Discover Gene Specific Local Co-regulations Using Progressive Genetic Algorithm

Ji Zhang, Qigang Gao
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
{jiz, qggao}@cs.dal.ca

Hai Wang
Sobey School of Business
Saint Mary's University
Halifax, Nova Scotia, Canada
hwang@smu.ca

Abstract

The problem of gene specific co-regulation discovery is that, for a particular gene of interest, identify its closely co-regulated genes and the associated subsets of experimental conditions in which such co-regulations occur. The co-regulations are local in the sense that they occur in some subsets of full experimental conditions. In this paper, we propose an innovative method for finding gene specific co-regulations using genetic algorithm (GA). Two novel ad hoc GAs, the single-stage and two-stage progressive GA, are proposed. They are called progressive because the initial population for the GA in a window position inherits the top-ranked individuals obtained in the preceding window position, enabling them to achieve better accuracy than the non-progressive algorithm. Experimental results with real-life gene expression data demonstrate the efficiency and effectiveness of our technique in discovering gene specific co-regulations.

1 Introduction

DNA microarray provides us with a global view of gene expression and has been used in a number of different ways. One interesting research direction is to study co-regulatory relationships among genes under different temporal conditions that are the experimental time points along the course of some biological activity when the expression of genes are extracted. It has been known that a gene may be regulated by multiple regulators along the full timeline and the phenomenon of partial (or local) co-expression between genes has been identified, meaning that gene profiles may simultaneously change in a sub-range of the time course rather than the overall time course [16]. An interesting problem is to *find the regulators of a given gene and the associated sets of experimental conditions in which such co-regulations occur*. This is called *Single Gene Approach* for gene microarray analysis [13]. The discovered co-regulated genes and the associated subsets of conditions are gene specific. The answer to this question is very helpful for human users to better understand and characterize the given gene by means of its co-regulations with other genes in the discovered sets of experimental conditions during the biological activity involved. As early DNA microarray experiments have shown that genes of similar function yield similar expression patterns [12], gene-specific co-regulations are therefore able to assist in function prediction of unknown genes through in-depth study on its correlated genes whose function has been known. In this paper, we are interested in studying the *local* co-regulations of the given gene that occur in a few neighboring, but not necessarily consecutive, conditions. Those co-regulations among conditions located far apart from each other in the timeline are disregarded. The biological rationale behind this is that genes are more likely to display biologically meaningful co-regulations at neighboring conditions. These co-regulations may experience time-lag [8], but such lagged co-regulation still often occur within a relatively short time period compared to the entire timeline involved.

Even though the problem of gene co-regulation discovery has been studied intensively in recent years, there is rare research work on single-gene co-regulation discovery. In this paper, we propose an approach for discovering local gene-specific co-regulation using genetic algorithm. The basic idea of our approach is to first find the subsets of conditions in which the given 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 subsets of conditions. Specifically, a sliding window is used to scan all the experimental conditions sequentially and the search of subsets of conditions are performed within each window position. A single-stage progressive genetic algorithm is presented, in which the top-ranked subsets of conditions

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

obtained in one window position will be used to find other good subsets in the subsequent window position. Heuristics of offline random sampling is proposed to remarkably boost the efficiency of the genetic algorithm by speeding up the fitness evaluation of individuals. A two-stage progressive genetic algorithm is also proposed to enhance the accuracy of the algorithm that may be adversely affected by using the sampling technique.

