Fast and Accurate Hierarchical Clustering Based on Growing Multilayer Topology Training

Yiu-ming Cheung, *Fellow, IEEE*, and Yiqun Zhang

***Abstract*— Hierarchical clustering has been extensively applied for data analysis and knowledge discovery. However, the scalability of hierarchical clustering methods is generally limited due to their time complexity of $O(n^2)$, where n is the size of the input data. To address this issue, we present a fast and accurate hierarchical clustering algorithm based on topology training. Specifically, a trained multilayer topological structure that fits the spatial distribution of the data is utilized to accelerate the similarity measurement, which dominates the computational cost in hierarchical clustering. Moreover, the topological structure also guides the merging steps in hierarchical clustering to form a meaningful and accurate clustering result. In addition, an incremental version of the proposed algorithm is further designed so that the proposed approach is applicable to the streaming data as well. Promising experimental results on various data sets demonstrate the efficiency and effectiveness of the proposed algorithms.**

***Index Terms*— Data analysis, hierarchical clustering, incremental algorithm, time complexity, topology.**

I. Introduction

CLUSTERING methods can be classified into two types: partitional clustering [5]–[9], [26], [40] and hierarchical clustering [10], [18], [19], [31]. Partitional clustering separates a set of data points into a certain number of clusters to minimize the intracluster distance and maximize the intercluster distance, while hierarchical clustering views each data point as an individual cluster and builds a nested hierarchy by gradually merging the current most similar pair of them. Compared with partitional clustering, hierarchical clustering offers more information regarding the distribution of the data set. Often, the hierarchy is visualized using dendrograms, which can be "cut" at any level to produce the desired number of clusters. Due to the rich information it offers, hierarchical clustering has been extensively applied to different fields, e.g., data analysis, knowledge discovery, pattern recognition, image processing, bioinformatics, and so on [4], [11], [21].

Manuscript received April 18, 2017; revised November 2, 2017 and April 15, 2018; accepted June 27, 2018. Date of publication July 31, 2018; date of current version February 19, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61672444 and Grant 61272366, in part by the SZSTI under Grant JCYJ20160531194006833, and in part by the Faculty Research Grant of Hong Kong Baptist University under Project FRG2/16-17/051 and FRG2/17-18/082. *(Corresponding author: Yiu-ming Cheung.)*

Y. Cheung is with the Department of Computer Science, Hong Kong Baptist University (HKBU), Hong Kong, and also with the HKBU Institute of Research and Continuing Education, Shenzhen 518057, China (e-mail: ymc@comp.hkbu.edu.hk).

Y. Zhang is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: yqzhang@comp.hkbu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2853407

In general, a traditional hierarchical clustering framework can be summarized as follows.

Step 1: Each single data point is assigned to an individual cluster.

Step 2: The most similar pair of clusters is found according to a certain linkage strategy.

Step 3: The most similar pair of clusters is merged to form a new cluster.

Step 4: Steps 2 and 3 are repeated until only one cluster exists or a particular stop condition is satisfied.

In the above-mentioned steps, the commonly used linkage strategies are single linkage (SL), average linkage (AL), and complete linkage (CL), which compute the maximum, average, and minimum similarity between the data points of two clusters, respectively [27]. The traditional hierarchical clustering frameworks with SL, AL, and CL linkages are abbreviated as T-SL, T-AL, and T-CL hereinafter. Although these three traditional approaches are parameterless and simple to use, they have three major problems.

1) Their performance is sensitive to different data distribution types. T-SL "has a tendency to produce clusters that are straggly or elongated" [17]; T-CL and T-AL tend to produce compact and spherical-shaped clusters, respectively.
2) All three only consider the local distance between the pairs of data points during clustering. When overlapped clusters exist, their performances will be influenced [37].
3) Their time complexity is $O(n^2)$, which limits their applications, particularly for large-scale data and streaming data.

