

Consolidating Software for DNA Microarray Design and Manufacturing*

M. Atlas, N. Hundewale, L. Pereyagina, A. Zelikovsky

Department of Computer Science and Department of Biology, Georgia State University, Atlanta, GA, USA
 matlas@iman.cs.gsu.edu, nisar@cs.gsu.edu, biolmp@panther.gsu.edu, alexz@cs.gsu.edu

Abstract—As the human genome project progresses and some microbial and eukaryotic genomes are recognized, a novel technology, DNA microarray (also called gene chip, biochip, gene microarray, and DNA chip) technology, has attracted increasing number of biologists, bioengineers and computer scientists recently. This technology promises to monitor the whole genome at once, so that researchers can study the whole genome on the global level and have a better picture of the expressions among millions of genes simultaneously. Today, it is widely used in many fields- disease diagnosis, gene classification, gene regulatory network, and drug discovery.

In this paper, we present a concatenated software solution for the entire DNA array flow exploring all steps of a consolidated software tool. The proposed software tool has been tested on Herpes B virus as well as simulated data. Our experiments show that the genomic data follow the pattern predicted by simulated data although the number of border conflicts (quality of the DNA array design) is several times smaller than for simulated data. We also report a trade-off between the number of border conflicts and the running time for several proposed algorithmic techniques employed in the physical design of DNA arrays.

Keywords—DNA, microarray, probe selection, chip design

I. INTRODUCTION

In this paper we describe the main flow of the producing a DNA chip, methods employed to read a genomic data from database and producing ORF's in FASTA format. We illustrate the models used for probe selection, Promide and Tm checker, with specific details regarding the production of pool of probes that must optimize hybridization affinity, and satisfy the constraints on the melting temperature. We describe VLSI CAD models of chip-physical design, and how to engage CAD algorithms in probe placement and embedding. We discuss mask and array manufacturing process that is a combination of photolithography and combinatorial chemistry, as well as a hybridization experiment and the analysis of gene intensities. We discuss a specific application of the DNA chip production to analyze the genome of the Herpes virus B2, and we give the experiments results. We present the DNA Array flow (see Figure 1). In the next section we describe each step separately and also point to the corresponding software tools that were consolidated into a flow. In section three, we describe experimental design.

II. DNA ARRAY FLOW

In this section we discuss each stage of DNA Array Flow.

1. Reading Genomic Data and ORF Extraction: In this step, we will use ORF extraction programs to extract the desired ORF and we will check them. There are two approaches to accomplish this. The first one is ORF-Finder; however, the other approach is to use GenMark.

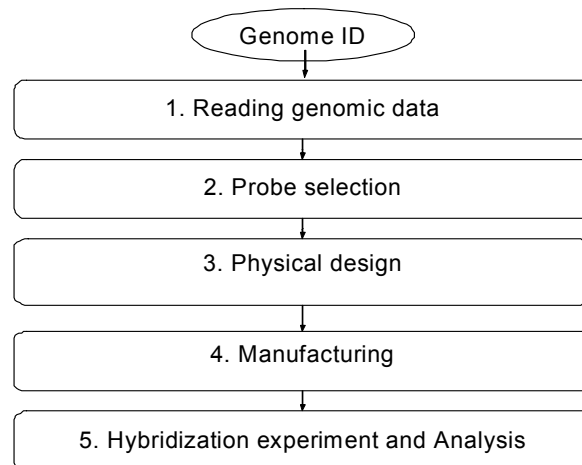


Figure 1. DNA Array Flow

1.1 ORF Extraction Using ORF Finder: Extracting ORF sequence from the genome sequence is straightforward using ORF Finder. ORF can be extracted by means of the genome's sequence or id. ORF Finder is an extremely simple program at the NCBI that can be used to visualize open reading frames in a DNA sequence. An open reading frame is simply a subsequence of DNA that could potentially be transcribed into mRNA. Because of the differences between prokaryotic and eukaryotic transcription systems, there are two definitions of an open reading frame. In prokaryotes, an open reading frame is defined as a DNA subsequence between a start and stop codon. In eukaryotes, an open reading frame is simply a list of subsequences that are terminated by a stop codon.

