Automated Construction of Diagnosis Rules from DNA Samples

Ha-Young Jang and Byoung-Tak Zhang

Biointelligence Laboratory, Seoul National University, Seoul 151-742, Korea
{hyjang, btzhang}@bi.snu.ac.kr
http://bi.snu.ac.kr/

Abstract. We propose a molecular computing algorithm for constructing diagnosis rules from blood sample automatically. Different to disease diagnosis based on microarray, proposed method can make a diagnosis without statistical analysis of sample. Every operator in the proposed method can be implemented with conventional wet-lab techniques such as Polymerase Chain Reaction (PCR), hybridization and affinity separation. Tested on a real disease data, simulation results show not only the feasibility of proposed method but also the possibility of biological information processing. The use of huge population in molecular evolutionary algorithm also can give various insights to evolutionary computation.

1 Introduction

The microarray technology has many applications such as gene discovery, disease diagnosis, drug discovery and toxicological research. For example, with the evolution of microarray technology, it will be possible for the researchers to further classify the types of cancer on the basis of the patterns of gene activity in the tumor cells. This will tremendously help the pharmaceutical community to develop more effective drugs as the treatment strategies will be targeted directly to the specific type of cancer [9]. Typical approach is the selection of discriminating genes in specific disease. But the selection of discriminating genes is not enough to make a diagnosis, because these genes are co-related with each other.

For this kind of microarray analysis, biological signal should be transformed into optical signal. Then optical signal should be transformed into numerical data. As a result of this procedure, the microarray data have noise or error inevitably. If we can process the biological signal directly without transformation, we can analysis more clean data without noise or error.

We propose novel method to construct diagnosis rules from patient's DNA Samples automatically and show that the diagnosis rules can be learned from training examples using DNA computing. The probabilistic nature of the computation performed by DNA computing makes it robust against uncertainty arising from both internal and external sources. The general setting for proposed method is similar to the genetic programming framework where the programs for digital computers are evolved using the principle of natural selection [5]. But our method

is much simpler than genetic programming, because we want to implement our method with real DNA molecules. Simulation result shows the feasibility of our method and the possibility of implementation with DNA molecules.

There are some similarities and differences between the standard genetic programming and the molecular programming. Molecular programming (MP) is similar to a standard GP in that its representation is of variable-length, which is a defining characteristic that distinguishes GP from other evolutionary computation methods. The use of decision lists as the representation of program structure is distinguished from other GP approaches, including the linear GP [8]. The use of DNA computing technology makes the design of the evolutionary operators very different from the conventional GP and other evolutionary computation methods. The possibility of synthetic DNA molecules and their manipulation by biochemical techniques in a test tube allows for the use of huge population size.

The paper is organized as follows. In section 2, representation of the individual is described. Learning algorithm of the proposed method is in section 3. In section 4 and 5, experimental design and simulation results are reported. Section 6 draws conclusions.

2 Decision List

Each individual represents not a tree but a decision list (rule). These decision lists constitute whole decision forest. In other words each decision list represents a certain path from root to leaf in the decision tree. And the decision is made by the decision forest which is composed of decision lists or decision trees. The aim is to find these decision lists and build a decision-making system f that makes a diagnosis. In the decision tree in Figure 1, possible decision lists are as follows: (36822_at = yes, 32815_at = yes, 1915_at =yes, class = yes), (36822_at = yes, 32815_at = no, 38982_at = no, class = no), (36822_at = no, 1915_s_at = yes, class = yes), (36822_at = no, 1915_s_at = no, 1894_f_at = yes, 38072_at = yes, 36950_at = yes, class = no), etc.

To be more specific, consider a DNA-based diagnosis problem. Given a training set D of K labeled DNA samples in the form

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K} \tag{1}$$

$$\mathbf{x}_i = (x_{i_1}, x_{i_2}, ..., x_{i_n}) \in \{0,1\}^n \tag{2}$$

$$y_i \in \{0,1\}. \tag{3}$$

Here $\mathbf{x}_i$ represents the DNA markers (subsequences of genes) in sample i and y_i is its associated diagnosis. For example, a training example $(10101, 1)$ means the sample is diagnosed positive ($y = 1$) if it contains the DNA markers numbered 1, 3, and 5 ($x_1 = 1, x_3 = 1, x_5 = 1$) and does not contain the rest ($x_2 = 0, x_4 = 0$).

