# Gene Specific Co-regulation Discovery: An Improved Approach *

Ji Zhang, Qing Liu and Kai Xu

CSIRO Tasmanian ICT Centre
Hobart, TAS, Australia 7001
Email: {*Ji.Zhang, Q. Liu, Kai.Xu*}*@csiro.au*

**Abstract.** Discovering gene co-regulatory relationships is a new but important research problem in DNA microarray data analysis. The problem of gene specific co-regulation discovery is to, for a particular gene of interest, called the *target gene*, identify its strongly co-regulated genes and the condition subsets where such strong gene co-regulations are observed. The study on this problem can contribute to a better understanding and characterization of the target gene. The existing method, using the genetic algorithm (GA), is slow due to its expensive fitness evaluation and long individual representation. In this paper, we propose an improved method for finding gene specific co-regulations. Compared with the current method, our method features a notably improved efficiency. We employ $k$NN Search Table to substantially speed up fitness evaluation in the GA. We also propose a more compact representation scheme for encoding individuals in the GA, which contributes to faster crossover and mutation operations. Experimental results with a real-life gene microarray data set demonstrate the improved efficiency of our technique compared with the current method.

## 1 Introduction

DNA microarray is an enabling technology to provide a global view of the expression of a large number of genes. A gene microarray data set is typically presented as matrix where each row represents a gene and each column is the experimental condition (such as time point) when the gene expression is extracted. Finding gene co-regulatory relationships is an important research focus in microarray data analysis. One interesting research problem, called *Single Gene Approach* for gene microarray analysis [14], was recently studied by [17]. This problem can be formulated as follows: *for a particular gene of interest, called target gene, identify its strongly co-regulated genes and the condition subsets where such strong gene co-regulations are observed*. The discovered co-regulated genes and the associated condition subsets are specific to the target gene, which can help biologists to better understand and characterize it. This is useful in many

* This research is supported by CSIRO Preventative-Health Flagship program and Tasmanian ICT Centre (TasICTC). TasICTC is jointly funded by the Australian Government through the Intelligent Island Program (administered by the Tasmanian Department of Economic Development and Tourism) and CSIRO.

applications such as the investigation of the most differentially expressed genes in a disease study or the function prediction of unknown genes.

This paper proposes two boosting techniques aiming to achieve a noticeable efficiency improvement for the method proposed in [17]. The major technical contributions of this paper are summarized as follows:

1. First, we employ $k$NN Search Table [16] to find the lower and upper bounds of the distance between the target gene and its $k^{th}$ most co-regulated genes in the data set. These bounds can be utilized to substantially speed up fitness evaluation in the GA and facilitate the procurement of the top condition subsets where the target gene is most co-regulated with other genes in the microarray data set. We have also devised a better way to specify the parameter values that are used in the approximation scheme;
2. We also propose a more compact representation scheme for encoding condition subsets in the GA, which contributes faster crossover and mutation operations.
3. Experimental results with a real-life gene microarray data demonstrate the better efficiency of our technique than that of the existing method in discovering gene specific co-regulations.

The remainder of this paper is organized as follows. Section 2 presents the approach to efficiently find the top co-regulated condition subsets for the target gene using $k$NN Search Table. A more compact individual representation scheme is proposed in Section 3 to speedup the GA by reducing the overhead of crossover and mutation operations. We report the experimental results in Section 4 and the last section concludes this paper.

## 2 Related Work

Clustering analysis is probably the most commonly used technique for studying gene co-regulations by grouping closely co-regulated genes together. The major clustering algorithms in discovering gene co-regulations include hierarchical clustering method [11], $k$-means algorithm [11], Self-Organization Maps (SOMs) [12] and SVD-based clustering algorithm [8]. Because they perform clustering based on the entire set of conditions (*i.e.*, full dimensionality), thus they miss out those interesting co-regulations embedded in the lower dimensional condition subsets. To find the gene co-regulations in some subsets of conditions, a few subspace clustering methods for gene expression data, such as Coupled Two-Way Clustering [7], bi-cluster [3] and $\delta$-cluster [15], are also proposed. However, a common key drawback for clustering methods, no matter whether they find co-regulations on the full or partial dimensionality, is that there is no guarantee that the target gene's most-co-regulated genes are in the same cluster where the target gene is located. Quite likely, they are located in a few different clusters because the results of clustering are quite sensitive to the parameters such as the number of clusters to be obtained or the value of the inter-cluster dis-similarity metric that users choose. In addition, the clustering analysis needs to be performed in a large number of condition subsets, which is rather expensive. As such, clustering is not a direct nor efficient way for discovering gene-specific co-regulations.