ORF Finder uses the prokaryotic definition of an open reading frame, and therefore is only useful in detecting the initial exons in a eukaryotic gene sequence. Each of these ORFs represents a potential initial exon in a gene. The interface to ORF Finder is extremely functional, allowing the user to select and BLAST individual ORFs, which can be a somewhat effective way of finding unknown genes with initial exons that are similar to known genes. Also, the documentation and support available from the NCBI is good, especially considering the simplicity of ORF Finder. However, ORF Finder is a very limited tool for predicting

* The work is partially supported by NIH Award 1 P20 GM065762-01A1 and P40 RR05162.

genes within DNA sequences, and there are many more other programs available. As one of these other programs GenMark.

1.2 ORF Extraction Using GenMark: GenMark server provides e-mail based identification of protein coding regions in DNA sequences from prokaryotic and eukaryotic species. GeneMark server accepts a formatted message containing a DNA sequence in text format (numbers and spaces allowed, see below). Orflist is a function in GenMark, which is responsible for producing a list of evaluated open reading frames. The option 'Orflist' instructs to return a list of areas of the DNA sequence where GeneMark has identified a coding region.

In the ORF list output, the “left end” and “right end” columns denote the physical position of the ORF in the sample sequence in absolute nucleotides relative to the beginning of the sample sequence. “DNA strands” indicates the strand in which the ORF lies. “Coding frame” indicates the reading frame in which the ORF lays relative beginning to the sequence. “Avg prob” denotes the mean coding potential over the indicated range and reading frame. “Start Prob” denotes the start probability, which is a measure of the likelihood that indicated start marks the true beginning of the ORF. There are several outputs; however, our concern will focus on the physical position and the name of the ORFs.

1.3 ORF Parser: In this step, we used the ORFs produced by GenMark because of the accuracy of GenMark over ORF Finder. In this step we parse the ORFs produced by GenMark to suit the input format acceptable by Promide. The output of GenMark is lists of start and end positions of each ORF along with their names. The parser locates each ORF using their positions within the entire sequence and creates a file with all ORF's in FASTA format. The content of pool of ORF's using provided information from GenMark is parsed to extract the target of the sequence in question.

Although, Perl has strong native string matching capability; however, working at the character level is not as efficient compared to C++ or Java that is why we choose to implement the first parser using C++. The convenience here is that each time the genome sequence changes, we do not have to rewrite our parser.

2. Probe Selection: This step is analogous to the logic synthesis in VLSI design; the probe selection step is responsible for implementing the desired functionality of the DNA array. Although probe selection is application dependent, many underlying criteria are common all DNA micro-array technologies. The challenge for probe selection is how to identify the optimum probes for each gene. The basic criteria for the probes are the following:

The probe should be a unique sequence in the all-coding sequences of the target genome, even with allowable number of mismatches, should not be self complementary, be close to the 3' end of that gene, does not contain single

nucleotide multiple repeat regions, such as AAAAAA, does not a representative of low complexity region, and should have almost the same GC percentage as that of the target genome.

Empirically, the optimum probe for a gene would be the one with minimum hybridization free energy for that gene (under the appropriate hybridization conditions) and, maximum hybridization free energy for all other genes in the hybridization pool.

Probe selection presents the first algorithm that can select suitable custom oligonucleotide probes (e.g. 25-mers) for microarray experiments on a truly large scale. For example, oligos for human genes can be found within 50 hours. This becomes possible by using the longest common substring as a specificity measure for candidate oligos. Promide [4] presents an algorithm based on a suffix array with additional information that is efficient both in terms of memory usage and running time to rank all candidate oligos according to their specificity. Promide also introduce the concept of master sequences to describe the sequences from which oligos are to be selected. Constraints such as oligo length, melting temperature, and self-complementarity are incorporated in the master sequence at a preprocessing stage and thus kept separate from the main selection problem. We want to select 25-mers for every known ORF of budding sequence using Promide. We assume that the oligonucleotide probe sequence is the coding sequence of the gene; this choice depends on the type of assay being used. We already obtain a FASTA file with all genome ORF sequences using GenMark and Parser1.

