Probe Selection with Fault Tolerance

Sheng-Lung Peng^{1,*}, Yu-Wei Tsay¹, Tai-Chun Wang¹, and Chuan Yi Tang²

¹ Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 974, Taiwan

² Department of Computer Science
National Tsing Hua University, Hsinchu 300, Taiwan
slpeng@mail.ndhu.edu.tw.

Abstract. Microarray techniques play an important role for testing some reactions of diseases which are caused by viruses. Probes in microarray are one kind of the most important materials. Usually, scientists use a unique probe for marking a special target sequence. Thus, for identifying n different viruses, we need n different probes. Recently, some researchers study non-unique probes to identify viruses by using less number of probes. In this case, a virus can be identified by a combination of some probes. In this paper, we study the problem of finding a set of probes that can identify all the given targets. We consider the k -fault tolerance selection of probes. That is, if any k probes fail, then we still can identify each target. We propose a practical algorithm for this k -fault tolerance probe selection problem. Some experiments are studied on SARS, H5N1, and so on.

1 Introduction

Research for the human nosography and immunology is important today [1]. SARS, AIV, cervical cancer, acquired immunodeficiency syndrome (AIDS), and so on, are all infected from viruses. Many researchers use microarray for clinical experiments [4,8]. A microarray usually uses parallel analysis for an expression of thousands of genes. Because it can be formed by different types (*e.g.*, oligonucleotide and cDNA), the applications of microarray are very wide [5].

Recent works or softwares try to find unique probes for the target sequences [2,11,12]. That is, each probe can only hybridize with one target sequence. In this case, for n different targets, we need n different probes to identify these targets. If we allow one probe fault, then for the premise of unique sequences, each target will need two unique sequences as probes. It is not hard to see that the number of probes will be doubled. It is very impractical. Thus, some researchers focus on non-unique probes [9,10]. Since each of non-unique probes can hybridize with more than one target, fewer probes are needed. In this case, a target is identified by a combination of selected probes.

Klau *et al.* proposed an integer linear programming algorithm to find a minimum set of non-unique probes which can identify all the input targets [6,7].

* Corresponding author.

In this paper, we propose a simple greedy algorithm to find a minimal set of non-unique probes for recognizing the input targets.

2 Preliminaries

A *probe* is a single-strand DNA or RNA molecule with a specific sequence labeled either radioactively or immunologically. It is used to detect the complementary sequence by hybridization for identifying unknown species or gene functions.

When designing probes, we need to get a single-strand DNA or RNA sequence. Then we use primers to help us produce the complement subsequences of the ss-DNA (RNA). Each of them is between 25 and 80 bp. To obtain probes, we need to consider the temperature, GC contents rate, DNA secondary structure, and cross-hybridization. Most known probes are stored in NCBI GeneBank. There are several softwares can help us to find probes, such as MAPS [2], GoArray [11], OligoArray [12], and so on.

Figure 1 shows an example. For unique probes, each target needs one unique probe to identify itself. In the non-unique probes, we use less probes to identify all the targets.

Fig. 1. Unique and non-unique probes

Table 1 shows a *relation table* according to Figure 1. In this table, if target T_i can hybridize to probe P_j , then the (i, j) entry is 1. According to this table, it is clear that we need four probes to identify all the targets by using unique probes. But for non-unique probes, two probes, namely, P_1 and P_3 , are sufficient. That is an advantage of using non-unique probes. Furthermore, in large transcript families, such as alternative splice variants of a gene, or in a large family of closely homologous genes, it is often impossible to find a unique probe of 25 bp as a signature for a specific variant [10]. So we consider about non-unique probes.

A *microarray* is an array containing thousands of probes. These probes are printed on a solid surface. Sometimes it is a glass or nylon membrane.

