Analysis of Gene Expression Data **Spring Semester, 2002**

Lecture 5: April 26, 2002

Lecturer: Zohar Yakhini *Scribe: Eran Shalom, Anat Rapoport*

5.1 Introduction

5.1.1 Motivation

Custom-designed DNA arrays offer the possibility of simultaneously monitoring thousands of hybridization reactions. These arrays show great potential for many medical and scientific applications, such as polymorphism analysis and genotyping. Relatively high costs are associated with the need to specifically design and synthesize problem-specific arrays. Recently, an alternative approach was suggested that utilize fixed, universal arrays. This approach presents an interesting design problem - the arrays should contain as many probes as possible, while minimizing experimental errors caused by cross-hybridization. This lecture presents the work of Ben-Dor et al. [1] on the design problem. They use a simple thermodynamical model to cast this design problem in a formal mathematical framework. Employing new combinatorial ideas, they derive an efficient construction for the design problem and prove that the construction is near-optimal.

5.1.2 Universal DNA Tag Arrays

A DNA tag is a sequence of oligonucleotides. Oligonucleotides can hybridize with their Watson-Crick complements to form a stable DNA strand. A new approach of designing a DNA chip was suggested by S. Brenner and others [2, 3]. Their approach suggested using a fixed size array with a large set of tags, such that each tag will hybridize with its complement anti-tag, with minimal probability of hybridizing with the others. In general, the chip holds the anti-tags. An example of working with the chip is given below.

One application of working with universal arrays is SNP genotyping. Single nucleotide polymorphisms (SNPs), are DNA variations that occur on a single base in a specific site [5].

Assume that a polymorphic site exists, and we want to find out its content. The following can be done in order to achieve this goal:

- Let 'S' be the upstream sequence that immediately precedes the polymorphic site of the SNP. We create a reporter molecule consisting of two parts:

  1. A sequence $\overline{S}$, complementary to 'S'.

2. A tag 'T' that is complementary to an anti-tag $\overline{T}$ that is located on the chip.

- When an individual is to be genotyped, a sample is prepared that fully contains the SNP position (with 'S' preceding it, and some bases after it) (Figure 5.1). The sample is mixed with the reporter molecules in a solution. The hybridization then takes place (Figure 5.2).

- Polymerase is added to the solution. The polymerase needs a primer molecule (our reporter), and a pattern (the target being examined) in order for it to work. The polymerase is used in order to extend the reporter molecule with the right nucleotide i.e the one that is complementary with the polymorphic site on the target.

- Single dideoxy-nucleotides (A, T, G, C) are added to the solution. Each nucleotide is marked with a specific fluorescent color. We use dideoxy-nucleotides, because they will cause the polymerase to extend the reporter molecule, by a single nucleotide. The reporter molecule is being extended with one dideoxy-nucleotide (Figure 5.3).

- The reporter molecule are separated from the target, and brought into contact with the universal chip. The tags hybridize with the anti-tags (Figure 5.4). And finally the chips are being scanned to find out which nucleotide extended the reporter molecule, deducing the polymorphic site nucleotide.

When working with a universal chip, we wish to examine more than one SNP, thus we have to create multiple tags and anti-tags. When scanning the chip we have to know which anti-tag we are scanning, and to which reported molecule its tag is attached.

5.2 TAG system design problem

The problem we wish to solve is designing a set of tags such that the probability for cross-hybridization is minimized. That is, the probability of hybridization between a tag and a different anti-tag, or between tags is minimized.

In order to create such a set we need to consider the melting temperature of the DNA strand created by the hybridization of two sequences. As can be seen on Figure 6, the higher hybridization percent the lower temperature. The melting point of the molecule is higher as the number of mismatches is lower. Therefore, we can use temperature in order to decrease the number of mismatches. (Because when we heat the lotion a bit, the strands with the

Figure 5.1: Determining SNP site: the polymorphic site.

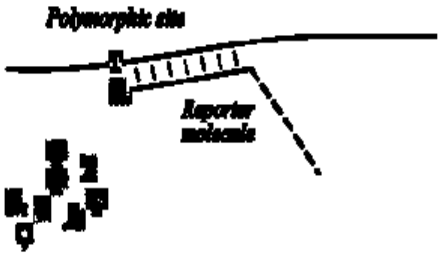

Figure 5.2: Determining SNP site: Reporter molecule attaches to target.