2 Related Work

Clustering analysis is currently the most used technique for gene expression data. It is able to identify genes that are co-regulated in a similar manner, forming groups or clusters, under a set of specific experimental conditions. The commonly used clustering methods in discovering gene co-regulations include hierarchical clustering method [9], k-means algorithm [9], Self-Organization Maps (SOMs) [10], SVD-based clustering algorithm [6]. Yet, most of them perform clustering based on the entire set of conditions (i.e. full dimensionality), which causes them to miss out those interesting co-regulations embedded in the lower dimensional subsets of conditions. Recently, a few subspace clustering methods for gene expression data, such as Coupled Two-Way Clustering [5], bi-cluster [4] and δ-cluster [15], are proposed. They try to find sub-matrices/blocks defined by a subset of genes on a subset of conditions that satisfy some user-defined clustering criterion. Since gene expression dataset is high-dimensional by nature, thus finding all these coherent blocks is NP-hard due to the curse of dimensionality. In the aspect of dealing with time-course gene expression data whose conditions have explicit temporal meaning, Kwon *et al.*[11] marks the changes of gene expression as an event [Rising (*R*), Constant (*C*) or Falling (*F*)] by calculating the slope of the expression value at each time interval, resulting in a string of events. Then a global sequence alignment algorithm, the Needleman−Wunsch algorithm, is employed to match the corresponding events of two genes, based on which a numerical score is generated as an indicator of the likelihood of a regulatory relationship existing between those two genes. Being a full-dimensionality method, this method may underscore the local time-lag patterns and miss them out. Ji *et al.*[8] recently proposed a method for identifying local time-lagged gene clusters. In this method, each gene will be clustered into a few so-called q-clusters whose members share the same local change pattern for q consecutive conditions. Although this method does not suffer the problem of full dimensionality, it can only identify co-regulations occurring in a few consecutive conditions. The conditions in which the gene exhibit co-regulation may not consecutive in the sense that one or a few conditions may be skipped in practice. In addition, the local patterns are rigidly restricted to have a fixed length q so this method cannot find those patterns with smaller variable lengths.

The most salient drawback of the abovementioned clustering-based methods, regardless of relying on full or partial dimensionality, lies in that they cannot provide efficient support to the single gene co-regulation problem. It will be computationally prohibitive to extract single gene co-regulations directly from the gene clusters. Furthermore, the clustering methods are less appropriate when only a few genes are likely to co-regulated [13] as these co-regulated genes may not form a strong cluster. Thus, a new method to find gene specific co-regulations is desirable.

3 Problem Formulation

To define local gene co-regulations, we need to delimit the allowable length of a subset of condition. To this end, a *window* with a fixed size, ω, will be used. The size of this window is specified by human users a priori. This may require some biological knowledge to decide the maximum possible number of conditions under which meaningful co-regulations of the given genes are to be studied. A large window allows for a study on gene co-regulations within a wider span of conditions and vice versa. To examine all the possible local subsets of conditions, this window will be slided from the leftmost to the rightmost position, with one condition offset each time. Therefore, for a gene expression dataset with M dimensions, there will be $M-\omega+1$ different positions for the sliding window with a size of ω.

Having discussed the sliding window, we can now formulate the problem to be studied in this paper mathematically. let $D = N \times M$ be the table with N genes and M conditions, representing the given microarray data, and $C = \{d_1, d_2, \ldots, d_M\}$ be the full set of experimental conditions. The set of top n subsets of conditions, denoted as S, in which the gene g of our interest is most significantly co-regulated with some other genes are defined as follows:

$$S = \{s_1, s_2, \ldots, s_n\}$$

where each element s_i $(1 \leq i \leq n)$ of S is subject to the following constraints:

1. $s_i \subseteq C$ and $|s_i| \leq \omega$;
2. For any other subset of conditions $s \notin S$, $s \subseteq C$ and $|s| \leq \omega$, we have $f(g, s_i) \geq f(g, s)$.

$|s|$ denotes the number of conditions in subset s. $f(g, s)$ is the function computing the averaged similarity (absolute value) between g and each of its k nearest neighbors in a given subset of conditions s, i.e.

$$f(g, s) = \frac{1}{k}\sum_{i=1}^{k} |sim(g, g_i)|, gi \in KNNSet(g, s, D) \quad (1)$$

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

where $KNNSet(g, s, D)$ is the set of k nearest neighbors of g from dataset D in s and $sim(g, g_i)$ is the metric used to compute the similarity between g and g_i. For gene expression data, Pearson's Correlation Coefficient (*PCC*) is often used as the similarity metric. Suppose $x = \{x_1, x_2, \ldots, x_{|s|}\}$ and $y = \{y_1, y_2, \ldots, y_{|s|}\}$ are the projections of two genes x and y in s. The similarity between x and y in s is formulated as follows:

$$sim(x, y, s) = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n \sum x_i^2 - (\sum x_i)^2][n \sum y_i^2 - (\sum y_i)^2]}} \quad (2)$$

By averaging the absolute values of PCC of all the k nearest neighbors of g, $f(g, s)$ is able to measure the overall co-regulations of g within its neighborhood in s.

The set of co-regulated genes for the given gene g in a subset of conditions s are simply those genes, out of the kNNs of g in s, that are significantly co-regulated with g.