There are also some work on gene co-regulation discovery from time series perspective by considering the explicit temporal nature of the features in the gene microarray data [1][2][4][5][6]. However, there are cases that the features do not have explicit temporal meaning and gene co-regulations may occur in the subsets of features that are far apart from each other in the gene microarray data set.

In [17], the authors proposed an approach for mining local gene-specific co-regulation using genetic algorithm (GA) [9]. The basic idea of this approach is to first find the condition subsets in which the target gene $g$ is most significantly co-regulated with others and the co-regulated genes of $g$ are then selected from its nearest neighbors in these condition subsets. A sliding window is used to scan all the conditions sequentially and the search of condition subsets is performed within each window position. This method is able to find the closely co-regulated genes for the target gene and the associated condition subsets where such co-regulation occur. However, this method is slow. The major speed bottleneck of this approach is the fitness computation in the GA, which involves a $k$NN search for the target gene in each condition subset. The typically large number of genes in the microarray data and the number of condition subsets evaluated in the GA lead to a slow $k$NN search.

## 3 Searching Co-regulated Condition Subsets Using *k*NN Search Table

In [17], the fitness function of the target gene in each condition subset is defined as the distance between it and its $k^{th}$ nearest gene in the microarray data set. Such $k$NN search has to be performed in all the subspaces that are explored by the GA, leading to a slow speed for the whole method. Using indexing methods to speed up $k^{th}$NN search is not efficient due to two major reasons. First, since a large number of condition subsets may be evaluated in the GA, it will be expensive to index the genes in each possible condition subset. Second, for gene-specific co-regulation discovery problem, we may be only interested in a small number of target genes, thus the high cost associated in building indexing cannot be amortized by the one-time performance gain by using the indexing.To solve this problem, we draw on the *kNN Search Table* proposed in [16] to speed up computation of fitness function. A $k$NN Search Table for a target gene $g$, denoted as $\mathcal{T}^g$, is a $M \times k$ table containing its $k$ nearest neighbors in *each* single dimension of full data space with $M$ dimensions. The entry $x_{ij}$ of the table represents the $j^{th}$ nearest neighbor of $g$ in the $i^{th}$ dimension, where $1 \leq i \leq M$ and $1 \leq j \leq k$. Using $k$NN Search Table, we can compute very efficiently the lower and upper bounds of the distance between the target gene and its $k^{th}$ nearest neighbor in any condition subset $s$, denoted by $f^k_{min}(g, s)$ and $f^k_{max}(g, s)$, respectively. Interested readers can refer to [16] for the computation of $f^k_{min}(g, s)$ and $f^k_{max}(g, s)$ using $k$NN Search Table and the proof of their correctness.

The idea of using $k$NN Search Table for speeding up the fitness computation of condition subset $s$ with respect to the target gene $g$ is to approximate the

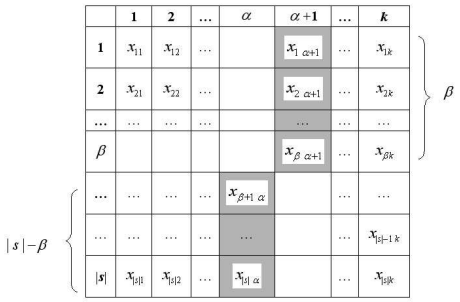


**Fig. 1.** $k$NN Search Table

fitness by using the linear combination of $f^k_{min}(g,s)$ and $f^k_{max}(g,s)$ as follows:

$$fitness_{app}(g,s) = \alpha f^k_{min}(g,s) + \beta f^k_{max}(g,s)$$

where $\alpha + \beta = 1$. Unlike setting $\alpha = \beta = \frac{1}{2}$ as in [16], we calculate the accurate fitness, $f^k_{min}$ and $f^k_{max}$ for a specified number of condition subsets in the GA to set the values for $\alpha$ and $\beta$ for each condition subset $s$ such that the following equation is satisfied:

$$fitness_{accurate}(g,s) = \alpha f^k_{min}(g,s) + \beta f^k_{max}(g,s)$$

subject to $\alpha + \beta = 1$. The average of $\alpha$ and $\beta$ of the test condition subsets are computed and used for the fast fitness approximation of all the other condition subsets in the GA.