2.1 Checking Temperature of Hybridization: In this step, we check the temperature of our probe that we produce from Promide. We calculate the temperature using the formula, $2 * \text{number of A and number of T} + 4 \text{ times the number of G and C}$. Using this formula, we find that the temperature chosen before in Ocand is closed to the one calculated.

3. Physical Design: This is the main step of the flow. Its inputs are pools of probes from the probe selection step. In this step also, even though we face the conflict problem of the nucleotides, the Chip program introduces new algorithms to reduce the number of conflicts when we place and embed the probes. The Chip program also gives running time for each algorithm.

Physical design for DNA arrays is equivalent to the physical design phase in VLSI design. It consists of four steps: *deposition sequence design*, which is a basic optimization in DNA array design, is minimizing the number of synthesis steps, or, equivalently, minimizing the number of photo-lithographic masks used in the manufacturing process; *test control*, which is equivalent of built-in self-test (BIST) structures in VLSI design, and aim at detecting catastrophic manufacturing defects, i.e., defects that irrevocably compromise the functionality of the DNA chip. Additionally, DNA chip designs incorporate control

structures for ensuring reliable interpretation of results; *probe placement*, which is responsible for mapping selected probes onto locations on the chip, and *probe embedding*, which embeds each probe into the deposition sequence (i.e., determines synthesis steps for all nucleotides in the probe). The result of physical design is the complete description of the reticles (photomasks) used to manufacture the microarray.

3.1 Deposition Sequence Design: Fundamental optimization in DNA array design is minimizing the number of synthesis steps, or, equivalently, minimizing the number of photolithographic masks used in the manufacturing process. Current methodologies use a predefined deposition sequence, typically periodic. Chip authors propose to optimize the deposition sequence with respect to a given set of selected probe pools, and add a feedback loop to provide updated design rules and parameters to the probe selection step. So, the number of synthesis steps affects manufacturing time and the cost of the mask set, and also directly affects the quantity of defective probes synthesized on the chip. Therefore, a basic optimization in DNA array design is to minimize the number of synthesis steps. In the simplest model, this optimization has been reformulated as the classical shortest common super sequence (SCS) problem [9, 10]

3.2 2-D Probe Placement: Under ideal manufacturing conditions, the functionality of a DNA array is not affected by the placement of the probes on the chip, or the particular order in which nucleotides of each probe are synthesized. In practice, since manufacturing process is prone to errors, probe placement and synthesis schedules affect to a great degree the hybridization sensitivity and ultimately the functionality of the array. There are several types of synthesis errors that take place during array manufacturing.

3.3 3-D Probe embedding: Recently, in [2], Kahng et al introduced the border minimization problem for the asynchronous synthesis regime, which allows arbitrary probe embeddings. Asynchronous synthesis has identical technological requirements with synchronous synthesis. Asynchronous synthesis is already mandated by minimization of the number of synthesis steps. At the same time, asynchronous embedding offers more flexibility for reducing total border length.

4. Manufacturing: In this step, DNA array goes through a combination of photolithography and combinatorial chemistry process, resulting in many of the arrays' powerful capabilities. With a calculated minimum number of synthesis steps, DNA arrays with hundreds of thousands of different probes are packed at an extremely high density. This feature enables researchers to obtain high quality, genome-wide data using small sample volumes. Manufacture is scalable because the length of the probes,

determines the number of synthesis steps required. This automated production process yields arrays with highly reproducible properties, which reduces user set-up time by eliminating the need for individual labs to produce and test their own arrays.

