Clustering Distributed Short Time Series with Dense Patterns

Josenildo C. da Silva*, Gustavo H. B. S. Oliveira*, Stefano Lodi† and Matthias Klusch‡
* Instituto Federal do Maranhão (IFMA), Depto. de Computação,
Av. Getúlio Vargas, 04, Monte Castelo, CEP 65030-005, São Luís, MA, Brasil
Email: {jcsilva,gustavo.oliveira}@ifma.edu.br
† Dipartimento di Informatica - Scienza e Ingegneria, Viale Risorgimento 2, Bologna, Italy
Email: stefano.lodi@unibo.it
‡ DFKI GmbH Stuhlsatzenhausweg 3, Campus D3.2, D-66123, Saarbrücken, Germany
Email: klusch@dfki.de

***Abstract*—The clustering of genes with similar temporal profiles is an important task in gene expression data analysis. Current approaches to the clustering of sparse gene expression data with temporal information suffer from their at least quadratic complexity in the number of clusters, the number of genes, or both, and are not distributed. In this paper, we present the first distributed and density-based approach to short time series clustering, called DTSCluster, which is suitable for gene expression data. DTSCluster identifies dense patterns in the distributed datasets and uses them to generate the time series clusters. The comparative experimental results revealed that DTSCluster is scalable in the dataset size with linear complexity in time and space, and outperforms other representative approaches in terms of cluster validation with the silhouette index as well. The distributed scenario also opens up the opportunity for collaborative data mining between different gene expression data holders.**

***Index Terms*—Time series clustering, short time series, distributed data clustering**

I. Introduction

Time series is an abundant form of data found on all areas of human activity. Stock options closing prices, electro cardiograms, number of sun spots, are but a few examples of data collected with time dimension. In fact, there has been much investigation on how to extract knowledge from time series [14], [13], resulting in a large body of research on time series similarity [23], [33], [29], pattern discovery [31], [9], [34], [28], classification [15], [37], [6], [2] and clustering [45], [25], [29], [20], [40].

Time course gene expression experiments capture the interactions of genes through a specific period of time. Typical gene expression data only contain a limited number of time points, giving rise to short-time-series data. One of the major challenges with time course gene expression data is data sparsity, which may lead to weak statistics due to limited sampling [47]. Sparsity is a consequence of high costs of experiments or a limited number of samples in animal or clinical studies.

An important task in gene expression data analysis is the clustering of genes with similar temporal profiles. Given data from time course gene expression experiments, researchers are often interested in identifying groups of differentially expressed genes which are significantly correlated with each other, since such genes might be part of the same causal mechanism or pathway [32]. Since data is sparse and includes temporal information, classical clustering algorithms usually do not perform well. STEM [12], [11] is the first software application designed specifically for the analysis of short time series from gene expression datasets (3 to 8 time points). Another and representative approach for interpolation-based consensus clustering (IBCC) with B-splines interpolation, affinity propagation, and consensus clustering has been proposed in [41]. However, these approaches suffer from their high computational complexity. For example, STEM is quadratic on the number of clusters, and IBCC is quadratic on the number of genes and the number of clusters. Besides, to the best of our knowledge, none of the current approaches address clustering for the distributed genomic data scenario [44], [38], [46], [51].

In this paper, we present a first approach to density-based distributed time series clustering with special focus on short time series, such as time course microarray [12] or RNA-sequencing [49], [7] experiments data. The basic idea of this method is to identify dense patterns in the distributed dataset and use them to generate time series clusters. The results of a comparative experimental evaluation reveal that this approach is scalable in the dataset size with only linear complexity in time and space, and improves cluster quality compared with relevant alternative approaches. Besides, the underlying distributed scenario also allows for collaborative research between different gene expression data holders.

The remainder of this paper is structured as follows. Related work is briefly summarized in Section II, and the addressed research problem is introduced in Section III. The proposed solution is presented in detail in Section IV, while the results of the comparative experimental evaluation are discussed in Section V. Section VI concludes this contribution.

II. Related Work

Time series clustering problem has been extensively investigated and applied to a large variety of fields [25], [35], [13]. For instance, it can be used as a pre-processing step for

time series classification, pattern discovery [18], forecasting [21], etc. Interestingly, a recent survey showed that most of the studies are focusing on representation methods, distance measurement and prototypes while less than 10% focused on enhancing cluster approaches [1]. In general, a well-known algorithm such as k-means, or self-organizing maps (SOM) [24], is modified to work with a transformed version or a model of the time series, e.g. k-means applied to shaplets [50]. Most time series from gene expression datasets contain less than 9 points, and there are few tools available geared towards the analysis of this type of data [12]. For short time series data, classical clustering algorithms are expected to perform less optimally due to data overfitting caused by the small number of sampled time points [42]. Moreover, short-time series data are usually very noisy and, therefore, algorithms that are designed to analyze either steady state data or long time-series data do not perform well due to their relatively small number of time points [43]. The main approaches for short time series clustering are: prototypes followed by clustering [27], [12], [11], feature extraction followed by clustering [16], [41], and statistical modeling [48], [3].

The FCV-TSD (Fuzzy-C Varieties with transitional stated discrimination) method in [27] for short time series clustering produces a set of prototypes from the original data, and then uses these prototypes to produce a cluster solution. However, the prototypes generation takes $O(k^2S)$ steps, where k is the number of clusters and S is the number of possible states in the time series. The clustering phase of FCV-TSD is an iterative algorithm that runs until it reaches a given stop criterion.

