International Journal of Innovative Computing, Information and Control
Volume **10**, Number **3**, June **2014**

ICIC International ©2014 ISSN 1349-4198
pp. 1051-1069

DYNAMIC CORE BASED CLUSTERING OF GENE EXPRESSION DATA

Maria-Iuliana Bocicor, Adela Sîrbu and Gabriela Czibula

Faculty of Mathematics and Computer Science
Babeş-Bolyai University
1, M. Kogălniceanu Street, Cluj-Napoca 400084, Romania
{ iuliana; adela; gabis }@cs.ubbcluj.ro

Received April 2013; revised October 2013

Abstract. *Modern microarray technology allows measuring the expression levels of thousands of genes, under different environmental conditions and over time. Clustering is, often, a first step in the analysis of the huge amounts of biological data obtained from these microarray based experiments. As most biological processes are dynamic and biological experiments are conducted during longer periods of time, the data is continuously subject to change and researchers must either wait until the end of the experiments to have all the necessary information, or analyze the data gradually, as the experiment progresses. If the available data is clustered progressively, using clustering algorithms, as soon as new data emerges, the algorithm must be run from scratch, thus leading to delayed results. In this paper, we approach the problem of dynamic gene expression data sets and we propose a dynamic core based clustering algorithm, which can handle newly collected data, by starting from a previously obtained partition, without the need to rerun the algorithm from the beginning. The experimental evaluation is performed on a real-life gene expression data set and the algorithm has proven to perform well in terms of a series of evaluation measures.*

Keywords: Bioinformatics, Gene expression, Unsupervised learning, Dynamic clustering

1. Introduction.

Gene expression refers to the process by which the information from a gene is converted into functional gene products: proteins or RNA having different roles in the life of a cell. Modern microarray technology is nowadays used to experimentally detect the levels of expressions of thousand of genes, across different conditions and over time. Once the gene expression data has been gathered, the next step is to analyze it and extract useful biological information. Data mining is the field that offers the necessary methods to accomplish this task and one of the most used algorithms dealing with the analysis of gene expression data approaches the *clustering problem* [1].

Clustering involves partitioning a certain data set in groups. The components of each group resemble each other according to a certain similarity measure [2]. In the case of gene expression data sets, each gene is represented by its expression values (features), at distinct points in time, under the monitored conditions. The process of gene clustering is at the foundation of genomic studies that aim to analyze the functions of genes [3], because it is assumed that genes that are similar in their expression levels are also relatively similar in terms of biological function.

A great number of clustering algorithms have been introduced in the literature, most of which deal with a given static data set (that is not subject to change during the clustering process). There are also some incremental approaches, meaning that the clustering algorithm was designed to take into consideration new instances of data as well, as these are added to the existing data set. These approaches will be briefly described in Section 3.

In this paper we tackle the dynamic problem of clustering gene expression data. In our case, the term *dynamic* indicates that the data set is not static, but it is subject to change. Still, as opposed to the incremental approaches from the literature, where the data set is enriched with new genes (instances) during the clustering process, our approach tackles the cases when new features (expression levels for new points in time) are added to the genes already existing in the data set. To our knowledge, there is no approach in the literature that deals with the problem of dynamic clustering of gene expression data, defined as above.

The rest of the paper is structured as follows. Section 2 presents the problem tackled by this paper. A previously proposed incremental clustering algorithm, as well as different approaches existing in the literature for clustering of gene expression data are presented in Section 3. The dynamic core based clustering model that we propose is introduced in Section 4. Experimental evaluations of our algorithm are given in Section 5, while analysis and comparisons of the results with related work from the literature are presented in Section 6. Section 7 contains some conclusions of our paper and indicates future research directions.

2. **Problem Statement and Relevance.**

In this section we aim at presenting the problem of dynamic clustering of gene expression data and its relevance, as well as some practical applications of solutions to this problem.

2.1. **Problem definition and motivation.**

The emergence of microarray technology, that allows measuring the levels of expression of thousands of genes has led to an exponential increase in the amount of gene expression data. Still, all this data would be useless unless relevant biological information was extracted from it, therefore thorough exploratory analyses are usually required and performed. One of the most widely used techniques for these analyses and, most frequently, the first step, is clustering.

Clustering refers to creating a set of groups (clusters) and assigning each instance of a data set to one of these groups, according to a certain similarity measure. In what concerns gene clustering, the goal is twofold: firstly, by dividing the huge amount of gene expression data into clusters, this data becomes easier to process and analyze; secondly, but not less important, it is assumed that genes having similar expression patterns over time (during the experiments) are likely to have similar biological functions and therefore clustering can also be considered an initial stage in the process of determining gene functions.

Gene expression data is usually collected with the goal of investigating the progress of different biological processes, as they evolve under different conditions and over time. Since biological processes are dynamic and time varying, they are best described by time series gene expression data [4]. A time series data set is a collection of data resulted from a specific type of biological experiment: samples of cells or tissues are extracted from the same individual at different, known points in time, during the progression of the biological process. Thus, for each of the targeted genes, the level of expression is measured at several distinct points in time. The data set will then be composed of thousands of genes (instances), each gene being characterized by a set of features: its expression levels (which can be quantified as real numbers) at different points in time.

However, there are some processes worth studying that may take days, or even months (e.g., diseases that progress over time), as well as experiments that are conducted over longer durations of time. For such cases, researchers must either wait until the experiment finishes and the expression levels of the genes are available at all time points, or analyse the data gradually, as the experiment progresses. Clustering of gene expression data might be performed during the evolution of the experiment, at intermediate time steps, when genes would be characterized by only a subset of the entire set of features (time points). The main disadvantage is that as the experiments advance and new data becomes available (expression levels of the targeted genes for new points in time), the clustering process must once again start from scratch, which requires considerable time (especially as the number of the genes in such data sets is extremely high), in the end leading to a slower and more inefficient processing of the data.

To overcome this drawback, we propose a *dynamic clustering algorithm*, based on the idea of incremental clustering introduced in [5]. Given a previously obtained partition of a data set and new features for all the genes in this data set, the dynamic clustering algorithm is able to re-cluster the set of instances, without the need to start from the beginning, but by using the existing partition. This way, the clustering of genes at intermediate time points during the experiment can be more efficiently exploited and the final result could be achieved in smaller amounts of time.

2.2. **Practical applications.** Nowadays, in this postgenomic era, one of the greatest concerns of scientists is to deeply understand the functioning of organisms. One step towards this understanding is the analysis of genes and clustering offers one way to achieve it.

Among the most important practical applications of gene clustering is the identification of gene functions. Genes with similar expression profiles will be clustered together, therefore facilitating the prediction of the functions of unknown genes or the identification of gene sets that are regulated by the same mechanism [6]. The information offered by gene clustering is often used for genome annotation [7]. In practice, gene clustering is used for different applications. Among these, we mention the study of molecular mechanisms of plant-pathogen interaction to the goal of determining those genes in certain plants or vegetables that are resistant to different pathogens [8]. Further, another application is in the pharmaceutical industry, where biosynthetic gene clusters identified in microbial genomes could lead to the development of novel pharmaceuticals [9].