To solve the diagnosis problem, a test tube of DNA molecules representing the genetic programs or diagnosis rules is maintained. Given is a set D of training data consisting of pairs of DNA-sample and its associated label. The goal is to find a population of decision lists (or a decision forest) that can predict the

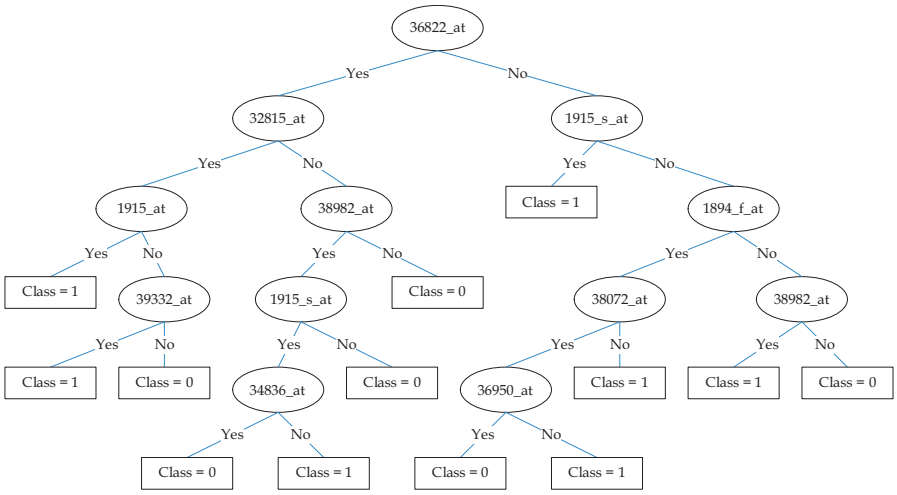

Fig. 1. Decision tree using ID3: each path from root to leaves represents the decision list.

correct diagnosis label for a future DNA-sample. The generic form of a decision list is $(x_1 = 1, x_3 = 1, x_5 = 1, y = 1)$, where the commas are interpreted as logical ANDs. Each attribute-value pair is encoded as a sequence of nucleotides (A, T, G, and C). The output label can also be encoded as a DNA sequence.

The whole population consists of multiple copies of the decision lists and the number of copies is proportional to the importance of the decision list. That is to say, the number of copies means the weight of each decision tree in decision forest. The goal of molecular genetic programming is to find the probabilistic distribution of the decision lists to solve the diagnosis problem. Since each decision list represents a conjunction, the population represents a disjunction of conjunctions where each conjunction may have multiple copies. This representation has some similarity with the decision tree [7] or decision forest [4] in that the whole population represents a disjunction of conjunctions of attribute-value pairs. However, our ensemble representation allows for probabilistic computation of decision labels rather than deterministic as in conventional decision tree methods.

3 Design for Wet-lab Experiment

Due to the limitation of handling DNA molecules, variation operators are not used. We only use selection and amplification to change the probability distribution of decision lists. Some recent genetic and evolutionary algorithms build explicit probabilistic models for the population [6]. These distribution-estimation algorithms (EDAs) generate offspring by sampling from the probabilistic model rather than using crossover and mutation. Like EDAs, our method generates