The STEM (Short Time-series Expression Mining) method in [12], [11] uses a predefined set of model profiles to capture the potential distinct patterns that can be expected from the experiment. The number of possible profiles for n time points is $p = (2S+1)^{n-1}$, where S is the number of possible states in the time series. A given profile shows how the original expression changes over time. First, a subset of possible profiles is selected and, then, STEM evaluates the significance levels of each profile using a permutation test based method. Finally, it assigns genes to its most similar profile, measured by a Pearson correlation based distance metric. STEM is one of the first methods designed to address short time series. However, STEM takes $O(k^2(2S+1)^{n-1})$ steps to select k clusters profiles with S possible states, and n is the size of the short time series. Similar profiles are grouped together via a graph-theoretical algorithm, where profiles originate large cliques of very similar profiles. Profile grouping takes $O(p'^4)$ steps, where $p' \leq p$ is the number of significant profiles.

The FBPA (Feature-Based PAM) method in [16] extracts features from each short time series and feeds them into the partition around medoids (PAM) clustering algorithm [22]. PAM is similar to k-means but uses median instead of mean to describe clusters. The features describing the time course are the vector of slopes between adjacent time points, maximum and minimum expression, time of maximum and minimum expression, and the steepest positive and negative slope. The feature extraction part is straightforward. However, the PAM algorithm used at the clustering step has time complexity of $O(k(Gn-k)^2)$ [30], where k is the number of clusters, and n is the size of the short time series and G is the number of genes. Therefore, FBPA does not scale well with the size of datasets.

The IBCC (Interpolation-Based Consensus Clustering) method in [41] is based on B-splines interpolation, affinity propagation, and consensus clustering. The coefficients of B-splines are used as features describing the time series. It requires no prior knowledge, such as the number of clusters or the cluster exemplars. However, IBCC has $O(G^2n^2 + k^2)$ time complexity, where G is the number of genes, n is the size of short time series, and k is the number of clusters.

The two-stages fuzzy clustering algorithm SiMM [3] first identifies all points that cannot be clustered with high certainty. This is followed by employing a variable length genetic algorithm to find a cluster solution without these points. For the purpose of the latter, a multiobjective genetic algorithm called NSGA-II [8] is used by SiMM running over a fixed number of iterations. SiMM runs for 100 generations with two objectives $M = 2$, and population $N = 50$. However, each iteration of the core algorithm NSGA-II is of quadratic complexity $O(MN^2)$, where M is the number of objectives to optimize and N the population size.

In summary, although current solutions improved the quality of gene expression clustering, they still present at least quadratic runtime complexity. Furthermore, all existing solutions assume a centralized dataset. In contrast, our approach assumes a distributed dataset and focus on scalability both of dataset size and number of peers in the mining group. Our approach also improves the quality of cluster solutions measured by average silhouette index.

III. Preliminaries

This section briefly introduces the general notations used in subsequent sections, and the problem of distributed short time series clustering that is being solved by our approach. Let $f : \mathbb{N} \to \mathbb{R}$ be a real-valued measurement function mapping timestamps to reals, and a *time series* T be a sequence of reals x_t coming from f. A time series is denoted as $T = \langle x_1, x_2, x_3, \ldots, x_n \rangle$, with $x_t = f(t)$, $1 \leq t \leq n$. With $|T|$ we denote the *length* of a time series T. A time series with small n is commonly called a *short time series*. For example, about 80% of the time series in the Stanford Microarray Database (SMD) are short by having less than 9 points only [12]; in the following, we assume $n \leq 20$. A *set of time series* is a denoted $\mathcal{T} = \{T_1, T_2, T_3, \ldots T_G\}$, where $|T_i| = n$, with $i = 1, \ldots, G$. Furthermore, let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a function measuring the *distance* between two vectors in $\mathbb{R}^n$.

For the distributed scenario, we assume a group of networked peers $\mathcal{L} = \{L_i\}, i = 1, \ldots, p$, each of which having an individual local dataset $\mathcal{T}_i$. The time series data collected at different sites refers to the same genes and has the same time spacing. The peer sites are organized into an unstructured peer-to-peer network, such that each party may act as an initiator or as an arbitrary party in any given clustering session. In a

given clustering session, peers organize themselves by means of a circular list, such that peer L_i knows neighbors L_{i-1} and L_{i+1}. The problem of distributed short time series clustering considered in this paper is as follows: Given a group of peers $\mathcal{L} = \{L_i\}$ and local datasets $\mathcal{T}_i = \{T_{i1}, T_{i2}, \ldots, T_{iG_i}\}$, the task is to find a partition $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of $\mathcal{T} = \bigcup_{i=1}^{p} \mathcal{T}_i$, such that similar time series are grouped together based on a given distance measure d. Each C_i is called a cluster, where $\mathcal{T} = \bigcup_{i=1}^{k} C_i$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. Additionally, no local dataset should be transferred among the peers during a clustering session. The basic idea of our approach to solve this problem is to transform the original series in a discretized form and detect the most frequent patterns in the new discrete space. These patterns are then utilized to group time series based on the observation that similar time series generate similar discrete transformations.

IV. Clustering Distributed Short Time Series

In this section, we present an approach to distributed time series clustering, called DTSCluster, that is particularly suitable for short time series analysis. The basic idea of DTSCLuster is inspired by the work reported in [26] on the discovery of frequent patterns in time series. While the latter first attempts to reduce the dimension of original time series and then discretizes the original time series values into a smaller set of symbols, DTSCluster does not need to perform the dimension reduction step but computes the set of most dense sequences found in the discretized time series. The relevant methods 1 and 2 together with the embedded auxiliary methods of DTSCluster are described in pseudo-code in more detail in the following.