A significant application field of the results obtained through clustering of gene expression data is clinical research – patient classification and diagnosis, classification of different types of diseases (for instance, cancer) and development of gene expression-based tools for the personalized treatment and prevention. In the cancer research domain, tumor prediction and classification is a difficult problem and clustering could offer a manner of finding similar gene expression patterns, which could be used to discover new cancer subtypes [10].

Biological systems are inherently dynamic, therefore the gene information is extracted at different moments in time. Clustering can be used for analysis at any point and thus the conclusions extracted from the obtained partitions can be practically used as mentioned in the previous paragraph. The main contribution of our algorithm is that it can obtain partitions as new data is gathered, using the partitions obtained in a former phase of data collection, without having to re-apply the clustering algorithm from the beginning and without losing the quality of the partitions, but even improving it in several cases. The final goal of the method is to cluster genes. The clustering result can be used to analyze gene expression data and to apply this analysis for any of the practical purposes described above.

3. **Background.**

An incremental clustering technique, which serves as the starting point of our proposal is briefly presented in this section. Moreover, we shortly describe some of the existing clustering techniques that approach the problem of clustering gene expression data.

3.1. **Incremental clustering.**

Generally, existing clustering methods, such as the k-means algorithm [11], start with a known set of objects, measured against a known set of features and all features are considered simultaneously when calculating objects' similarity. However, there are numerous applications where the feature set characterizes the objects evolves, thus, re-clustering is required. Şerban and Câmpan introduce in [5] an incremental, k-means based clustering method, *Core Based Incremental Clustering (CBIC)*, that is capable to re-partition the objects set, when the features set increases. The method starts from the partitioning into clusters that was established by applying k-means before the feature set changed. This partitioning is adapted considering the newly added features. The authors [5] show that the result is reached more efficiently than running k-means again from scratch on the feature-extended object set.

The idea of incremental clustering introduced in [5] is used in this paper and extended to handle the problem of *dynamic clustering* of gene expression data.

3.2. **Clustering of gene expression data.**

A great number of algorithms have been proposed for clustering gene expression data sets that are not subject to change. Among these, we mention approaches based on the k-means [12] or the fuzzy k-means algorithms [13], on artificial neural networks [14] or methods using self organizing maps in conjunction with hierarchical clustering [15], with k-means clustering [16] or with particle swarm optimisation [17].

Concerning clustering of dynamic gene expression data sets, to our knowledge, there are no approaches in the literature that deal with the dynamic clustering problem as it was presented in Section 2.1. Although, as mentioned, no techniques exist that cluster gene expression data containing instances with an increasing number of features, there are a series of incremental clustering methods that are designed to work for data sets in which the number of instances may increase over time. We will present in the following some of these incremental approaches, as well as other studies that use dynamic or incremental clustering methods.

Sarmah and Bhattacharyya [18] present a density based approach technique for clustering gene expression data, which can also be applied for incremental data. Their algorithm, GenClus [18], obtains a hierarchical cluster solution and has as an advantage the fact that it does not require the number of clusters as input. InGenClus, the incremental version of GenClus, uses the result offered by GenClus and is able to update it when new genes are added to the data set, therefore decreasing the computational time. Both algorithms are evaluated using real-life data sets and the reported results prove a good performance.

In [19] the authors propose an incremental clustering algorithm for gene expression data – incDGC, based on a clustering algorithm they had previously introduced. The main idea is that when a new gene is introduced into the data set, the current clustering should only be affected in the neighborhood of this gene. This algorithm does not require as input the number of clusters and it helps avoiding performing the clustering each time the data set is updated. The algorithm was tested on three data sets and it proved to outperform other clustering algorithms, such as k-means or hierarchical clustering.

In addition to these techniques, which were designed to handle incremental data sets, the literature offers other examples of dynamic, or incremental algorithms. Lu *et al.* [20] introduce an Incremental Genetic K-means Algorithm (IGKA), which computes an objective value that the authors define and clusters centroids incrementally under the condition that the mutation probability from the genetic algorithm part is small, which leads to high costs of centroid calculation. Bar-Joseph *et al.* present in [21] a clustering algorithm for time series gene expression data that performs the clustering on continuous curve representations of the data, obtained using statistical spline estimation. In [22] the authors present a clustering method that uses a dynamically computed threshold value to determine the membership of a node to a cluster. An and Doerge [4] introduce a novel dynamic clustering approach that deals with the time dependent nature of the genes so that the genes in the same cluster may have different starting and ending points or different time durations.

4. Methodology.

In the following, we introduce our approach for dynamic clustering of gene expression data. The starting point for our proposal is the incremental clustering idea previously introduced in [5] that is extended to handle the problem of dynamic clustering of gene expression data.

4.1. Model.

Let $X = \{G_1, G_2, \ldots, G_n\}$ be the set of genes to be classified. Each gene is measured m times and is therefore described by an m-dimensional vector $G_i = (G_{i1}, \ldots, G_{im})$, $G_{ik} \in \Re$, $1 \le i \le n$, $1 \le k \le m$. An element G_{ik} from the vector characterizing the gene G_i represents the expression level of gene G_i at time point k.

Let $\{K_1, K_2, \ldots, K_p\}$ be the set of clusters discovered in data by applying the k-means algorithm. Each cluster is a set of genes, $K_j = \{G_1^j, G_2^j, \ldots, G_{n_j}^j\}$, $1 \le j \le p$. The centroid (cluster mean) of the cluster K_j is denoted by f_j, where $f_j = \left(\frac{\sum_{k=1}^{n_j} G_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} G_{km}^j}{n_j} \right)$.

Two of the most used similarity measures or distances for gene expression data are the *Euclidean distance* and the *Pearson correlation* [23]. The measure used in our paper for discriminating genes is the *Euclidian distance*: $d(G_i, G_j) = d_E(G_i, G_j) = \sqrt{\sum_{l=1}^{m} (G_{il} - G_{jl})^2}$. We have chosen this type of distance for the present study because it takes into account the magnitude of the changes in gene expression [23], therefore preserving more data.

The set of features consisting of the m expression levels of the genes coming from m consequent measurements is afterwards extended with s ($s \ge 1$) new features, coming from new measurements, numbered as $(m+1), (m+2), \ldots, (m+s)$. After extension, the genes' vectors become $G_i' = (G_{i1}, \ldots, G_{im}, G_{i,m+1}, \ldots, G_{i,m+s})$, $1 \le i \le n$.

We want to analyze the problem of recalculating the genes' grouping into clusters, after the extension of the genes and starting from the current partitioning. Our aim is to achieve an increased performance in comparison with the process of partitioning from scratch.

We start from the fact that, at the end of the initial *k-means* clustering process, all genes are closer to the centroid of their cluster than to any other centroid. So, for any cluster j and any gene $G_i^j \in K_j$, Inequality (1) below holds

$$d(G_i^j, f_j) \le d(G_i^j, f_r), \ \ \forall j, r, \ 1 \le j, \ r \le p, \ r \ne j. \tag{1}$$

We denote by K_j', $1 \le j \le p$, the set containing the same genes as K_j, after the extension. By f_j', $1 \le j \le p$, we denote the mean (center) of the set K_j'. These sets K_j', $1 \le j \le p$, will not necessarily represent clusters after the feature set extension. The newly arrived features can change the genes' arrangement into clusters, formed so that the intra-cluster similarity is high and inter-cluster similarity is low. Taking into account that the genes' features have equal weights (and normal data distribution) there is a considerable change when adding one or few features to the genes, for the old arrangement in clusters to be close to the actual one. The actual clusters could be obtained by applying the

k-means classification algorithm on the set of extended genes. However, we try to avoid this process and replace it with one less expensive but not less accurate. In the following, we continue to refer the sets K'_j as clusters.