The sensitive of a microarray will be influenced by the reaction of target's concentration, temperature, and reaction time. Moreover, we also need to compute

Table 1. The relation matrixes for Figure 1 (A) and (B) respectively

	P_1	P_2	P_3	P_4
T_1	1	0	0	0
T_2	0	1	0	0
T_3	0	0	1	0
T_4	0	0	0	1

(A)

	P_1	P_2	P_3	P_4
T_1	1	0	0	1
T_2	1	0	1	0
T_3	0	1	1	0
T_4	0	1	0	1

(B)

the diffusion coefficient, adhesive force coefficient, the uniform level of reaction targets, and so on. Some researches show that a long oligonucleotide is more sensitive than short one ([3]). However, the design problem is very complicated. Chung *et al.* try to use a hierarchical method to design probes [3]. For exon chip, the length of the probes on the chip can be different. But for GeneChip, its length has the limitation of 25 bp.

Sometimes probes will fail because of the pollution during producing a microarray. So we need to add extra probes in order to make sure if some probe fails, then there still has another probe can take care of it. Table 2 shows the case that if any probe fails, then the probe set still can identify the four targets.

Table 2. Probe sets with 1-fault tolerance for Figure 1 (A) and (B) respectively

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
T_1	1	1	0	0	0	0	0	0
T_2	0	0	1	1	0	0	0	0
T_3	0	0	0	0	1	1	0	0
T_4	0	0	0	0	0	0	1	1

(A)

	P_1	P_2	P_3	P_4
T_1	1	0	0	1
T_2	1	0	1	0
T_3	0	1	1	0
T_4	0	1	0	1

(B)

3 Methods

In this section, we will introduce our greedy algorithm for finding a set of non-unique probes with k -fault tolerance.

3.1 Data Preprocessing

In our method, we use the software **Promide** proposed by Rahmann to find all the possible probes [9]. The input file is in FASTA format. After finding the

Table 3. An example of target-probe relation table

	P_1	P_2	P_3	P_4	P_5	P_6
T_1	1	0	0	1	1	0
T_2	0	1	0	0	1	1
T_3	1	1	1	1	1	1
T_4	0	1	1	1	0	0

Table 4. A distinguishable table obtained from Table 3

Target pairs	(T_1, T_2)	(T_1, T_3)	(T_1, T_4)	(T_2, T_3)	(T_2, T_4)	(T_3, T_4)
P_1	1	0	1	1	0	1
P_2	1	1	1	0	0	0
P_3	0	1	1	1	1	0
P_4	0	0	0	1	1	1
P_5	0	0	1	0	1	1
P_6	1	1	0	0	1	1

probes, we construct a relation table between probes and targets. Table 3 shows an example. For each target, if a probe can bind the target, we then assign the value 1 in the corresponding entry. Thus, we know which probe can affect which target. For distinguishing the targets, we translate the relation table into the *distinguishable table* (see Table 4). In the case that different probes have the same distinguishable ability, we only keep one. In Table 4, T_1 can hybridize to probe P_1 , but T_2 cannot. That is, P_1 can distinguish T_1 and T_2 . Thus, we record the value 1 in the $(P_1-(T_1, T_2))$ entry.

3.2 Relation Graph

Take Table 4 as an example. In order to illustrate the relationship between probes and target pairs, we use the distinguishable table to draw a *relation graph* (see Figure 2). In this graph, each edge denotes that the probe can differentiate the two targets in the target pair. By using non-unique probes, each target pair has more than two incident edges. That is, if we remove any one node from the probe set, any node in the target pairs still has at least one incident edge. It means that each target pair still can be distinguished by one probe. Based on this fact, it is not hard to see that a probe set is k -fault tolerance if every target pair in the graph has at least $k + 1$ edges. Figure 3 shows an example of 1-fault tolerance. In this graph, every target pair has at least two edges and any failure probe will not affect the distinguishable ability.