A. Discretization

Each time series $T_i \in \mathcal{T}$ is transformed in a discretized version $S'_i \in \Sigma^w$. Thus, discretization generates the set $\mathcal{S} \subseteq \Sigma^w$ given $\mathcal{T}$. Each time series $T = \langle x_1, x_2, \ldots, x_n \rangle$ is transformed in a discretized sequence S', which is a sequence of symbols in a given alphabet Σ. For each element x of T, the corresponding string S' will have a symbol $\sigma_a \in \Sigma$. The substitution procedure first chooses breakpoints $\{\beta_a\}$ in the values dimension of original series T, such that $|\{\beta_a\}| = |\Sigma|+1$, and such that each occurrence of a given value s' of S' has the same probability [26], assuming they are normally distributed. For example, considering a 4-symbol alphabet, we need five break points, where each region will have probability 0.25 of appearing in T. Then, the substitution rule is applied for each $s'_j \in S'$:

$$s'_j = \sigma_a \text{ if } \beta_{a-1} < x_j \leq \beta_a, \text{with } 1 < a < |\Sigma|$$

B. Estimating Density of sequences

Local density estimation. Each local site L_j estimates the density of each sequence $S'_i \in \mathcal{S}$ corresponding to time series $T_i \in \mathcal{T}_j$, the local dataset. An important requirement is that the density estimate function $\hat{\varphi}$ be a non-negative function over $\mathbb{R}$. A general approach to compute data density function is kernel-based density estimation. For a given kernel function K such that $\int_{-\infty}^{+\infty} K(x)dx = 1$, an estimate of the density, for a specific dataset D, is given by:

$$\hat{\varphi}[D, r](x) = \frac{1}{Nh} \sum_{x_i \in Neigh(x,r)} K\left(\frac{d(x, x_i)}{h}\right) \qquad (1)$$

where N is the total number of points, d is a distance function. The parameter h is a bandwidth parameter and controls the smoothness of the density estimates. $Neigh(x, r)$ is the set of points close to x, in a given dataset D, which are inside a ball of radius r computed with distance d. Here, an arbitrary data point x_i in (1) is a discrete sequence $S'_i \in \mathcal{S}$. In the sequel, we will not need to compute values of the estimate at arbitrary space points, but only at the discrete sequences S'_i. Therefore, we represent the density estimate as a sample lookup table holding sequences S'_i and their density values.

Global density computation. The mining group of p peers cooperatively sums up all local density estimates. Initially, peer L_1 sends its local density to L_2. After that, each peer L_i receives partial density estimate $\hat{\varphi}_i$ from its neighbor L_{i-1}, $2 < i \leq p$. L_i adds its local density to the received partial global density estimates and sends the new partial global density $\hat{\varphi}_i$ to the next neighbor L_{i+1} in the mining group. This protocol continues until L_p sends the partial sum to L_1, which broadcasts the global density estimate $\hat{\varphi}$ to all members of the mining group. Notice that density estimates are additive, therefore, summing up all local densities produces the same result as estimating a global density from a centralized dataset.

C. Clustering Local Series with Global Profiles

The main idea is to use hill climbing on a global density estimate function to find clusters of strings $\mathcal{P} = \{P_i \subseteq \mathcal{S}\}$, with $i = 1, \ldots, k$. It is called cluster profile because the strings in P_i are discrete representations of original time series in $\mathcal{T}$. The set $\mathcal{P}$ is a cluster map in the string space and can be used to identify $\mathcal{C}$, the cluster map in the original time series space.

Label assignment. (Algorithm 3) Given the density estimate $\hat{\varphi}$ and a distance function d, the set of local maxima is defined as:

$$\mathcal{M} = \{\, S'_m \in \mathcal{S} \mid \forall S'_i \in \mathcal{S} : d(S'_m, S'_i) \leq r \rightarrow \hat{\varphi}(S'_m) > \hat{\varphi}(S'_i)\}.$$

Local maxima are found through hill climbing. For each string S'_i we choose a neighboring string S'_j that has a higher density than S'_i. If there is no such neighbor, the S'_i is a local maximum and receives a new label. Otherwise, the search continues from S'_j until a local maximum is found and its label is assigned to S'_j. Each point S'_i reaching a local maximum S'_m is thus assigned the same label as S'_m.

Building of cluster profiles on global density. (Algorithm 4) First, the local maxima in the global density estimate are identified, i.e., centers of the densest regions in the lookup table that stores the density estimates. Each string $S'_m \in \mathcal{M}$ is added to a different singleton set called $P_m = \{S'_m\}$, $m = 1, \ldots, k$. Then, each string $S'_i \in \mathcal{S}$ that is connected to a maximum $S'_m \in P_m$ will be included in the profile set P_m together with S'_m. The idea is that local maxima correspond

Algorithm 1 DTSCluster: initiator L_1

Require: A group of peers $\mathcal{L}$
Ensure: A local cluster map $\mathcal{C}_1$
At the initiator L_1:
$\mathcal{S}_1 \leftarrow$ DISCRETIZE($\mathcal{T}_1$, Σ) ▷ Cf. Sec. IV-A
$\hat{\varphi}_1 \leftarrow$ ESTIMATEDENSITY($\mathcal{S}_1$,d,r,K,h) ▷ Cf. Sec. IV-B
SEND($\hat{\varphi}_1$, L_2)
RECEIVE($\hat{\varphi}_p$, L_p)
$\hat{\varphi} \leftarrow \hat{\varphi}_p$
for all $S'_i \in \hat{\varphi}.getKeys()$ **do**
    ASSIGNLABEL(S'_i, $\hat{\varphi}$, r)
end for
$\mathcal{P} \leftarrow$ BUILDCLUSTERPROFILES($\hat{\varphi}$, r)
for all $L_i \in \mathcal{L}$ **do**
    SEND($\mathcal{P}$,L_i)
end for
$\mathcal{C}_i \leftarrow \emptyset$
for each $T_j \in \mathcal{T}_1$ **do**
    $l \leftarrow$ ASSIGNTOCLUSTER(T_j,$\mathcal{P}$, Σ)
    $C_l \leftarrow C_l \cup \{T_j\}$
    $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup C_l$
end for
return $\mathcal{C}_1$