Considering the partitioning into clusters obtained on the set of genes before the feature set extension, our focus is to identify conditions in which an extended gene $G_i^{j\prime}$ is still correctly placed into its cluster K'_j. For that, we express the distance of $G_i^{j\prime}$ to the center of its cluster, f'_j, compared with the distance to the center f'_r of any other cluster K'_r.

Starting from the approach introduced in [5] it can be easily proven that when Inequality (2) holds for an extended gene $G_i^{j\prime}$ and its cluster K'_j, for each new added feature ($\forall l \in \{m+1, m+2, \ldots, m+s\}$), then the gene $G_i^{j\prime}$ is closer to the center f'_j than to any other center f'_r, $1 \le j$, $r \le p$, $r \ne j$.

$$G_{il}^j \geq \frac{\sum\limits_{k=1}^{n_j} G_{kl}^j}{n_j}. \tag{2}$$

We have to notice that the inequality in (2) imposes only intra-cluster conditions. A gene is compared against its own cluster in order to decide its new affiliation to that cluster.

4.2. The *core based dynamic clustering of gene expression* (*CBDCGE*) approach.

4.2.1. *Centroids identification.* It is well known that a problem with the *k-means* algorithm is that it is sensitive to the selection of the initial centroids and may converge to a local minimum of the squared error value if the initial centroids are not properly chosen. In order to accurately evaluate our algorithm, we considered the same initial centroids when running *k-means* for the initial and feature-extended gene set (m and $m+s$ number of features). For identifying the optimal number of clusters in the data set with m features, as well as the initial centroids, the heuristic proposed below will be used.

In order to determine the appropriate number p of clusters, we are focusing on determining p representative entities, i.e., a representative entity for each cluster.

The main idea of *CBDCGE*'s heuristic for choosing the representative genes and the number p of clusters is the following:

(i) The initial number p of clusters is set to 0.
(ii) The first representative gene chosen is the most "distant" gene from the set of all genes (the gene that maximizes the average distance from all other genes). The number p of chosen representatives becomes 1.
(iii) In order to choose the next representative gene we have reason as follows. For each remaining gene (that was not already chosen), we compute the average distance (*davg*) from the gene and the already chosen representative genes. The next representative gene is chosen as the gene g that maximizes *davg* and this distance is greater than a positive given threshold (*distMin*), p is increased, and step (iii) is performed again. If such a gene does not exist, it means that g is very close to all the already chosen representatives and should not be chosen as a new representative. In this case, the iterative process of selecting the initial centroids stops.

We have to notice that step (iii) described above assures that near genes (with respect to the given threshold *distMin*) will be merged into a single cluster, instead of being distributed in different clusters.

4.2.2. *The algorithm.* We will use Inequality (2) in order to identify inside each cluster K'_j, $1 \leq j \leq p$, those genes have a considerable chance to remain stable in their cluster, and do not move into another cluster as a result of the feature set extension. These objects form the *core* of their cluster. The idea of our approach is to compute, for each cluster K_j, its core, denoted by $Core_j$.

Let us denote $StrongCore_j = \{G_i^{j'} | G_i^{j'} \in K'_j, G_i^{j'}$ satisfies Inequality (2) $\forall l \in \{m+1, m+2, \ldots, m+s\}\}$. We denote $WeakCore_j$ the set of genes in K'_j satisfies Inequality (2) for at least the average number of features (computed from all genes belonging to K'_j) for which (2) holds.

For each new feature l, $m+1 \leq l \leq m+s$, and each cluster K'_j there is at least one gene that satisfies Inequality (2) with respect to the feature l. Namely, the object that has the greatest value for feature l between all genes in K'_j certainly satisfies the relation (the maximum value in a set is greater than or equal to the mean of the values in the set). However, it is not sure that there is in cluster K'_j any gene that satisfies relation (2) for all new features $m+1, \ldots, m+s$. If there are such genes ($StrongCore_j \neq \emptyset$), we know that they are closer to the cluster center f'_j than to any other cluster center f'_r, $1 \leq r \leq p$, $r \neq j$. Then, $Core_j$ will be taken to be equal to $StrongCore_j$ and will be the seed for cluster j in the incremental algorithm. However, if $StrongCore_j = \emptyset$, then we will choose as seed for cluster j other genes, the most stable ones between all genes in K'_j. These genes ($WeakCore_j$) could be less stable than the genes in $StrongCore_j$. This is not, however, a certain fact: the genes in the "weaker" set $WeakCore_j$ can be as good as those in $StrongCore_j$. This comes from the fact that Inequality (2) holding for each new added feature, gives a *sufficient* condition for the genes in K'_j to be closer to f'_j than to any other f'_r, but not a *necessary* condition too.

The *cluster cores*, chosen as we described, will serve as seed in the incremental clustering process. All genes in $Core_j$ will surely remain together in the same group if clusters do not change. This will not be the case for all core genes, but for most of them.

We give next the *Core Based Dynamic Clustering of Gene Expression* algorithm.

We mention that the algorithm stops when the clusters from two consecutive iterations remain unchanged or the number of steps performed exceeds the maximum allowed number of iterations.

```
Algorithm Core Based Dynamic Clustering of Gene Expression is
Input: - the set $X = \{G_1, \ldots, G_n\}$ of $m$-dimensional genes
       - the set $X' = \{G'_1, \ldots, G'_n\}$ of $(m+s)$-dimensional extended genes
         $G'_i$ has the same first $m$ components as $G_i$,
       - the metric $d$ between objects in a multi-dimensional space,
       - $K = \{K_1, \ldots, K_p\}$ the partitioning of $X$,
       - $noMaxIter$ the maximum number of iterations allowed.
Output: - the re-partitioning $K' = \{K'_1, \ldots, K'_p\}$ for the genes in $X'$.
Begin
   For all clusters $K_j \in K$
      Calculate $Core_j = (StrongCore_j \neq \emptyset) ? StrongCore_j : WeakCore_j$
      $K'_j = Core_j$
      Calculate $f'_j$ as the mean of objects in $K'_j$
   EndFor
   While ($K'$ changes between two consecutive steps) and
         (there were not performed $noMaxIter$ iterations) do
      For all clusters $K'_j$ do
```

```
        $K'_j := \{G'_i \mid d(G'_i, f'_j) \leq d(G'_i, f'_r), \forall r, 1 \leq r \leq p, 1 \leq i \leq n\}$
        If $K'_j = \emptyset$ then
         @ remove $K'_j$ from the partition $K'$
        EndIf
      EndFor
      For all clusters $K'_j$ do
        $f'_j :=$ the mean of objects in $K'_j$
      EndFor
    EndWhile
End.
```

The algorithm starts by calculating the old clusters' cores. The cores will be the new initial clusters from which the iterative process begins. Next, the algorithm proceeds in the same manner as the classical k-means method does. We have to mention that if at a certain iteration a cluster from the partition becomes empty, it is removed from the partition, and consequently the number of clusters in the partition is decreased.