Finally, by putting together S and the co-regulated genes of g in each member of S, we can obtain the *complete answer set* $\mathcal{A}$ to the problem, which can be represented as

$$\mathcal{A} = \{< g_i, s_j > | g_i \in KNNSet(g, s_j, D) \text{ AND } s_j \in S \text{ AND } \quad sim(g, g_i) \geq \delta\} \quad (3)$$

The answer to the problem is basically a set of pairs whose first element represents a closely co-regulated gene g_i of g and the second element represents the subset of conditions in which co-regulation occur between g_i and g. The threshold δ is used to specify the desired significance of co-regulations.

4 Genetic Algorithm for Discovering Gene Co-regulations

The evolutionary algorithm [7], such as genetic algorithm, is inspired by the Darwinian theory of evolution that a competition among the various species lead to survivals of the only fittest after a natural selection process. The fitter individuals tend to mate each other more often, resulting in better individuals [2]. Often, typically general-purpose black-box GA software on straightforward string encodings does not work very well [1]. Therefore, an *ad hoc* genetic algorithm needs to be designed to well suit the specific problem under study basing on a good understanding of the problem. This involves choosing appropriate individual representation, fitness function, selection operators and search operators. In the sequel, we will elaborate on the designing details of the genetic algorithm for single gene co-regulation discovery.

4.1 Individual Representation

To prevent the terminological ambiguity arisen from the "gene" in the microarray dataset and the "gene" of individual used in GA domain, we will call the the "gene" of individual used in GA domain as "*bit*" instead for the rest of this paper. Our GA technique uses standard binary individual encoding; all individuals are represented by strings with fixed and equal length ω, where ω is the window size. Using binary alphabet $\Sigma = \{0, 1\}$ for gene alleles, each bit in the individual will take on the value of "0" and "1", indicating whether or not its corresponding condition is selected, respectively ("0" indicates the corresponding condition is absent and vice versa for "1"). For a simple example, the individual "100101" when $\omega = 6$ means that the 1^{st}, 4^{th} and 6^{th} conditions in the current window are selected, which is a 3-dimensional subset.

4.2 Fitness Function

Recall that the subset search involves finding those subsets of conditions s that are able to maximize the following function (referring to Eq. (1)):

$$f(g, s) = \frac{1}{k} \sum_{i=1}^{k} |sim(g, g_i)|, gi \in KNNSet(g, s, D) \quad (4)$$

The fitness function of a subset s, in terms of the given gene g, is defined as follows:

$$f_{fit}(g, s) = \begin{cases} f(g, s), & \text{if } f(g, s) > 0 \\ f(g, s) + \epsilon, & \text{if } f(g, s) = 0 \end{cases} \quad (5)$$

Because it is likely that $f(g, s) = 0$, so a small positive *adjusting factor* ϵ (say 0.05) is added to $f(g, s)$ in this case to make it still possible for this individual to be selected into the population for next generation. This is to help ensure the necessary diversity of individuals in the populations. A higher value for adjusted fitness function indicates a fitter solution and vice versa.

4.3 Selection

In our work, *fitness-proportionate selection*, also known as roulette-wheel selection, is used to select fitter solutions in each step of the evolution. Fitness-proportionate selection is a stochastic selection method where the selection probability of a subset of conditions, given a gene g, is proportional to the value of its adjusted fitness function $f_{fit}(s, g)$, i.e.,

$$Pr(s, g) = \frac{f_{fit}(s, g)}{\sum_{i=1}^{P} f_{fit}(s_i, g)} \quad (6)$$

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

where P is the population size. Since $f_{fit}(s, g) > 0$, then each individual stands a chance of being selected for the next generation.

4.4 Crossover and Mutation

Crossover and mutation are two most commonly used search operators in genetic algorithm. Following Holland's canonical GA specification [7], the crossover and mutation used in this paper is *single-point crossover* and *bit-wise mutation*. In single-point crossover, a crossover locus on two parent individuals is selected and all the bits beyond that locus in the strings are swapped between the two parents, producing two new children. The bit-wise mutation involves flipping each bit randomly and leads to generating a new children. In our work, all the new individuals generated by crossover and mutation are of the same length, i.e. ω, as their parent(s). There are two associated probabilities, p_c and p_m, used to determine the frequencies for applying crossover and mutation, respectively. Normally, we have $p_c >> p_m$, meaning that crossover is performed in a much higher frequency than mutation.