Algorithm 2 DTSCluster: party L_i

Require: Peer group $\mathcal{L}$, local time series $\mathcal{T}_i$, alphabet Σ, distance d, radius r, Kernel function K, kernel width h
Ensure: Local cluster map $\mathcal{C}_i$
At an arbitrary party L_i:
RECEIVE($\hat{\varphi}_{i-1}$, L_{i-1})
$\mathcal{S}_i \leftarrow$ DISCRETIZE($\mathcal{T}_i$, Σ) ▷ Cf. Sec. IV-A
$\hat{\varphi}_i \leftarrow$ ESTIMATEDENSITY($\mathcal{S}_i$,d,r,K,h) ▷ Cf. Sec. IV-B
SEND($\hat{\varphi}_i$, $L_{i+1 \mod |\mathcal{L}|}$)
RECEIVE($\mathcal{P}$,L_h)
$\mathcal{C}_i \leftarrow \emptyset$
for each $T_j \in \mathcal{T}_i$ **do**
    $l \leftarrow$ ASSIGNTOCLUSTER(T_j,$\mathcal{P}$, Σ)
    $C_l \leftarrow C_l \cup \{T_j\}$
    $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup C_l$
end for
return $\mathcal{C}_i$

to strings that reoccur more frequently than others do in their cluster. Additionally, all neighboring strings S'_i that contributed to the density of a given maximum S'_m are also included in the cluster description since they represent variations of the maximum. We consider only the strings that are closer than radius r from S'_m to avoid clusters with strings that are too dissimilar to S'_m. The set of all profiles $\mathcal{P}$ represents a partition of all strings found during the density estimation. Thus, each string is in only one cluster profile.

Algorithm 3 assignLabel

Require: Discrete seq. S'_i, density estimate $\hat{\varphi}$, radius r
Ensure: Cluster label l
if $\neg S'_i.labeled()$ **then**
    $N \leftarrow \hat{\varphi}.getNeighbors(S'_i, r)$
    $S'_m \leftarrow S'_i$
    for all $S'_j \in N$ **do**
        if $\hat{\varphi}(S'_j) > \varphi(S'_m)$ **then**
            $S'_m \leftarrow S'_j$
        end if
    end for
    if $(i \neq m)$ **then**
        $l \leftarrow$ ASSIGNLABEL(S'_m,$\hat{\varphi}$,r)
    end if
    if $\neg S'_m.labeled()$ **then**
        $l \leftarrow newLabel()$
        $S'_m.setLabel(l)$
    end if
    $S'_i.setLabel(l)$
    return l
else
    return $S'_i.getLabel()$
end if

Algorithm 4 buildClusterProfiles

Require: Density estimates $\hat{\varphi}$
Ensure: Set of cluster profiles $\mathcal{P}$
$\mathcal{S} \leftarrow \varphi.getKeys()$
for all $S'_i \in \mathcal{S}$ **do**
    if $S'_i.getLabel() = j$ **then**
        $P_j \leftarrow P_j \cup \{S_i\}$
    end if
end for
return $\mathcal{P}$

Algorithm 5 assignToCluster

Require: Local time series T_i, profiles $\mathcal{P}$, alphabet Σ
Ensure: Cluster label j
$S'_i \leftarrow$ DISCRETIZE(T_i,Σ)
for all $P_j \in \mathcal{P}$ **do**
    if $S'_i \in P_j$ **then**
        return j
    end if
end for
return -1 ▷ labeled as noise

Grouping of local time series in global clusters. (Algorithm 5) Once the global set of profiles $\mathcal{P}$ is found by L_1 and broadcast to $L_i, i \neq 1$, each local peer L_i can use it to cluster its local time series. For each sequence S'_i, L_i compares it with strings in a profile $P_j \in \mathcal{P}$. Since the strings do not repeat in the profiles, S'_i will match with only one string in a particular

profile P_j. T_i will be assigned to the cluster C_j represented by the profile P_j containing the match.

D. Complexity Analysis

Computational complexity. In DTSCluster, the function ESTIMATEDENSITY() computes the density of each sequence in $\mathcal{S}$. Considering v as the number of neighbors (vicinity) for each sequence, and the cost of accessing these neighbors, it takes $O(|\mathcal{S}|vF)$ steps in the worst case, where F is the cost of fetching each neighbor and $v \ll |\mathcal{S}|$. Function ASSIGNLABELS() has computational complexity of $O(|\mathcal{S}|)$, since it visits each sequence in $\mathcal{S}$ and its neighbors. There are $2wr$ neighbors considering only the variations of one symbol (before and after a given position) with size w. Note that each sequence is labeled only once, and the search stops when the sequence is already labeled. Similarly, the construction of cluster profiles with function BUILDCLUSTERPROFILES() requires the visiting of each point in $\mathcal{S}$, which yields a complexity of $O(|\mathcal{S}|)$. In the last stage of DTSCluster, the function ASSIGNTOCLUSTER() examines each time series in $\mathcal{T}$, and assigns a cluster to it by looking up all possible profiles in $\mathcal{P}$, which results in a computational complexity of $O(|\mathcal{T}||\mathcal{P}|)$, with $|\mathcal{P}| \ll |\mathcal{T}|$. Finally, DTSCluster has a space complexity of $O(|\mathcal{S}|)$, which is the amount of space necessary to store all entries of a lookup table for the density estimates; in practice, this structure is sparse.