5. **Experimental Evaluation.**

In this section we aim at experimentally evaluating our dynamic core based clustering algorithm on gene expression data. The case study used in our experiment, the evaluation measures, as well as the obtained results are presented in the following subsections.

5.1. **Case study.**

For the computational experiments developed in order to test the performance of our method we used a real-life data set, taken from [24] and chosen for the following reasons:

- It is publicly available.
- It is a time series gene expression data set.
- It has been experimented by several works approaching the clustering problem, thus giving us the possibility to compare our results with other results existing in the literature.

Microarray technology was used by the authors of [24] to measure the levels of expression of 6400 genes belonging to the organism *Saccharomyces cerevisiae*, during its metabolic shift from fermentation to respiration. Gene expression levels were measured at seven time points during the diauxic shift: 0, 9, 11.5, 13.5, 15.5, 18.5 and 20.5 hours.

Before proceeding with the evaluation of the dynamic clustering algorithm, a pre-processing step must be applied on this data. First, we exclude those genes that have missing expression levels for certain time points. Then, we filter out the genes that are not expressed or whose expression values do not change. To this purpose, we used the MATLAB Bionformatics Toolbox [25], which offers functions that allow us to remove genes having small variance over time or very low absolute expression values, as well as genes whose profiles have low entropy. Following this pre-processing, the data set is reduced to a total number of 614 genes.

5.2. **Evaluation measures.**

We present in the following a set of *evaluation measures* that will be further used to compute the quality of the partitions provided by the clustering algorithms used in our approach. The first three measures (*IntraD*, *Dunn* and *Dist*) evaluate a partition from the clustering point of view and the last one (*Z-score*) evaluates a partition from a biological point of view.

In the following, let us consider a partition $K = \{K_1, \ldots, K_p\}$, where each cluster consists of a set of genes.

A. Intra-cluster distance of a partition – IntraD. The *intra-cluster distance* of a partition K, denoted by *IntraD(K)*, is defined as:

$$IntraD(K) = \sum_{j=1}^{p} \sum_{i=1}^{n_j} d(G_i^j, f_j)$$

where the cluster K_j is a set of genes $\{G_1^j, G_2^j, \ldots, G_{n_j}^j\}$ and f_j is the centroid (mean) of K_j.

From the point of view of a clustering technique, smaller values for $IntraD$ indicate better partitions, meaning that $IntraD$ has to be minimized.

B. Dunn Index – Dunn. The *Dunn index* [26] of a partition K is defined as:

$$Dunn(K) = \frac{d_{\min}}{d_{\max}}$$

where $d_{\min}$ represents the smallest distance between two genes from different clusters and $d_{\max}$ is the largest distance among two genes from the same cluster. The *Dunn index* takes values from the interval $[0, \infty]$. The greater the value of this index, the better a partition is, therefore the *Dunn index* should be maximized.

C. Overall distance of a partition – Dist. The *overall distance* of a partition K, denoted by *Dist(K)*, is defined as:

$$Dist(K) = \sum_{j=1}^{p} d_j$$

where d_j is defined as the distance between all pair of genes from the cluster K_j, i.e.,

$$d_j = \sum_{G_1, G_2 \in K_j} d(G_1, G_2)$$

From the point of view of a clustering technique, smaller values for *Dist* indicate better partitions, meaning that *Dist* has to be minimized.

D. Z-score. Z-score [27] is a figure of merit, indicating the relationship between a clustering result and the functional annotation of the used genes, within the gene ontology developed by the Gene Ontology Consortium [28]. A higher value of the Z-score indicates that the obtained clusters are more biologically relevant and therefore a more accurate clustering. To compute the Z-score for a partition we used the ClusterJudge software, which implements the algorithm described in [27].

5.3. **Results.** Considering an initial number of features (denoted by m) characterizing the genes from the considered data set (Subsection 5.1), and different values for the threshold $distMin$ used for determining the initial centroids in the k-means process (see Subsection 4.2.1), the experiments are conducted as follows:

1. The number of clusters nc and the initial centroids are identified in the data set using the heuristic presented in Subsection 4.2.1. The k-means clustering algorithm is applied on the data set consisting of m-dimensional genes, starting from the identified centroids and a partition $\mathcal{K}$ is provided. In our implementation if at a certain iteration in the k-means clustering algorithm a cluster from the partition becomes empty, it is removed from the partition, and consequently the number of clusters in the partition is decreased.

2. The set of features is now extended with s ($s \geq 1$) new features, numbered as $(m+1), (m+2), \ldots, (m+s)$. The *CBDCGE* adaptive algorithm (Subsection 4.2.2) is now applied, by adapting the partition $\mathcal{K}$ and considering the instances extended with the newly added s features.
3. The partition into clusters provided by *CBDCGE* algorithm (denoted by $\mathcal{K}_{CBDCGE}$) is compared with the one provided by the k-means algorithm applied from scratch on the $m+s$-dimensional instances (denoted by $\mathcal{K}'$). We mention that the initial centroids considered in the clustering process are the centroids identified at step 1. The comparison of the obtained partitions is made considering the evaluation measures presented in Subsection 5.2 (both from the clustering and biological point of view), as well as the number of iterations are performed by the clustering algorithms.

5.3.1. *First experiment.* In our first experiment, we are initially considering 5 features (i.e., $m = 5$) and afterwards the set of features is extended with 2 features (i.e., $s = 2$). Table 1 presents the results obtained in our experiment. For different values of the *distMin* threshold we indicate the initial number nc of clusters (heuristically determined as indicated above), and for the partitions $\mathcal{K}_{CBDCGE}$ and $\mathcal{K}'$ we indicate: the number of clusters in the partition, the number of iterations performed by the algorithm and the values of the evaluation measures (indicated in Subsection 5.2). We mention that the values reported for Z-score are averaged over 10 repeated experiments, for each value of *distMin*.

Analyzing the results indicated in Table 1, we observe the following:

TABLE 1. Results for the first experiment

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 3.231$ $nc = 63$					
$\mathcal{K}'$	62	21	411.7653	0.1345	13604.5546	5.4240
$\mathcal{K}_{CBDCGE}$	49	12	421.0210	0.1632	11319.7105	7.1500
2	$distMin = 3.233$ $nc = 62$					
$\mathcal{K}'$	61	21	417.6670	0.1472	15449.8604	5.2920
$\mathcal{K}_{CBDCGE}$	49	25	417.4004	0.1945	11119.4377	8.0740
3	$distMin = 3.26$ $nc = 61$					
$\mathcal{K}'$	60	19	423.3145	0.1356	15811.1460	5.6460
$\mathcal{K}_{CBDCGE}$	49	20	423.1767	0.1957	11976.0106	7.3780
4	$distMin = 3.44$ $nc = 47$					
$\mathcal{K}'$	47	24	440.9141	0.1586	17209.7174	6.2650
$\mathcal{K}_{CBDCGE}$	43	23	437.4579	0.18087	16150.0690	7.3010
5	$distMin = 3.45$ $nc = 46$					
$\mathcal{K}'$	46	26	444.9301	0.1586	18100.0294	6.0810
$\mathcal{K}_{CBDCGE}$	43	20	433.2875	0.1923	15451.7017	8.0780
6	$distMin = 3.47$ $nc = 44$					
$\mathcal{K}'$	44	21	448.1514	0.1586	17251.2868	7.8660
$\mathcal{K}_{CBDCGE}$	43	14	445.0609	0.1848	17734.6491	9.2680
7	$distMin = 3.51$ $nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	19597.3680	7.7190
$\mathcal{K}_{CBDCGE}$	40	15	436.4608	0.1664	15665.8960	10.5900