Fig. 2. The relation graph of Table 4

Fig. 3. An induced subgraph of Figure 2

Table 5. The main variables used in our algorithm

Input variables	
<i>relationArray</i>	It is a $n \times m$ matrix, where n is the number of possible probes and m is the number of target pairs.
<i>ftValue</i>	The number of fault tolerance.
Output variables	
<i>coverArray</i>	It is an array of size n . Each entry stores the number of target pairs that can be distinguished by the corresponding probe.
<i>ftArray</i>	It is an array of size m . Initially, every entry is 0. If every entry is at least $k + 1$, then the found probe set is k -fault tolerance.
<i>identifyArray</i>	It is an array of size m . Each entry stores the number of probes that can be used to identify the target pair.
<i>ProbeSet</i>	A probe set outputted by our algorithm.

3.3 Probe Selection Algorithm

After constructing the relation graph, we start to find a probe set. We give some definition first for the variables which are used in our algorithm in Table 5.

Our algorithm is shown in Algorithm 1. It can be divided into three parts. The first part is from step 7 to step 10. This part helps us to find the smallest number in the *identifyArray*. By our experience, if the target pair that can be distinguished by few probes, then considering it first will obtain a good solution.

After finding a candidate target pair in the first part, we then go to the second part from step 13 to step 16. In this part, we find the largest number in the *coverArray*. That is, we select the probe that can distinguish the maximum number of target pairs.

The final part of our algorithm is to update all the related information after we select a probe. If the requirement of the fault tolerance does not fulfill, we then repeat the process again until we fulfill the requirement.

Algorithm 1. ProbeSelect(*relationArray*, *n*, *m*, *ftValue*)

```

1: ftArray[i] = 0 for all  $i \in \{1, \dots, m\}$ 
2: ProbeSet =  $\phi$ 
3: loop = .T.
4: while loop do
5:   temp =  $\infty$ 
6:   targetPair = 0
7:   for  $i = 1$  to  $m$  do
8:     if ftArray[i] < ftValue and identifyArray[i] < temp then
9:       targetPair = i
10:      temp = identifyArray[i]
11:   temp = 0
12:   probe = 0
13:   for  $i = 1$  to  $n$  do
14:     if relationArray[i][targetPair] = 1 and temp < coverArray[i] then
15:       probe = i
16:       temp = coverArray[i]
17:   ProbeSet = ProbeSet  $\cup$  {probe}
18:   loop = .F.
19:   for  $i = 1$  to  $m$  do
20:     if relationArray[probe][i] = 1 then
21:       identifyArray[i] = identifyArray[i] - 1
22:       ftArray[i] = ftArray[i] + 1
23:     if ftArray[i] < ftValue then
24:       loop = .T.
25: return ProbeSet

```

The algorithm first finds the minimum number of *identifyArray* (e.g., *identifyArray*[1]). Then it finds the maximum number in *coverArray* for target pair 1. Since probe 1 and probe 6 have the same value, we choose probe 1. After choosing probe 1, we update the remaining information. The selected probes are shown in Table 6, i.e., $\{P_1, P_3, P_6\}$. It is not hard to check that any failure of selected probes does not loss the distinguishing ability for the input targets.

Table 6. The final result of our algorithm

(T_1, T_2)	(T_1, T_3)	(T_1, T_4)	(T_2, T_3)	(T_2, T_4)	(T_3, T_4)
P_1	P_6	P_1	P_1	P_6	P_1
P_6	P_3	P_3	P_3	P_3	P_6

4 Results

In this section, we show the result by using our algorithm to find a set of non-unique probes for some viruses (e.g., SARS, HIV, avian's influenza A virus, and so on) with the ability of k -fault tolerance. Our virus data were downloaded from

Table 7. An experimental result of our algorithm

Dataset	No. of targets	No. of probes	0-FT selected	1-FT selected	2-FT selected	4-FT selected
SARS	121	1041	75	140	207	336
H5N1	136	1156	79	142	222	387
AIV	484	4356	216	252	730	1203
Retroviruses	54	10169	53	105	157	262

NCBI Database (<http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html>). The avian's influenza A virus was downloaded from the Influenza Virus Project web site which is maintained by NCBI. All the data are stored in FASTA format.

In the experiment of AIV's data, we mix all the H5N1's PB2 sequences and other PB2 sequences of different subtypes of avian's influenza A virus. Table 7 shows the result. The number k in k -FT (k from 0 to 4) is the fault tolerant number.

