An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis

Patrick C. H. Ma, Keith C. C. Chan, Xin Yao, *Fellow, IEEE*, and David K. Y. Chiu

***Abstract*—Clustering is concerned with the discovery of interesting groupings of records in a database. Many algorithms have been developed to tackle clustering problems in a variety of application domains. In particular, some of them have been used in bioinformatics research to uncover inherent clusters in gene expression microarray data. In this paper, we show how some popular clustering algorithms have been used for this purpose. Based on experiments using simulated and real data, we also show that the performance of these algorithms can be further improved. For more effective clustering of gene expression microarray data, which is typically characterized by a lot of noise, we propose a novel evolutionary algorithm called evolutionary clustering (EvoCluster). EvoCluster encodes an entire cluster grouping in a *chromosome* so that each *gene* in the *chromosome* encodes one cluster. Based on such encoding scheme, it makes use of a set of reproduction operators to facilitate the exchange of grouping information between *chromosomes*. The fitness function that the EvoCluster adopts is able to differentiate between how relevant a feature value is in determining a particular cluster grouping. As such, instead of just local pairwise distances, it also takes into consideration how clusters are arranged globally. Unlike many popular clustering algorithms, EvoCluster does not require the number of clusters to be decided in advance. Also, patterns hidden in each cluster can be explicitly revealed and presented for easy interpretation even by casual users. For performance evaluation, we have tested EvoCluster using both simulated and real data. Experimental results show that it can be very effective and robust even in the presence of noise and missing values. Also, when correlating the gene expression microarray data with DNA sequences, we were able to uncover significant biological binding sites (both previously known and unknown) in each cluster discovered by EvoCluster.**

***Index Terms*—Bioinformatics, clustering, DNA sequence analysis, evolutionary algorithms (EAs), gene expression microarray data analysis.**

I. Introduction

GIVEN a database of records each characterized by a set of attributes, the clustering problem is concerned with the discovering of interesting groupings of records based on the values of the attributes. Many algorithms have been developed to tackle different clustering problems in a variety of application domains and they have been proven to be very effective [1]–[3]. Recently, some of these algorithms have been used to uncover hidden groupings in gene expression microarray data [4]–[6].

Gene expression is the process by which a gene's coded information is converted into the structures present and operating in a cell. Gene expression occurs in two major stages: transcription and translation. During transcription, a gene is copied to produce an RNA molecule (a primary transcript) with essentially the same sequence as the gene; and during translation, proteins are synthesized based on the RNA molecule. If one would like to prevent an undesirable genes, such as cancerous genes, from expressing, the transcription process should be prevented as much as possible from taking place so that the corresponding undesirable functional proteins will not be synthesized [7], [8].

To prevent the transcription process of undesirable genes from taking place, a set of transcription factor binding sites must be located. These sites consist of untranscribed nucleotide sequences located on the promoter regions of the genes and are responsible for activating and regulating the process. If they are located, we can then bind appropriate molecules, such as protein repressors, to these sites so that the genes they correspond to cannot be activated [8]. To locate these transcription factor binding sites, coexpressed genes may need to be identified. Coexpressed genes are genes that have similar transcriptional responses to the external environment (e.g., temperature, pH value, pressure, etc). This can be an indication that they are coregulated by the same transcription factors and therefore have common binding sites. To identify coexpressed genes, one can cluster the gene expression microarray data obtained by performing DNA microarray experiments [9]–[15] on these genes. Genes that are grouped into a cluster are likely to be coexpressed genes. By analyzing the promoter regions of these genes, we may be able to discover patterns, which have relatively high occurring frequencies compared with other sequence fragments, that are possible binding sites of these genes [16].

Three popular clustering algorithms have been used to cluster gene expression microarray data. They include the hierarchical clustering algorithm [4], the k-means algorithm [5] and the self-organizing map (SOM) [6]. Using simulated and real microarray data, we compared and contrasted the performance of these algorithms (in Section IV). Since various types of noise are normally introduced at different experimental stages during data collection, such as when producing the DNA array, preparing biological samples, and extracting of the results, etc. [17], the performance of these algorithms are less than ideal [18]. To effectively handle noisy data, we need an algorithm that is able to overcome noise. Here, we propose such an algorithm for clustering gene expression microarray data. This algorithm, which

Manuscript received April 29, 2004; revised February 2, 2005; July 20, 2005. This work was supported in part by the Hong Kong Polytechnic University under Grant RG1E.

P. C. H. Ma and K. C. C. Chan are with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong, China (e-mail: cschma@comp.polyu.edu.hk; cskcchan@comp.polyu.edu.hk).

X. Yao is with CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

D. K. Y. Chiu is with the Biophysics Interdepartmental Group and the Department of Computing and Information Science, University of Guelph, Guelph, N1G 2W1 ON, Canada (e-mail: dchiu@snowhite.cis.uoguelph.ca).

Digital Object Identifier 10.1109/TEVC.2005.859371

we call the EvoCluster algorithm, is based on the use of an evolutionary approach. Compared with other evolutionary and nonevolutionary based clustering algorithms, EvoCluster has a number of desirable characteristics.

1) It encodes the entire cluster grouping in a *chromosome*[1] so that each *gene* encodes one cluster and each cluster contains the labels of the data records grouped into it.
2) Given the above encoding scheme, it has a set of crossover and mutation operators that facilitates the exchange of grouping information between two *chromosomes* on one hand and allows variation to be introduced to avoid trapping at local optima on the other.
3) It makes use of a fitness function that measures the interestingness of a particular grouping of data records encoded in a *chromosome*.
4) Unlike similarity measures that are based on local pairwise distances [19] that may not give very accurate measurements in the presence of very noisy data, the proposed interestingness measure is probabilistic and it takes into consideration global information contained in a particular grouping of data.
5) It is able to distinguish between relevant and irrelevant feature values in the data during the clustering process.
6) It is able to explain clustering results discovered by explicitly revealing hidden patterns in each cluster in an easily understandable if-then rule representation.
7) There is no requirement for the number of clusters to be decided in advance.

For performance evaluation, we have tested EvoCluster using both simulated and real data. Experimental results show that it can be very effective and robust even in the presence of noisy and missing values.

The rest of this paper is organized as follows. In Section II, we provide an overview of the existing clustering algorithms used for identification of coexpressed genes. In Section III, the proposed evolutionary algorithm called EvoCluster is described in details. In Section IV, we discuss how EvoCluster can be evaluated and compared with some existing clustering algorithms using both simulated and real data. The evaluation results and the biological interpretation of the clusters discovered by EvoCluster are then presented and discussed. In Section V, we give a summary of this paper and make a proposal on the directions for future work.

II. Clustering Algorithms for Identification of Coexpressed Genes

Of the many existing clustering algorithms, some of them, including the hierarchical agglomerative clustering algorithm [1], the k-means algorithm [2] and the SOM [3], have been used to cluster gene expression microarray data. These algorithms can be described briefly below.

[1]Since the development of evolutionary algorithms have, in some instances, taken great liberty with biological terms and theory. Such terms as *chromosomes* and *genes*, when used in a computational context, may not have the same meanings as their biological counterparts. In order to avoid possible confusion, when referring to these terms in the contexts of evolutionary computation, they are made italic.

Let us assume that we are given a set of gene expression microarray data, $\mathbf{G}$, consisting of the data collected from N genes in M experiments each carried out under different sets of conditions. Let us represent the data set as a set of N records, $\mathbf{G} = \{\boldsymbol{g_1}, \ldots, \boldsymbol{g_i}, \ldots, \boldsymbol{g_N}\}$ with each record $\boldsymbol{g_i}$, $i = 1, \ldots, N$, characterized by M attributes, $E_1, \ldots, E_j, \ldots, E_M$ whose values, $e_{i1}, \ldots, e_{ij}, \ldots, e_{iM}$, where $e_{ij} \in \text{domain}(E_j)$ represents the expression value of the ith gene under the jth experimental condition.

To discover clusters in the data, the hierarchical agglomerative clustering algorithm [1] performs a series of successive fusion of records into clusters. The fusion process is guided by a measure of similarity between clusters so that clusters that are similar to each other are merged. This fusion process is repeated until all clusters are merged into a single cluster. The results of the fusion process are normally presented in the form of a two-dimensional (2-D) hierarchical structure, called dendrogram. The records falling along each branch in a dendrogram form a cluster. Depending on user preferences, a specific number of clusters can be obtained from the dendrogram by cutting across the branches at a specific level.

Comparing to the hierarchical agglomerative clustering algorithm that does not require users to specify the number of clusters ahead of time, users of the k-means algorithm [2] are required to do so. Given a data set $\mathbf{G}$, the k-means algorithm can group the records, $\boldsymbol{g}_1 \ldots, \boldsymbol{g_i} \ldots, \boldsymbol{g_N}$, into k clusters by initially selecting k records as centroids. Each record is then assigned to the cluster associated with its closest centroid. The centroid for each cluster is then recalculated as the mean of all records belonging to the cluster. This process of assigning records to the nearest clusters and recalculating the position of the centroids is then performed iteratively until the positions of the centroids remain unchanged.

The SOM algorithm [3] is one of the best known artificial neural network algorithms. It can be considered as defining a mapping from M-dimensional input data space onto a "map"—a regular 2-D array of neurons—so that every neuron of the map is associated with an M-dimensional reference vector. The reference vectors together form a codebook. The neurons of the map are connected to adjacent neurons by a neighborhood relation, which dictates the topology of the map. The most common topologies used are rectangular and hexagonal. In the basic SOM algorithm, the topology and the number of neurons remain fixed from the beginning. The number of neurons determines the granularity of the mapping, which has an effect on the accuracy and generalization of the SOM. During the training phase, the SOM forms an elastic net that folds onto the "cloud" formed by input data. The algorithm controls the net so that it strives to approximate the density of the data. The reference vectors in the codebook drift to the areas where the density of the input data is high. Eventually, only few codebook vectors lie in areas where the input data is sparse. After the training is over, the map should be topologically ordered. This means that n topologically close (based on, say, the Euclidean distance or the Pearson correlation coefficient) input data vectors map to n adjacent map neurons or even to the same single neuron. With this idea, SOM has been used successfully in various application areas [3].

Despite some successes with existing clustering algorithms in gene expression data analysis [4]–[6], [20]–[22], there is still no single clustering algorithm that is the most "dominant" gene expression microarray data clustering algorithm. This may be a result of their use of such metrics and functions as the Euclidian distance measure or the Pearson correlation coefficient, etc. that do not differentiate between the importances of different variables when measuring similarities. They also do not give very accurate measurements when the data concerned are noisy and contain missing values. Since these metrics and functions measure only pairwise distances, the measurements obtained could be too "localized." Clustering algorithms based only on the local pairwise information may, therefore, miss important "global" information.

In addition to these deficiencies, clustering results obtained with the use of many clustering algorithms could be difficult to interpret. For example, although one can visualize the result of hierarchical clustering as a tree-like dendrogram and note correlations between genes, it is the users' responsibilities to discover the similarities and differences between various clusters and to decide on the number of clusters and the cluster boundaries to form. To do so, users need to have prior knowledge about the data. Similarly, for the k-means algorithm and SOM, users have to decide on the number of clusters to partition a data set into. They also have to use a separate technique to uncover underlying patterns in the clusters.

Given the increased popularity of DNA microarray technologies [23]–[26] in the study of gene variations and given that more and more such data are generated as a result of advances in these technologies, there is a great need to develop clustering algorithms that can overcome the limitations of the existing ones. Toward this goal, we propose a novel clustering algorithm that is based on evolutionary computation techniques.

III. An Evolutionary Clustering Algorithm

For a clustering algorithm to discover the best data grouping, it has to consider

$$n(N,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^N \tag{1}$$

possibilities, where N is the total number of records and k is the number of clusters [27]. To find the optimal grouping among the very large number of possibilities, we propose to use an evolutionary clustering algorithm called EvoCluster.