1. Excepting the second case and third case (lines 2 and 3 in Table 1) the number of iterations performed by the *CBDCGE* method is smaller than the number of iterations performed by k-means applied from scratch.
2. The partitions obtained by the *CBDCGE* method are better (considering all the evaluation measures presented in Subsection 5.2) than the partitions obtained applying k-means from scratch. In all the cases the *Dunn index* computed on the results obtained by the *CBDCGE* is greater than the one obtained by applying k-means from scratch, which denotes more compact and well separated clusters. The same holds for the Z-score measure, implying that the partitions obtained by *CBDCGE* are biologically more relevant. In what concerns the *IntraD* and *Dist* measures, they also indicate better partitions, exception of only two cases when either one or the other are greater for the k-means algorithm applied from scratch (line 1 – for *IntraD* and line 6 for *Dist*).

Considering the previous analysis, we can conclude that for the first experiment the partitions obtained adaptively (applying *CBDCGE* method) are better than the ones obtained by applying k-means from scratch. Also, the number of iterations performed by the clustering algorithm (excepting one case) is smaller for the *CBDCGE* method.

5.3.2. *Second experiment.* In the second experiment we have performed the evaluation of the *CBDCGE* method, by initially considering 6 features (i.e., $m = 6$) and afterwards extending the set of features with 1 feature (i.e., $s = 1$). Table 2 presents the results obtained in this experiment. For different values of the *distMin* threshold we indicate the initial number *nc* of clusters (heuristically determined as indicated in Subsection 4.2.1) and for the partitions $\mathcal{K}_{CBDCGE}$ and $\mathcal{K}'$ we indicate: the number of clusters in the partition, the number of iterations performed by the algorithm and the values of the evaluation measures (indicated in Subsection 5.2). As in the case of the first experiment, the values reported for Z-score are averaged over 10 repeated experiments, for each value of *distMin*.

Analyzing the results indicated in Table 2 we observe the following:

1. Excepting the first case (line 1 in Table 2) and the fourth case (line 4 in Table 2) the number of iterations performed by the *CBDCGE* method is smaller than the number of iterations performed by k-means applied from scratch.

Table 2. Results for the second experiment

No.	**No. of clusters**	**No. of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	$distMin = 4.38$ $nc = 63$					
$\mathcal{K}'$	62	21	411.7653	0.1345	13604.5546	5.6500
$\mathcal{K}_{CBDCGE}$	54	22	401.7995	0.1763	9276.8868	8.5700
2	$distMin = 4.401$ $nc = 62$					
$\mathcal{K}'$	61	21	417.6670	0.1472	15449.8604	5.5350
$\mathcal{K}_{CBDCGE}$	53	18	406.4398	0.1525	10003.3949	7.5120
3	$distMin = 4.6$ $nc = 45$					
$\mathcal{K}'$	45	26	446.8450	0.1586	18359.2485	6.5640
$\mathcal{K}_{CBDCGE}$	40	12	437.5520	0.1896	14658.3710	9.6970
4	$distMin = 4.66$ $nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	18757.3680	7.8520
$\mathcal{K}_{CBDCGE}$	40	23	438.2164	0.1615	14074.2680	8.2860

2. The partitions obtained by the *CBDCGE* method are better (considering all the evaluation measure presented in Subsection 5.2) than the partitions obtained applying k-means from scratch. Excepting the second case (line 4 in Table 2) the *Dunn index* computed on the results obtained by the *CBDCGE* is greater than the one obtained by applying k-means from scratch. The Z-score is always greater and both the *IntraD* and *Dist* measures are lower, all these indicating a better clustering result obtained by *CBDCGE*.

Considering the previous analysis, we can conclude that for the second experiment the partitions obtained adaptively (applying *CBDCGE* method) are better than the ones obtained by applying k-means from scratch. Also, the number of iterations performed by the clustering algorithm (excepting two cases) is smaller for the *CBDCGE* method.

6. **Discussion.**

Considering the experimental results presented in Section 5, an analysis of the method proposed in this paper is provided in this section. Furthermore, we present a study on the relevance of the considered features, as well as a comparison of our method with similar approaches in the literature.

6.1. **Analysis of the results.**

The clustering technique that we proposed in this paper is generally suitable for dynamic data sets, in which the features characterizing the instances are continuously subject to change. Particularly, as biological processes and experiments are dynamic, clustering the gene expression data resulting from these led to the need of a dynamic approach.

Our *dynamic core based clustering* algorithm has two main advantages over the traditional k-means algorithm applied from the beginning: less iterations and better clustering accuracies. As can be seen from Tables 1 and 2, the number of iterations used by our dynamic algorithm for finding the solution is, on average, smaller than the one needed by k-means, run from the beginning for the whole set of features.

In what concerns the clustering accuracy, we used four evaluation measures (see Subsection 5.2) to help us evaluate each clustering result. Regarding the measures *IntraD* and *Dist* we mention that a decrease in their values signifies better partitions, while for the *Dunn* and for the Z-score greater values result from better clustering. The fact that our *CBDCGE* algorithm leads, in most cases, to better partitions than the k-means algorithm applied on the whole set of features, is illustrated in Figures 1 and 2. By analysing the top two plots of each of these figures, it can be clearly observed that, except for one case (Figure 2, *Dunn*, case 4), both the *Dunn* and the Z-score of our algorithm are higher than those obtained for k-means. The last two plots of the same figures show that, in almost all cases (except for Figure 1, *IntraD*, case 7 and *Dist*, case 6), the values of *IntraD* and *Dist* for *CBDCGE* are lower than those computed for k-means.

Another benefit of our approach, is that by using the heuristic described in Subsection 4.2.1, it does not require apriori knowledge about the number of clusters. This number is automatically computed by the algorithm, using a positive threshold, *distMin*, which represents the minimum distance to be used when deciding whether to assign genes to the same or to different clusters. By its definition, an increase in the threshold *distMin* leads to a decrease in the number of clusters. From Table 1 we note that the most biologically relevant clustering for the first experiment is obtained for $distMin = 3.51$, while Table 2 indicates that for the second experiment, the best value of *distMin* is 4.6.