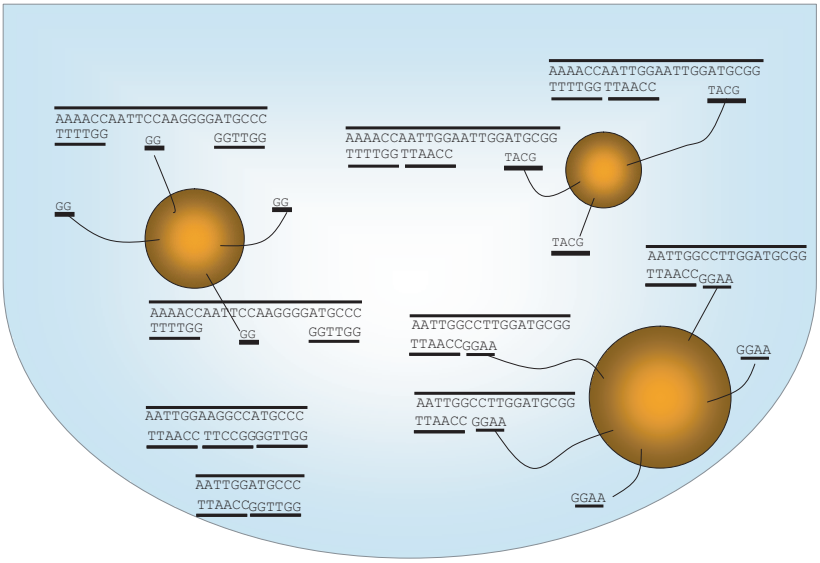

Fig. 2. Affinity separation by magnetic beads: Selection operator to the specific DNA sequence can be implemented with its complement sequence attached to magnetic bead. After the DNA sequence hybridize with its complement, we can separate it with magnetic.

the offspring by sampling from a probability distribution. Unlike in EDAs, no extra probabilistic model is built. The population of linear lists itself represents a probability distribution. The use of a huge number of molecules (10^{15} or more) enables the test tube to represent the empirical probability distribution.

Polymerase chain reaction (PCR) is a molecular biology technique, for enzymatically replicating DNA without using a living organism, such as E. coli or yeast. Like amplification using living organisms, the technique allows a small amount of the DNA molecule to be amplified exponentially. However, because it is an *in vitro* technique, it can be performed without restrictions on the form of DNA and it can be extensively modified to perform a wide array of genetic manipulations. PCR is commonly used in medical and biological research labs for a variety of tasks, such as the detection of hereditary diseases, the identification of genetic fingerprints, the diagnosis of infectious diseases, the cloning of genes, paternity testing, and DNA computing.

Hybridization is the process of combining complementary, single-stranded nucleic acids into a single molecule. Nucleotides will bind to their complement under normal conditions, so two perfectly complementary strands will bind to each other readily. Conversely, due to the different geometries of the nucleotides, a single inconsistency between the two strands will prevent them from binding. The process can be reversed by heating the molecule.

Figure 2 shows implementation of selection operator using complement DNA attached to magnetic beads.

4 Evolving the Decision List

In this section we describe the molecular algorithm for learning the decision list. The goal is to learn a decision list that best fits to a data set. To learn the formula we initialize a library of DNA molecules representing random combinatorial decision list. Let this library at the step t be L_t. Given a query pattern $\mathbf{x}_q$ we extract from the library all the molecules (terms) that match the query. The extraction can be implemented using hybridization reaction in the same way to check which markers exist. The idea is to chop the query sequence into subsequences for individual variables. These chopped query sequences hybridize with the decision list in the population. Only the fully double-stranded sequences are then separated.

These molecules will have class labels from which we decide the majority label as the class of the query pattern. To perform the matching between $\mathbf{x}_i$ and $\mathbf{x}_q$ for $i = 1, ..., N$ in parallel, we present multiple copies (up to the number of the library size) of it. That is, we generate a collection

$$Q = \{\Delta c(x_1), \Delta c(x_2), ..., \Delta c(x_n), \Delta c(y)\}, \tag{4}$$

where $\Delta c(\cdot)$ denotes copies made by PCR (learning rate). The class decision is made by comparing the number of elements in class 1, N_1, with that in class 0, N_0: where y takes 0 or 1.

For learning, we prepare two collections, M_+ and M_-, consisting of library elements that correctly (or incorrectly) classifies the query sample as follows:

- $M_+ = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of x_i for $i = 1, ..., n$ and $v_i = y\}$
- $M_- = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of x_i for $i = 1, ..., n$ and $v_i \neq y\}$.

Now, we describe how the library is revised to learn from newly observed data. As a new training example $(\mathbf{x}, y)$ is given, we extract from the library the terms whose x-part matching with $\mathbf{x}$. The class y^* of $\mathbf{x}$ is determined by the classification procedure described above. Then, the matching terms (library patterns) are modified in their frequency depending on their contribution to the correct or incorrect classification of $\mathbf{x}$. If the label v of the library pattern $(\mathbf{u}, v)$ matching $\mathbf{x}$ is correct, i.e. $v = y$, it is reproduced:

$$L_t \leftarrow L_t + \Delta c(M_+). \tag{5}$$

If the label v is incorrect, i.e. $v \neq y$, the matching library pattern is removed from the library:

$$L_t \leftarrow L_t - \Delta c(M_-). \tag{6}$$

The update of the library in this way is more or less like evolutionary computation with the additional feature that the presentation of a training example