4.5 Single-stage Progressive Genetic Algorithm

Genetic algorithm is applied for each sliding window position independently in order to identify subsets of conditions in the window. The initial population for each window position is generated randomly. The top n subsets of conditions within the window will be maintained as the *candidate individuals*. Therefore, the total number of subsets of conditions obtained after a window scan on all conditions will be $n * (M - \omega + 1)$, given that there are $M - \omega + 1$ different window positions. The top n subsets of conditions are selected from these $n * (M - \omega + 1)$ individual candidates, together with the closely co-regulated genes in respective subsets of conditions relative to the given gene.

Considering the fact that the windows locating at two consecutive positions are highly overlapped with each other and the results of the previous window position are very useful for the subsequent one, thus a more effective method is to adapt a *progressive* fashion in the search process. In contrast to the naive GA-based method, the progressive method includes into the initial population for each window position, except the first one, the *modified* individuals produced in the previous window as part of respective initial population. In other words, the initial population comes from two sources, the modified individuals from previous window and some other individuals generated randomly with a bias. Please note that the entire initial population for the window in the first position is generated randomly without any bias. Next, we will elaborate on individual modification and biased random population generation.

Individual Modification. Let us suppose that a top-ranked subset of conditions s is obtained in the window at the i^{th} position, denoted as W_i. Two cases will be considered here in which different modification schemes will be applied:

- **Case 1**: The first bit of s is "1", meaning that s contains the first condition in window W_i. An example of such a subset s can be "101101";
- **Case 2**: The first bit of s is "0", suggesting that s does not contain the first condition in window W_i. An example of such a subset s can be "001101".

Two modification operations, *deletion* and *insertion*, will be performed on s to generate new individuals for the next window as follows:

1. The first bit of s will be deleted;
2. If the first bit of s is "1", then *two* new individuals will be generated by inserting "0" and "1" respectively into the tail position of the string obtained in the first substep. If the first bit of s is "0", then only *one* new individual will be generated by inserting "1" into the tail position.

Now, let we denote by regular expressions $\Omega\{\omega - 1\}0$ and $\Omega\{\omega - 1\}1$ the strings with the format of $\underbrace{\Omega \ldots \Omega}_{\omega-1}0$ and $\underbrace{\Omega \ldots \Omega}_{\omega-1}1$, respectively. Ω is a "don't care" symbol that can be instantiated by either "0" or "1". Based on the above discussion, we can see that the number of modified individuals matching $\Omega\{\omega - 1\}1$ is no less than the number of those matching $\Omega\{\omega - 1\}0$.

The motivation for modifying the top-ranked individuals of the preceding window is because they preserve the segment of strings that are potentially contained in the good individuals in the current window due to the high degree of overlap between two consecutive window positions. Thus, these modified individuals, if properly modified, can provide useful guidance to the search process for the current window. The reason why we do not add "0" to those individuals in Case 2 is because the new modified individuals, with "0" being added at end, has been evaluated in the previous window and therefore should be excluded from the initial population of current window.

Random Population Generation with Bias. Besides modifying the top-ranked individuals obtained from the preceding window, a random population will be generated to create the initial population for the current window. To achieve the necessary diversity of the initial population, it is desired to have approximately equal change for the presence and absence of each condition in every individual of the initial population for each window. Unfortunately, random

population generation without any bias may not be able to meet this need due to inclusion of modified individuals from the preceding window. As discussed earlier, among those individuals coming from the preceding window, there is usually a higher number of modified individuals matching $\Omega\{\omega-1\}1$ than those matching $\Omega\{\omega-1\}0$. Therefore, it is necessary to have a mechanism to offset this unbalance by means of introducing some *bias* in the initial population generation for each window. Probabilistically speaking, such bias will help generate more individuals with the format of $\Omega\{\omega-1\}0$ than those with the format of $\Omega\{\omega-1\}1$. To this end, we first generate random binary strings with a length of $\omega-1$ and then add "0" and "1" to the end of these strings with a probability of p_0 and p_1, respectively, $p_0 \geq p_1$. p_0 and p_1 are defined in three cases as follows:

$$\begin{cases} p_0 = \frac{0.5P-n_0}{P-n_0-n_1}, p_1 = \frac{0.5P-n_1}{P-n_0-n_1}, & \text{if} \quad n_0 \leq n_1 < 0.5P \\ p_0 = 1, p_1 = 0, & \text{if} \quad n_0 < 0.5P \leq n_1 \\ p_0 = 0, p_1 = 0, & \text{if} \quad n_0 + n_1 = P \end{cases} \tag{7}$$

where n_0 and n_1 denote the number of modified individuals from the preceding window matching $\Omega\{\omega-1\}0$ and $\Omega\{\omega-1\}1$, respectively. n_0 and n_1 are subject to $n_0 \leq n_1$ and $n_0 + n_1 \leq P$.