5. Hybridization Experiments and Analysis: In this step, hybridization experiments are performed. During the hybridization experiment label of each probe is quantified, and probes are diluted so that all are at an equal concentration. Usually, a duplicate filter or micro-array is prepared for each probe to be assayed. Probes are hybridized separately with each array. Filter arrays are incubated with probe and washed in much the same way as is done for Southern or Northern blotting. For glass microarrays, hybridization is done under a cover slip. Dipping into wash solutions washes slides along with cover slips. Commercially produced arrays come in cassettes, in which hybridization, washing, and detection is done.

III. EXPERIMENTAL STUDY

Herpes B virus is a member of the subfamily *Alphaherpesvirinae* from the genus Simplex virus. Herpes B (HB) virus is mild localized or asymptotic infection in its internal hosts [7]. In contrast HB virus infection in foreign host, humans or monkeys species other than macaques often result in encephalitis, encephalomyelitis, and death. Herpes B virus genome is 156,798 bps long and includes 74 genes [7]. In this study, we design a chip to carry on experiments to study HB virus. The chip should contain 20 to 50 probes for each of 74 genes. To do that we follow the following step:

First, we extract the genome sequence from GenBank using BioPerl [1] tools by giving the genome id to Genbank [8]. After getting the genomic data, we feed it to GenMark [8], which gives a set of ORF for Herpes B Virus. The ORF generated from GenMark is going to be the input for Probe Selection step, Promide [4]. However, the format provided by GenMark is not suitable for Promide input; therefore we use our parser Parser1 as tool to change the format of the input. Parser1 get as input the ORF's name, their left and right physical position. After parsing the data, Parser1 produces a set of ORF in FASTA format. The ORF generated from Parser1 will be provided to Promide for probe selection [4] that is to generate a set of pools of probes. Since melting temperature play very important role in DNA arrays, Promide gives a freedom of choosing the melting temperature for probes. Checking melting temperature [5] for probes will be offered by Sigma-Genosys. We use the temperature 60 and 65 as best melting temperature to carry on our experiments. [4, 6], the generated pool of probes from both temperatures has the same targets but the average pool size for each temperature is different. The pool of probes spawned will be nosh to the Chip program [2]. However, the pool of probes given is not in the proper format of Chip's input. For this reason we

create the second parser for DNA array flow, "Parser2". Parser 2 will read probes from Promide- which in the form of ACTG- and output them in the format-0123- where 0 stand for A, 1 for C, 2 for T and 3for G. Beside converting the format of Promide, our parser can choose from the pool of probes the number of candidate for each probe, which make a powerful tools when we want to change the chip size. The pool of probes produce by Parser will be given to Chip program to produce a chip and compare the number of conflict [3] for each algorithm, CPU time usage of Herpes B virus and simulated data. We will repeat the experiment for different chip size, temperature and number of candidate chosen from each probe.

The Chip Program does not allow us to read a real data [3]; it uses random generated pool of probes. We added a module to the Chip program that can read data from a file "Readpool". This module read real data file chooses a number of candidate wanted for each probe. Readpool model measures a number of conflicts between probes and gives CPU time for each algorithm.

The goal of this study is to design a chip for HB virus. To measure the quality of our design we have to minimize the number of conflict between the probes. For this reason we perform the following experiments in HB virus using the chip program.

The results of the experiment, using following parameters, are presented in Tables 1 and 2.

Melting Temperature: In our experiment, we choose 65°C as the melting temperatures for our DNA probe array.

Number of Candidates (K): We experimented with different values of K (number of candidates) for each pools of probes: 1 and 2. **Chip Size:** We ran our Experiments with chip size 60x60. **Pool Size:** In order to design a chip of 60x60, we selected 47 probes from each set of probes.

The algorithms in Chip Program as well as Promide were implemented in C. The parser1 was implemented in C++; however, parser 2 was implemented in Perl. We run all the code on Linux server. The experiments were run on randomly generated data and herpes B Virus genome data.