- 1. Let the library $L_0 = \{(\mathbf{u}_i, v_i)\}$ contain the initial decision lists. Let $t = 0$.
- 2. Let $t \leftarrow t + 1$.
- 3. Get a training example $(\mathbf{x}, y) = (x_1, x_2, ..., x_n, y)$.
- 4. Let $Q = \{\Delta c(x_1), \Delta c(x_2), ..., \Delta c(x_n), \Delta c(y)\}$.
- 5. Classify $\mathbf{x}$ using L_t as described in the text and construct the following:
  - $M_+ = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of x_i for $i = 1, ..., n$ and $v_i = y\}$.
  - $M_- = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of x_i for $i = 1, ..., n$ and $v_i \neq y\}$.
- 6. Update the library L as follows:
  - $L_t \leftarrow L_{t-1} + Q$.
  - $L_t \leftarrow L_t + \Delta c(M_+)$.
    Optionally, $L_t \leftarrow L_t - \Delta c(M_-)$.
- 7. Go to Step 2 if not terminated.

Fig. 3. Learning procedure of decision lists from training examples [12].

proceeds one generation of the library (as a population). This is also a learning procedure since the library improves its classification performance as new examples are presented. The molecular algorithm for the whole evolutionary learning procedure is summarized in Figure 3 and the cycle of learning procedure is shown in Figure 4. Addition operation can be implemented by PCR and removal can be done by extraction of the corresponding molecules. The update process relies upon the reliability of DNA extraction technology.

Note also that the learning rule has a parameter Δc that reflects the strength of learning for each training example. This is also related to the reproduction rate. Big Δc imposes high reproduction rate while small Δc forces a low reproduction rate. How to set this parameter is an important issue for the stability and the adaptability of the algorithm. δ is expressed as the amplification ratio of the number $\Delta c(\mathbf{x}, y)$ of additional copies of molecules to the number of current copies $c_{n-1}(\mathbf{x}, y)$, i.e.

$$\delta = \frac{\Delta c(\mathbf{x}, y)}{c_{n-1}(\mathbf{x}, y)}. \tag{7}$$

Since $\Delta c(\mathbf{x}, y)$ is determined by the number of PCR cycles for signal amplification, the reproduction rate δ can be set indirectly by controlling the number of PCR cycles or its fraction.

5 Experimental Results

5.1 Data

Microarray gene expression data are used to evolve molecular genetic programs for making diagnosis based on DNA. Since molecular programming is based on DNA molecules, it is very natural to solve the problem based on DNA. Microarrays or DNA chips are a new technology for measuring gene expression intensities at the cDNA (i.e. the DNA sequence complementary to the mRNA

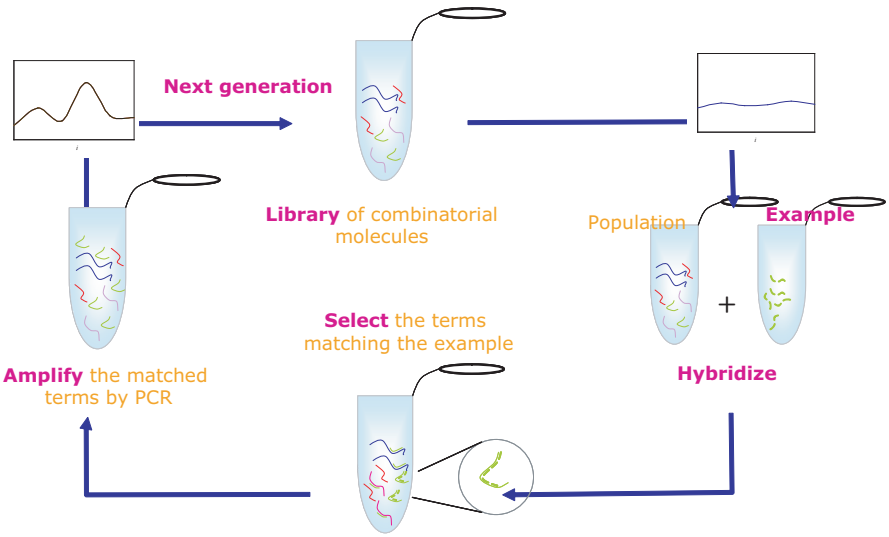

Fig. 4. Schematic diagram of learning procedure

sequence) level. Gene expression data are collected from microarray experiments for ALL/AML leukemia [3]. It should be noted that the microarray gene expression data contain much noise and this application can be a good test bed problem for the molecular programming approach against uncertainty.