To tackle the above-mentioned three problems, various types of hierarchical clustering approaches have been proposed in the literature. To solve the first two problems, potential-based hierarchical clustering approaches based on potential theory [33] have been proposed (see [22] and [23]) where the potential field is utilized to measure the similarity between data points. Because this type of approach merges the data points by considering both the global distribution, i.e., potential fields of data points and local relationship, i.e., the exact distance between neighbors, they show robustness when processing data sets with different data distribution types and overlapped clusters. Nevertheless, their time complexity is still $O(n^2)$. To cope with the third problem, locality-sensitive hashing-based hierarchical clustering [20] has been proposed with a time complexity of $O(nm)$ to speed up the closest pair search procedure of T-SL, where m is the bucket size. However, the setting of parameters for this approach is nontrivial, and its clustering accuracy is generally lower than that of T-SL. Furthermore, hierarchical clustering based on random projection (RP) [30] with time complexity of $O(n(\log n)^2)$ has also been proposed. It accelerates T-SL and T-AL by iteratively projecting data points into different lines for splitting. In this manner, the data set is partitioned into small subsets, and the similarity can be measured locally to reduce computation cost. However, RP-based approaches inherit the drawbacks of T-SL and T-AL, i.e., they have a bias for certain data distribution types, and they cannot distinguish overlapped clusters well, due to approximation. To simultaneously tackle the three problems, summarization-based hierarchical clustering frameworks have been proposed in the literature. Specifically, data bubble-based hierarchical clustering and its variants [2], [3], [25], [29], [41] have been proposed to summarize the data points by randomly initializing a set of seed points to incorporate nearby data points into groups (data bubbles). Subsequently, the hierarchical clustering is performed on the bubbles only to avoid the similarity measurement for a large number of original data points. In general, the performance of the data bubble and its variants is sensitive to the compression rate and the initialization of seed points. Our preliminary work in [39] has addressed the sensitivity problem by training the seed points to better summarize the data points. Nevertheless, a common shortcoming of the summarization-based approaches is that the hierarchical relationship between data points is lost due to summarization. In addition, none of the above-mentioned approaches are fundamentally designed for streaming data. Specifically, the entire clustering process should be executed to update the hierarchy structure for each new input, which may sharply increase the computational cost. To solve this problem, the incremental hierarchical clustering (IHC) approach [34] has been proposed. It saves a large amount of computational cost by dynamically and locally restructuring the inhomogeneous regions of the present hierarchy structure. Therefore, this approach performs hierarchical clustering with a time complexity as low as $O(n \log n)$ when the hierarchy structure is completely balanced. However, the balance of the constructed hierarchy is not guaranteed, which makes its worst-case time complexity still $O(n^2)$. Furthermore, because IHC is an approximation of T-SL, it will also have a bias for certain data distribution types.

In this paper, we concentrate on: 1) addressing with the three above-mentioned problems of traditional hierarchical clustering frameworks and linkage strategies, and 2) proposing a new hierarchical clustering framework for streaming data. We first propose a growing multilayer topology training (GMTT) algorithm to dynamically learn the spatial distribution of data and construct the corresponding topological structure. In the literature, topology training has been widely utilized for partitional clustering [1], [14], [15], [28], [32], [36], [38]. However, to the best of our knowledge, it has yet to be utilized for hierarchical clustering. We make the topology grow by creating new layers with new seeds based on existing seeds if the existing seeds cannot represent the data set well. The growth is continued until each node can appropriately represent the local data distribution. As a result, the GMTT algorithm assigns more layers and seeds to finely describe the high-density region of data sets. Accordingly, a hierarchical clustering framework based on GMTT is formed. Differing from our preliminary work in [39], this framework can dynamically create and train seeds to form a multilayer topology. With the topology, the merging steps of hierarchical clustering are performed under its guidance. Moreover, the similarity between data points is only measured within each seed's corresponding subset, which can significantly reduce the computational cost. In general, most of the traditional linkage strategies, i.e., SL, AL, and CL, can be applied to the GMTT-based framework. To achieve better clustering performance, a new density-based linkage strategy is also presented. Because it simultaneously considers the global and local data distribution information, its clustering performance is promising. In addition, an incremental version of the GMTT framework, denoted as the IGMTT framework, is also presented to cope with streaming data. In the IGMTT framework, each new input can easily find its nearest neighbor by searching the topology from top to bottom. Then, both the existing topology and hierarchy are locally updated to recover the influence caused by the input. Both the GMTT and the IGMTT frameworks have competent performance in terms of clustering quality and time complexity, i.e., $O(n^{1.5})$. Their effectiveness and efficiency have been empirically investigated. The main contributions of our work are summarized as follows.