Communication. DTSCluster needs two rounds of communication, one to send the local density estimates to the helper, and a second one to communicate the cluster profiles. The messages from local peers to the helper have size $O(|\mathcal{S}_i|)$, and the number of messages sent from the helper to the other peers are $O(k)$, where k is the number of clusters.

V. EXPERIMENTAL EVALUATION

A. Setup

The experimental evaluation has been conducted on an Intel Core i7 (2.40 GHz) with 8GB memory running on Windows 8.1 64 bits. The core library of DTSCluster was developed in Java 7, and the distributed version was implemented with the JADE framework [4].

Kernel function. All experiments were performed with a Gaussian kernel with kernel bandwidth h equal to the radius ($h = r$).

Distance function. We used a Pearson-based distance for dissimilarity measurement $PD(x, y) = 1 - r_{xy}$, where r_{xy} is the Pearson correlation of two vectors x and y. Pearson correlation is defined as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean, and $\overline{y}$ is similarly defined. Pearson based distance is one of the most widely employed measure in gene expression data [19], [10]. Although there are studies indicating that Pearson-based distance may not be the best measure for clustering [39], all algorithms for short time series clustering we investigate utilize this measure [3], [41], [12], [16].

Cluster validation. The silhouette index (SI) is a cluster validity index that is used to assess the quality of any clustering map $\mathcal{C}$. Given the average distance $a(i)$ of a point i from the other points in the same cluster, and the minimum average distance $b(i)$ from point i to other points in other clusters, the silhouette width $s(i)$ of point i is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

The silhouette index SI of a cluster solution is the averaged silhouette width for all data points taking a value between -1 and 1, where higher values indicate better cluster solutions.

Alphabet, number of clusters and radius. Alphabet Σ, number of clusters k, and radius are user given parameters. We ran several experiments with different configurations of alphabet size, number of clusters k and radius values r, and restrict the reporting of results to configurations that maximized the average silhouette index for each dataset.

Datasets. The following data sets were used for the comparative performance evaluation of DTSCluster:

- The **Galactose** dataset is composed of 205 genes involved in galactose use in *Saccharomyces cerevisiae*. Expression profiles correspond to the mean of four replicates of 20 time points (20 perturbations in the galactose pathway)[17].
- The **Yeast sporulation** dataset contains gene expression measurements during sporulation for more than 6400 genes of budding yeast. The measurements were taken at seven time points (0h, 0.5h, 2h, 5h, 7h, 9h, and 11h). Genes with missing expression values and genes that showed no significant changes in expression during the process were excluded from the experimental analyses [3]. The final dataset is composed of 474 genes with 7 time point.
- The **Yeast cell-cycle** dataset [41], also known as **Y5** dataset, includes more than 6000 yeast genes. The expression levels were measured during two cell cycles at 17 time points as the 5-phase of the cell-cycle: early G1 (G1E), late G1 (G1L), S, G2, and M. In this paper, we used a subset of the Yeast cell-cycle dataset [36]. It consists of 384 genes with 17 time points.
- The **G27** dataset is initially composed of 24192 genes with 5 time points measuring the genetic expression of the wild-type G27 strain of *Helicobacter pylori*, a human pathogenic bacteria [12]. The Data was obtained from two replicates on the same biological sample in which time series data was collected at five time points (0h, 0.5h, 3h, 6h, and 12h).
- The **Yeast Lithium** dataset, contains 6.6678 genes expression measurements of wild-type strain CEN.PK113-7D, which was grown with 20 g/L galactose, with three samples analyzed at 0 minute and one sample at 20, 40, 60 and 140 minutes after addition of Lithium (LiCl, 10mM) [5].

Dataset	#Genes	n	k	Σ	r	SI
Galactose	205	20	4	$\{a, b, c\}$	0.20	0.79698
Sporulation	474	7	4	$\{a, b, c, \ldots, e\}$	0.05	0.79108
Y5	384	17	5	$\{a, b, c, \ldots, h\}$	0.15	0.46752
G27	2243	5	9	$\{a, b, c, \ldots, f\}$	0.01	0.62725
Lithium	6670	7	8	$\{a, b, c, \ldots, f\}$	0.01	0.74132

TABLE I: Performance of DTSCluster on various datasets

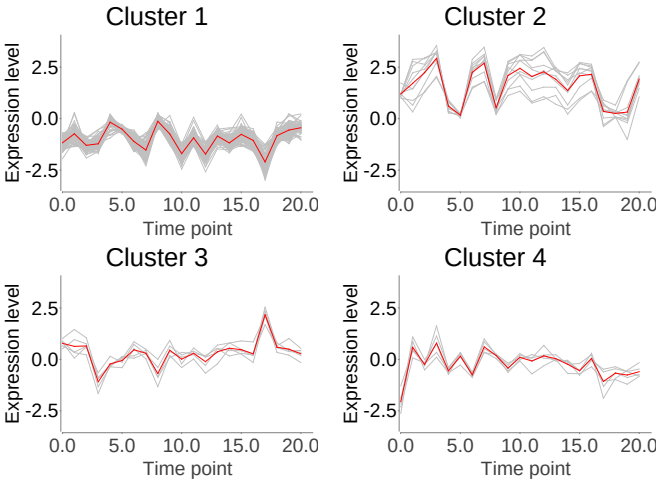


Fig. 1: Clusters from galactose dataset

B. Results

Cluster validation. Table I shows a summary of the silhouette index (SI) obtained by DTSCluster with several datasets, and expression profiles for each experiment. Each short time series from genes in the same cluster is plotted in gray, while the central red line indicates the arithmetic mean computed at each time point. The configuration used for each experiment and SI achieved by DTSCluster per dataset are as follows.