Evolutionary algorithms have been successfully used to solve different data mining problems [28]–[31]. They have, in particular, been used for clustering [32]–[39]. In [32] and [33], for example, the data records are encoded as *genes* in a *chromosome* and are given a label from one to k, where k is the maximum number of clusters to be discovered. Such an approach is relatively easy to implement as they do not require special evolutionary operators. Unfortunately, they are not very scalable. As the length of each *chromosome* is exactly the size of the training set, these algorithms are not very practical when handling large data sets.

An alternative data and cluster representation was proposed in [34], where the clustering problem is formulated as a graph-partitioning problem. Based on it, each data record is represented as a node in a graph and each node is mapped to a certain position in a *chromosome* and is encoded as a *gene*. The indexes of other records are encoded as *alleles* so that if a *gene* i contains value j, an edge is created in the graph to link the nodes i and j. The *alleles* in each *gene* i are therefore the nearest neighbors of i, and the users are required to specify the number of nearest neighbors as input parameter ahead of time. With this representation, an evolutionary algorithm is used to find clusters which are represented as connected subgraphs. This approach is again not very scalable. Other than the length of the *chromosomes* being again the same as the size of a dataset, there is an additional need for the nearest neighbors of data records to be computed. It also suffers from the same problems as other clustering algorithms that are based on the need to compute pairwise distance measures.

One other popular use of evolutionary algorithms (EAs) in clustering is to use them to identify the best cluster centers. In [35], [36], each *chromosome* encodes the coordinates of k centers and the standard genetic algorithm (GA) is used to find the best ones. A similar approach to identifying the best centers is to use an EA to search for optimal initial seed values for cluster centroids [37]. As in other problems, in clustering we can use domain knowledge in several ways to try to improve the performance of the algorithm. For example, we could design specialized evolutionary operators or we can hybridize the evolutionary algorithm with a conventional clustering algorithm such as the k-means algorithm. In [38], [39], each *chromosome* represents the coordinates of the cluster centroids and different crossover methods are used to generate the offspring. After crossover each *chromosome* undergoes several iterations of the k-means clustering algorithm. The authors observed that adding the k-means iterations is crucial for obtaining good results, although there can be a considerable increase of the computation time if many iterations are used. This kind of hybridization raises the question of how to allocate the computing time—for example, using many generations of the EAs and a few iterations of the local methods or running the EAs for a few generations and using the local methods to improve the solutions. In principle, the centroid-based representation has the advantage that the *chromosomes* are shorter because they only need to encode the coordinates of the k centroids. This means that the length of the *chromosome* is proportional to the dimensionality of the problem and not the size of the training set. However, just like many EA-based clustering methods, the drawback of the centroid-based representation is that the number of clusters needed to be specified in advance. Moreover, the similarity functions used such as Euclidean distance or correlation coefficient for measuring the similarity of the records do not differentiate between the importances of different attributes. Therefore, they do not give accurate measurements when the data concerned are noisy and contain missing values. In addition, these similarity functions measure only pairwise distances, the measurements obtained could be too "localized."

Clustering gene expression data as a new area of research poses new challenges due to its unique problem nature that the previous EA-based clustering algorithms were not originally designed to deal with. As discussed in [40], there are some new challenges in dealing with gene expression data. For example, the presence of both biological and technical noise inherent in the data set, the presence of large number of irrelevant attributes, and the explanatory capability of an algorithm to help biologists in gaining more understanding of the underlying biological process. And also, the clustering structure of gene expression data is usually unknown.

To effectively tackle the challenges posed by gene expression data, the proposed EvoCluster algorithm has several characteristics that make it different from existing clustering algorithms. The differences can be summarized as follows.

1) It encodes the entire cluster grouping in a *chromosome* so that each *gene* encodes one cluster and each record label of a *gene* encodes the number of a data record.
2) Given the above encoding scheme, it makes use of crossover and mutation operators that facilitates the exchange of grouping information between two *chromosomes* on one hand and allows variation to be introduced to avoid trapping at local optima on the other.
3) It makes use of a fitness function that measures the interestingness of a particular grouping of data records encoded in a *chromosome*.
4) This measure of interestingness, unlike many similarity measures that is based on local pairwise distances [19] which do not give accurate measurements when the data concerned are noisy and contain missing values, takes into consideration global information contained in a particular grouping of data.
5) It is able to distinguish between relevant and irrelevant feature values in the data during the clustering process.
6) It is able to explain clustering results discovered by explicitly revealing hidden patterns in each cluster in an easily understandable if-then rule representation.
7) There is no requirement for the number of clusters to be decided in advance.

Like other evolutionary algorithms [41]–[46], the EvoCluster algorithm consists of the following steps.

1) Initialization of a population of *chromosomes* with each representing a unique cluster grouping.
2) Evaluate the fitness of each *chromosome*.
3) Select *chromosomes* for reproduction using the roulette wheel selection scheme.
4) Apply crossover and mutation operators.
5) Replace the least fit *chromosomes* in the existing population by the newly generated offspring.
6) Repeat Steps 2)–5) until the stopping criteria are met.

A. Cluster Encoding in Chromosomes and Population Initialization

To evolve the best cluster grouping, EvoCluster encodes different grouping arrangements in different *chromosomes* so that one *chromosome* encodes one particular cluster grouping. In each such *chromosome,* each *gene* encodes one cluster

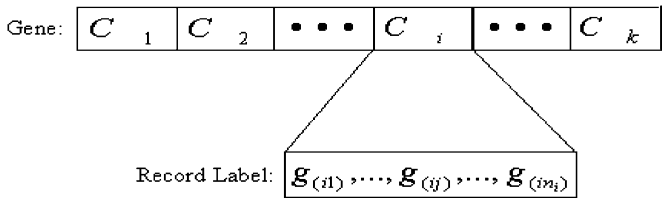

Fig. 1. Chromosome encoding scheme.

[43]. Hence, if a particular *chromosome* encodes k clusters, $C_1, \ldots, C_i, \ldots, C_k$, it has k *genes*. Since each cluster contains a number of data records, a *gene* encoding a cluster can be considered as being made up of the labels of a number of data records in gene expression data. For example, assume that C_i contains n_i, $\boldsymbol{g}_{(i1)}, \ldots, \boldsymbol{g}_{(ij)}, \ldots, \boldsymbol{g}_{(in_i)}$, where $\boldsymbol{g}_{(ij)} \in \mathrm{G} = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_i, \ldots, \boldsymbol{g}_N\}$, the unique labels of these records, $\boldsymbol{g}_{(i1)}, \ldots, \boldsymbol{g}_{(ij)}, \ldots, \boldsymbol{g}_{(in_i)}$, can be encoded in each *gene* C_i so that a *chromosome* that encodes a particular cluster grouping can then be represented diagrammatically, as shown in Fig. 1.

For the initial population, each *chromosome* is randomly generated by EvoCluster in such a way that the number of clusters k to be encoded in a *chromosome* is first generated randomly from within a certain range of acceptable numbers. Each of the records in $\mathbf{G} = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_i, \ldots, \boldsymbol{g}_N\}$ is then assigned, also randomly, to one of the k clusters [41], [42].

B. Selection and Reproduction

Reproduction in EvoCluster consists of the application of both the crossover and mutation operations. As the evolutionary process enters into reproduction, two *chromosomes* are selected as parents for crossover using the roulette-wheel selection scheme [42] so that each parent's chance of being selected is directly proportional to its fitness.

Since it is the cluster grouping encoded in each *chromosome* that conveys the most important information, our crossover operators are designed to facilitate the exchange of grouping information. And since this process can be "guided" or "unguided," our crossover operators are also classified in the same way. We have a "guided" operator and an "unguided" operator. For the "guided" crossover (GC) operator, the exchange of grouping information is not totally random in the sense that the grouping information of the "best formed" clusters is preserved during the crossover process. For the "unguided" crossover (UGC) operator, the exchange of the grouping information between clusters takes place randomly.

Assume that two parent *chromosomes* $P1$ and $P2$ are chosen so that $P1$ encodes k_1 *genes,* $C_1^{P1}, \ldots, C_i^{P1}, \ldots, C_{k_1}^{P1}$ (with each corresponding to a cluster), and $P2$ encodes k_2 *genes,* $C_1^{P2}, \ldots, C_i^{P2}, \ldots, C_{k_2}^{P2}$, i.e., the number of clusters encoded in each *chromosome* can be different. Assume also that MIN is a user-defined minimum number of clusters encoded in a *chromosome* and MAX is a user-defined maximum number of clusters encoded in a *chromosome*. Then the following are the steps taken by the guided and unguided operators when crossover is performed. (Note: The probability for a *gene* or a record label in a *gene* to be selected by both crossover and mutation operators can be randomly generated from within a certain range $[L_g, U_g]$ or $[L_r, U_r]$, respectively, where $0.0 \leq L_g \leq U_g \leq 1.0$ and $0.0 \leq L_r \leq U_r \leq 1.0$, and L_g, U_g, L_r, and U_r can be set by users or also generated randomly. Moreover, in guided crossover operator and also mutation operators (b), (d), and (f), the interestingness of each *gene* is determined based on the interestingness measure described in Section III-C.)

1) The Guided Crossover (GC) and Unguided Crossover (UGC) Operators:

1) Set $P_{\text{g}-\text{rpl}}$, the probability for a *gene* to be selected for crossover.
2) Set $P_{\text{r}-\text{rpl}}$, the probability for a record label in a *gene* to be replaced by another record label from another *gene* in another parent.
3) *Gene* selection procedure.
    a) For the UGC, based on $P_{\text{g}-\text{rpl}}$ and using a random number generator, each *gene* in $P1$ is scanned to decide if it should be selected. Those selected are then represented as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, where $l_1 < k_1, k_2$ and $C_{(i)}^{P1}$, $i = 1, \ldots, l_1$ is in $\{C_1^{P1}, \ldots, C_i^{P1}, \ldots, C_{k_1}^{P1}\}$.
    b) For the GC, based on $P_{\text{g}-\text{rpl}}$, $N_{\text{g}-\text{rpl}} < k_1, k_2$, the number of interesting *genes* to be selected can be determined. Then select $N_{\text{g}-\text{rpl}}$ of the most interesting ones in $P1$ and $P2$, respectively. Rank them in descending order of interestingness and represent them as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$ and $\{C_{(1)}^{P2}, \ldots, C_{(i)}^{P2}, \ldots, C_{(l_1)}^{P2}\}$ so that $C_{(1)}^{P1}$ and $C_{(1)}^{P2}$ is the most interesting in $P1$ and $P2$, respectively.
4) Record label replacement procedure.
    a) For the UGC, for each *gene* in $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, say $C_{(1)}^{P1}$, randomly select one *gene* from $P2$, say, C_i^{P2} that has not previously been selected, based on $P_{\text{r}-\text{rpl}}$ and using a random number generator, each record label in $C_{(i)}^{P1}$ can be scanned to identify those that should be replaced by a record label in C_i^{P2}. Record labels that are selected for replacement are then represented as $\{g_{(i1)}^{P1}, \ldots, g_{(ij)}^{P1}, \ldots, g_{(in_i)}^{P1}\}$ and removed from $C_{(i)}^{P1}$. Randomly select n_i or $|C_i^{P2}|$ record labels, whichever is smaller, from C_i^{P2} and replace those removed. Repeat the above steps for all the other selected *genes* in $P1$.
    b) For the GC, begin with the most interesting $C_{(1)}^{P1}$, select the corresponding most interesting *gene* $C_{(1)}^{P2}$ from $P2$. Based on $P_{\text{r}-\text{rpl}}$, scan through each record label in $C_{(1)}^{P1}$ to select those that should be replaced by that in $C_{(1)}^{P2}$. Those record labels to be replaced are represented as $\{g_{(il)}^{P1}, \ldots, g_{(1j)}^{P1}, \ldots, g_{(1n_1)}^{P1}\}$ and removed from $C_{(1)}^{P1}$. Randomly select n_i or $|C_{(1)}^{P2}|$ record labels, whichever is smaller, from $C_{(1)}^{P2}$ and replace those removed. Repeat the above steps for all the other selected *genes* in $P1$.
5) Repairing procedure (for producing child $Ch1$): scan through all *genes* to remove duplicates in such a way that if a record label is found in another *gene,* other than the one containing the replacement, it is removed. For those record labels that have not been assigned to any *gene* after their removals.
    a) For the UGC, they are randomly assigned to one of the *genes*.
    b) For the GC, they are reclassified into one of the *genes* using the reclassification algorithm given in the section below.
6) Repeat Steps 1)–5) with $P2$ to produce child $Ch2$.