1) The GMTT algorithm is proposed for seed point training. The topology of the seed points can appropriately represent the structural data distribution. The training is automatic without prior knowledge of the data set, e.g., number of clusters, proper number of seeds, and so on.
2) A fast hierarchical clustering framework has been proposed based on GMTT. According to the topology trained through GMTT, distance measurement is locally performed to reduce computational cost. Merging is also guided by the topology to make the constructed hierarchy able to distinguish the borders of real clusters.
3) A new linkage strategy called density linkage (DL) is presented, which simultaneously considers the local and global data distribution information to make the clustering results robust to different data distribution types and overlapping phenomena.
4) An incremental version of the GMTT framework, i.e., the IGTMM framework, is provided for streaming data hierarchical clustering. Similar to the GMTT framework, it is also fast and accurate.

The rest of this paper is organized as follows. Section II gives an overview of the existing relevant hierarchical clustering approaches. In Section III, the details of the proposed GMTT framework, IGMTT framework, and DL linkage are described. Then, Section IV presents the experimental results for various benchmark and synthetic data sets. Finally, we draw a conclusion in Section V.

II. Overview of Existing Relevant Hierarchical Clustering Methods

A. Potential-Based Hierarchical Clustering

The approach proposed in [23] converts the distance between data points into potential values to measure the

Algorithm 1 Potential-Based Hierarchical Clustering

```
Input: Data set $X$
Output: Dendrogram $D$
1: for $i = 1$ to $n$ do
2:     compute the potential $\varphi_{x_i}$ by Eq.(2);
3: end for
4: for $i = 1$ to $n$ do
5:     find $x_i$'s parent $x_p$;
6:     link the pair as $x_p \rightarrow x_i$ to form a part of $EWT$;
7: end for
8: for $i = 1$ to $n - 1$ do
9:     find and merge the pair with the shortest edge in $EWT$;
10:    eliminate the edge between the merged pair;
11: end for
```

density levels of data points. Having the potential value of each data point, an Edge Weighted Tree (EWT) is constructed, and the hierarchy can easily be read off from it. Suppose that we have a data set with n data points, denoted as $X = \{x_1, x_2, \ldots, x_n\}$. The distance between two data points x_a and x_b is denoted as $\text{dist}(x_a, x_b)$. The potential value of point x_a received from point x_b is calculated by

$$\Phi_{x_a,x_b} = \begin{cases} -\dfrac{1}{\text{dist}(x_a, x_b)} & \text{if } \text{dist}(x_a, x_b) \geq \lambda \\ -\dfrac{1}{\lambda} & \text{if } \text{dist}(x_a, x_b) < \lambda \end{cases} \tag{1}$$

where the parameter λ is used to avoid the singularity problem when $\text{dist}(x_a, x_b)$ is too small. The total potential value of a data point x_a is defined as the sum of the potential values it has received from all of the other data points

$$\varphi_{x_a} = \sum_{i=1, i\neq a}^{n} \Phi_{x_a,x_i}. \tag{2}$$

According to the potential values and the distances between data points, an EWT is constructed by linking data points to its closest point with a higher potential value than it. The hierarchy of the data set can be read off from the EWT by sequentially merging the linked pair with the closest distance. The algorithm of the potential-based approach is summarized in Algorithm 1.