The detailed algorithm of progressive GA for single gene co-regulation discovery is given in Figure 1. We call it single-stage progressive GA to distinguish it from the two-stage progressive GA we will present later in this paper. $CandidateSet$ is the set for storing the top n best individuals obtained in all generations and it is generation-wisely updated (Line 7-8). There are two nested while loops (Line 2 and 4). The outer while loop examines all the possible window positions, whereas the inner loop performs GA-based subset search within each window. The single-stage progressive GA differs the naive GA in the initial population generation for each different window position (Line 3), in which individual modification and biased random population generation are performed.

Algorithm: Single-stage_Progressive_GA(the given gene g)

1. $CandidateSet \leftarrow \emptyset$;
2. WHILE (the window does not reach the last position) DO {
3. $S_{pop} \leftarrow$ initial population of P strings;
4. WHILE (evolution_stop_criterion=false) DO {
5. FOR each individual s in S_{pop} DO
6. evaluate fi tness of s using the whole dataset;
7. $CandidateSet \leftarrow CandidateSet \cup$ top n individuals in
8. this generation;
9. $S_{pop} \leftarrow selection(S_{pop})$;
10. $S_{pop} \leftarrow crossover(S_{pop}, p_c)$;
11. $S_{pop} \leftarrow mutation(S_{pop}, p_m)$; } }
12. $\mathcal{S} \leftarrow$ Top n individuals in $CandidateSet$;
13. $\mathcal{G} \leftarrow$ Co-regulated kNNs of g in $\mathcal{S}$;
14. Return $(\mathcal{G}, \mathcal{S})$;

Figure 1. Single-stage Genetic Algorithm

5 Speed Up Genetic Algorithm

5.1 Random Sampling

Like many other GA applications, the most computationally expensive step performed in our genetic algorithm lies in the fitness evaluation of individuals. The problem of slow fitness evaluation in our work is because the fitness evaluation for each individual (i.e. subset of conditions) involves scanning all the genes in the dataset in order to find the k nearest neighbors of the given gene for computing $f(g,s)$. This will be slow as the number of genes in the microarrary data is usually large. To speed up fitness evaluation, we draw on the sampling technique and evaluate fitness of individuals only based on the random samples, rather than on the entire dataset.

We generate a moderate number of samples and adopt a *round-robin* paradigm in utilizing the samples in different generations. Specifically, suppose t different random samples are generated for a GA with N_g generations, where $t < N_g$. Each random sample is assigned an index number, starting from 0 to $t-1$. The i^{th} generation will use the sample with the index number of (i mod t), for $1 \leq i \leq N_g$.

Using the random samples, $f_{fit}(g,s)$ can now be computed as

$$f_{fit}(g,s) = \frac{1}{k}\sum_{i=1}^{k} |sim(g,g_i)|, g_i \in KNNSet(g,s,G_s) \tag{8}$$

where G_s denotes the set of genes contained in a random sample.

5.2 Two-Stage Progressive Genetic Algorithm

Sampling enables the genetic algorithm to be performed much faster. However, if the size of sample is small then the fitness value may not be a reliable indicator of the quality of individuals. To address this problem, we employ a *two-stage GA* technique in achieving good quality of individuals in our work. In the first stage, GA is performed on the sampling data to find the n best individuals in each window. The second stage, functioning as a refining step, finds the n best individuals as the final result amongst ALL the candidate individuals obtained in the first stage using the whole dataset. Of course, evaluating each individual in the second stage is more costy compared to that in the first