Regarding the number of new features, corresponding to new measurements at different time points, as the data set we used is composed of genes which were measured during a total of seven time points, we chose to use the first five time points for the initial partition and then incrementally add two features, in the first experiment or use the first

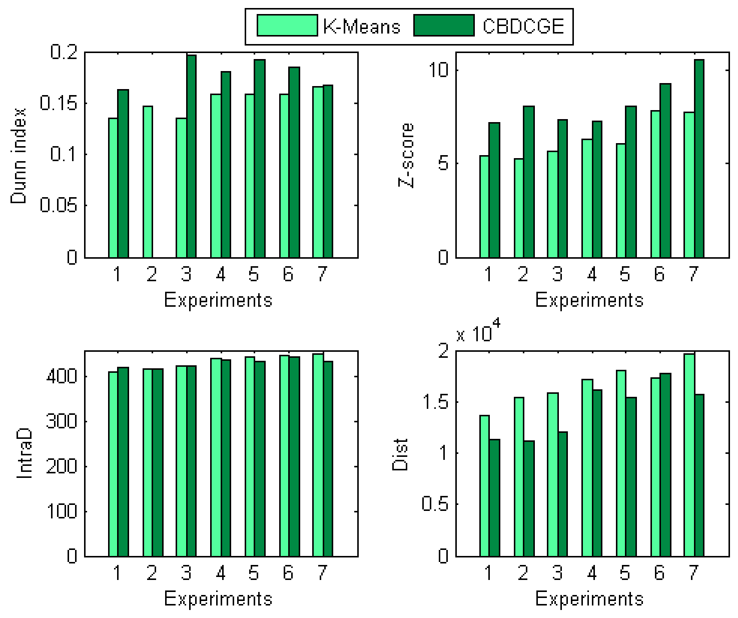

FIGURE 1. Illustration of the evaluation measures' values for the first experiment

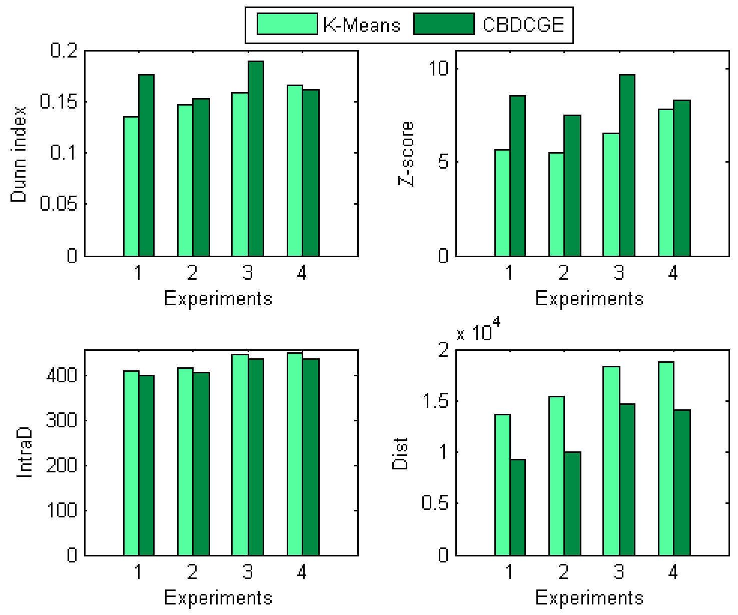

FIGURE 2. Illustration of the evaluation measures' values for the second experiment

six time points and then add one feature, in the second experiment. Table 3 presents the average values of the four evaluation measures for both algorithms (k-means applied from scratch and our adaptive algorithm *CBDCGE*) and for each of the two experiments. We remark that for the second experiment three out of the four considered evaluation measures, computed for *CBDCGE*, indicate that when only one feature is added the obtained clustering is more accurate: apart from the *Dunn*, whose value is lower for the second experiment, compared with the first, the values of both the *IntraD* and *Dist* measures decrease and the Z-score is greater. The same table also demonstrates that the *CBDCGE* algorithm outperforms the k-means applied from scratch, as all the evaluation

Table 3. Average values of the considered evaluation measures obtained for both the k-means and the *CBDCGE* algorithms, after the two experiments

Experiment	**Algorithm**	**IntraD**	**Dunn**	**Dist**	**Z-score**
First experiment	$\mathcal{K}'$	434.0442	0.1513	16717.7089	6.3276
	$\mathcal{K}_{CBDCGE}$	430.5522	0.1826	14202.4964	8.2627
Second experiment	$\mathcal{K}'$	431.9611	0.1515	16542.7579	6.4003
	$\mathcal{K}_{CBDCGE}$	421.0019	0.1700	12003.2302	8.5163

measures' values indicate better partitions: the values for *IntraD* and *Dist* are smaller, while those for the *Dunn* and the Z-score are greater.

6.2. **Study on features' relevance.**

6.2.1. *Information gain.* In the following we aim at analyzing how the *information gain (IG)* of the newly added features influences the efficiency of the dynamic clustering process. The *information gain* measure is a measure from *information theory* and expresses the expected reduction in entropy caused by partitioning the instances according to a given feature [29]. In order to compute the information gain of the features the partition obtained by applying k-means on the extended ($m + s$-dimensional) genes is used.

As gene expression levels take values in the $\Re$ space, in order to compute the information gain of the features we have to discretize their values. This was achieved by dividing their interval of variation into several sub-intervals.

In Table 4 we present, for each experiment and each different number of sub-intervals we used, the features in decreasing order of their information gain (the new features are emphasized), as well as a percentage indicating the information gain of the new features with respect to the one of the existing features.

Table 4. The information gain measure for the features

Experiment	No. of sub-intervals	Order of features	IG of new features / IG of old features (%)
First experiment	3	**7 6** 5 4 1 2 3	92.20%
	4	**7 6** 4 5 1 2 3	88.20%
	5	**7 6** 5 4 1 2 3	82.45%
	6	**7 6** 4 5 1 2 3	76.69%
Second experiment	3	**7** 6 5 4 1 2 3	51.37%
	4	**7** 6 4 5 1 2 3	50.78%
	5	**7** 6 5 4 1 2 3	51.25%
	6	**7** 6 4 5 1 2 3	50.66%

From Table 4 we can notice that the *IG* of the newly added features is rather high, when compared with the *IG* of the first set, especially for the first experiment. This may lead to a greater difficulty in adapting the partition (obtained by using the first set of features) for the *CBDCGE* algorithm. As mentioned before, the second experiment indicates that when only one feature is added the obtained clustering is more accurate and this could be explained by the fact that in the second experiment the information gain introduced in the system is lower. Another conclusion is that the *IG* of the features is monotonically related to the number of sub-intervals considered for the variation, as for both the existing and the new features the *IG* generally increases as the number of sub-intervals increases.

We have to mention two characteristics of the *CBDCGE* algorithm. First, the time complexity for calculating the cores in the clustering process does not grow the complexity of the global calculus. The second characteristic is that the method for calculating the core of a cluster (using Inequality (2)) depends only on the current cluster (does not depend on other clusters).

Considering the above analysis of the experimental results, we can conclude that applying the adaptive *CBDCGE* method is effective for dynamic clustering of gene expression data.

6.2.2. *Features' correlation.* To study how the sets of features are correlated with each other and to analyze how the correlation of the newly added attributes to the existing ones could influence the result of the clustering algorithm, we used the Pearson correlation coefficient [30]. The Pearson correlation is a statistical measure of the linear correlation between two random variables indicating how highly correlated the variables are. A Pearson correlation of 0 between two variables X and Y indicates that there is no linear relationship between the variables. A Pearson correlation of 1 or −1 results when the two variables being compared are linearly monotonically related. A Pearson correlation of 1 implies that a linear equation describes the relationship between X and Y, with all data points lying on a line for which Y increases as X increases. A correlation of −1 implies that all data points lie on a line for which Y decreases as X increases.