B. RP-Based Hierarchical Clustering

RP-based hierarchical clustering approaches aim to partition the entire data set into small enough subsets in which the data points are very close to each other. In this manner, the similarity can be measured within each subset to reduce computational cost. In this approach, data points are randomly projected onto different lines for splitting. After each projection, the original subset is split into two smaller subsets. After a certain amount of splitting, each subset will contain a small number of data points that are highly likely to be very close to each other, and each pair of the closest data points will stay in at least one of the subsets. The splitting is stopped when the size of each subset is smaller than a parameter $minPts$. Finally, all the similarity values of each subset are ranked together, and pairs of data points are merged according to

Algorithm 2 RP-Based Hierarchical Clustering

```
Input: Data set $X$
Output: Dendrogram $D$
1: perturb the data points;
2: while subsets with size larger than $minPts$ exist do
3:     partition these subsets using random projection;
4: end while
5: compute distance between data points within each subset;
6: sort all the computed distances together;
7: for $i = 1$ to $n - 1$ do
8:     merge the closest pair;
9: end for
```

the ranking. Because the procedures of the RP-based framework with SL linkage and RP-based framework with AL linkage are similar, both of them are summarized in Algorithm 2.

In the algorithm, improper selection of the parameter $minPts$ may lead to the failure of building a dendrogram. Therefore, parameter-free versions of RP-based approaches have also been proposed in [30], which solves the parameter selection problem by repeatedly performing the RP-based approaches with the different values of $minPts$ until the dendrogram can be correctly constructed.

C. Incremental Hierarchical Clustering

The IHC approach was proposed for streaming data, and it processes each input and maintains the hierarchy in three steps: 1) search the existing data points to find the one with the shortest distance to the new input; 2) detect the hierarchy in a bottom-up manner and insert a new input under a proper node; and 3) detect and restructure the hierarchy in a top-down manner. In the IHC approach, we can judge if a node is homogeneous or not according to its upper and lower limitation. For a new data point x_a, its nearest neighbor x_b is first located over the leaf nodes. Then, the upward detection is performed to x_b's parent node υ_p. If the distance $\text{dist}(x_a, x_b)$ between x_a and x_b is smaller than the upper limitation and larger than the lower limitation of υ_p, υ_p is judged to be homogeneous after accepting x_a as its child. In this case, x_a and x_b are said to form a normal density region under υ_p, and x_a is simply inserted into the hierarchy as υ_p's children. Similarly, if $\text{dist}(x_a, x_b)$ is smaller than the lower limitation, x_a and x_b will form a higher density region under υ_p. Therefore, the hierarchy should be restructured by inserting a new node with the child nodes x_a and x_b and parent node υ_p to maintain the homogeneity of the hierarchy. If x_a and x_b form a lower density region under υ_p, detection should be performed upward to υ_p's parent node, grandparent node, and so on until x_a is properly inserted into the hierarchy. Due to the incorporation of x_a, the homogeneity of the nodes in layers lower than x_a may also be influenced. Therefore, a downward detection and recovery are also necessary to detect and recover the inhomogeneous regions of the hierarchy until no inhomogeneous region is detected. The IHC approach is summarized in Algorithm 3.

Algorithm 3 IHC

```
Input: streaming data set $X$
Output: Dendrogram $D$
1: initialize the hierarchy;
2: for each new input $x_a$ do
3:     find $x_a$'s parent $v_p$;
4:     if $x_a$ cause a higher-density region then
5:         create new node as the parent node of $x_a$ and $x_b$;
6:     end if
7:     if $x_a$ cause a normal-density region then
8:         insert $x_a$ as $v_p$'s child node;
9:     end if
10: end for
11: detect and recover inhomogeneous regions;
```

III. The Proposed Fast Hierarchical Clustering Approach

This section will propose a topology training algorithm that can gradually and automatically make a topology grow to better represent the distribution of data. Subsequently, a framework based on it is presented to achieve fast and accurate hierarchical clustering. Furthermore, an incremental version of the framework will also be presented for streaming data hierarchical clustering.