$k$NN Search Table is advantageous in the following two aspects: 1) The construction of $k$NN Search Table w.r.t. the target gene only involves finding its $k$NNs in each one-dimensional condition subset, therefore its construction cost is only $\mathcal{O}(NM)$, which is linear with respect to both number and dimensionality of genes in the gene microarray data set. For the whole gene microarray data, only one $k$NN Search Table needs to be pre-computed for a target gene; 2) The total complexity for computing $f^k_{min}(g,s)$ and $f^k_{max}(g,s)$ is $O(k^2|s|^2)$, which becomes independent of $N$ and $M$.

Since $fitness(g,s)$ is approximated in the GA, the accuracy of computation is thus somehow affected. To address this problem, we can perform a *refinement step* on the top condition subsets we obtain in the GA that are stored in the so-called *CandidateSet*. Instead of using $f^k_{min}$ and $f^k_{max}$ for a fast fitness approximation, the refinement step computes the accurate fitness for top candidate condition subsets and the top-$k$ condition subsets among them will be returned. Admittedly, evaluating each condition subset in this refinement process is more expensive than the approximation as it involves more accurate computations. However, the number of candidate condition subsets is typically much smaller than the total number of condition subsets approximated in the GA. In addition, a pruning optimization strategy can be further devised to speed up the computation, which operates as follows. After the fitness of $n$ candidate condition subsets have been accurately evaluated, we start to maintain the minimum $fitness(g,s)$

for the top-$k$ condition subsets we have found thus far, denoted as $MinFit$. Those unevaluated condition subsets that satisfies $f^k_{max}(g, s) < MinFit$ cannot become the top-$k$ condition subsets and can therefore be safely pruned. This is because that $MinFit$ is monotonically increasing as we examine more condition subsets in the refinement step.

Once the top-ranked condition subsets have been found by the GA, the next step will be finding the co-regulated condition subsets from these top-ranked condition subsets. This step will be much trivial than the step of finding the top co-regulated condition subsets. It only involves finding the $k$ most co-regulated genes of the target gene in each of those top-ranked condition subsets.

## 4 Shorter Individual Representation

A straightforward representation scheme for condition subsets in the GA is the standard binary encoding; all individuals are represented by strings with fixed and equal length $M$, where $M$ is the number of dimensions of the Microarry data set. Using binary alphabet $\Sigma = \{0, 1\}$ for gene alleles, each bit in the individual will take on the value of "0" and "1", respectively, indicating whether or not its corresponding attribute is selected

However, for a high-dimensional microarray data set with $M$ dimensions, we will end up with a long binary string with a length of $M$ for representing each condition subset. A high computational overhead is involved in the frequently preformed crossover and mutation of long binary strings. Therefore, a short representation of individuals is desirable. To this end, we employ integer string, instead of binary string, to represent each condition subset in the GA that features a much shorter length. Each integer can represent a few binary bits, which contributes a remarkable reduction of the length of individual representation. If we assume that the number of binary bits needed for representing each integer is $L$, $L \leq M$ (integers are in the range of $[0, 2^L - 1]$), then length of an integer string for representing a condition subset will be only approximately $\frac{M}{L}$ of the length of the binary string used to represent the same condition subset.

Even though it features a more compact representation and enables faster crossover and mutation operations, the integer string representation changes the behavior of mutation and tends to result in abrupt mutations more frequently, which may lead to loss of useful segments of bits that might just turn out to be part of potentially good solutions.

To solve these problems while, at the same time, preserve the desired advantages of integer representation, we propose efficient methods to seamlessly simulate the genetic operations of binary string using integer string representation. We derive ways for quickly obtaining crossover and mutation results between any pair of integers that are consistent with those of binary strings.

For crossover operations, we propose *Crossover Lookup Table (CLT)*. It is a $2^L \times 2^L$ table with each entry being a pair of integers corresponding to the crossover result of a given pair of integers. The crossover locus $l_c$ is generated randomly in the range of $[1, M-1]$. Notice that, from an integer string's perspective, the crossover locus will only be located on the boundary of two adjacent integers or within a single integer. In the former case, two integer strings can

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | (0,0) | (1,0) | (2,0) | (1,2) |
| **1** | (0,1) | (1,1) | (0,3) | (1,3) |
| **2** | (2,0) | (3,0) | (2,2) | (3,2) |
| **3** | (2,1) | (3,1) | (2,3) | (3,3) |