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

Algorithm: Two-stage_Progressive_GA(the given gene g)
1. $CandidateSet \leftarrow \emptyset$;
2. WHILE (the window does not reach the last position) DO {
3. $S_{pop} \leftarrow$ initial population of P strings;
4. WHILE (evolution_stop_criterion=false) DO {
5. FOR each individual s in S_{pop} DO
6. evaluate fitness of s using a sampling dataset;
7. $CandidateSet \leftarrow CandidateSet \cup$ top n individuals in
8. this generation;
9. $S_{pop} \leftarrow selection(S_{pop})$;
10. $S_{pop} \leftarrow crossover(S_{pop}, p_c)$;
11. $S_{pop} \leftarrow mutation(S_{pop}, p_m)$; } }
12. FOR each individual s in $CandidateSet$ DO
13. evaluate fitness of s using entire dataset;
14. $\mathcal{S} \leftarrow$ Top n individuals in $CandidateSet$;
15. $\mathcal{G} \leftarrow$ Co-regulated kNNs of g in $\mathcal{S}$;
16. Return ($\mathcal{G}$, $\mathcal{S}$);

Figure 2. Two-stage Genetic Algorithm

stage, but this leads to a more accurate evaluation. Our two-stage method is advantageous as it enables the algorithm to find the good solutions appearing before the last generation by keeping track of the good solutions appearing along the course of the genetic algorithm.

Figure 2 presents the detailed algorithm for the two-stage progressive GA. It is similar to the algorithm of single-stage progressive GA. The major differences between them are 1) Two-stage progressive GA evaluates fitness of individuals using sampling datasets in the first stage (Line 6), while single-stage progressive GA uses the entire dataset to do so (Line 6); 2) The top-ranked individuals obtained in different generations are re-evaluated using the whole dataset in the second stage of two-stage progressive GA (Line 12-13), but this will not be performed in single-stage progressive GA.

6 Experimental Results and Evaluation

In our experiments, we use the CDC28 dataset for our experiments, as did in [8] and [11]. The dataset used for experimental purpose contains 6178 genes at 35 time points, forming a 6178×35 matrix (this dataset can be downloaded at *http://www.comp.nus.edu.sg/ jiliping/p2/YeastData.xls*).

As the experimental setup, we set the size of the sliding window $\omega = 10$, the nearest neighbors considered $k = 10$, the number of subsets of individuals returned in the end (as well as the number of top individuals kept as individual candidates in each window) $n = 10$, the number of generations for the GA in each window position $N_g = 20$, the population size in each generation $P = 30$, the frequency of applying crossover $p_c = 0.8$ and the frequency of applying mutation $p_m = 0.2$. All the experimental evaluations are carried out on a Pentium 4 PC with 256MB RAM.

6.1 Efficiency Study

We start the performance evaluation with the investigation of the effect of parameters such as the number of genes and the window size on the efficiency of our method.

Effect of number of genes. The number of genes directly affects the efficiency of evaluating fitness of individuals. Fitness evaluation involves scanning the genes in the microarray dataset and computing the Pearson's Correlation Coefficient between the given gene and each other gene in a given subset of conditions. Thus, fitness evaluation is expected to be linear with respect to the number of genes in the dataset. The two-stage progressive GA employs random sampling to enhance the speed. Different sampling ratios, 0.1, 0.25 and 0.5, are tested in this experiment. Here, the sampling ratio refers to as the ratio of the number of genes in the sample against the total number of genes in the whole dataset. Figure 3 shows the running time of the single-stage and two-stage progressive GA under varying number of genes. The results verify the linear behavior of the running time as we expect. Moreover, this result also demonstrates that two-stage GA, using sampling as an effective means for speedup, is more efficient than the single-stage GA.

Effect of window size. The window size determines the size of the search space for subsets of conditions within each window, which is in an exponential order of the window size. This does not necessarily mean that the running time of our algorithm will be exponential with respect to the window size whatsoever. The actual running time is depended on how the search workload within each window is specified. More precisely, if we use the fixed number of generations and population size for each generation in the GA, i.e. *fixed number of individuals* to be evaluated in each window, then the total search workload for each window will be the same. In this case, increase in window size will consequently lead to a decrease, rather than an increase, in the number of window positions and therefore a drop of running time. A *fixed ratio search* scheme, in contrast, performs a search workload that is proportional to the size of search space in this window. The time complexity now becomes quadratic with respect to the window size. The running time of these two search schemes are presented in Figure 4. For each window position, the search workload with a fixed number of individuals is set to be 400 and that of a fixed ratio is 50% of the search space of the window.