After crossover, the children produced undergo mutation in order to avoid getting trapped at local optima on one hand and to ensure diversity on the other. EvoCluster makes available six different mutation operators that can be selected at random when a *chromosome* undergoes mutation. These operators can be classified according to whether or not the mutation process involves just the removal and reclassification of record labels or the merging and splitting of the whole *gene*. They can also be classified according to whether or not they are "guided" or "unguided." Based on these classification schemes, EvoCluster makes use of six operators.

a) The unguided remove-and-reclassify-record mutation (UGRRM) operator.
b) The guided remove-and-reclassify-record mutation (GRRM) operator.
c) The unguided merge-*gene* mutation (UGMGM) operator.
d) The guided merge-*gene* mutation (GMGM) operator.
e) The unguided split-*gene* mutation (UGSGM) operator.
f) The guided split-*gene* mutation (GSGM) operator.

The merge and split mutation operators (i.e., UGMGM, GMGM, UGSGM, and GSGM) were specifically designed to allow the length of *chromosomes* to be changed dynamically as the evolutionary process progresses. The advantage with this feature is that the number of clusters that need to be formed does not need to be specified by the users ahead of time. In the following, the details of these operators are given.

2) The Guided (GRRM) and Unguided Remove-and-Reclassify-Record Mutation (UGRRM) Operators:

1) Set $P_{\text{g}-\text{rr}}$, the probability for a *gene* to be selected.
2) Set $P_{\text{r}-\text{rr}}$, the probability for a record label in a *gene* to be removed.
3) *Gene* selection procedure.
    a) For the UGRRM, based on $P_{\text{g}-\text{rr}}$, scan through each *gene* in the *chromosome* to decide if it should be selected. Those selected are then represented as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, where $l_1 < k_1$ and $C_{(i)}^{P1}$, $i = 1, \ldots, l_1$ is a member of $\{C_1^{P1}, \ldots, C_i^{P1}, \ldots, C_{k_1}^{P1}\}$.
    b) For the GRRM, based on $P_{\text{g}-\text{rr}}$, determine $N_{\text{g}-\text{rr}} < k_1$, the number of uninteresting *genes* to be selected. Then select the $N_{\text{g}-\text{rr}}$ least interesting *genes* and represent them in ascending order of interestingness as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$ so that $C_{(1)}^{P1}$ is the least interesting *gene*.
4) Record label replacement procedure: For each *gene* in $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, based on $P_{\text{r}-\text{rr}}$, scan through each record label in each of $C_{(i)}^{P1}$ to select those

that should be removed. These record labels are represented as $\{g_{(i1)}^{P1}, \ldots, g_{(ij)}^{P1}, \ldots, g_{(in)}^{P1}\}$ and removed from $C_{(i)}^{P1}$.

5) Children repairing procedure: for those record labels that have not been assigned to any *gene* after their removals.
   a) For the UGRRM, they are randomly classified into one of the *genes*.
   b) For the GRRM, they are reclassified into one of the *genes* using the reclassification algorithm given in the section below.

3) The Guided (GMGM) and Unguided Merge-Gene Mutation (UGMGM) Operators:

1) Set $P_{\text{g-mrg}}$, the probability for a *gene* to be merged.
2) *Gene* selection procedure.
   a) For the UGMGM, based on $P_{\text{g-mrg}}$, scan through each *gene* in the *chromosome* to randomly select a *gene* for merging. Those selected are then represented as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, where $l_1 < k_1$ and $C_{(i)}^{P1}$, $i = 1, \ldots, l_1$ is a member of $\{C_1^{P1}, \ldots, C_i^{P1}, \ldots, C_{k_1}^{P1}\}$.
   b) For the GMGM, based on $P_{\text{g-mrg}}$, determine $N_{\text{g-mrg}} < k_1$, the number of uninteresting *genes* to be merged. Then select $N_{\text{g-mrg}}$ least interesting *genes* and represent them in ascending order of interestingness as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$ so that $C_{(1)}^{P1}$ is the least interesting.
3) Merging procedure: for each *gene* in $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, randomly select one other *gene* to be merged in this set. The number of *genes* remaining after merging should be greater than MIN. Otherwise, the mutation operator terminates.

4) The Guided (GSGM) and Unguided Split-Gene Mutation (UGSGM) Operators:

1) Set $P_{\text{g-splt}}$, the probability for a *gene* to be split.
2) *Gene* selection procedure.
   a) For the UGSGM. Based on $P_{\text{g-splt}}$, scan through each *gene* in the *chromosome* to decide if it should be selected for splitting. Those selected are then represented as $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$ where $l_1 < k_1$ and $C_{(i)}^{P1}$, $i = 1, \ldots, l_1$ is a member of $\{C_1^{P1}, \ldots, C_i^{P1}, \ldots, C_{k_1}^{P1}\}$.
   b) For the GSGM. Based on $P_{\text{g-splt}}$, determine $N_{\text{g-splt}} < k_1$ the number of uninteresting *genes* to be split. Then select $N_{\text{g-splt}}$ least interesting *genes*, and arrange them into ascending order of interestingness, $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$ so that $C_{(1)}^{P1}$ is the least interesting.
3) Splitting procedure: for each *gene* in $\{C_{(1)}^{P1}, \ldots, C_{(i)}^{P1}, \ldots, C_{(l_1)}^{P1}\}$, randomly split it into two clusters. The resulting number of *genes* has to be smaller than MAX. Otherwise, the mutation operator terminates.

After reproduction, EvoCluster constructs a classifier based on each of the two children *chromosomes* using a reclassification algorithm described in [47]. Record labels that have not been assigned to any *gene* are then reclassified into one of the *genes* encoded in a *chromosome*.

A simple evolutionary algorithm typically uses a generational replacement technique. This technique replaces the entire population after enough children are generated. However, the potential drawback of such a replacement approach is that many good *chromosomes* also get replaced, making it difficult for the good traits to survive. To overcome this problem, EvoCluster adopts a steady-state reproduction approach [45] so that only two least fit *chromosomes* are replaced whenever two new children are generated after each reproduction.

C. Fitness Function

To evaluate the fitness of each *chromosome,* we used an objective interestingness measure based on that described in [47]–[52]. This measure has the advantage that it is able to handle the potential noise and inconsistency resulting from the clustering process. It is able to take into consideration both local and global information by distinguishing between feature values that are relevant and irrelevant in a particular cluster grouping. This makes EvoCluster very robust even in the presence of noisy data.

The fitness evaluation procedure is invoked after new *chromosomes* are formed. The fitness function accepts a *chromosome* as a parameter and its fitness is evaluated in two steps. In Step 1, it attempts to discover statistically significant association patterns in the cluster grouping encoded in the *chromosome*. To do so, a subset of records from different clusters encoded in a *chromosome* is selected randomly to form a training set for pattern discovery. In Step 2, those records not selected in Step 1 are reclassified into one of the clusters based on the discovered patterns in order to determine the reclassification accuracy of the *chromosome*. As discussed above, if a clustering algorithm is effective, the clusters that are discovered should contain hidden patterns that can be used to accurately reclassify the records in the testing data. And if this is the case, the reclassification accuracy measure is an indication of how good the quality of the cluster grouping is. For this reason, the reclassification accuracy is then taken to be the fitness of each *chromosome*. The details of the two-step algorithm are given below.

- *Discovery of statistically significant association patterns:* We begin by detecting the association between the values of an attribute E_j (i.e., the gene expression values under a specific experimental condition) of the records that belong to a particular cluster and the cluster label itself. To do so, we let obs_{pk} be the total number of records in the dataset that belong to cluster c_p and are characterized by the same attribute value e_{jk}, where $e_{1j} = \cdots = e_{ij} = e_{jk}$ and $l < N$. We also let $\exp_{pk} = obs_{p+}obs_{+k}/N'$ be the expected total under the assumption that being a member of c_p is independent of whether a record has the characteristic e_{jk} (where $obs_{p+} = \Sigma_{k=1}^{k_{val}} obs_{pk}$, and $obs_{+k} = \Sigma_{p=1}^{P} obs_{pk}$, and K_{val} is the total number of distinct attribute values of an attribute E_j, P is the total number of clusters discovered, and $N' = \Sigma_{p,k} obs_{pk} \le N$ due to possible missing values in the data). The statistical significance of the association can be evaluated using the following test statistic:

$$z_{pk} = \frac{obs_{pk} - \exp_{pk}}{\sqrt{\exp_{pk}}} \tag{2}$$

where the maximum likelihood estimate of its asymptotic variance v_{pk} is defined by

$$v_{pk} = \left(1 - \frac{obs_{p+}}{N'}\right)\left(1 - \frac{obs_{+k}}{N'}\right). \tag{3}$$

Then

$$d_{pk} = \frac{\frac{(obs_{pk} - \exp_{pk})}{\sqrt{\exp_{pk}}}}{\sqrt{v_{pk}}} = \frac{z_{pk}}{\sqrt{v_{pk}}}. \tag{4}$$

This statistics has an approximate standard normal distribution and the attribute value can then be selected based on a statistically significant cluster-dependency.

- *Classification and reclassification using a weight-of-evidence information measure:* Using the test statistics given in (4), we can determine if e_{jk} of E_j is associated with cluster c_p, say, at the 95% confidence level. If it is the case, then we can compute an uncertainty measure called the *weight of evidence measure* W for a record characterized by e_{jk} to belong to c_p against its belonging to other clusters

$$W\left(\text{Cluster} = c_p/\text{Cluster} \neq c_p | e_{jk}\right) \tag{5}$$

where W is defined in terms of the mutual information $I(c_p : e_{jk})$ as follows:

$$W(\text{Cluster}=c_p/\text{Cluster}\neq c_p|e_{jk}) = I(c_p : e_{jk}) - I(\neq c_p : e_{jk}) \tag{6}$$

where

$$I(c_p : e_{jk}) = \log \frac{P(c_p|e_{jk})}{P(c_p)}. \tag{7}$$

The *weight of evidence* measures the amount of positive or negative evidence that is provided by e_{jk} supporting or refuting the labeling of the record as c_p. Given a collection of the selected attribute values $(M' \leq M)$, the weight of evidence from all observed attribute values is defined as a summation of the total weights

$$\begin{aligned} W&\left(\text{Cluster} = c_p/\text{Cluster} \neq c_p | e_{1k} \ldots e_{jk} \ldots e_{M'k}\right) \\ &= \sum_{j=1}^{M'} W\left(\text{Cluster} = c_p/\text{Cluster} \neq c_p | e_{jk}\right). \end{aligned} \tag{8}$$

Therefore, the cluster label c_p is inferred if W is maximized. Based on this, the predicted label can be compared with the original label of each record encoded in the *chromosome* to determine the reclassification accuracy and based on it, the fitness value of the cluster grouping encoded in a *chromosome* can be determined.

IV. Experimental Results

To evaluate the effectiveness of EvoCluster, we have tested it using both simulated and real data. In this section, we describe the experiments we carried out and present the results of these experiments.

A. Experimental Data

For performance evaluation, we used a set of simulated data consisting of 300 records each characterized by 50 different attributes that takes on values from [0.0, 1.0]. Initially, all these records were sampled from a uniform distribution and they were preclassified into one of three clusters so that each cluster contains 100 records. To embed hidden patterns in the data, 10% of the attributes in each cluster were randomly selected. For each selected attribute, 40% of its values in that cluster were randomly generated from within a certain range $[L, U]$, where $0.0 \leq L \leq \text{U} \leq 1.0$ so that L was selected uniformly from [0.0, 1.0] first, and U was then also selected uniformly from $[L, 1.0]$.