Because ID3 prunes nodes (attribute), the microarray data are preprocessed and 10 genes were selected out of 12600 genes for the purpose of comparing simulation results with the results of ID3. The genes are chosen according to the information gain measure for extracting features [7]. The training set consists of 120 examples each composed of 10 genes plus the associated leukemia class which is AML or ALL. A 6-fold cross-validation is used for testing the performance. That is, the whole data set of 120 examples is partitioned into 6 subsets and a total of six sessions were run, where each run used a subset of 20 examples for test and the remaining 100 examples (5 subsets) for training.

5.2 Simulation Results

For the simulation of in vitro evolution of the molecular genetic programs, the population size of $118,096 \times 10^6 \approx 1.2 \times 10^{11}$ was used, where 118,096 is the number of different library elements and 10^6 is the number of their copies. The library was initialized to contain each and every conjunction of order 1 through 10. These include $(x_1 = 0, y = 0), (x_1 = 0, y = 1), (x_1 = 1, y = 0), (x_1 = 1, y = 1), (x_1 = 0, x_2 = 0, y = 0), (x_1 = 0, x_2 = 0, y = 1), (x_1 = 1, x_2 = 0, y = 0), ...$ Thus, the total number of the different library elements is $N = \sum_{k=1}^{10} {}_{10}\mathrm{C}_k \cdot 2^k \cdot 2 = 118,096$, where ${}_{10}\mathrm{C}_k$ denotes the number of cases choosing k variables out of 10. For the simulation of in vitro computation, we used the library size

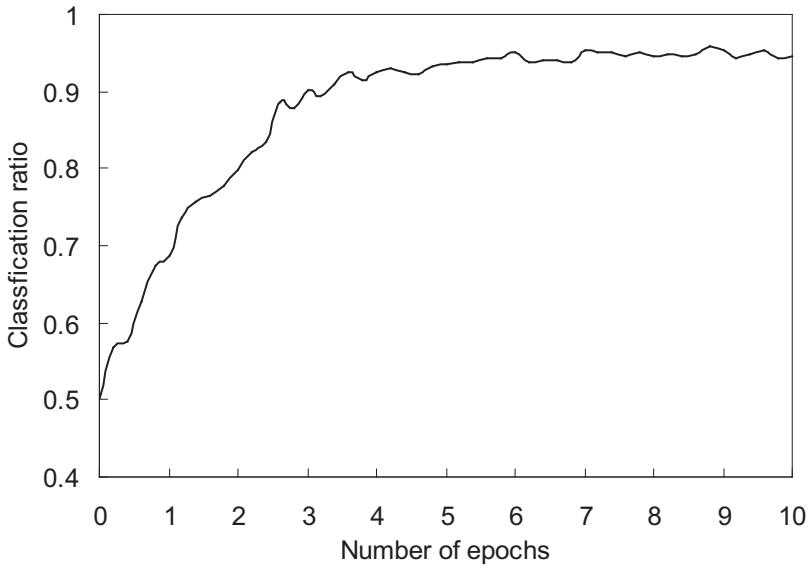

Fig. 5. Fitness evolution. Shown are the average classification rates over the 6-fold cross-validation runs. Though there are fluctuations the fitness values tend to converge 96 % accuracy. The reproduction rate was 0.01.

of $118,096 \times 10^6$, i.e. the initial library was generated by copying each element 10^6 times. Thus, the library consists of multiple copies of the same terms and we evolved the distributions of the terms through the molecular programming procedure.