**Fig. 2.** An example of Crossover Lookup Table ($identifier = 1$, $L = 2$)

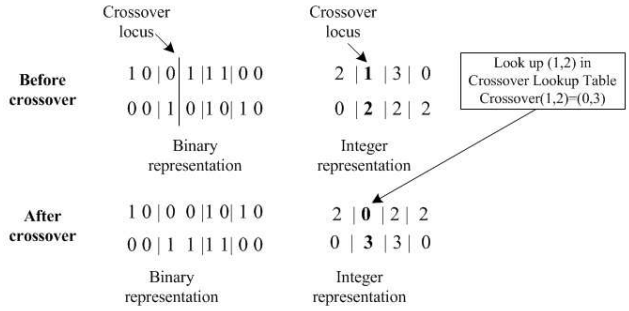


**Fig. 3.** A crossover example of two integer strings ($M = 8$, $L = 2$, $l_c = 3$)

be directly crossovered in the same way as the binary strings. In the later case, all the integers after the one that the crossover locus is located can also be crossovered in the same way as the binary strings. The crossover result of the pair of integers where crossover locus is located can be obtained by looking up the *appropriate* Crossover lookup Table based on the value of ($l_c$ *mod* $L$). As there are $L - 1$ different crossover locus inside an integer, thus we need to pre-computed $L - 1$ different CLTs. Each table is uniquely identified by an integer $i \in [1, L - 1]$. Figure 2 gives an example CLT with identifier=1 when $L = 2$. Figure 3 is a crossover example of two integers by means of the CLT given in Figure 2.

For mutation operations, we quantify the *Mutation Transition Probability Table (MTPT)*. It is also a $2^L \times 2^L$ table with each entry representing the mutation transition probability from one integer to another. An integer is mutated to another based upon their transition probability initiated from this integer. Let us suppose that two integers $a$ and $b$ differ in $l$ bites in their binary representations ($0 \leq l \leq L \leq M$), the mutation transition probability from $a$ to $b$, denoted as $Pr(a, b)$, is computed as $Pr(a, b) = p_m^l(1 - p_m)^{(L-l)}$. Unlike CLT, there is only one MTPT that needs to be pre-computed for a given gene microarray data set. Suppose $L = 2$ and $p_m = 0.1$, we need to compute the mutation transition probability from integer 2 to 3. As they differ in only 1 bit in their binary representations, thus $Pr(2, 3)$ can be computed as $Pr(2, 3) = 0.1^1 \cdot (1-0.1)^{2-1} = 0.09$.

When using a shorter integer representation for condition subsets in the GA, we can achieve an approximate $\frac{M}{L}$ times performance boost in crossover and mutation operations for $M$-dimensional gene microarray data if each integer in

the string is represented by $L$ binary bits. The pre-computed CLT and MTPT contribute to a remarkable performance gain of crossover and mutation in the GA by transforming them to simple lookup operations from lookup tables, without the frequent on-the-fly conversion between integer and binary strings.

## 5 Experimental Results

The Spellman's data set is used in our experiments that can be downloaded from http://genome-www.stanford.edu/cellcycle/data/rawdata. This data set contains all the data for the alpha factor, cdc15, and elutriation time courses, and includes the data for the Clb2 and Cln3 induction experiments. We used only the alpha-factor and CDC28 data set for our experiments. This data set contains 6178 genes under 35 experimental conditions (time points).

The main focus of our experimental evaluation is to investigate the efficiency boost of the existing method when the two strategies we propose (*i.e.*, fitness evaluation using $k$NN Search Table and shorter individual representation scheme using integer strings) are employed. Efficiency comparison is conducted in the experiment between the baseline method (*i.e.*, no boosting strategies are used) and the method with only one boosting strategy is used. In this way, we can obtain a clear idea as to the magnitude of improvement that can be achieved using each strategy. For all the experiments, the number of generations and population size are fixed as 200 and 50 respectively in the GA. The efficiency are investigated under varying number of genes and conditions. The original gene microarray data is properly sampled to obtain new data sets with desired number of genes and conditions for experimental purposes. At the end of this section, we will also investigate the effectiveness of our proposed improved method.

### 5.1 Efficiency Improvement Using $k$NN Search Table

We first study the efficiency boost of the method using $k$NN Search Table. Figure 4 and 5 show the execution time of the method using $k$NN Search Table, compared with the baseline method. By using $k$NN Search Table, the method can achieve a remarkable improvement of efficiency, especially when the number of genes and conditions of the microarray data set are high. The magnitude of speed improvement could be over 70% for the microarray data we use in the experiment. As we have analyzed earlier, the complexity of using $k$NN Search Table to approximate the fitness of condition subsets in the GA is independent of the number of genes and the number of condition of the microarray data set, thus approximately horizontal curves for the boosted method are observed in both figures. Note that the execution time of the improved method increases only slightly when dealing with data sets with higher number of genes or conditions. Such increase comes from the refinement step where the top co-regulated condition subsets for the target gene are obtained based on the top-ranked condition subsets returned by the GA. Insteading of using $k$NN Search Table, this step involves evaluating all the genes in the data set whose complexity is depended on the number of genes and conditions.