In addition to the simulated data, to test the effectiveness of EvoCluster, we also used two different sets of real gene expression microarray data given in [53] and [54] respectively. *Dataset 1* (Spellman's data) contains about 800 cell cycle regulated genes measured under 77 different experimental conditions. According to [11], these cell cycle regulated genes could be partitioned into six to ten clusters. Following this, we tried in our experiments to partition the data set also into different clusters from six to ten. *Dataset 2* (Cho's data) contains 384 genes measured under 17 different experimental conditions. According to [12], the data could be partitioned into four to eight clusters. Following this again, we tried to partition *Dataset 2* also into different clusters from four to eight. In our experiments, the data values were normalized using the *zero-mean normalization method* [55], which is the most commonly used method for microarray normalization [56].

B. Validating Cluster Grouping

With the above datasets, how effective EvoCluster is at its tasks is evaluated objectively based on three objective measures: i) the Davies–Bouldin validity index (DBI) measure [57]; ii) the F-measure [58]; and iii) a predictive power measure of the discovered clusters.

The DBI measure is a function of the inter and intracluster distances. These distances are considered good indicators of the quality of a cluster grouping as a good grouping should be reflected by a relatively large intercluster distance and a relatively small intracluster distance. In fact, many optimization clustering algorithms are developed mainly to maximize intercluster and minimize intracluster distances. The DBI measure combines these two distances in a function to measure the average similarity between a cluster and its most similar one. Assume that a cluster grouping consisting of k clusters has been formed. Its DBI measure is then defined as follows:

$$\text{DBI} = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j}\left\{\frac{d_{\text{intra}}(i) + d_{\text{intra}}(j)}{d_{\text{inter}}(i,j)}\right\} \tag{9}$$

where $d_{\text{intra}}(j) = (\sum_{x=1}^{n_j} ||g_x - g_{jc}||)/n_j$, $d_{\text{inter}}(i,j) = ||\boldsymbol{g}_{ic} - \boldsymbol{g}_{jc}||$, k denotes the total number of clusters, d_{intra} and d_{inter} denote the centroid intracluster and intercluster distances, respectively, and n_j is the number of records in cluster j. The intracluster distance for a given cluster is therefore defined to be the average of all pairwise distances between records in cluster j

TABLE I
SUMMARY OF THE NUMBER OF EXPERIMENTS PERFORMED BY EVOCLUSTER, k-MEANS, AND SOM

	No. of reproductions/iterations	No. of runs	No. of trials
EvoCluster	5000	-	10
k-means	5000	100	10
SOM	5000	100	10

and its centroid $\boldsymbol{g}_{jc}$ and the intercluster distance between two clusters i and j is computed as the distance between their centroids $\boldsymbol{g}_{ic}$ and $\boldsymbol{g}_{jc}$. For the experiments described below, the Euclidean distance was chosen as the distance metric when computing the DBI measure, and a low value of DBI therefore indicates good cluster grouping.

The F-measure that is typically used for cluster evaluation combines the "precision" and "recall" ideas from information retrieval [59]. When the correct classification of the data is known, the F-measure is useful in the sense that it would provide objective information on the degree to which a clustering algorithm is able to recover the original clusters. According to the F-measure, the discovered set of records in cluster j can be considered as if they are retrieved by a certain query; and the known set of records in cluster i can be considered as if they are the desired records that can be retrieved by the same query. The F-measure of each cluster j can then be calculated as follows:

$$F(i,j) = \frac{2\mathrm{Re}call(i,j)\mathrm{Pr}ecision(i,j)}{\mathrm{Re}call(i,j) + \mathrm{Pr}ecision(i,j)} \tag{10}$$

where $\mathrm{Re}call(i,j) = count_{ij}/count_i$ and $\mathrm{Pr}ecision(i,j) = count_{ij}/count_j$ and $count_{ij}$ is the number of records with cluster label i in the discovered cluster j, count_i is the number of records with cluster label i, and count_j is the number of records in discovered cluster j. The F-measure values are in the interval [0, 1], and the larger its values, the better the clustering quality is.

The predictive power measure is actually a measure of classification accuracy. If the clusters discovered are valid and of good qualities, we should expect patterns to be discovered in them. If these patterns are used to classify some testing data, the classification accuracy can reflect how valid and how good the qualities of the discovered clusters are. In order to determine the classification accuracy, a set of training samples can be randomly selected from each cluster to construct a decision-tree classifier using C4.5 [60]. C4.5 is a greedy algorithm that recursively partitions a set of training samples by selecting attribute that yield a *maximum information gain* measure at each step in the tree-construction process. After a decision tree is built, it then makes use of a postpruning procedure to compensate for any overfitting of training samples. Based on the pruned tree, the cluster memberships of those records that were not selected for training are then predicted. The percentage of accurate predictions can then be determined as classification accuracy. This accuracy measure is also referred to as the predictive power measure. If a clustering algorithm is effective, the clusters that are discovered should contain hidden patterns that can be used to accurately predict the cluster membership of the testing data. And if this is the case, the predictive power of a cluster grouping should be high. Otherwise, if a clustering algorithm is ineffective, the clusters it discovers are not expected to contain too many hidden patterns and the grouping is more or less random. And if this is the case, the predictive power is expected to be low. Hence, the greater the predictive power, the more interesting a cluster grouping is and vice versa. In our experiments, the predictive power measure was computed based on a tenfold cross validation approach. For each fold, 90% of the data records in each cluster were randomly selected for training and the remaining 10% used for testing. After ten experiments corresponding to the ten folds of data were performed, the average predictive power of the discovered clusters was computed as the average classification accuracy over the ten experiments.

C. The Results

The effectiveness of EvoCluster has been compared with a number of different clustering algorithms using both simulated and real data. Since the Pearson correlation coefficient is more commonly used for gene expression data [4] and is known to be better than the Euclidean distance in dealing with noise, it was used as the distance function in these clustering algorithms.

In our experiments, we used the SOM clustering algorithm [6] and we adopted the default settings for all the parameters as described in [61] (i.e., we used the bubble neighborhood function, the initial learning weight $(Alpha\_i)$ was set to 0.1, the final learning weight $(Alpha\_f)$ was set to 0.005, the initial sigma $(Sigma\_i)$ was set to five, and the final sigma $(Sigma\_F)$ was set to 0.5, etc.). Since a given number of clusters (say, six) can represent multiple SOM geometries (e.g., $1 \times 6, 2 \times 3$, etc.), we also tried all these geometries [61] in order to obtain the best cluster grouping with SOM. For both SOM and the k-means algorithms, 5000 iterations were performed. Also, to ensure that the best results for SOM and the k-means algorithm were obtained, 100 runs were performed for each of them with each run using different randomly generated initial cluster centroids. Only the best result from among these 100 runs was recorded. In order for enough samples to be collected for statistical testing purposes, such 100-run test was repeated ten times. The ten best results obtained from each 100-run test were then used in subsequent statistical tests (i.e., in Table VII).

In the case of EvoCluster, we also performed ten trials in our experiments. For each such trial, we randomly generated different initial populations of size fixed at 50. Using a steady-state reproduction scheme, the evolutionary process was terminated either when the maximum *chromosome* fitness converged or when the maximum number of reproductions reached 5000. As summarized in Table I, therefore, the total number of iterations performed with k-means and SOM is 100 times more than the total number of reproductions carried out using EvoCluster. This was done to ensure that EvoCluster would not use any more

TABLE II
PARAMETER SETTINGS OF EVOCLUSTER USED IN SIMULATED DATA (P_c/P_m REPRESENT THE PROBABILITY OF GUIDED OR UNGUIDED CROSSOVER/MUTATION OPERATOR SELECTED)

	Pop. Size	MIN	MAX	P_c	P_m	L_g	U_g	L_r	U_r
Simulated data	50	3	3	0.5	0.5	0.2	0.8	0.2	0.8

computational resources (in terms of the number of trial-and-errors through iterations/reproductions and in terms of computational time) than other clustering algorithms it was being compared against.

During the evolutionary process, the probabilities of selection of a *gene* or a record label in a *gene*, used by the crossover and mutation operators, were randomly generated from within [0.2, 0.8] using a random number generator. In order to evaluate the fitness of each *chromosome,* the data were first discretized using the approach given in [62] so as to ensure minimum loss of information during the discretization process. Discretization, it should be noted, was preformed only for EvoCluster but not for the other techniques.

In order to evaluate the effectiveness of EvoCluster, in addition to the traditional clustering algorithms, we also compared its performance with a clustering algorithm that represents one of the most successful attempts to use EA in clustering [36]. Each *gene* in it encodes one dimension of a cluster center and a *chromosome* encodes a fixed number of clusters. For our experiment with it, we set the crossover rate to 0.8 and the mutation rate to 0.001, i.e., the same as that used in [36]. Other parameter settings, including population size, number of reproductions, etc., were set exactly the same as that with EvoCluster.

Since one of the desirable features of EvoCluster is its ability to distinguish relevant from irrelevant feature values during the evolutionary process, we also compared its performance against various "hybrid" clustering algorithms that use a feature selection technique in combination with a clustering algorithm. Specifically, we used an effective feature selection technique together with the k-means algorithm, SOM, the hierarchical clustering algorithm, and the EA-based algorithm [36] to see how much improvement the performance of these algorithms can have when features were first filtered for clustering.

Among different feature selection techniques that can be used for this purpose [63]–[70], we chose to consider the one described in [70]. This is because this technique, which makes use of the t-statistic measure, has previously been used to reduce the number of attributes in gene expression data [66], [70]. Given an initial cluster grouping, the feature selection was performed in several steps as follows.

i) A cluster grouping is first determined using say, the k-means algorithm (or SOM, or the hierarchical clustering algorithm, the EA-based algorithm [36], or whatever other clustering algorithm) that the feature filtering method is hybridizing with.
ii) Given the initial cluster grouping, a t-statistic measure is then computed for each attribute to determine how well it is able to distinguish one cluster from the rest of the others.
iii) Based on the t-statistic, a new subset of attributes with the largest t-statistic values is obtained by first selecting 10% of the attributes that has the largest t-statistic values. With this new attribute subset, a classifier is then generated using C4.5 [60] and its classification accuracy is measured using ten-fold cross-validation. Afterward, the process of adding another 5% of the attributes with the largest t-statistic values to this new attribute subset and measuring the accuracy of the resulting new classifier was repeated. The final attribute subset is determined when the performance of the classifier converge [56].
iv) With this final attribute subset, a new and improved cluster grouping is then determined.

1) Simulated Data: Since the number of clusters ($k = 3$) to discover was known in advance for the simulated data, the length of the *chromosome* was fixed in our experiment to be three. Table II shows the parameter settings of EvoCluster used in the simulated data.

As discussed in the previous section, EvoCluster has a set of "guided" and "unguided" operators. For the "guided" operators, the exchange of grouping information is not totally random in the sense that the grouping information of the "best formed" clusters is preserved during the evolutionary process. For the "unguided" operators, the exchange of the grouping information between clusters takes place randomly. To determine if there is a real need for both types of operators, three separate experiments were carried out. In the first experiments, a 50/50 mixture of "guided" or "unguided" operators were used whereas in the second and third, only "guided" operators and only "unguided" operators were used respectively. The average number of reproductions performed by each algorithm until convergence, the average predictive power measures, DBI measures, and F-measure are given in Table III.

As shown in Table III and as expected, when only "guided" operators were used alone, it appeared that the results converged only to some local optima and when only "unguided" operators were used alone, not only a longer evolutionary process was required, the results obtained were unsatisfactory. The performance of EvoCluster is at its best when both "guided" and "unguided" operators were used together even though it required more reproductions to converge. Based on these results, we conclude that both the "guided" and "unguided" operators have a role to play in the evolutionary process. When they are used together, they can facilitate the exchange of grouping information in a way that such information in the "best formed" clusters is preserved as much as possible during the evolutionary process on one hand but variations can be introduced at the same time on the other so as to avoid trapping at local optima too early. The performance of EvoCluster has been compared with the other clustering algorithms and the results are given in Tables IV–VI.