For each feature, we computed the Pearson correlation coefficient with the rest of the features. Figure 3 illustrates the average correlations among each feature and all the other features. By computing the mean value M of the average correlations of initial features (the first five), we notice that the average correlations of the last two features (the new ones) are both higher than M. From this we can conclude that the adaptation process occurs in a simpler manner.

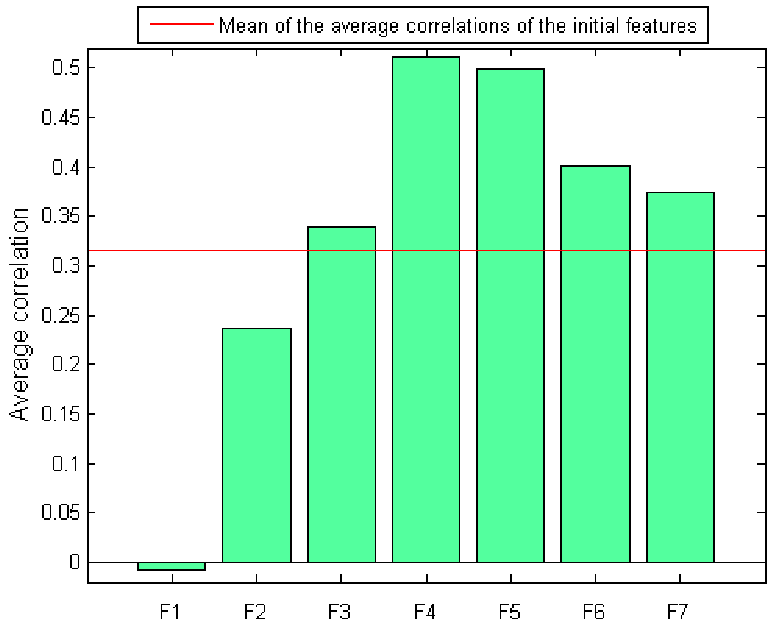

FIGURE 3. Illustration of the average correlations of the features

Table 5. Features' correlations

Experiment	$\frac{\text{Mean correlation of new features}}{\text{Mean correlation of old features}}$
First experiment	0.97
Second experiment	1.17

Table 5 illustrates the ratios between the mean of the average correlations of the newly added features and the mean of the average correlations of the existing ones. We notice that in the case of the second experiment this ratio is higher, thus indicating that when only one feature is added, the correlation between the new information and the existing one is stronger. This could be yet another reason for which the results obtained in the second experiment are more accurate than those obtained by the first.

6.3. **Comparison to related work.** To our knowledge, in the literature, there are no similar techniques that approach the problem of clustering a dynamic gene expression data set, that changes in the sense that new features (values of the gene expression levels at new time points) are added to the instances (genes). For this reason, we cannot provide a thorough comparison of our results to others. However, we note that the biological relevance of the partitions obtained using *CBDCGE*, quantified in the Z-score, is significant. Although our algorithm was designed with the purpose of providing an adaptive clustering technique for dynamic gene expression data sets, instead of a novel clustering method, we remark that, in terms of Z-score, it outperforms other existing incremental clustering algorithms proposed for gene expression data sets, which are subject to change, in the sense that they are enriched with new instances [18, 19]. Figure 4 illustrates the values of the Z-score reported by the algorithms *GenClus* [18] and *incDGC* [19] for the same data set that we used in our experiments. The same figure shows the averaged Z-score over all the experimental evaluations of our algorithm *CBDCGE*.

Compared with applying the traditional k-means algorithm from the beginning over and over each time new values of gene expression become available, our *dynamic core based clustering* algorithm generally obtained better clustering accuracies, in terms of

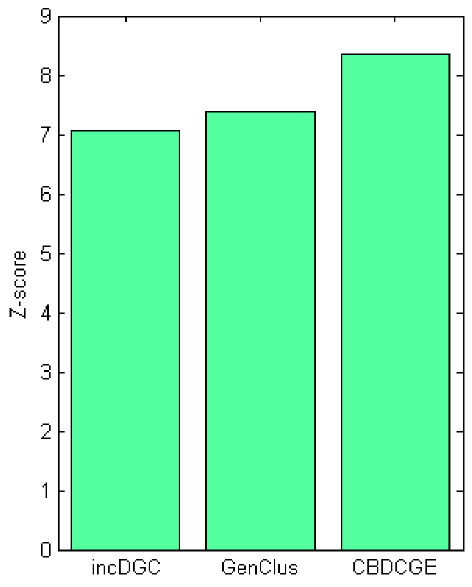

Figure 4. Comparative Z-score

TABLE 6. Reduction of the number of iterations

Experiment	Algorithm	No. of clusters	No. of iterations	Reduction of the no. of iterations (%)
Experiment 3	$\mathcal{K}_{CBDCGE}$	176	36	**2.77%**
	$\mathcal{K}'$	175	35	
	$\mathcal{K}_{CBDCGE}$	185	59	**47.45%**
	$\mathcal{K}'$	184	31	
Experiment 4	$\mathcal{K}_{CBDCGE}$	118	51	**7.84%**
	$\mathcal{K}'$	118	47	
	$\mathcal{K}_{CBDCGE}$	128	54	**12.96%**
	$\mathcal{K}'$	127	47	

the considered evaluation measures. Moreover, our algorithm needs a smaller number of iterations to achieve the solution, in most cases. This can be seen in Tables 1 and 2. Still, to see how the number of iterations is being modified even for larger gene expression data sets, we experimented on the data set described in Subsection 5.1, without applying all the pre-processing steps. More specifically, from the 6400 existing genes, we only eliminated the ones that have missing expression levels for certain time points, thus remaining 6276 instances. For this data set, we applied both the traditional k-means and our *CBDCGE* algorithms, similar to what we have presented in the two experiments in Subsection 5.3. In Experiment 3 we begun with 5 features and subsequently added the last 2 and in Experiment 4 we started with 6 features and then added the last one. Table 6 illustrates the percent by which the number of iterations is reduced, showing two considered cases for each experiment. From this table we can conclude that by using our *CBDCGE* algorithm the number of iterations is reduced in all four cases and in one case by up to almost 50%.

A large number of clustering algorithms that have been used to cluster gene expression data need the number of clusters as a priori information. Among these, we mention k-means [12], self organizing maps [31] or genetic algorithms [32]. Compared with these techniques, our approach has the advantage that the number of clusters is not required as input information, but is computed according to the heuristic described in Subsection 4.2.1. We remark, however, that there exist other algorithms in the literature that do not need this number of a priori: *GenClus* [18] and *incDGC* [19].

7. **Conclusions.**

In this paper we proposed a new method for adapting a clustering of gene expression data when expression levels for new points in time are added to the genes. To the best of our knowledge, our technique is the first in the literature which approaches the problem of dynamic clustering gene expression data sets from this point of view.