A. Growing Multilayer Topology Training

The GMTT algorithm is presented, which trains a set of seed points to represent the data distribution. In the beginning, only one seed point is initialized and trained to be the physical center of the entire data set. Obviously, one seed point alone cannot represent the spatial distribution of the entire data set well, especially for complex real-world data sets. To better represent the data distribution, a number of new seed points are initialized and trained to be the child seed points of the original one. The new seed points are the centers of their corresponding subsets, which are produced by splitting the entire data set according to them. All of the newly created seed points are linked to their parent with edges, which indicate their affiliation. Because the seed points and their nested affiliation structure are very similar to the neuron nodes and the topology of multilayer neural networks, respectively, we utilize the words "nodes" and "topology" to indicate the seed points and their affiliation structure hereinafter. For each new node, growing training should be performed repeatedly until all of the existing seed points represent their subsets well. It is expected that more nodes should be assigned to the regions that are hard to represent well in the data set. There are many criteria for defining a region that is hard to represent well, e.g., inhomogeneous data distribution, high-density data distribution, border region of clusters, and overlapped region of several clusters. From the perspective of hierarchical clustering, the merging of data points happens in high-density regions at the beginning and gradually moves to low-density regions. Moreover, the merging of the high-density region data points dominates the processing time. Based on this scenario, we choose to better represent the high-density region via the GMTT algorithm. Consequently, the structure of the trained topology is similar to the desired hierarchy and can offer guidance to accelerate the hierarchical clustering procedures.

Specifically, given a data set $X = \{x_1, x_2, \dots, x_n\}$ with n data points, the topology T is trained by randomly inputting data points from X to adjust the nodes. Each node in T is expressed in the form of $v_{l,p,h}$, where l indicates the layer of the node in T, p is the sequence number of its parent node, and h is its own sequential number. For simplicity, $v_{l,p,h}$ can be denoted as v_h if the information of its layer and parent node is not considered in some cases. The corresponding subset of a node $v_{l,p,h}$ is expressed as X_h, which contains s_h data points belonging to X. During the training, we need to decide if the topology should grow or not. In other words, we should decide if a node $v_{l,p,h}$ can represent its corresponding subset X_h well and when to make the topology grow by creating B child nodes for $v_{l,p,h}$ in layer $l + 1$. Here, B is a constant referred to as the branching factor, and it controls the number of child nodes created for each node. The nodes that cannot represent their subset well are defined as coarse nodes. The definitions of a full coarse node and semicoarse node are as follows.

Definition 1: Let $v_{l,p,h}$ be a leaf node with a s_h-point corresponding subset. Given the branching factor B and the upper limitation U_L, the node $v_{l,p,h}$ is a full coarse node if and only if $s_h > U_L \cdot (B - 1)$.

Definition 2: Let $v_{l,p,h}$ be a leaf node with a s_h-point corresponding subset. Given the branching factor B and the upper limitation U_L, the node $v_{l,p,h}$ is a semicoarse node if and only if $U_L < s_h \leqslant U_L \cdot (B - 1)$.