TABLE III
COMPARING THE CLUSTERING PERFORMANCE USING "GUIDED + UNGUIDED" OPERATORS, "GUIDED" OPERATORS, OR "UNGUIDED" OPERATORS (SIMULATED DATA)

	k	No. of reproduction (converged)	Predictive power	DBI	F-Measure
Unguided + Guided operators	3	3830	87.60%	1.84	0.91
Unguided operators	3	4561	61.18%	1.92	0.63
Guided operators	3	3050	76.93%	1.86	0.80

TABLE IV
COMPARING THE MEAN OF AVERAGE PREDICTIVE POWER USING SIMULATED DATA (TEN TRIALS) (A REPRESENTS THE FINAL NUMBER OF ATTRIBUTES USED AND FS REPRESENTS FEATURE SELECTION)

	EvoCluster (A:50)	EA [36] (A:50)	EA [36] + FS (A:11)	*k*-means (A:50)	*k*-means + FS (A:14)	SOM (A:50)	SOM + FS (A:14)	Hierarchical (A:50)	Hierarchical + FS (A:17)
k=3	87.60%	65.49%	75.23	58.73%	72.85%	55.61%	69.70%	44.37%	57.64%

TABLE V
COMPARING THE MEAN OF DBI USING SIMULATED DATA (TEN TRIALS)

	EvoCluster (A:50)	EA [36] (A:50)	EA [36] + FS (A:11)	*k*-means (A:50)	*k*-means + FS (A:14)	SOM (A:50)	SOM + FS (A:14)	Hierarchical (A:50)	Hierarchical + FS (A:17)
k=3	1.84	1.91	1.87	1.94	1.89	1.95	1.92	1.96	1.93

TABLE VI
COMPARING THE MEAN OF F-MEASURE USING SIMULATED DATA (TEN TRIALS)

	EvoCluster (A:50)	EA [36] (A:50)	EA [36] + FS (A:11)	*k*-means (A:50)	*k*-means + FS (A:14)	SOM (A:50)	SOM + FS (A:14)	Hierarchical (A:50)	Hierarchical + FS (A:17)
k=3	0.91	0.66	0.78	0.61	0.75	0.57	0.72	0.52	0.65

As shown in the above tables, compared with other clustering algorithms, EvoCluster performs better in terms of all measures. Moreover, it seems that none of the EA-based, k-means, SOM, and hierarchical algorithms (with or without feature selection) is particularly effective when handling very noisy data such as the simulated data. In order to decide if the differences between these clustering algorithms are significantly different, we performed one-sided pairwise t-test [71] on the null and alternative hypotheses of $H_0 : \mu_1 > \mu_2$ and $H_\mathrm{a} : \mu_1 \leq \mu_2$, respectively, for the case of the predictive power and the F-measure and on the null and alternative hypotheses of $H_0 : \mu_1 < \mu_2$ and $H_\mathrm{a} : \mu_1 \geq \mu_2$, respectively, for the case of the DBI measure. The results of the t-tests confirm that the differences are all statistically significant at the 95% confidence level (in Table VII).

This shows that EvoCluster is very robust in the presence of a very noisy environment. Being able to effectively discover hidden patterns, it should be noted that EvoCluster has the additional advantage of being able to provide a "justification" of the cluster grouping it discovers. EvoCluster can make explicit the hidden patterns in a set of rules characterizing each cluster. Examples of such rules are given in Table VIII. The hidden patterns are expressed in rules of the form "If $EX = [L, U]$, then CY [0.95]" where it should be understood as "If the gene expression value of a gene under experimental condition X is within the interval from L to U, then there is a probability of 0.95 that it belongs to cluster Y." As described above, the simulated data was generated by randomly selecting 10% of the attributes in each cluster and randomly making 40% of its values in that cluster to fall within a certain range $[L, U]$, where $0.0 \leq L \leq U \leq 1.0$. The rules EvoCluster discovered are therefore consistent with the way the patterns were generated.

2) Gene Expression Data:

a) Statistical analysis: In the following, the performance of EvoCluster is evaluated using real expression data. The minimum and maximum number of clusters considered for both *Datasets 1* and *2* were set at (MIN= 6, MAX= 10) and (MIN= 4, MAX= 8), respectively. Table IX shows the parameter settings of EvoCluster used in *Datasets 1* and *2*.

As with the simulated data, the experiments with *Datasets 1* and *2* were repeated three times with a mixture of guided and unguided operators, unguided operators alone, and guided operators alone, respectively. Based on the results shown in Tables X and XI, we found that using both "guided" and "unguided" operators together, once again, gave us the best clustering results. The performance of EvoCluster in comparison with other algorithms is given in Tables XII–XV. The F-measure was not computed for the experiments with real datasets as the "original" correct clustering results are not known. EvoCluster again performs better than others even with the combination of the feature selection method. It is worth noting that the performance of EvoCluster and most other clustering algorithms is at their best when k equals to six for *Dataset 1* and when k equals to five for

TABLE VII
RESULTS OF t-TEST (TEN TRIALS) (DEGRESS OF FREEDOM: $N1 + N2 - 2$) (SIMULATED DATA)

Test #	Measure	Null Hypothesis (H_0: $\mu_1 > \mu_2$)	t-test statistics	Accept/Reject
1	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]}}$	+20.38	Accept
2	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means}}$	+29.17	Accept
3	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM}}$	+38.29	Accept
4	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical}}$	+195.33	Accept
5	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]+FS}}$	+11.04	Accept
6	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means+FS}}$	+17.53	Accept
7	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM+FS}}$	+19.68	Accept
8	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical+FS}}$	+42.61	Accept
Test #	Measure	Null Hypothesis (H_0: $\mu_1 < \mu_2$)	t-test statistics	Accept/Reject
1	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]}}$	-16.81	Accept
2	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means}}$	-21.48	Accept
3	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM}}$	-26.55	Accept
4	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical}}$	-83.48	Accept
5	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]+FS}}$	-10.79	Accept
6	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means+FS}}$	-15.16	Accept
7	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM+FS}}$	-21.74	Accept
8	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical+FS}}$	-31.02	Accept
Test #	Measure	Null Hypothesis (H_0: $\mu_1 > \mu_2$)	t-test statistics	Accept/Reject
1	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]}}$	+25.23	Accept
2	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means}}$	+34.17	Accept
3	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM}}$	+42.54	Accept
4	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical}}$	+76.22	Accept
5	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]+FS}}$	+19.45	Accept
6	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means+FS}}$	+22.36	Accept
7	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM+FS}}$	+25.08	Accept
8	F-Measure	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical+FS}}$	+33.95	Accept

TABLE VIII
ASSOCIATION RULES DISCOVERED IN THE SIMULATED DATA

Rules Discovered for C0	Rules Discovered for C1	Rules Discovered for C2
If A3 = [0.011, 0.257] then C0 [0.98]	If A7 = [0.003, 0.252] then C1 [1.0]	If A5 = [0.004, 0.252] then C2 [1.0]
If A15 = [0.251, 0.5] then C0 [1.0]	If A12 = [0.253, 0.501] then C1 [1.0]	If A18 = [0.253, 0.501] then C2 [1.0]
If A31 = [0.498, 0.747] then C0 [1.0]	If A26 = [0.499, 0.749] then C1 [1.0]	If A22 = [0.501, 0.75] then C2 [1.0]
If A40 = [0.75, 0.999] then C0 [1.0]	If A34 = [0.748, 0.996] then C1 [1.0]	If A28 = [0.742, 0.989] then C2 [0.98]
If A46 = [0.001, 0.247] then C0 [1.0]	If A39 = [0.006, 0.253] then C1 [1.0]	If A36 = [0.015, 0.261] then C2 [0.96]

TABLE IX
PARAMETER SETTINGS OF EVOCLUSTER IN GENE EXPRESSION DATA

	Pop. Size	MIN	MAX	P_c	P_m	L_g	U_g	L_r	U_r
Dataset 1	50	6	10	0.5	0.17	0.2	0.8	0.2	0.8
Dataset 2	50	4	8	0.5	0.17	0.2	0.8	0.2	0.8

Dataset 2. This seems to indicate that cluster quality may deteriorate if k is not set properly.

To confirm that these differences are also statistically significant, we performed some statistical tests (as the sample size is large enough, the z-test is used rather than the t-test and they are carried out at the 95% confidence level) [71]. The results of these tests are shown in Tables XVI and XVII.

b) Biological interpretation: Based on the clustering results obtained by EvoCluster, we are able to discover some very interesting patterns that may have great biological significance. For example, for *Dataset* 1, when $k = 6$ (which gives the best results), and for *Dataset* 2, when $k = 5$ (which gives the best results), we discovered the rules, as shown in Tables XVIII and XIX, respectively.

These rules can be interpreted as follows. In Table XVIII, for the discovered rule "If alpha21 $= [0.45, 2.12]$, then C0 [0.86]," it states that if the gene expression value of a gene, under experimental condition alpha21, is within the interval from 0.45 to 2.12, then there is a probability of 0.86 that it belongs to cluster 0. In Table XIX, the rule "If Cond3 $= [-2.874, -1.413]$ then C2 [0.94]" means that if the gene expression value of a gene, under experimental condition Cond3, is within the interval from -2.874 to -1.413, then there is a probability of 0.94 that it belongs to cluster 2.

TABLE X
COMPARING THE CLUSTERING PERFORMANCE USING "GUIDED + UNGUIDED" OPERATORS, "GUIDED" OPERATORS, OR "UNGUIDED" OPERATORS (DATASET 1)

	k	No. of reproduction (converged)	Predictive power	DBI
Unguided + Guided operators	6	3325	84.88%	1.59
	7	3701	81.25%	1.64
	8	4088	78.19%	1.67
	9	4717	74.26%	1.67
	10	4779	73.26%	1.68
	Average	4122	78.37%	1.65
Unguided operators	6	4493	67.28%	1.69
	7	4989	65.36%	1.69
	8	4777	57.19%	1.79
	9	4815	58.34%	1.79
	10	4622	59.71%	1.78
	Average	4739	61.58%	1.75
Guided operators	6	2089	73.23%	1.64
	7	3385	71.02%	1.65
	8	2876	67.49%	1.68
	9	4002	66.33%	1.68
	10	4747	68.51%	1.70
	Average	3420	69.32%	1.67

TABLE XI
COMPARING THE CLUSTERING PERFORMANCE USING "GUIDED + UNGUIDED" OPERATORS, "GUIDED" OPERATORS, OR "UNGUIDED" OPERATORS (DATASET 2)

	k	No. of reproduction (converged)	Predictive power	DBI
Unguided + Guided operators	4	3104	87.10%	1.52
	5	3631	89.86%	1.48
	6	4703	80.90%	1.57
	7	4464	83.42%	1.58
	8	4405	77.33%	1.59
	Average	4061	83.72%	1.55
Unguided operators	4	4852	67.68%	1.62
	5	4197	70.41%	1.61
	6	4763	62.33%	1.65
	7	4945	64.75%	1.64
	8	4572	62.59%	1.65
	Average	4666	65.55%	1.63
Guided operators	4	2670	76.13%	1.57
	5	1978	78.47%	1.57
	6	2404	75.94%	1.59
	7	4204	72.16%	1.60
	8	3978	70.08%	1.61
	Average	3047	74.56%	1.59

TABLE XII
COMPARING THE MEAN OF AVERAGE PREDICTIVE POWER USING SPELLMAN'S GENE EXPRESSION DATA (DATASET 1) (TEN TRIALS) (AVG REPRESENTS AVERAGE)

	EvoCluster (A:77)	EA [36] (A:77)	EA [36] + FS (A:16)	k-means (A:77)	k-means + FS (A:16)	SOM (A:77)	SOM + FS (A:20)	Hierarchical (A:77)	Hierarchical + FS (A:20)
k=6	84.88%	72.26%	78.43%	67.23%	75.86%	62.88%	69.14%	65.30%	71.75%
k=7	81.25%	61.37%	69.55%	63.19%	72.37%	63.75%	70.04%	59.86%	66.32%
k=8	78.19%	67.12%	72.38%	61.40%	68.88%	56.21%	66.56%	63.11%	68.29%
k=9	74.26%	60.51%	65.43%	64.77%	70.12%	56.87%	68.38%	62.80%	67.74%
k=10	73.26%	64.86%	70.24%	62.59%	71.11%	48.38%	59.48%	58.21%	68.86%
Avg	78.37%	65.22%	71.21%	63.84%	71.67%	56.72%	66.72%	61.85%	68.59%

The discovery of these expression patterns is of biological significance in several ways.