In our experiment, if the data are highly conservative such as SARS, then it is easy for **Promide** to find many possible non-unique probes. On the other hand, if the data are not highly conservative, **Promide** will find a lot of unique probes. In this case, our algorithm obtains a lot of probes like Retroviruses.

5 Discussion

The production of microarrays is costly. An important factor is the number of probes used in the microarray. Besides, the precision of tests for microarrays is another factor. These motivate this research. By the experimental result, our method shows that we can reduce the number of probes used for identifying several groups of viruses. Further, our method can be easily extended to k -fault tolerance, *i.e.*, if there are k failure probes, the remaining probes can still be used to recognize the input viruses.

In fact, our probe selection algorithm is an approximation algorithm. If the input size is not too large, we may use a branch-and-bound algorithm to find an optimal solution. Second, for the tool **Promide**, it is unable in dealing with an oversize sequences. Thus, it is necessary to find or design a new tool for large scale input size. Finally, while the cost of building microarrays is down, fault tolerance becomes an important issue. Our method obtains only an approximated result. Designing new algorithms considering fault tolerance will be a new direction.

References

1. Abbas, A.K., Lichtman, A.: Cellular and Molecular Immunology, 5th edn. Saunders, Philadelphia (2005)
2. Bushel, P.R., Hamadeh, H., Bennett, L., Sieber, S., Martin, K., Nuwaysir, E.F., Johnson, K., Reynolds, K., Paules, R.S., Afshari, C.A.: Maps: a microarray project system for gene expression experiment information and data validation. *Bioinformatics* 17, 564–565 (2001)

3. Chung, W.H., Rhee, S.K., Wan, X.F., Bae, J.W., Quan, Z.X., Park, Y.H.: Design of long oligonucleotide probes for functional gene detection in a microbial community. *Bioinformatics* 21, 4092–4100 (2005)
4. Cooper, C.S.: Applications of microarray technology in breast cancer research. *The Institute of Cancer Research* 3, 158–175 (2001)
5. He, Y.D., Dai, H., Schadt, E.E., Cavet, G., Edwards, S.W., Stepaniants, S.B., Kleinhanz, S.D.R., Jones, A.R., Shoemaker, D.D., Stoughton, R.B.: Microarray standard data set and figures of merit for comparing data processing methods and experiment designs. *Bioinformatics* 19, 956–965 (2003)
6. Klau, G.W., Rahmann, S., Schliep, A., Vingron, M., Reinert, K.: Optimal robust non-unique probe selection using integer linear programming. *Bioinformatics* 20(Suppl. 1), i186–i193 (2004)
7. Klau, G.W., Rahmann, S., Schliep, A., Vingron, M., Reinert, K.: Integer linear programming approaches for non-unique probe selection. *Discrete Applied Mathematics* 155, 840–856 (2007)
8. Petricoin, E.F., Hackett, J.L., Lesko, L.J., Puri, R.K., Gutman, S.I., Chumakov, K., Woodcock, J., Feigal Jr., D.W., Zoon, K.C., Sistare, F.D.: Medical applications of microarray technologies: a regulatory science perspective. *Nature Genetics Supplement* 32, 474–479 (2002)
9. Rahmann, S.: Rapid large-scale oligonucleotide selection for microarrays. In: *The First IEEE Computer Society Bioinformatics Conference*, pp. 54–63 (2002)
10. Rahmann, S., Muller, T., Vingron, M.: Non-unique probe selection by matrix condition optimization. *Currents in Computational Molecular Biology* (2004)
11. Rimour, S., Hill, D., Milton, C., Peyret, P.: Goarrays: highly dynamic and efficient microarray probe design. *Bioinformatics* 21, 1094–1106 (2005)
12. Rouillard, J.M., Zuker, M., Gulari, E.: Oligoarray 2.0: design of oligonucleotide probes for dna microarrays using a thermodynamic approach. *Nucleic Acids Research* 31, 3057–3062 (2003)