K=1	Herpes B Virus		Simulated Data	
	# Conflicts	CPU Time(sec)	# Conflicts	CPU Time(sec)
Initial	107577		265992	
Tsort	98830	0.17	231526	0.08
Tsp	95640	0.22	227960	0.09
Lalign	79254	0.25	189272	0.1
Reptx 2	64830	4.45	154766	1.58
Chessboard	63594	15.58	150812	7.1

Table 1

K=2	Herpes B Virus		Simulated Data	
	# Conflicts	CPU Time(sec)	# Conflicts	CPU Time(sec)
Initial	54205		265328	
Tsort	49746	0.3	232954	0.14
Tsp	48541	0.34	227762	0.15
LAlign	42858	0.42	182972	0.16
Reptx 2	32098	7.84	149332	3.16
Chessboard	31498	20.93	146708	10.89

Table 2.

IV. CONCLUSION AND FUTURE WORK

Our experiments show that the genomic data follow the pattern predicted by simulated data. In case of Herpes B virus, like simulated data, increasing number of candidates per probe (k) decreases number of border conflicts during the probe placement algorithms. However, the number of border conflicts is several times smaller than for simulated data. We give the trade-off between number of border conflicts and the CPU time taken for the various algorithms that are defined in the physical design.

In this paper, we give a concatenate software solution for the entire DNA array flow, and we explore all steps in a single automated software suite of tools. We suggest that the entire software suite be made available through web services, so that users can enter name of organism, whose DNA sequence is already available at GenBank, and with an option of choosing to set the required parameters the suite will produce the DNA probe micro-array chip layout. Web Services have increasingly been of interest because of the ease and simplicity of usage. With Web Services the usage complexity of several tools can be abstracted. Users do not have to manually execute the tools and utilities. Considering the users could be biologists or those who may not be computer wizards, web interface would provide added simplicity for experimenters to effectively use the suite.

The consolidated software for DNA microarray design and manufacturing is available through web at the URL <http://alla.cs.gsu.edu/~alexz>.

REFERENCES

- [1] <http://www.bioperl.org>
- [2] A.B. Kahng, I.I. Mandoiu, P.A. Pevzner, S. Reda, and A. Zelikovsky, "Border Length Minimization in DNA Array Design", *Proc. 2nd International Workshop on Algorithms in Bioinformatics (WABI 2002)*, R. Guigo and D. Gusfield (Eds.), Springer-Verlag Lecture Notes in Computer Science Series 2452, pp. 435-448.
- [3] A.B. Kahng, I.I. Mandoiu, P.A. Pevzner, S. Reda, and A. Zelikovsky, "Engineering a Scalable Placement Heuristic for DNA Probe Arrays", *Proc. 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, April (2003), pp. 148-156.
- [4] S. Rahmann. "Rapid large-scale oligonucleotide selection for microarrays", *Proc. IEEE Computer Society Bioinformatics Conference (CSB)*, 2002.
- [5] Sigma Genosys www.sigma-genosys.com
- [6] L. Kaderali and A. Schliep, "Selecting signature oligonucleotides to identify organisms using DNA arrays", *Bioninformatics 18*, 2002, pp: 1340-1349.
- [7] L Perelygina, L., Zhu, L., Zurkühlen H., Mills, R., Borodovsky, M., Hilliard, J.K., "Complete Sequence and Comparative Analysis of the Genome of Herpes B Virus (Cercopithecine herpes virus 1) from a Rhesus Monkey. J", *Virology*, 77(11), (2003), pp.6167-6177.
- [8] GenBank <http://www.ncbi.nlm.nih.gov>
- [9] S. Kasif, Z. Weng, A. Derti, R. Beigel, and C. DeLisi, "A computational framework for optimal masking in the synthesis of oligonucleotide microarrays", *Nucleic Acids Research vol. 30* (2002), e106.
- [10] A.C. Tolonen, D.F. Albeanu, J.F. Corbett, H. Handley, C. Henson, and P. Malik, "Optimized in situ construction of oligomers on an array surface", *Nucleic Acids Research vol. 30* (2002), e107.