1) Based on the discovered rules, we found that genes within a cluster share similar expression patterns. For example, for *Dataset 1,* genes in Cluster 1, expressed very similarly to each other, under the conditions of cdc15_170 and alpha35 and genes in Cluster 2, expressed very similarly to each other, under the conditions of elu360 and cdc15_30, etc. This kind of cluster-specific expression patterns discovered by EvoCluster is new and has not

TABLE XIII
COMPARING THE MEAN OF DBI USING SPELLMAN'S GENE EXPRESSION DATA (DATASET 1) (TEN TRIALS)

	EvoCluster (A:77)	EA [36] (A:77)	EA [36] + FS (A:16)	*k*-means (A:77)	*k*-means + FS (A:16)	SOM (A:77)	SOM + FS (A:20)	Hierarchical (A:77)	Hierarchical + FS (A:20)
k=6	1.59	1.65	1.62	1.67	1.63	1.74	1.69	1.71	1.70
k=7	1.64	1.70	1.69	1.70	1.68	1.77	1.72	1.76	1.74
k=8	1.67	1.69	1.68	1.71	1.70	1.78	1.75	1.73	1.72
k=9	1.67	1.72	1.69	1.70	1.68	1.79	1.75	1.76	1.75
k=10	1.68	1.73	1.70	1.72	1.70	1.85	1.79	1.79	1.76
Avg	1.65	1.69	1.67	1.70	1.68	1.79	1.74	1.75	1.73

TABLE XIV
COMPARING THE MEAN OF AVERAGE PREDICTIVE POWER USING CHO'S GENE EXPRESSION DATA (DATASET 2) (TEN TRIALS)

	EvoCluster (A:17)	EA [36] (A:17)	EA [36] + FS (A:6)	*k*-means (A:17)	*k*-means + FS (A:5)	SOM (A:17)	SOM + FS (A:7)	Hierarchical (A:17)	Hierarchical + FS (A:8)
k=4	87.10%	70.23%	77.41%	76.43%	80.02%	72.38%	78.21%	74.39%	75.37%
k=5	89.86%	73.92%	80.16%	70.68%	77.48%	68.53%	75.51%	75.22%	76.82%
k=6	80.90%	68.34%	72.77%	66.98%	75.92%	64.26%	69.23%	73.23%	75.16%
k=7	83.42%	69.12%	75.53%	64.77%	72.37%	63.84%	70.06%	64.66%	66.79%
k=8	77.33%	60.02%	67.58%	63.12%	69.55%	61.39%	70.84%	62.30%	64.23%
Avg	83.72%	68.33%	74.69%	68.40%	75.07%	66.08%	72.77%	69.96%	71.67%

TABLE XV
COMPARING THE MEAN OF DBI USING CHO'S GENE EXPRESSION DATA (DATASET 2) (TEN TRIALS)

	EvoCluster (A:17)	EA [36] (A:17)	EA [36] + FS (A:6)	*k*-means (A:17)	*k*-means + FS (A:5)	SOM (A:17)	SOM + FS (A:7)	Hierarchical (A:17)	Hierarchical + FS (A:8)
k=4	1.52	1.61	1.57	1.59	1.56	1.62	1.58	1.61	1.60
k=5	1.48	1.54	1.51	1.56	1.52	1.64	1.55	1.59	1.56
k=6	1.57	1.62	1.60	1.62	1.60	1.62	1.59	1.58	1.57
k=7	1.58	1.61	1.60	1.66	1.61	1.70	1.67	1.68	1.61
k=8	1.59	1.69	1.65	1.68	1.64	1.79	1.72	1.69	1.67
Avg	1.55	1.61	1.58	1.62	1.59	1.67	1.62	1.63	1.60

TABLE XVI
RESULTS OF z-TEST (TEN TRIALS) (DATASET 1—OVER ALL kS)

Test #	Measure	Null Hypothesis (H_0: $\mu_1 > \mu_2$)	z-test statistics	Accept/Reject
1	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]}}$	+14.87	Accept
2	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means}}$	+16.13	Accept
3	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM}}$	+23.48	Accept
4	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical}}$	+21.32	Accept
5	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]+FS}}$	+8.66	Accept
6	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means+FS}}$	+8.53	Accept
7	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM+FS}}$	+13.36	Accept
8	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical+FS}}$	+14.55	Accept
Test #	Measure	Null Hypothesis (H_0: $\mu_1 < \mu_2$)	z-test statistics	Accept/Reject
1	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]}}$	-11.58	Accept
2	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means}}$	-12.26	Accept
3	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM}}$	-18.75	Accept
4	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical}}$	-17.16	Accept
5	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]+FS}}$	-8.58	Accept
6	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means+FS}}$	-9.32	Accept
7	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM+FS}}$	-12.17	Accept
8	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical+FS}}$	-11.84	Accept

previously been discovered before in microarray data analysis by the existing clustering algorithms [4]–[6], [20]–[22].

2) The patterns discovered in each cluster can lead to the discovery of functionally similar genes. For example, by closely examining the results with *Dataset 1,* we found that genes such as YBL023C, YEL032W, YLR103C, YLR274W, YBR202W, and YPR019W, etc., which are directly involved in DNA replication [72], [73], satisfied the rule "If alpha42 $= [-2.24, -0.21]$ then C3 [0.88]." We also found that many genes that are involved in mitosis, such as YPR119W, YGR108W, YDR146C, YGR109C, and YML027W [73] satisfied the rules "If elu270 $= [0.32, 1.32]$ then C5 [0.9]."

3) Since it is well known that functionally similar genes have similar expression patterns [4], biologists can make

TABLE XVII
RESULTS OF z-TEST (TEN TRIALS) (DATASET 2—OVER ALL kS)

Test #	Measure	Null Hypothesis (H_0: $\mu_1 > \mu_2$)	z-test statistics	Accept/Reject
1	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]}}$	+13.12	Accept
2	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means}}$	+13.04	Accept
3	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM}}$	+16.18	Accept
4	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical}}$	+13.67	Accept
5	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{EA [36]+FS}}$	+10.72	Accept
6	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{k\text{-means+FS}}$	+9.88	Accept
7	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{SOM+FS}}$	+13.37	Accept
8	Predictive Power	$\mu_{\text{EvoCluster}} > \mu_{\text{Hierarchical+FS}}$	+11.94	Accept
Test #	Measure	Null Hypothesis (H_0: $\mu_1 < \mu_2$)	z-test statistics	Accept/Reject
1	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]}}$	-7.69	Accept
2	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means}}$	-8.25	Accept
3	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM}}$	-12.72	Accept
4	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical}}$	-8.77	Accept
5	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{EA [36]+FS}}$	-6.28	Accept
6	DBI	$\mu_{\text{EvoCluster}} < \mu_{k\text{-means+FS}}$	-6.87	Accept
7	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{SOM+FS}}$	-9.93	Accept
8	DBI	$\mu_{\text{EvoCluster}} < \mu_{\text{Hierarchical+FS}}$	-7.11	Accept

TABLE XVIII
RULES DISCOVERED FOR SPELLMAN'S GENE EXPRESSION DATA (DATASET 1)

If alpha21 = [0.45, 2.12] then C0 [0.86]	If cdc28_20 = [0.56, 3.35] then C4 [0.85]
If elu270 = [0.32, 1.32] then C5 [0.9]	If cdc15_120 = [-2.5, -0.42] then C3 [0.92]
If elu360 = [-1.27, -0.29] then C2 [0.92]	If alpha42 = [-2.24, -0.21] then C3 [0.88]
If cdc28_150 = [-1.83, -0.32] then C4 [0.9]	If alpha35 = [0.04, 0.24] then C1 [0.86]
If alpha49 = [-2.17, -0.33] then C5 [0.84]	If cdc28_50 = [0.56, 3.35] then C0 [0.88]
If cdc15_170 = [0.43, 1.96] then C1 [0.83]	If cdc15_30 = [0.53, 2.54] then C2 [0.84]

TABLE XIX
RULES DISCOVERED FOR CHO'S GENE EXPRESSION DATA (DATASET 2)

If Cond4 = [-2.564, -1.275] then C0 [0.96]	If Cond11 = [-1.53, -0.267] then C3 [0.9]
If Cond8 = [-0.284, 1.01] then C4 [0.9]	If Cond10 = [1.481, 2.974] then C0 [0.92]
If Cond12 = [0.872, 1.953] then C1 [0.9]	If Cond17 = [-0.23, 0.861] then C2 [0.86]
If Cond11 = [0.995, 2.258] then C1 [0.86]	If Cond1 = [-0.139, 1.247] then C4 [0.88]
If Cond3 = [-2.847, -1.413] then C2 [0.94]	If Cond3 = [0.021, 1.454] then C3 [0.92]

TABLE XX
KNOWN TRANSCRIPTION FACTOR BINDING SITES IN BIOLOGICAL CELL CYCLE

Binding Site Name	Pattern
MCB	ACGCGT
SCB	CRMSAAA = C(A/G)(A/C)(C/G)AAA
Mcm1	DCCYWWNNRG = (A/G/T)CC(C/T)(A/T)(A/T)(A/G/C/T)(A/G/C/T)(A/G)G
Swi5;Ace2	RRCCAGCR = (A/G)(A/G)CCAGC(A/G)
SFF	GTMAACAW = GT(A/C)AACA(A/T)
Met31;Met32	AAAACTGTGG
Met4/Met28/Cbf1	TCACGTGA

TABLE XXI
KNOWN TRANSCRIPTION FACTOR BINDING SITES REVEALED FROM THE CLUSTERS IN SPELLMAN'S GENE EXPRESSION DATA (DATASET 1)

Cluster	Sequence revealed	Binding Site Name
C0	ACGCGT	MCB
C1	CGCGAA	SCB
C2	CCAGCA	Swi5;Ace2
C3	CCCAAA	Mcm1
C4	CTGTGG	Met31;Met32
C5	AAACAA	SFF

use of the expression patterns discovered in each cluster to classify other newly found genes of the same organism in order to infer their potential biological functions [74].