- **Galactose**: We used alphabet $\{a, b, c\}$ with radius $r = 0.2$. Figure 1 shows cluster profiles of 4 clusters found on the galactose dataset. DTSCluster achieved SI of 0.79108 with $k = 4$.
- **Yeast sporulation**: We used alphabet $\{a, b, c, d, e\}$ with radius $r = 0.05$. Figure 2 shows cluster profiles of 4 clusters found on the sporulation dataset. DTSCluster achieved an average SI of 0.79108 with $k = 4$.
- **Y5**: We used alphabet $\{a, b, c, d, e, f, g, h\}$ with radius $r = 0.15$. Figure 3 shows cluster profiles of 5 clusters found on the Y5 dataset. DTSCluster achieved an average SI of 0.46752 with $k = 5$. This dataset is difficult due to the presence of noise, which makes it hard to find good cluster solutions.
- **G27**: We used alphabet $\{a, b, c, d, e, f\}$ and radius $r = 0.01$. DTSCluster achieved average SI of 0.62725 with $k = 8$.
- **Yeast Lithium**: We used alphabet $\{a, b, c, d, e, f\}$ and radius $r = 0.01$. DTSCluster achieved average SI of 0.74132 with $k = 8$.

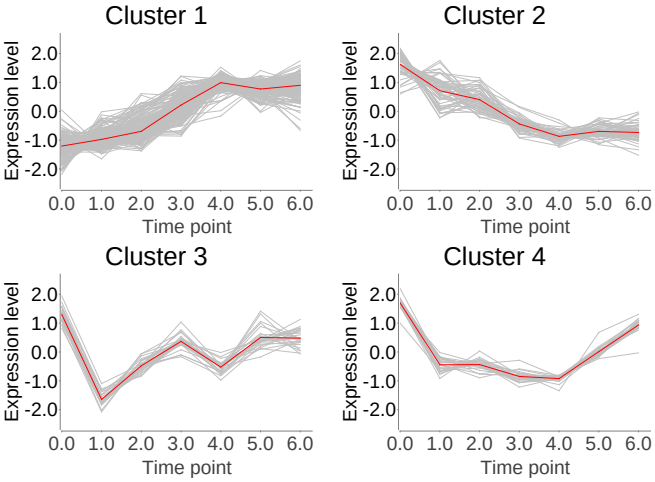


Fig. 2: Clusters from sporulation dataset

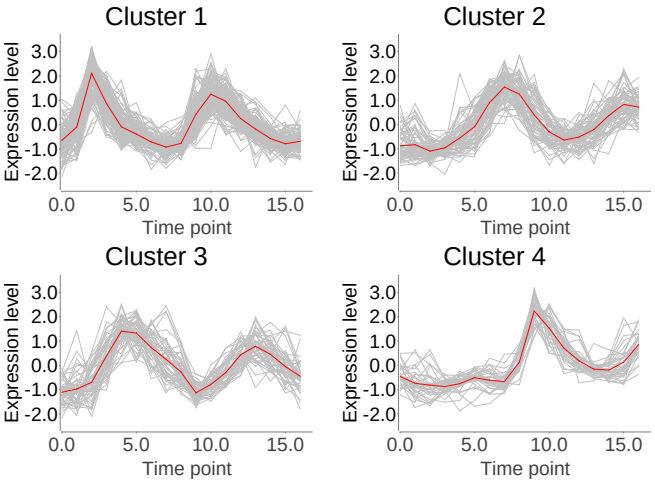


Fig. 3: Clusters from yeast cell-cycle (y5) dataset

Dataset	DTSCluster		IBCC	
	best k	SI	best k	SI
Galactose	4	**0.79698**	5	0.75451
Sporulation	4	**0.79108**	5	0.75976
Y5	5	**0.46752**	5	0.24416

TABLE II: Cluster validation: DTSCluster compared to IBCC [41]

Table II shows the comparative evaluation of DTSCluster with IBCC with respect to cluster validation on three different datasets. Note that in [41] the IBCC was applied to neither the Lithium nor the G27 dataset. DTSCluster significantly outperformed IBCC in terms of the silhouette index (SI) for all three datasets. Table III shows the results of DTSCluster compared to other representative clustering algorithms on the sporulation dataset.

Runtime performance. The runtime of DTSCluster has been measured for experiments with different dataset sizes. For this purpose, the G27 dataset was split into subsets with 1000

Algorithm		SI
DTSCluster		0.79108
IBCC	[41]	0.75976
SiMM-TS	[3]	0.62470
SOM	[24]	0.58450

TABLE III: Cluster validation: DTSCluster vs. IBCC, SiMM-TS, and SOM for the Yeast Sporulation dataset

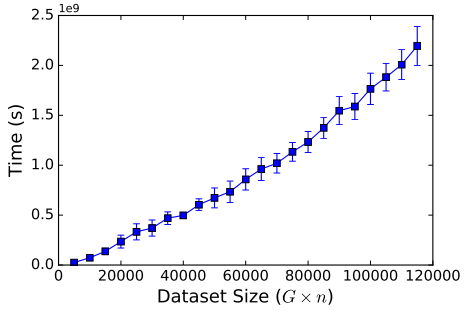


Fig. 4: DTSCluster execution time vs. dataset size

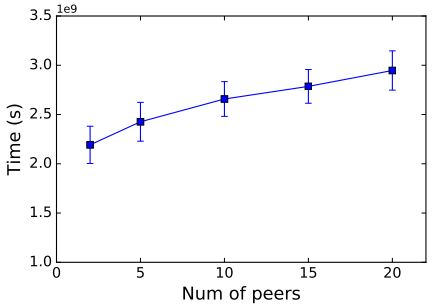


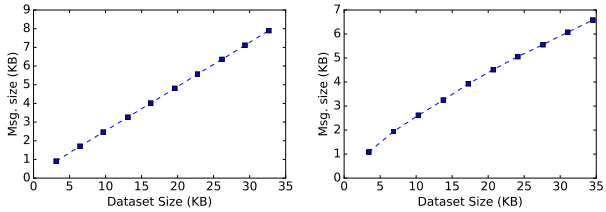
Fig. 5: DTSCluster execution time vs. number of peers