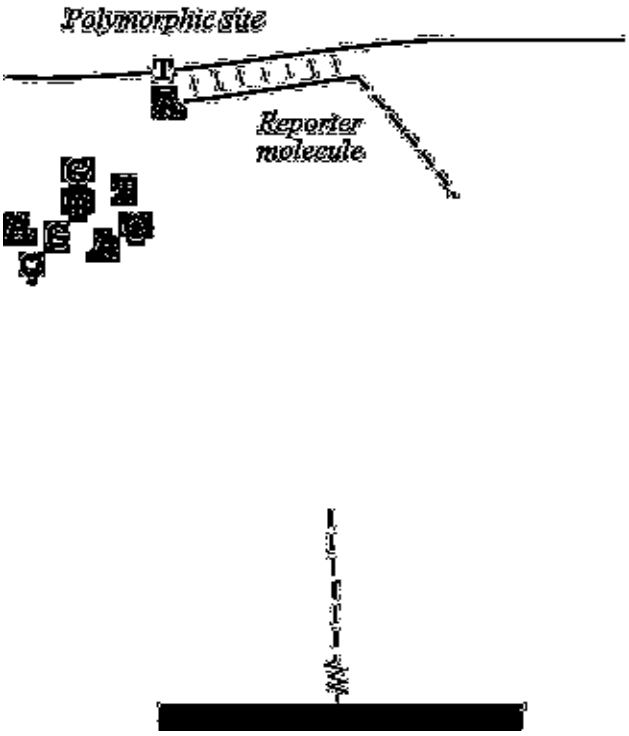

Figure 5.3: Determining SNP site: Polymerase connects dideoxy-A.

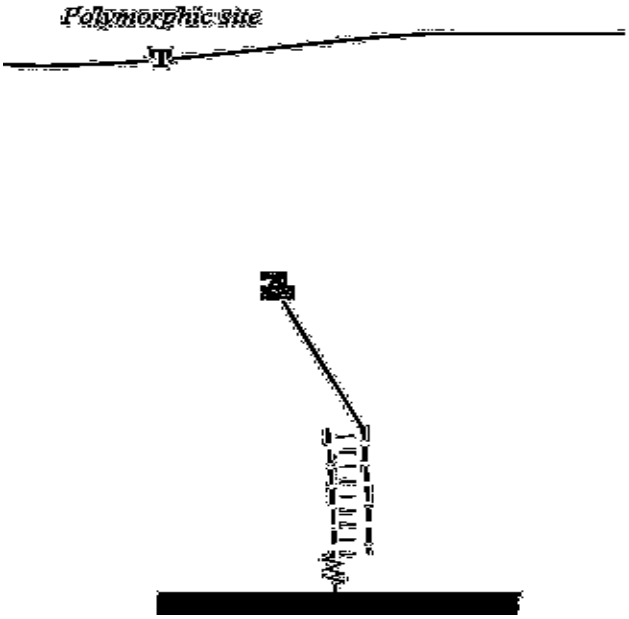

Figure 5.4: Determining SNP site: Tag attaches to anti-tag.

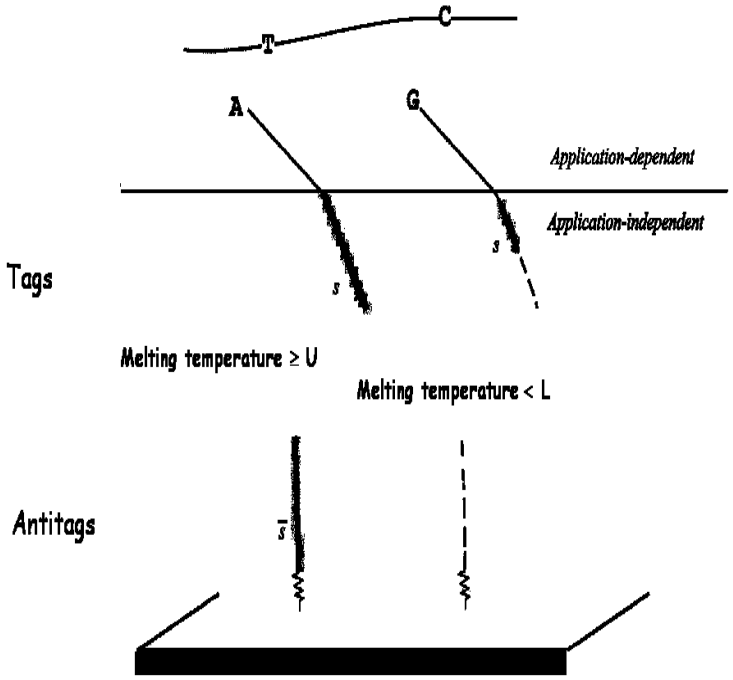

Figure 5.5: A universal chip.

least mismatches will remain connected with their anti-tag, and the other will disconnect, enabling another tag to connect with its own anti-tag).