6.2 Effectiveness Study

To analyze the effectiveness of our method, experiments are performed to test fitness enhancement of progressive

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

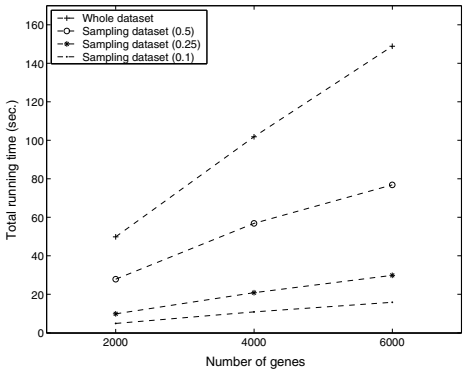

Figure 3. Running time for single-stage and two-stage GA (with different sampling ratios) under varying number of genes

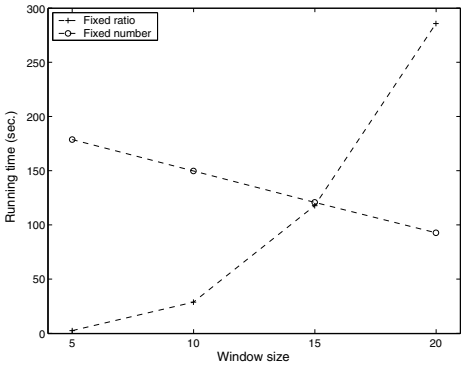

Figure 4. Running time under varying window sizes

GA versus non-progressive scheme and the convergence of our method.

Fitness enhancement by using progressive GA. We first study the contribution of progressive GA used in our method to enhancing the fitness of individuals, compared to the case when non-progressive GA is used. They primarily differ in that the progressive GA inherits the top-ranked individuals obtained from the previous window position with appropriate modifications and bias is introduced in the initial random population generation for the current window position, while the non-progressive GA evaluates each window independently and the entire initial population for each window position is generated randomly. Figure 5 presents the *averaged* fitness of top 10 individuals for each window position (from No. 1 to 26). The result demonstrates that the progressive GA outperforms non-progressive scheme in term of fitness in up to 88% of the window positions and fitness improvement by over 10% is observed at about 25% of the window positions. This result indicates that progressive GA is more capable of finding fitter individuals than the non-pregressive GA.

Convergence study. GA tends to produce an increasing number of fitter individuals as evolution proceeds, referring to as the phenomenon of convergence. In this experiment, we investigate the convergence of our technique. Without losing generality, three window positions, the first, middle and last (1^{st}, 13^{th} and 26^{th}), are picked up for this study. For each generation, the number of individuals with relative high fitness (> 0.7 in this experiment) are counted. As we can see from Figure 6 that the number of individuals with high fitness is increased as the GA evolves, which indicates a good convergence of our method. In addition, the good individuals do not only appear in the last generation, though the overall convergence of the GA has been observed. A small number of good individuals have been observed in the earlier generations of the GA. This verifies the validity of keeping track of the top-ranked individuals in each generation of GA in our approach to prevent the loss of good individuals appearing in different, particularly the early, generations of the GA.

7 Conclusions

This paper investigates the problem of gene co-regulation discovery problem in DNA microarray data. Unlike most of the existing methods that find gene co-regulations utilizing clustering analysis, our approach aims to discover co-regulations from a single gene perspective. We are interested in finding the regulators of a given gene and the associated sets of experimental conditions in which such co-regulations occur. The basic idea of our approach is to first find the subsets of conditions in which the given gene g is most significantly co-regulated with other genes and the co-regulated genes of g are reported by selecting from its nearest neighbors in these subsets of conditions.

Considering the search space of subsets of conditions is typically large for microarray dataset, genetic algorithm (GA) is employed. Due to inapplicability of the general-purpose black-box GA for our problem, two ad-hoc GA-based algorithms are proposed, i.e. the single-stage and two-single progressive GAs. Their salient feature is that they try to make use of the top-ranked solutions found in each window position in guiding the search process for its subsequent window position to boost the overall accuracy of the algorithm. Experiments are conducted for performance evaluation. The experimental results show that our method

Proceedings of the 18th IEEE International
Conference on Tools with Artificial Intelligence (ICTAI'06)
0-7695-2728-0/06 $20.00 © 2006 IEEE

