 1989.