Another parameter influencing the stability of a connection between a tag and its anti-tag is their length. The longer the sequences are the stronger the connection (yielding higher melting temperature). We use the following rule when computing the melting temperature [1, 4]: The melting temperature of a sequence and its complement is approximately the sum of twice the number of A-T pairs and four times the number of G-C pairs.

5.2.1 The Problem

We can now phrase the design problem as follows: Let t_M denote the melting temperature, and let H and C be two temperature thresholds with $H > C$ we require:

1. For each tag,anti-tag $(U, \overline{U})$, $t_M(U, \overline{U}) \geq H$.

2. For distinct tags U and V , and for any sequence of oligonucleotides λ, that occurs in both U and V, $t_M(\lambda, \overline{\lambda}) < C$.

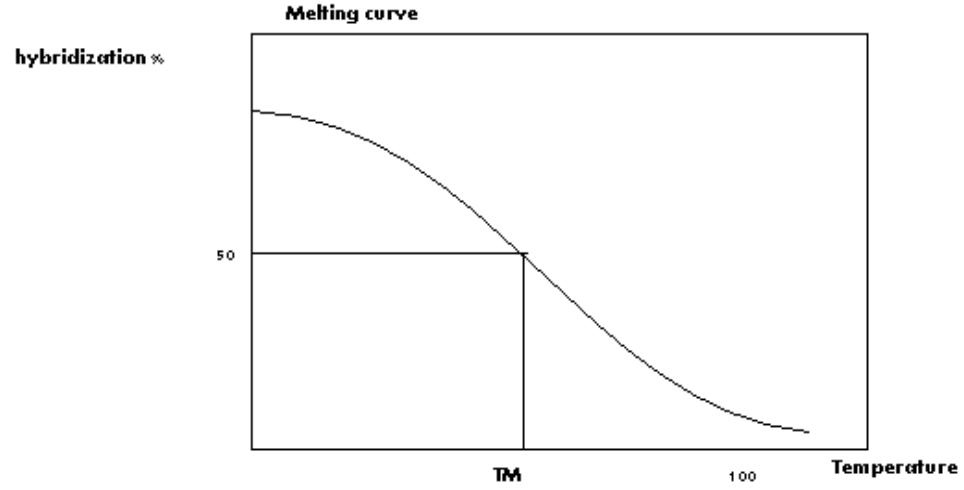

Figure 5.6: Melting curve.

3. For every λ, if $t_M(\lambda, \overline{\lambda}) \geq C$ then λ may occur only on one tag, and it may not repeat on that tag as well.

5.2.2 De-Brujin Sequences

Requirements 2 and 3 are quiet equivalent to a de-brujin sequence problem of rank λ: On an alphabet of length z, create a set of strings of length r, when no sub-string of length λ is repeated more than once. For Example:

- z = 2, r = 2, $\lambda = 2$: 00,01, 10, 11.
- z = 2, r= 3, $\lambda = 2$: 110, 001.

Combining requirement 1 we arrive at the following pronlem: Given two integers $\lambda < r$, find the largest set of tags such that:

- The length of each tag equals r.
- Each string of length $\geq \lambda$ occur at most once.

The solution would be to chop a de-brujin sequence of rank λ. Using the following process will get us $4^\lambda/(r-\lambda+1)$ tags (assuming an alphabet of size 4): Let S be a sequence of length 4^λ, when 4 is the alphabet size (A, T, G, C).

1. Start at the beginning of S, cur = 1.
2. Add a new tag $S(cur)...S(cur + r - 1)$.
3. Go back (λ - 1) positions (from cur + r - 1) in S.
4. Update cur.
5. If cur$\leq 4^{\lambda} - \lambda$ go to 2.

5.2.3 Thermodynamical Formalization

We now present a formalization of the above tag design problem. We not only forbid that a string x with $t_M(x, \overline{x}) \geq C$ occurs in any two distinct tags, but we also forbid that it occurs twice in the same tag. We model the oligonucleotides as strings over the alphabet $\sum = A, C, T, G$. We assume that the parameters C and H are fixed. To keep it simple, we assign to each string the number that corresponds to half of its melting temperature in degrees Celsius.

Definition The weight $w(s)$ of a string $s = a_1 a_2 ... a_k$ is $\sum_{i=1...k} w(a_i)$, where $w(A) = w(T) = 1$ and $w(C) = w(G) = 2$. Given two parameters c and h, we call a set T of strings or "tags" a **valid** $c - h$ code if the following two conditions are satisfied: **Condition 1.** Each tag has a weight of h or more. **Condition 2.** Any substring of weight c or more occurs at most once.