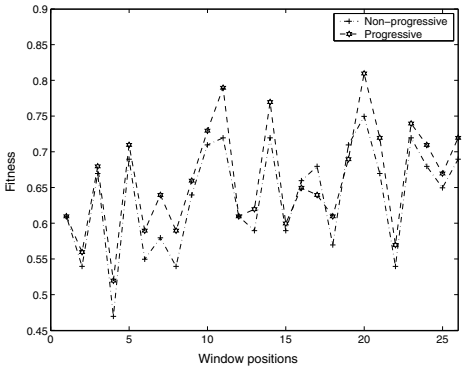

Figure 5. Fitness of non-progressive and progressive GA

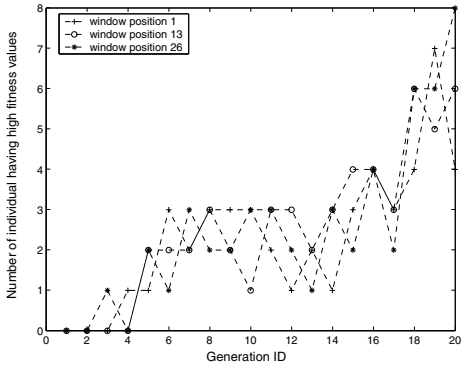

Figure 6. Number of individuals having high fitness values in three window positions

is effcient and effective in discovering gene-specific local co-regulations from gene expression data.

Acknowledgement

The authors would like to thank Dr. Malcolm I. Heywood and Dr. Christian Blouin, both from Faculty of Computer Science at Dalhousie University, for their useful suggestions on the draft of this paper. This research work is supported in part by research grant of Natural Sciences and Engineering Research Council of Canada (Grant No.: 312423).

References

[1] C. C. Aggarwal, J. B Orlin and R. P Tai. Optimized Crossover for the Independent Set Problem. *Operational Research* 45(2):226-234, 1997.

[2] C. C. Aggarwal and P.S. Yu. An Effective and Effi cient Algorithm for High-dimensional Outlier Detection. *VLDB Journal*, 14, pp 211-221, 2005.

[3] S. Bhattcharrya. Direct Marketing response Meodels Using Genetic Algorithms. *KDD'98*, pp 144-148, 1998.

[4] Y. Cheng and G.M. Church, Biclustering of Expression Data. In *Proc. International Conference on Intelligent Systems for Molecular Biology (ISMB)*, vol. 8, pp. 93-103, 2000.

[5] G. Getz, E. Levine, and E. Domany, Coupled Two-Way Clustering Analysis of Gene Microarray Data, in *Proc. Natioal Academy of Science*, vol. 97, no. 22, pp. 12079-12084, 2000.

[6] G. H Golub and C. F. Van Loan. *Matrix Computations*, Johns Hopkins University Press, 1983.

[7] J. Holland. *Adaption in Natural and Artifi cial Systems*. MIT Press, Cambridge, 1992.

[8] L. Ji and K. L. Tan. Identifying Time-Lagged Gene Clusters on Gene Expression Data. *Bioinformatics*, Vol. 21, No. 4, pp. 509-516, 2005.

[9] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*, Prentice Hall International, USA, 1998.

[10] T. Kohonen. *Self-Organization Maps*. Springer-Verlag, Berlin Heidelberg, 1995.

[11] A. T. Kwon, H. H. Hoos, and R. Ng. Inference of transcriptional regulation relationships from gene expression data. *Bioinformatics*, 19, 905-912, 2003.

[12] C. A. Orengo, D. T. Jones and J. M. Thornton. *Bioinformatics. Genes, proteins and Computers*. BIOS Scientifi c Publishers ltd, Oxford, UK, 2003.

[13] T. Speed, J. Fridlyand, Y. H. Yang and S. Dudoit. Discrimination and clustering with microarray gene expression data. *2001 Spring Meeting of International Biometric Society Eastern North American Region (ENAR'01)*, Charlotte NC, 2001.

[14] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Bostein and P. Brown. Clustering Methods for the Analysis of DNA Microarray Data. *Technical Report*, Stanford, 1999.

[15] J. Yang, W. Wang, H. Wang, and P.S. Yu, δ-Cluster: Capturing Subspace Correlation in a Large Data Set. In *Proc. 18th International Conference on Data Engineering (ICDE'02)*, pp. 517-528, 2002.

[16] Y. Zhang, H. Zha, J. Z. Wang, C. Chu. Gene Co-regulation vs. Co-expression 8^{th} *Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004)*, poster, San Diego, CA, 2004.