series, 2000 series up to 20000 series, with five time points for each series. Figure 4 displays the average execution time of DTSCluster for each dataset size. Figure 5 shows the results of measuring the execution time of DTSCluster for different numbers of peers, where the G27 dataset is split into subsets uniform at random and same size for each peer. It turns out that the execution time of DTSCluster grows sublinearly with the number of peers.

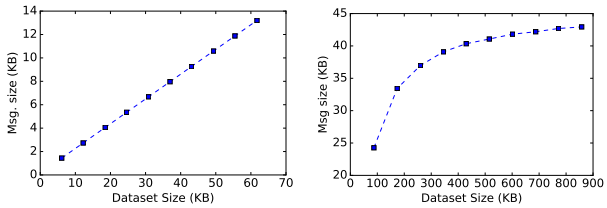
Message size and space. The lookup table which stores a sample of the density estimates is the largest data structure used by DTSCluster. Thus, we use it measure space complexity. The lookup table is also sent over the network from one peer to another and, therefore, accounts for the size of messages during the protocol. We run a series of experiments with subsets of datasets and performed 30 runs for each subset. Figure 6 shows the message size for different dataset sizes. Notice that in the worst case space grows linearly with the size of the dataset.

VI. Conclusions

We presented a novel distributed and density-based approach, called DTSCluster, for short time series clustering, which is



(a) Yeast Galactose. (b) Sporulation.



(c) Yeast (Y5) (d) H. pylori (G27).

Fig. 6: Message size vs. dataset size

particularly suitable for time course gene expression data. The two stages of DTSCluster are concerned with the identification of dense patterns in the distributed datasets and the utilization of these patterns for generating the time series clusters. Results of our comparative experimental evaluation revealed that DTSCluster scales in the dataset size and number of peers with linear complexity in time and space, and outperforms other representative approaches in terms of cluster validation with the silhouette index. Finally, DTSCluster allows for research cooperation among different parties without the necessity of transmitting original gene expression time series to a central location. Ongoing work is the integration of DTSCluster with the Gene Ontology[1] for integrated evaluation of computed clusters by DTSCluster with respect to biological relevance of found cluster solutions. DTSCluster currently provides a crisp cluster solution, which is a first step in a wider ongoing project. An interesting line of investigation is to allow for soft density based cluster solutions. We also plan to investigate other gene expression related problems, like bi-clustering and pathway analysis.

References

[1] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah. Time-series clustering: A decade review. *Information Systems*, 53(C):16–38, 2015.

[2] A. Bagnall, A. Bostrom, J. Large, and J. Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. In *Proc. of the 2nd Works. on Mining and Learning from Time Series (MiLeTS)*, 2016.

[3] S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik. An improved algorithm for clustering gene expression data. *Bioinformatics*, 23(21):2859–2865, 2007.

[4] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. *Jade — A Java Agent Development Framework*, pages 125–147. Springer, 2005.

[5] C. Bro, B. Regenberg, G. Lagniel, J. Labarre, et al. Transcriptional, proteomic, and metabolic responses to lithium in galactose-grown yeast cells. *Journal of Biological Chemistry*, 278(34):32141–32149, 2003.

[6] Z. Chen, W. Zuo, Q. Hu, and L. Lin. Kernel sparse representation for time series classification. *Information Sciences*, 292:15 – 26, 2015.

[1] Gene ontology consortium: http://www.geneontology.org/

[7] Y. Chu and D. R. Corey. Rna sequencing: platform selection, experimental design, and data interpretation. *Nucleic Acid Therapeutics*, 22(4):271–274, 2012.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolut. Computation*, 6(2):182–197, 2002.

[9] A. Denton, C. Besemann, and D. Dorr. Pattern-based time-series subsequence clustering using radial distribution functions. *Knowledge and Information Systems*, 18(1):1–27, 2009.

[10] R. Deshpande, B. VanderSluis, and C. L. Myers. Comparison of profile similarity measures for genetic interaction networks. *PLOS ONE*, 8(7):1–11, 07 2013.

[11] J. Ernst and Z. Bar-Joseph. Stem: a tool for the analysis of short time series gene expression data. *Bioinformatics*, 7(1):1–11, 2006.

[12] J. Ernst, G. J. Nau, and Z. Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(suppl 1):159–168, 2005.

[13] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):12:1–12:34, 2012.

[14] T.-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[15] P. Geurts. Pattern extraction for time series classification. In *Proc. of the 5th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 115–127. Springer, 2001.

[16] S. A. Ghandhi, A. Sinha, M. Markatou, et al. Time-series clustering of gene expression in irradiated and bystander fibroblasts: an application of fbpa clustering. *BMC Genomics*, 12(1):1–23, 2011.

[17] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, et al. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292(5518):929–934, 2001.

[18] F. Iglesias and W. Kastner. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. *Energies*, 6(2):579, 2013.

[19] P. A. Jaskowiak, R. J. Campello, and I. G. Costa. On the selection of appropriate distances for gene expression data clustering. *BMC Bioinformatics*, 15(2), Jan 2014.