After the learning procedure, only the rules whose concentration (weight) shows a rise of 10 times over initial concentration are selected. In order to compare with ID3, those whose length is shorter than maximum depth of decision tress are selected once more. 426 rules are selected finally. The rules agree exactly with the rules found by ID3 are as below:

- (36822_at = no, 1915_s_at = yes, class = yes)
- (36822_at = yes, 32815_at = no, 38982_at = no, class = no)
- (36822_at = yes, 32815_at = no, 38982_at = yes, 1915_s_at = no, class = no)
- (36822_at = no, 1915_s_at = no, 1894_f_at = no, 38982_at = yes, class = yes)

It is not so surprise that only 4 rules agree with those of ID3. ID3 executes greedy local search. On the other hand, our method executes global search. It does not mean low performance that only 4 rules agree with ID3's. As shown in Figure 5, weighted sum of rules shows high classification accuracy. Because our method maintains huge population, the efficiency of learning process is much worse than ID3. But the objective of our algorithm is automatic construction

of rules not *in silico* but *in vitro*. Its simplicity in the learning process makes it possible to implement molecular algorithm with DNA test tube.

Figure 5 shows the fitness evolution of population. The accuracy converges about 96%. This result is competitive to the result of ID3 (96.7%). The fitness is evaluated by the classification performance to the training set. For decision making, we used a sigmoid squashing function:

$$f(x) = \frac{1}{1 + \exp(-\beta x)} \tag{8}$$

where β is a constant which reflects the level of noise and sets the decision boundary. We count the number of each term which answers positive or negative. Then, the proportion of the positives and the negatives is calculated. This result is the input to the sigmoid function. We make a decision probabilistically based on the output of the sigmoid function. In the test tube, we cannot count the exact number of DNA molecules. We can only know the rough estimation of them. In order to simulate it, we use sigmoid function.

6 Conclusion

We presented an evolutionary method to construct diagnosis rules from DNA samples. Our simulation results on the leukemia problem shows the feasibility of automatic construction of diagnosis rules based on DNA computing. Recent research of Benenson et al. [1] also demonstrates the possibility of diagnosis based on DNA computing. Our results on DNA-based diagnosis also suggest the possibility of biological information processing.

The simplicity of our algorithm shows its potential molecular implementation. It needs no computation and all of the operators can be implemented with bio-lab techniques easily. We are preparing real DNA experiment based on proposed algorithm. It is also an interesting future work to study the connection between the huge size of the population and the complexity of the learning algorithm.

7 Acknowledgements

This research was supported by the Molecular Evolutionary Computing (MEC) Project of MICE and by Seoul Research and Business Development Program 0534-200.

References

1. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E. "An autonomous molecular computer for logical control of gene expression," *Nature*, 429, 423-429, 2004.
2. L. Breiman, Bagging predictors, *Machine Learning*, 24:123-140, 1996.

3. M.ᵣ Cheok et al., Treatment-specific changes in gene expression discriminate in vivo drug response in human leukemia cells,*Nature Genetics*, 34:85-90, 2003.
4. T. K. Ho, The Random Subspace Method for Constructing Decision Forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832-844, 1998.
5. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992.
6. P. Larranaga, : A Review on Estimation of Distribution Algorithms. In: Larranaga, P., Lozano, J.A. (eds.), *Estimation of Distribution Algorithms: A New Tools for Evolutionary Computation.* Kluwer Academic Publishers (2001) 57-100
7. T. M. Mitchell, *Machine Learning*, The McGrow-Hill Companies, Inc., 1997.
8. P. Nordin, W. Banzhaf, and F. Francone, Efficient evolution of machine code for CISC architectures using blocks and homologous crossover, *Proc. Third Annual Genetic Programming Conference (GP-99)*, L. Spector, W. Langdon, U.-M. O'Reilly and P. Angeline (eds.), 275-299, Morgan Kaufmann, 1999.
9. J. Petrik, Diagnostic applications of microarrays, *Transfusion Medicine*, 16(4):233-247, 2006.
10. B.-T. Zhang, A unified Bayesian framework for evolutionary learning and optimization, *Advances in Evolutionary Computation*, Chapter 15, pages 393-412, Springer-Verlag, 2003.
11. B.-T. Zhang and H.-Y. Jang, A Bayesian algorithm for in vitro molecular evolution of pattern classifiers, *Lecture Notes in Computer Science*, 3384:458-467, 2005.
12. B.-T. Zhang and H.-Y. Jang, Molecular learning of wDNF formulae, *Lecture Notes in Computer Science*, DNA 11, 3892:427-437, 2006.