The experimental evaluations we performed on a real-life gene expression data set show that, in most cases, the clustering result is reached more efficiently and is also more accurate by using the proposed method than by running the k-means algorithm from scratch on the feature-extended data. As opposed to some clustering algorithms existing in the literature, our algorithm does not need the number of clusters as a priori information. In addition, when compared with other incremental clustering algorithms, the method that we introduced proves to obtain partitions that are more biologically relevant.

Further work can be done in order to study how the results described for dynamic clustering could be applied for other clustering techniques. We will also investigate an extension of the *CBDCGE* method to a fuzzy clustering [33] approach. Moreover, we plan to examine practical applications of the proposed method.

Acknowledgment. The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the paper.

REFERENCES

[1] D. Stekel, *Microarray Bioinformatics*, Cambridge University Press, Cambridge, UK, 2006.

[2] J.-Y. Jiang, W.-H. Cheng and S.-J. Lee, A dissimilarity measure for document clustering, *ICIC Express Letters*, vol.6, no.1, pp.15-21, 2012.

[3] B. Song and H. Lee, Prioritizing disease genes by integrating domain interactions and disease mutations in a protein-protein interaction network, *International Journal of Innovative Computing, Information and Control*, vol.8, no.2, pp.1327-1338, 2012.

[4] L. An and R. W. Doerge, Dynamic clustering of gene expression, *ISRN Bioinformatics*, vol.2012, pp.1-12, 2012.

[5] G. Şerban and A. Câmpan, Incremental clustering using a core-based approach, *Proc. of the 20th International Conference on Computer and Information Sciences, ISCIS'05*, pp.854-863, 2005.

[6] Y. Luan and H. Li, Clustering of time-course gene expression data using a mixed-effects model with b-splines, *Bioinformatics*, vol.19, no.4, pp.474-482, 2003.

[7] V. R. Pejaver, H. Lee and S. Kim, Gene cluster prediction and its application to genome annotation, *Protein Function Prediction for Omics Era*, pp.35-54, 2011.

[8] L. Lopez-Kleine, J. Romeo and F. Torres-Avils, Gene functional prediction using clustering methods for the analysis of tomato microarray data, *The 7th International Conference on Practical Applications of Computational Biology and Bioinformatics Advances in Intelligent Systems and Computing*, vol.222, pp.1-6, 2013.

[9] H.-J. Frasch, M. H. Medema, E. Takano and R. Breitling, Design-based re-engineering of biosynthetic gene clusters: Plug-and-play in practice, *Current Opinion in Biotechnology*, 2013.

[10] M. C. P. de Souto, I. G. Costa, D. S. A. de Araujo, T. B. Ludermir and A. Schliep, Clustering cancer gene expression data: A comparative study, *BMC Bioinformatics*, vol.9, no.497, 2008.

[11] A. K. Jain, Data clustering: 50 years beyond k-means, *Pattern Recogn. Lett.*, vol.31, no.8, pp.651-666, 2010.

[12] A. M. Bagirov and K. Mardaneh, Modified global k-means algorithm for clustering in gene expression data sets, *Proc. of the 2006 Workshop on Intelligent Systems for Bioinformatics*, pp.23-28, 2006.

[13] C. Arima, T. Hanai and M. Okamoto, Gene expression analysis using fuzzy k-means clustering, *Genome Informatics*, vol.14, pp.334-335, 2003.

[14] Y. Yao, L. Chen, A. Goh and A. Wong, Clustering gene data via associative clustering neural network, *Proc. of the 9th Intl. Conf. on Information Processing*, pp.2228-2232, 2002.

[15] J. Herrero and J. Dopazo, Combining hierarchical clustering and self-organizing maps for exploratory analysis of gene expression patterns, *Journal of Proteome Research*, vol.1, no.5, pp.467-470, 2002.

[16] N. Yano and M. Kotani, Clustering gene expression data using self-organizing maps and k-means clustering, *Proc. of SICE 2003 Annual Conference*, vol.3, pp.3211-3215, 2003.

[17] X. Xiao, E. R. Dow, R. C. Eberhart, Z. B. Miled and R. J. Oppelt, Gene clustering using self-organizing maps and particle swarm optimization, *Proc. of the 17th Intl. Symposium on Parallel and Distributed Processing*, 2003.

[18] S. Sarmah and D. K. Bhattacharyya, An effective technique for clustering incremental gene expression data, *International Journal of Computer Science Issues*, vol.7, no.3, pp.31-41, 2010.

[19] R. Das, D. K. Bhattacharyya and J. K. Kalita, An incremental clustering of gene expression data, *World Congress on Nature and Biologically Inspired Computing*, pp.742-747, 2009.

[20] Y. Lu, S. Lu, F. Fotouhi, Y. Deng and S. J. Brown, Incremental genetic k-means algorithm and its application in gene expression data analysis, *BMC Bioinformatics*, vol.5, no.172, 2004.

[21] Z. Bar-Joseph, G. Gerber, D. K. Gifford and T. S. Jaakkola, A new approach to analyzing gene expression time series data, *Proc. of the 6th Annual International Conference on Computational Biology*, pp.39-48, 2002.

[22] R. Das, J. K. Kalita and D. K. Bhattacharyya, A new approach for clustering gene expression time series data, *International Journal of Bioinformatics Research and Applications*, vol.5, no.3, pp.310-328, 2009.

[23] K. Kim, S. Zhang, K. Jiang, L. Cai, I. B. Lee, L. J. Feldman and H. Huang, Measuring similarities between gene expression profiles through new data transformations, *BMC Bioinformatics*, vol.8, no.29, 2007.

[24] J. L. DeRisi, P. O. Iyer and V. R. Brown, Exploring the metabolic and genetic control of gene expression on a genomic scale, *Science*, vol.278, no.5338, pp.680-686, 1997.

[25] R. Henson and L. Cetto, *The MATLAB Bioinformatics Toolbox. Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics*, The MathWorks Inc., Natick, Massachusetts, 2005.

[26] M. K. Pakhira, S. Bandyopadhyay and U. Maulik, Validity index for crisp and fuzzy clusters, *Pattern Recognition*, vol.37, no.3, pp.487-501, 2004.

[27] F. D. Gibbons and F. P. Roth, Judging the quality of gene expression-based clustering methods using gene annotation, *Genome Research*, vol.12, no.10, pp.1574-1581, 2002.

[28] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig et al., Gene ontology: Tool for the unification of biology. The gene ontology consortium, *Nature Genetics*, vol.25, no.1, pp.25-29, 2000.

[29] T. Mitchell, *Machine Learning (Mcgraw-Hill International Edit)*, 1st Edition, McGraw-Hill Education, 1997.

[30] S. Tuffery, *Data Mining and Statistics for Decision Making*, John Wiley and Sons, 2011.

[31] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander and T. Golub, Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation, *Proc. of National Academy of Sciences*, vol.96, no.6, pp.2907-2912, 1999.

[32] V. Gesu, R. Giancarlo, G. L. Bosco, A. Raimondi and D. Scaturro, Genclust: A genetic algorithm for clustering gene expression data, *BMC Bioinformatics*, vol.6, no.289, 2005.

[33] J. Yang and J. Watada, Fuzzy clustering analysis of data mining: Application to an accident mining system, *International Journal of Innovative Computing, Information and Control*, vol.8, no.8, pp.5715-5724, 2012.