[20] A. Jha, S. Ray, B. Seaman, and I. S. Dhillon. Clustering to forecast sparse time-series data. In *Proc. of the 31st Intl. Conf. on Data Engineering (ICDE)*, pages 1388–1399. IEEE, 2015.

[21] A. Jha, S. Ray, B. Seaman, and I. S. Dhillon. Clustering to forecast sparse time-series data. In *31st Intl. Conf. on Data Engineering (ICDE)*, pages 1388–1399, 2015.

[22] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[23] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2000.

[24] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, 2000.

[25] T. W. Liao. Clustering of time series data: a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[26] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In *Proc. of the 2nd Work. on Temporal DM at the 8th KDD*. ACM, 2002.

[27] C. Möller-Levet, K. Cho, and O. Wolkenhauer. Microarray data clustering based on temporal variation: FCV with TSD preclustering. *Applied Bioinformatics*, 2(1):35–45, 2003.

[28] Y. Mohammad and T. Nishida. Approximately recurring motif discovery using shift density estimation. In *Proc. of 26th Intl. Conf. on Industrial, Engineering and Applications of Intelligent Systems (IEA/AIE)*, pages 141–150. Springer, 2013.

[29] U. Mori, A. Mendiburu, and J. A. Lozano. Similarity measure selection for clustering time series databases. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):181–195, 2016.

[30] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, VLDB, pages 144–155. Morgan Kaufmann Publishers Inc., 1994.

[31] P. Patel, E. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *Proc. of Intl. Conf. on Data Mining (ICDM)*, pages 370–377. IEEE, 2002.

[32] K. Pollard and M. van der Laan. A method to identify significant clusters in gene expression data. In *Proc. of Conf. on Systemics, Cybernetics and Informatics (SCI), Vol. II*, pages 318–325, 2002.

[33] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, et al. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proc. of the 18th Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 262–270. ACM, 2012.

[34] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proc. of the Intl. Conf. on Data Mining (SDM)*, pages 668–676. SIAM, 2013.

[35] S. Rani and G. Sikka. Recent techniques of clustering of time series data: A survey. *Intl. Journal of Comp. Appl.*, 52(15):1–9, 2012.

[36] A. Schliep, I. G. Costa, C. Steinhoff, and A. Schonhuth. Analyzing gene expression time-courses. *IEEE/ACM Trans. on Comp. Biology and Bioinformatics*, 2(3):179–193, 2005.

[37] P. Senin and S. Malinchik. Sax-vsm: Interpretable time series classification using sax and vector space model. In *Proc. of the 13th Intl. Conf. on Data Mining (ICDM)*, pages 1175–1180. IEEE, 2013.

[38] H. Shi, C. Jiang, W. Dai, X. Jiang, Y. Tang, et al. Secure multiparty computation grid logistic regression (smac-glore). *BMC Medical Informatics and Decision Making*, 16(3):89, Jul 2016.

[39] A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah. A comparison study on similarity and dissimilarity measures in clustering continuous data. *PLOS ONE*, 10(12):1–20, 12 2015.

[40] K. Sirinukunwattana, R. S. Savage, M. F. Bari, et al. Bayesian hierarchical clustering for studying cancer gene expression data with unknown statistics. *PLoS ONE*, 8(10):e75748, 2013.

[41] C.-C. Y. Tai-Yu Chiu, Ting-Chieh Hsu and J.-S. Wang. Interpolation based consensus clustering for gene expression time series. *Bioinformatics*, pages 1–17, 2015.

[42] A. B. Tchagang, K. V. Bui, T. McGinnis, and P. V. Benos. Extracting biologically significant patterns from short time series gene expression data. *BMC Bioinformatics*, 10(1):1–11, 2009.

[43] A. B. Tchagang, S. Phan, F. Famili, H. Shearer, P. Fobert, Y. Huang, J. Zou, D. Huang, A. Cutler, Z. Liu, and Y. Pan. Mining biological information from 3d short time-series gene expression data: the optricluster algorithm. *BMC Bioinformatics*, 13(1):1–17, 2012.

[44] P. M. Thompson, J. L. Stein, S. E. Medland, Hibar, et al. The enigma consortium: large-scale collaborative analyses of neuroimaging and genetic data. *Brain Imaging and Behavior*, 8(2):153–182, Jun 2014.

[45] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *Proc. of Workshop on Clustering High Dimensionality Data and Its Applications, at the 3rd Intl. Conf. on Data Mining (SDM)*, pages 23–30. SIAM, 2003.

[46] S. Wang, Y. Zhang, W. Dai, K. Lauter, M. Kim, et al. Healer: homomorphic computation of exact logistic regression for secure rare disease analysis in GWAS. *Bioinformatics*, 32(2):211–218, 2016.

[47] X. Wang, M. Wu, Z. Li, and C. Chan. Short time-series microarray analysis: Methods and challenges. *BMC Syst. Biology*, 2(1):1–6, 2008.

[48] Y. Wang, M. Xu, Z. Wang, M. Tao, J. Zhu, L. Wang, R. Li, et al. How to cluster gene expression dynamics in response to environmental signals. *Briefings in Bioinformatics*, 13(2):162–174, 2012.

[49] Z. Wang, M. Gerstein, and M. Snyder. RNA-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetitcs*, 10(1):57–63, 2009.

[50] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised-shapelets. In *Proc. of the 12th Intl. Conf. on Data Mining (ICDM)*, pages 785–794. IEEE Computer Society, 2012.

[51] Y. Zhang, M. Blanton, and G. Almashaqbeh. Secure distributed genome analysis for GWAS and sequence comparison computation. *BMC Medical Informatics and Decision Making*, 15(5):S4, Dec 2015.