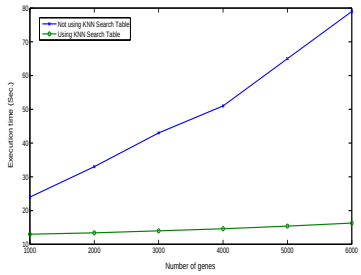


**Fig. 4.** Efficiency improvement using $k$NN Search Table (under varying number of genes)

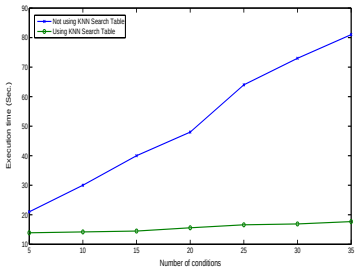


**Fig. 5.** Efficiency improvement using $k$NN Search Table (under varying number of conditions)

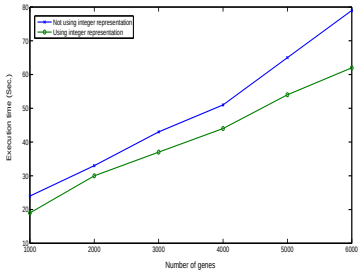


**Fig. 6.** Efficiency improvement using integer representation (under varying number of genes)

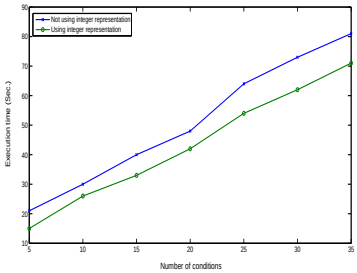


**Fig. 7.** Efficiency improvement using integer representation (under varying number of conditions)

### 5.2 Efficiency Improvement Using Shorter Representation

The second experiment investigates the efficiency improvement of the method by using integer strings for representing condition subsets in the GA. The experimental results are presented in Figure 6 and 7, respectively. As these results show, employing integer string representation is able to contribute to another 10-15% speed improvement. Also, as the number of genes or conditions increase, the magnitude of speed inprovement using integer representation is somewhat decrease. This is because that the cost of fitness function dominates the whole algorithm more severely when the number of genes or conditions increase, making the efficiency contirubtion of integer representation relatively less remarkable.

### 5.3 Effectiveness Study

Besides efficiency improvement, we are also interested in studying the effectiveness of the method in gene specific co-regulations discovery when the boosting strategies are used. A general rationale is that the effectiveness of the method

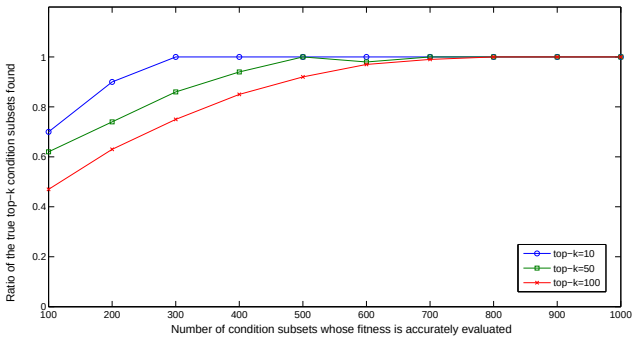


**Fig. 8.** Effectiveness of the improved method

should not be severely compromised when its speed can be improved noticeably. In this experiment, we study the final co-regulated condition subsets of the target gene obtained using the improved method, compared with the baseline method. As we know, as long as the condition subsets for gene co-regulation of the target gene can be correctly identified, it will then become trivial to find its co-regulated genes in these condition subsets. Therefore, we only need to evaluate how good are the final co-regulated condition subsets obtained by the GA. The ground-truth result, *i.e.*, the true top-$k$ condition subsets where the strongest co-regulation for the target gene, needs to be found first using the brute-force search method. Based on the ground-truth result, the result of the improved method can be evaluated.