4) In addition to the possible identification of functionally related genes, the discovered patterns are expected to help biologists better understanding their expression data. For example, they can help biologists better planning and de-

TABLE XXII
KNOWN TRANSCRIPTION FACTOR BINDING SITES REVEALED FROM THE CLUSTERS IN CHO'S GENE EXPRESSION DATA (DATASET 2)

Cluster	Sequence revealed	Binding Site Name
C0	TAAACA	Mcm1
C1	CTGTCC (potential variant of CTGTGG)	Met31;Met32
C2	CCAGCA	Swi5;Ace2
C3	AAGAAA	SCB
C4	ACGCGT	MCB

TABLE XXIII
SIGNIFICANT POTENTIAL TRANSCRIPTION FACTOR BINDING SITES REVEALED FROM THE CLUSTERS IN SPELLMAN'S GENE EXPRESSION DATA (DATASET 1)

Cluster	Sequence revealed	Cluster	Sequence revealed
C0	AACTCG	C3	GGTCAA
	ACGCGA		TTGGGT
	GCGTTT		TAGGAA
C1	GACGCG	C4	GGCCCA
	TCATGG		TGGATG
	ATCGTC		TCCAAG
C2	GAGCCA	C5	GCTAGA
	TGGTTT		CCACAG
	GGCTGG		GTGTGC

TABLE XXIV
SIGNIFICANT POTENTIAL TRANSCRIPTION FACTOR BINDING SITES REVEALED FROM THE CLUSTERS IN CHO'S GENE EXPRESSION DATA (DATASET 2)

Cluster	Sequence revealed	Cluster	Sequence revealed
C0	GATGCC	C3	AGGAAA
	CTCGAC		AGACCA
	AGAAAC		CTCTAA
C1	TGGACA	C4	GTCGCG
	GGTGAT		CGCGTT
	TGTCCA		CGACGC
C2	CCAGCC		
	AGATCG		
	AGGTGA		

TABLE XXV
DISCOVERY OF KNOWN TRANSCRIPTION FACTOR BINDING SITES IN EACH CLUSTER BY DIFFERENT CLUSTERING ALGORITHMS—DATASET 1 (SITE NAME AND THE NUMBER OF OCCURRENCES)

	C0	C1	C2	C3	C4	C5
EvoCluster	MCB (139)	SCB (86)	Swi5;Ace2 (44)	Mcm1 (62)	Met31;Met32 (102)	SFF (207)
EA [36]	MCB (95)	-	-	Mcm1 (55)	-	SFF (116)
k-means	MCB (79)	-	-	Mcm1 (40)	-	SFF (134)
SOM	-	-	-	Mcm1 (36)	-	SFF (89)
Hierarchical	MCB (61)	SCB (31)	-	-	-	SFF (107)

signing their experiments by focusing on the transcriptional responses of genes in one cluster at a time and by reducing the number of experimental tests required [75].

5) The discovered patterns can also help biologists closely investigating into the relationships between the experimental conditions and the corresponding expression levels under which genes express so that the gene networks [76], which describe the gene–gene interactions, could be discovered and better modeled.

6) With the fact that the gene expression patterns discovered in each cluster are different, we attempted to see if there are any well-known binding sites in each discovered cluster. To do so, we looked at the corresponding promoter regions for the genes in each cluster by downloading the sequences from the Saccharomyces Genome Database [73], [77]. We used a popular motif-discovery algorithm described in [78] to try to search for transcription factor binding sites in the DNA sequences. A motif is a DNA sequence pattern that is widespread and has biological significance. Since the essential element of most binding sites found is in the form of six-character motif such as ATTCGT [78], we also tried to detect six-char-

TABLE XXVI
DISCOVERY OF KNOWN TRANSCRIPTION FACTOR BINDING SITES IN EACH CLUSTER BY DIFFERENT CLUSTERING ALGORITHMS—DATASET 2 (SITE NAME AND THE NUMBER OF OCCURRENCES)

	C0	C1	C2	C3	C4
EvoCluster	Mcm1 (64)	Met31;Met32 (99)	Swi5;Ace2 (32)	SCB (138)	MCB (101)
EA [36]	Mcm1 (30)	Met31;Met32 (58)	-	SCB (102)	MCB (73)
k-means	Mcm1 (23)	-	Swi5;Ace2 (21)	SCB (119)	MCB (85)
SOM	-	-	Swi5;Ace2 (25)	SCB (114)	MCB (65)
Hierarchical	Mcm1 (26)	-	-	SCB (96)	MCB (57)

TABLE XXVII
TOTAL NUMBER OF KNOWN AND POTENTIAL TRANSCRIPTION FACTOR BINDING SITES DISCOVERED IN EACH CLUSTER BY DIFFERENT CLUSTERING ALGORITHMS—DATASET 1

	C0	C1	C2	C3	C4	C5
EvoCluster	12	21	9	4	24	15
EA [36]	5	9	6	4	8	11
k-means	8	7	3	3	13	10
SOM	4	5	-	3	7	6
Hierarchical	5	12	-	-	4	6

TABLE XXVIII
TOTAL NUMBER OF KNOWN AND POTENTIAL TRANSCRIPTION FACTOR BINDING SITES DISCOVERED IN EACH CLUSTER BY DIFFERENT CLUSTERING ALGORITHMS—DATASET 2

	C0	C1	C2	C3	C4
EvoCluster	14	11	8	17	19
EA [36]	9	5	3	7	15
k-means	6	2	5	12	9
SOM	4	3	3	9	13
Hierarchical	4	-	2	3	10

acter patterns in the DNA sequences of each cluster. All discovered sites in each cluster were then checked against the well-known binding sites given in Table XX (Note: R is A or G; M is A or C; S is C or G; D is A, G or T; Y is C or T; W is A or T; N is any base). As shown in Tables XXI and XXII, we did discover the patterns that are well-known transcription factors binding sites for being involved in the cell cycle [11], [79]. Moreover, in addition to known binding sites, we were able to discover some other potentially important sites (Tables XXIII and XXIV). The validity of these sites can be confirmed by biologists using different biochemical methods [80].

Comparing against EvoCluster, other clustering algorithms are only able to discover some of the known binding sites in some of the clusters they discovered (Tables XXV and XXVI). This is an indication that the cluster groupings discovered by EvoCluster are more biologically meaningful and significant than the groupings discovered by others. The total numbers of confirmed and suspected binding sites discovered in the clusters found by the various clustering algorithms are also given in Tables XXVII and XXVIII for *Datasets 1* and *2,* respectively. In both data sets, EvoCluster is able to find many more such binding sites.

V. CONCLUSION

With the advent of microarray technology, we are now able to monitor simultaneously the expression levels of thousands of genes during important biological processes. Due to the large number of data collected everyday and due to the very noisy nature in the data collection process, interpreting and comprehending the experimental results has become a big challenge. To discover hidden patterns in gene expression microarray data, we presented a novel clustering algorithm, EvoCluster.

EvoCluster encodes an entire cluster grouping in a *chromosome* so that each *gene* encodes one cluster. Based on such a structure, it makes use of a set of reproduction operators to facilitate the exchange of grouping information between *chromosomes*. The fitness function it adopts is able to differentiate between how relevant a feature value is in determining a particular cluster grouping. As such, instead of just local pairwise distances, it also takes into consideration how clusters are arranged globally. EvoCluster does not require the number of clusters to be decided in advance. Patterns hidden in each cluster can be explicitly revealed and presented for easy interpretation. We have tested EvoCluster using both simulated and real data. Experimental results show that EvoCluster is very robust in the presence of noise. It is able to search for near optimal solutions effectively, and discover statistically significant association patterns/rules in the noisy data for meaningful groupings. The results also show that, under some common performance measures, our proposed method is better than other algorithms used in gene expression data analysis, and the discovered clusters by EvoCluster contain more biologically meaningful patterns. In particular, we could correlate the clusters of coexpressed genes discovered by EvoCluster to their DNA sequences, and found that we were able to uncover significant well-known and new biological binding sites in each cluster of sequences. Since DNA microarrays only measure mRNA levels rather than protein levels of genes, it should be noted that gene expression microarray data alone does not present researchers with a complete picture of the underlying gene expression process. However, although it may be incomplete, gene expression data are still worth exploring as they contains a significant amount of information pertaining to the actual protein levels.

Compared with other clustering algorithms such as k-means or SOM, EvoCluster is about 12–18 times slower than other clustering algorithms when the time it takes for a reproduction to be performed is compared against the time it takes for performing an iteration in these algorithm. However, as shown in the experiments above, if the same computational resources are given to these clustering algorithms. EvoCluster will likely be giving the best clustering results. For microarray analysis, since the results are normally not required immediately, the relatively longer evolutionary process that EvoCluster takes to find a better solution is not very important. In order to cope with very large gene expression data sets, the inherently parallel nature of the problem solving process of EvoCluster can be exploited. In addition, for future research, we intend also to investigate into the effect of varying the parameters used by EvoCluster has on its performance. We believe that both its efficiency and effectiveness can be used with an adaptive learning process.

Acknowledgment

The authors are grateful to the acting Editor-in-Chief of this paper, Dr. G. Fogel, and anonymous associate editor and referees for their constructive comments.

References

[1] J. H. Ward, "Hierarchical grouping to optimize an objective function," *J. Amer. Stat. Assoc.*, vol. 58, pp. 236–244, 1963.

[2] J. MacQueen, "Some methods for classification and analysis of multivariate observation," in *Proc. Symp. Math. Stat. Prob. Berkeley*, vol. 1, 1967, pp. 281–297.

[3] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, 1989.

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. Nat. Acad. Sci. USA*, vol. 95, no. 25, pp. 14 863–14 868, Dec. 1998.

[5] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nat. Genet.*, vol. 22, no. 3, pp. 281–285, Jul. 1999.

[6] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Nat. Acad. Sci. USA*, vol. 96, no. 6, pp. 2907–2912, Mar. 1999.

[7] D. L. Hartl and E. W. Jones, *Genetics: Analysis of Genes and Genomes*. Sudbury, MA: Jones & Bartlett, 2001.

[8] T. M. Cox and J. Sinclair, *Molecular Biology in Medicine*. Oxford, U.K.: Blackwell Science, 1997.

[9] P. M. Fernandes, T. Domitrovic, C. M. Kao, and E. Kurtenbach, "Genomic expression pattern in Saccharomyces cerevisiae cells in response to high hydrostatic pressure," *FEBS Lett.*, vol. 556, no. 1–3, pp. 153–160, Jan. 2004.

[10] J. Lapointe, C. Li, J. P. Higgins, M. Van De Rijn, E. Bair, K. Montgomery, M. Ferrari, L. Egevad, W. Rayford, U. Bergerheim, P. Ekman, A. M. DeMarzo, R. Tibshirani, D. Botstein, P. O. Brown, J. D. Brooks, and J. R. Pollack, "Gene expression profiling identifies clinically relevant subtypes of prostate cancer," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. 3, pp. 811–816, Jan. 2004.

[11] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Mol. Biol. Cell*, vol. 9, no. 12, pp. 3273–3297, Dec. 1998.

[12] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell*, vol. 2, no. 1, pp. 65–73, Jul. 1998.

[13] R. J. Cho, M. Huang, M. J. Campbell, H. Dong, L. Steinmetz, L. Sapinoso, G. Hampton, S. J. Elledge, R. W. Davis, and D. J. Lockhart, "Transcriptional regulation and function during the human cell cycle," *Nat. Genet.*, vol. 27, no. 1, pp. 48–54, Jan. 2001.

[14] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, pp. 680–686, Oct. 1997.

[15] D. A. Lashkari, J. L. DeRisi, J. H. McCusker, A. F. Namath, C. Gentile, S. Y. Hwang, P. O. Brown, and R. W. Davis, "Yeast microarrays for genome wide parallel genetic and gene expression analysis," *Proc. Nat. Acad. Sci. USA*, vol. 94, no. 24, pp. 13 057–13 062, Nov. 1997.

[16] J. Zheng, J. Wu, and Z. Sun, "An approach to identify over-represented CIS-elements in related sequences," *Nucleic Acids Res.*, vol. 31, no. 7, pp. 1995–2005, Apr. 2003.

[17] D. P. Berrar, W. Dubitzky, and M. Granzow, *A Practical Approach to Microarray Data Analysis*. Boston, MA: Kluwer Academic, 2003.

[18] G. Sherlock, "Analysis of large-scale gene expression data," *Curr. Opin. Immunol.*, vol. 21, pp. 201–205, 2000.

[19] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[20] J. Quackenbush, "Computational analysis of microarray data," *Nat. Rev. Genet.*, vol. 2, no. 6, pp. 418–427, Jun. 2001.

[21] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *J. Comp. Biol.*, vol. 6, no. 3–4, pp. 281–297, Fall-Winter 1999.

[22] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, no. 2, pp. 126–136, Feb. 2001.