Example: weight(AACTTG) = 1+1+2+1+1+2 = 8, and melting temperature(AACTTG) = $2 \cdot$ weight.

Let's look at an example of a $4 - 10$ code:

```
GACCAAT     CAGCTAT     GTCGATA     CTGGTTA
CATTATCA    GAAATTCT    CTTAATGA    GTATTTGT
ATATAGTG    TAAAACTC    AATAAGAG    TTTTACAC
```

As will be shown in Theorem 6.4, this code is optimal. There exists no 4-10 code with more than 12 tags. In the next section we derive an upper bound that proves this claim.

5.2.4 Upper bound

We now describe the method of [1] for computing a tight upper bound on the number of tags in a valid $c - h$ code. The idea is to associate a numerical resource with each tag. We will show that any tag has to use a certain minimum amount of resource and the global resource usage over all tags cannot exceed a certain maximum. The upper bound on the number of

possible tags then follows from dividing the global upper bound by the minimum amount of resource used by each tag. The limited resource we consider consists of those substrings in a tag with a weight of c or more that can occur only once in a valid $c-h$ code. The following definition captures the minimal suffixes that can occur only once in a valid $c-h$ code.

Definition A string t is a c-token if :

1. $W(t) \geq c$.
2. No proper suffix of t has $weight \geq c$.

Definition The **tail weight** of a tag a is the sum of the weights of all characters that terminate tokens in a. For an example see Figure 5.7.

Tag T:	GACCAAT	Token's tail weight
	GAC	2
	CC	2
Tokens:	CCA	1
	CAA	1
	CAAT	1
		Tail weight of T: 7

Figure 5.7: Tokens and tail-weight. Here $c = 4$. Note that the first two characters do not terminate a token because they do not terminate a suffix of $weight \geq c$.

Lemma 5.1 *Any tag a in a valid $c-h$ code has tail weight of at least $h-c+1$.*

Proof: Consider a's prefix of weight smaller then $c-1$. All characters later then this contributes to tail-weight(a). The total weight is bigger then h. Only less then $c-1$ is lost. ■

We will now derive a bound on the total tail-weight afforded by a valid code T. Based on the fact that any c-token occurs at most once and that each tag has a weight of h or more, we now derive the upper bound on the total tail weight of a valid $c-h$ code. We use $< n >$ to denote the set of strings with weight $n \in N$, and G_n to denote the number of such strings. (Or formally: $E_n = \{t \in < A, C, T, G >^* : w(t) = n\}$ and $G_n = |E_n|$). It is straightforward to

derive the recurrence $G_1 = 2$, $G_2 = 6$ and $G_n = 2 \cdot G_{n-2} + 2 \cdot G_{n-1} + 2 \cdot G_{n-1}$ for $n \geq 3$, and for the sake of simplicity we define $G_0 := 1$.

Using standard techniques for solving recurrences, it can be shown that for all $n \in N, G_n$ is the nearest integer to

$$(\frac{3+\sqrt{3}}{6}) \cdot (1+\sqrt{3})^n$$

To compute the maximal total tail weight of the tokens in a valid code we will divide the tokens into four classes and compute the contribution of each class. We denote any character with weight 1 (A or T) by W ("weak") and a character with a weight of 2 (C or G) by S ("strong"). To see how the set of tokens is partitioned, observe that any token is terminated by either a strong or weak character, and it has a weight of either c or $c+1$. Tokens of weight $c+1$ begin with a strong character, since a string of weight $c+1$ that begins with a weak character probably contains a suffix of weight c. Table 1 contains the four classes of tokens, their maximal cardinalities, and the maximal total tail weight they can contribute.

Table 1:

Token class	*Max.occurrences invalidcode*	*Max.tail weight*
$< c-2 > S$	$2 \cdot G_{c-2}$	$4 \cdot G_{c-2}$
$S < c-3 > S$	$4 \cdot G_{c-3}$	$8 \cdot G_{c-3}$
$< c-1 > W$	$2 \cdot G_{c-1}$	$2 \cdot G_{c-1}$
$S < c-2 > W$	$2 \cdot G_{c-2}$	$2 \cdot G_{c-2}$

The total tail weight of each class is computed by multiplying its size by the weight of the terminal character of it's members. In the last row we can see that any token in the class $S < c-2 > W$ contains a token of form $S < c-2 >$ as a substring, and thus only $2 \cdot G_{c-2}$ tokens of this form can exist in a valid code. Therefore the number of tokens of the form $S < c-2 > W$ cannot exceed $2 \cdot G_{c-2}$. If we look at the rightmost column we get the following lemma:

Lemma 5.2 *The total weight of all tags contained in a valid $c-h$ code is at most:*

$$2 \cdot G_{c-1} + 6 \cdot G_{c-2} + 8 \cdot G_{c-3}$$

Combined with Lemma 1 we get the following upper bound:

Theorem 5.3 *The number of tags in any valid c-h code is at most*

$$\frac{2 \cdot G_{c-1} + 6 \cdot G_{c-2} + 8 \cdot G_{c-3}}{h - c + 1}$$

Example: For h=10 and c=4 the upper bound is $\frac{2*16+6*6+8*2}{10-4+1} = 12$ which proves: The 4-10 code which was shown previously is optimal.

5.2.5 Construction using Circular Strings

In this section we outline a method of constructing a nearly optimal $c-h$ code for arbitrary values c and h. The method consists of two stages. In the first, we construct a set of circular strings ("cycles") that is valid (each token occurs at most once). In the second we chop each cycle into overlapping tags. See Figure 5.8.

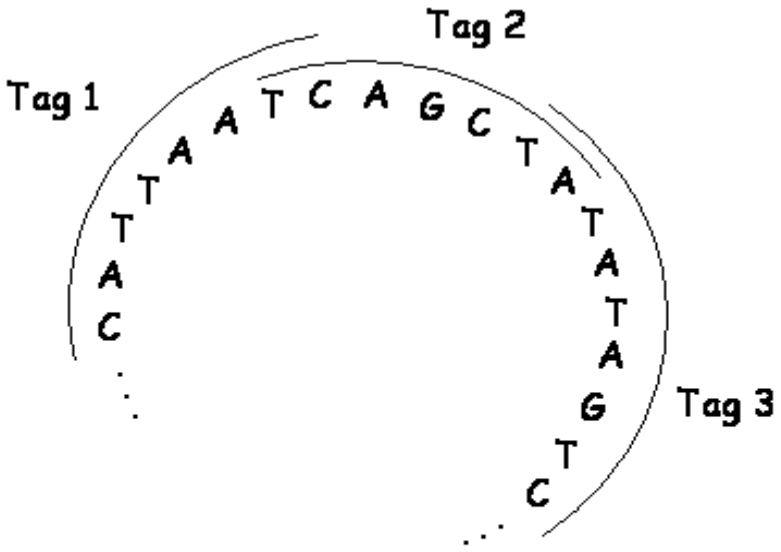

Figure 5.8: Circular strings.

- Each tag uses at most $(c - h + 3)$ in tail-weight.
- Loses tail weight of one partial tag per cycle.

This code contains at least $\frac{2 \cdot G_{c-1} + 6 \cdot G_{c-2} + 4 \cdot G_{c-3}}{h-c+3} - 1$ tags. Comparing this result with the upper bound, and using the recurrence of G_n, we find out that the method achieves at least a factor of approximately $0.89 \cdot (h-c+1)/(h-c+3)$ relative to the upper bound. For example for $c = 12$ and $h = 30$: the constructed code size is 12119, and the upper bound is 13840.

5.2.6 Universal DNA Tag Systems - Ongoing Work

- Generalization of the approach to nearest-neighbor thermodynamics. The construction can be done in heuristic methods, but in order to evaluate it we need a tight upper bound.
- Avoid repeating a fixed given set of tokens.

5.2.7 Summary

In this lecture we have shown:

- First analytic upper bound for universal DNA tag systems using common thermodynamics approximation.
- Constructive approach using circular strings for all values of c and h.
- Efficient constructive solution that is near optimal.

Bibliography

[1] A. Ben-Dor, R.M. Karp, B. Schwikowski, and Z. Yakhini. Universal dna tag systems: A combinatorial design scheme. *Journal Of Computational Biology.*, 7 Numbers 3/4:503–519, 2000.

[2] S. Brenner. Methods for sorting polynucleotides using oligonucleotide tags. *US Patent 5,604,097.*, 1997.

[3] M.S. Morris, D.D. Shoemaker, R.W. Davis, and M.P. Mittmann. Methods and compositions for selecting tag nucleic acids and probe arrays arrays. *European Patent Application 97302313.*, 1999.

[4] Strachan T. Humen molecular genetics. *John Wiley and Sons, New York.*, 1997.

[5] D.G et al. Wang. Large scale identification, mapping and genotyping of single nucleotide polymorphism in the human genome. *Science 280, 1077-1082.*, 1998.