In this experiment, we vary the number of top condition subsets to be returned by the GA, denoted by $N_s$, and study the percentage of the true top-$k$ that are found using the boosting strategies under three top-$k$ values, 10, 50 and 100. Intuitively, the larger $N_s$ is, the higher chance will be for the final results of the GA to include the true top-$k$ condition subsets. As approximation is used in the GA, $N_s$ should be set relatively large (yet still much smaller than the total number of condition subsets that are evaluated in the GA) in order to provide a sufficiently large room to include the true top-$k$ condition subsets. We evaluate the different values for $N_s$, ranging from 100 to 1000, in this experiment. The results, as presented in Figure 8, show that we will be able to obtain all the true top-$k$ condition subsets as long as the $N_s$ is reasonably large (the required value for $N_s$ is increased when top-$k$ goes up). In other words, we only need to evaluate the fitness of a small number of condition subsets accurately to find the true top-$k$ condition subsets, and the fitness of all the other condition subsets can be quickly approximated using $k$NN Search Table.

## 6 Conclusions

In this paper, we propose an improved method for discovering gene specific co-regulations based on genetic algorithm (GA). We employ $k$NN Search Table to substantially speed up fitness evaluation in the GA. We also propose a more

compact representation scheme for encoding condition subsets in the GA, which contributes faster crossover and mutation operations. Our method features a better efficiency than the current method (up to a 70%-80% speed boost). The experimental results demonstrate the good efficiency and effectiveness of our technique in discovering gene specific co-regulations.

## References


1. R. Amato, A. Ciaramella, N. Deniskina, C. Del Mondo, D. di Bernardo, C. Donalek, G.Longo, G. Mangano, G. Miele, G. Raiconi, A. Staiano and R. Tagliaferri. 2006. A Multi-step Approach to Time Series Analysis and Gene Expression clustering. *Bioinformatics*, 22(5): 589-596.
2. Z. Bar-Joseph. Analyzing time series gene expression data. 2004. *Bioinformatics*, 20(16):2493-2503.
3. Y. Cheng and G.M. Church, Biclustering of Expression Data. 2000. In *Proc. International Conference on Intelligent Systems for Molecular Biology (ISMB)*, vol. 8, pp. 93-103.
4. S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W. Ray. 2004. A time series analysis of microarray data. In *4th IEEE International Symposium on Bioinformatics and Bioengineering*.
5. J. Feng, P. E. Barbano, and B. Mishra. 2004. Time-frequency feature detection for timecourse microarray data. In *2004 ACM Symposium on Applied Computing (SAC'04)*.
6. V. Filkov, S. Skiena, and J. Zhi. 2001. Analysis techniques for microarray time-series data. In *5th Annual International Conference on Computational Biology*.
7. G. Getz, E. Levine, and E. Domany. 2000. Coupled Two-Way Clustering Analysis of Gene Microarray Data, in *Proc. Natioal Academy of Science*, vol. 97, no. 22, pp. 12079-12084.
8. G. H Golub and C. F. Van Loan. 1983. *Matrix Computations*, Johns Hopkins University Press.
9. J. Holland. 1992. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge.
10. L. Ji and K. L. Tan. 2005. Identifying Time-Lagged Gene Clusters on Gene Expression Data. *Bioinformatics*, Vol. 21, No. 4, pp. 509-516.
11. R. A. Johnson and D. W. Wichern. 1998. *Applied Multivariate Statistical Analysis*, Prentice Hall International, USA.
12. T. Kohonen. 1995. *Self-Organization Maps*. Springer-Verlag, Berlin Heidelberg.
13. M. F. Ramoni, P. Sebastiani, and I. S. Kohane.2002. Cluster analysis of gene expression dynamics. Proceedings of *the National Academy of Sciences*, USA, 99(14):9121-9126.
14. T. Speed, J. Fridlyand, Y. H. Yang and S. Dudoit. 2001. Discrimination and clustering with microarray gene expression data. *2001 Spring Meeting of International Biometric Society Eastern North American Region (ENAR'01)*.
15. J. Yang, W. Wang, H. Wang, and P.S. Yu. 2002. $\delta$-Cluster: Capturing Subspace Correlation in a Large Data Set. In *Proc. 18th International Conference on Data Engineering (ICDE'02)*, pp 517-528.
16. J. Zhang, Q. Gao, and H. Wang. 2006. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *ICDM'06*, pp 731-740.
17. J. Zhang, Q. Gao and H. Wang. 2006. Discover Gene Specific Local Co-regulations Using Progressive Genetic Algorithm. *ICTAI'06*, pp783-790.