[23] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative monitoring of gene expression patterns with a complementary DNA microarray," *Science*, vol. 270, no. 5235, pp. 467–470, 1995.

[24] D. J. Lockhart and E. A. Winzeler, "Genomic, gene expression and DNA arrays," *Nature*, vol. 405, no. 6788, pp. 827–836, 2000.

[25] A. Brazma, A. Robinson, G. Cameron, and M. Ashburner, "One-stop shop for microarray data," *Nature*, vol. 403, no. 6771, pp. 699–700, 2000.

[26] D. J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J. M. Trent, "Expression profiling using cDNA microarrays," *Nat. Genet.*, vol. 21, no. 1, pp. 10–14, Jan. 1999.

[27] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill, 1968.

[28] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 183–196, Apr. 2004.

[29] Z. Chi, X. Weimin, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 519–531, Dec. 2003.

[30] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 561–575, Dec. 2003.

[31] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 321–332, Aug. 2002.

[32] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms," *Pattern Recognit. Lett.*, vol. 17, no. 8, pp. 825–832, 1996.

[33] K. C. C. Chan and L. L. H. Chung, "Discovering clusters in databases containing mixed continuous and discrete-valued attributes," in *Proc. SPIE AeroSense'99 Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, 1999, pp. 22–31.

[34] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proc. 3rd Genetic Programming Annual Conf.*, San Francisco, CA, 1998, pp. 568–575.

[35] L. O. Hall, I. B. Ozyurt, and J. C. Bezdek, "Clustering with a genetically optimized approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 103–112, Jul. 1999.
[36] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, pp. 1455–1465, 2000.
[37] G. P. Babu and M. N. Murty, "Clustering with evolution strategies," *Pattern Recognit.*, vol. 27, no. 2, pp. 321–329, 1994.
[38] P. Franti, J. Kivijarvi, T. Kaukoranta, and O. Nevalainen, "Genetic algorithms for large-scale clustering problems," *Comput. J.*, vol. 40, no. 9, pp. 547–554, 1997.
[39] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no. 3, pp. 433–439, Jun. 1999.
[40] Y. Lu and J. Han, "Cancer classification using gene expression data," *Inf. Syst.*, vol. 28, no. 4, pp. 243–268, 2003.
[41] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley, 1989.
[42] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.
[43] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. Chichester, U.K.: Wiley, 1998.
[44] T. Baeck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, U.K.: Institute of Physics, 1997.
[45] ——, *Evolutionary Computation 1: Basic Algorithms and Operators*. Bristol, U.K.: Institute of Physics, 2000.
[46] ——, *Evolutionary Computation 2: Advanced Algorithms and Operators*. Bristol, U.K.: Institute of Physics, 2000.
[47] K. C. C. Chan and A. K. C. Wong, "A statistical technique for extracting classificatory knowledge from databases," in *Knowledge Discovery in Databases*, G. Piatesky-Shapiro and W. J. Frawley, Eds. Menlo Park, CA: AAAI/MIT Press, 1991, pp. 107–123.
[48] Y. Wang and A. K. C. Wong, "From association to classification: Inference using weight of evidence," *IEEE Trans. Knowl. Data Eng.*, vol. 15, pp. 764–767, May–Jun. 2003.
[49] W. H. Au, K. C. C. Chan, and X. Yao, "A novel evolutionary data mining algorithm with applications to churn modeling," *IEEE Trans. Evol. Comput. (Special Issue on Data Mining and Knowledge Discovery With Evolutionary Algorithms)*, vol. 7, pp. 532–545, Dec. 2003.
[50] W. H. Au and K. C. C. Chan, "Discovering fuzzy association rules in a bank database," *IEEE Trans. Fuzzy Syst. (Special Issue on Fuzzy Systems in Knowledge Discovery and Data Mining)*, vol. 11, pp. 238–248, Apr. 2003.
[51] K. C. C. Chan and W. H. Au, "Mining fuzzy association rules," in *Proc. ACM 6th Int. Conf. Inf. Knowl. Management*, Las Vegas, NV, 1997, pp. 209–215.
[52] K. C. C. Chan, A. K. C. Wong, and D. K. Y. Chiu, "Learning sequential patterns for probabilistic inductive prediction," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 10, pp. 1532–1547, Oct. 1994.
[53] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Mol. Biol. Cell.*, vol. 9, no. 12, pp. 3273–3297, Dec. 1998. [Online]. Available: http://genome-www.stanford.edu/cellcycle.
[54] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell.*, vol. 2, no. 1, pp. 65–73, Jul. 1998. [Online]. Available: http://genome-www.stanford.edu.
[55] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann, 2001.
[56] D. Stekel, *Microarray Bioinformatics*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[57] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 1, no. 2, pp. 224–227, 1979.
[58] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proc. KDD-99 Workshop*, San Diego, CA, 1999.
[59] G. Kowalski, *Information Retrieval Systems—Theory and Implementation*. Norwell, MA: Kluwer Academic, 1997.
[60] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1993.
[61] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Nat. Acad. Sci. USA*, vol. 96, no. 6, pp. 2907–2912, Mar. 1999. [Online]. Available: http://www.broad.mit.edu/cancer/software/genecluster2/gc2.html.
[62] D. K. Y. Chiu, B. Cheung, and A. K. C. Wong, "Information synthesis based on hierarchical maximum entropy discretization," *J. Exp. Theor. Artif. Intell.*, vol. 2, pp. 117–129, 1990.
[63] W. Li and Y. Yang, "How many genes are needed for a discriminant microarray data analysis," in *Critical Assessment Techniques Microarray Data Mining Workshop*, 2000, pp. 137–150.
[64] M. Xiong, X. Fang, and J. Zhao, "Biomarker identification by feature wrappers," *Genome Res.*, vol. 11, pp. 1878–1887, 2001.
[65] E. P. Xing and R. M. Karp, "CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts," *Bioinformatics*, vol. 17, no. 1, pp. S306–S315, 2001.
[66] J. Jaeger, R. Sengupta, and W. L. Ruzzo, "Improved gene selection for classification of microarrays," in *Pacific Symp. Biocomputing*, vol. 8, 2003, pp. 53–64.
[67] T. R. Golub, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, 1999.
[68] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Nat. Acad Sci. USA*, vol. 96, pp. 6745–6750, 1999.
[69] C. Ding, "Analysis of gene expression profiles: Class discovery and leaf ordering," in *Proc. 6th Int. Conf. Research Computational Molecular Biology (RECOMB)*, 2002, pp. 127–136.
[70] Y. Su, T. M. Murali, V. Pavlovic, M. Schaffer, and S. Kasif, "RankGene: Identification of diagnostic genes based on expression data," *Bioinformatics*, vol. 19, no. 12, pp. 1578–1579, Aug. 2003. [Online]. Available: http://genomics10.bu.edu/yangsu/rankgene/.
[71] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*. Boston, MA: Addison-Wesley, 2002.
[72] S. Chevalier and J. J. Blow, "Cell cycle control of replication initiation in eukaryotes," *Curr. Opin. Cell Biol.*, vol. 8, pp. 815–821, 1996.
[73] C. A. Ball, H. Jin, G. Sherlock, S. Weng, J. C. Matese, R. Andrada, G. Binkley, K. Dolinski, S. S. Dwight, M. A. Harris, L. Issel-Tarver, M. Schroeder, D. Botstein, and J. M. Cherry, "Saccharomyces genome database provides tools to survey gene expression and functional analysis data," *Nucleic Acids Res.*, vol. 29, no. 1, pp. 80–81, Jan. 2001.
[74] R. Matoba, K. Kato, C. Kurooka, C. Maruyama, Y. Sakakibara, and K. Matsubara, "Correlation between gene functions and developmental expression patterns in the mouse cerebellum," *Eur. J. Neurosci.*, vol. 12, no. 4, pp. 1357–1371, Apr. 2000.
[75] M. Schena, *DNA Microarrays: A Practical Approach*. Oxford, U.K.: Oxford Univ. Press, 1999.
[76] A. de la Fuente, P. Brazhnik, and P. Mendes, "Linking the genes: Inferring quantitative gene networks from microarray data," *Trends Genet.*, vol. 18, no. 8, pp. 395–398, Aug. 2002.
[77] K. Dolinski, R. Balakrishnan, K. R. Christie, and M. C. Costanzo. (2003) Saccharomyces Genome Database. [Online]. Available: ftp://ftp.yeastgenome.org/yeast/
[78] J. V. Helden, B. Andre, and J. Collado-Vides, "Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies," *J. Mol Biol.*, vol. 281, no. 5, pp. 827–842, Sep. 1998.
[79] J. V. Helden, A. F. Rios, and J. Collado-Vides, "Discovering regulatory elements in noncoding sequences by analysis of spaced dyads," *Nucleic Acids Res.*, vol. 28, no. 8, pp. 1808–1818, Apr. 2000.
[80] M. Suzuki, S. E. Brenner, M. Gerstein, and N. Yagi, "DNA recognition code of transcription factors," *Protein Eng.*, vol. 8, no. 4, pp. 319–328, Apr. 1995.

Patrick C. H. Ma received the B.A. degree in computer science from the Hong Kong Polytechnic University, Hong Kong, China. He is currently working towards the Ph.D. degree at the Department of Computing, Hong Kong Polytechnic University.

His research interests are in bioinformatics, data mining, and computational intelligence.

Keith C. C. Chan received the B.Math. degree in computer science and statistics and the M.A.Sc. and Ph.D. degrees in systems design engineering from the University of Waterloo, Waterloo, ON, Canada.

He was with IBM Canada Laboratory, Toronto, ON, where he was involved in the development of software engineering tools. In 1993, he joined the Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON, Canada, as an Associate Professor. In 1994, he joined the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China, where he is now Professor and Head. He is also a Guest Professor of the Graduate School and an Adjunct Professor of the Institute of Software, Chinese Academy of Sciences, Beijing. He has served as a Consultant to government agencies and various companies in Hong Kong, China, Singapore, Malaysia, and Canada. His research interests are in data mining, computational intelligence, bioinformatics, and software engineering.

Xin Yao (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, in 1982, the M.Sc. degree from North China Institute of Computing Technology, Beijing, in 1985, and the Ph.D. degree from USTC in 1990, all in computer science.

He worked in China and Australia before joining the University of Birmingham, Birmingham, U.K., in 1999, as a Professor of Computer Science. He is a Distinguished Visiting Professor at USTC, and a Visiting Professor at three other universities. He is the Director of The Centre of Excellence for Research in Computational Intelligence and Applications, which is focused on applied research and knowledge transfer to the industry. He is an Associate Editor or an Editorial Board Member of ten international journals. He is the Editor of the World Scientific Book Series on *Advances in Natural Computation*, and a Guest Editor of several journal special issues. He has been invited to present 40 keynote or plenary speeches at conferences worldwide. He has more than 200 research publications. His research interests include evolutionary computation, neural network ensembles, global optimization, data mining, computational time complexity of evolutionary algorithms, and real-world applications. In addition to basic research, he works closely with many industrial partners on various real-world problems.

Dr. Yao won the 2001 IEEE Donald G. Fink Prize Paper Award and several other best paper awards. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

David K. Y. Chiu received the M.Sc. degree in computing and information science from Queen's University, Kingston, ON, Canada, and the Ph.D. degree in systems design engineering from the University of Waterloo, Waterloo, ON.

Currently, he is a Full Professor in the Department of Computing and Information Science and Graduate Faculty of the Biophysics Interdepartmental Group, University of Guelph, Guelph, ON, Canada. He has done consulting work with NCR Canada, Ltd., and VIRTEC Vision, Intelligence, Robotics Technology Canada, Ltd., on unconstrained character recognition. He was a Visiting Researcher to Electrotechnical Laboratory (currently National Institute of Advanced Industrial Science and Technology). He has broad research interests in knowledge discovery and pattern analysis, and in particular, bioinformatics and comparative genomics. He has published more than 100 technical papers in these areas.

Prof. Chiu received the Science and Technology Agency Fellowship of Japan.