In the above-mentioned two definitions, U_L controls the upper bound of the size s_h of $v_{l,p,h}$'s subset. For a full coarse node, B new child nodes should be trained by randomly selecting data points from X_h. For a semicoarse node, B_s new child nodes should be created and trained in the same manner, where B_s is the branching factor of a semicoarse node. During the training, the value of B_s will dynamically change according to the size of the semicoarse node's corresponding subset

$$B_s = \left\lceil \frac{s_h}{U_L} \right\rceil. \tag{3}$$

Supposing that $v_{l,p,h}$ is a full coarse node, B child nodes $\{v_{l+1,h,t+1}, v_{l+1,h,t+2}, \dots, v_{l+1,h,t+B}\}$ should be initialized from $v_{l,p,h}$'s subset $X_h = \{x_{h,1}, x_{h,2}, \dots, x_{h,s_h}\}$, where t is the total number of nodes before the initialization of B new child nodes. After the initialization, the value of t is updated by $t^{(new)} = t^{(old)} + B$. For an input $x_{h,i}$, the winner child node $v_{l+1,h,w}$ is determined among B child nodes by

$$w = \operatorname*{argmin}_{t-B+1 \leqslant j \leqslant t} \gamma_j \|x_{h,i} - v_{l+1,h,j}\|_2 \tag{4}$$

where γ_j is the winning frequency of node $v_{l+1,h,j}$ among B new child nodes. After the winner child node $v_{l+1,h,w}$ is selected out, it is adjusted with a small step toward $x_{h,i}$ by

$$v_{l+1,h,w}^{(\text{new})} = v_{l+1,h,w}^{(\text{old})} + \eta \cdot \left(x_{h,i} - v_{l+1,h,w}^{(\text{old})}\right) \tag{5}$$

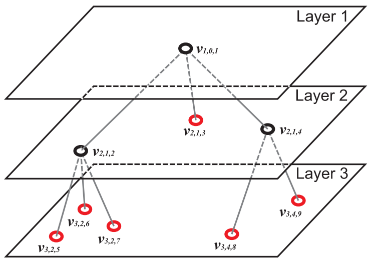
where η is the learning rate. The child nodes are iteratively trained through (4) and (5) until convergence. The training

Algorithm 4 Nodes Training

```
Input: Data set $X_h$, learning rate $\eta$, upper limitation $U_L$ and
  branching factor $B$ ($B_s$)
Output: $B$ ($B_s$) new child nodes
1: initialize $B$ ($B_s$) new nodes from subset $X_h$;
2: while $Convergence = false$ do
3:     randomly select a data point $x_{h,i}$ from $X_h$;
4:     find the winner node according to Eq.(4);
5:     adjust the winner node according to Eq.(5);
6: end while
```

Algorithm 5 GMTT Algorithm

```
Input: Data set $X$, learning rate $\eta$, upper limitation $U_L$ and
  branching factor $B$
Output: Topology $T$
1: initialize a node $\upsilon_{1,0,1}$ from $X$ to be the top node of $T$;
2: while existing full-coarse or semi-coarse node do
3:     find a coarse node $\upsilon_{l,p,h}$ in $T$;
4:     generate $B$ ($B_s$) new nodes through Algorithm 4;
5: end while
```



Node Information		
Node	Subset	Size
$v_{1,0,1}$	X_1	20
$v_{2,1,2}$	X_2	9
$v_{2,1,3}$	X_3	4
$v_{2,1,4}$	X_4	7
$v_{3,2,5}$	X_5	3
$v_{3,2,6}$	X_6	2
$v_{3,2,7}$	X_7	4
$v_{3,4,8}$	X_8	4
$v_{3,4,9}$	X_9	3

Fig. 1. Topology trained for a 20-points data set.

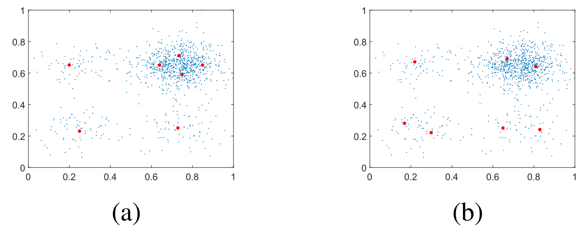


Fig. 2. Results of (a) GMTT and (b) its one-layer version.

procedure of child nodes can be summarized in Algorithm 4. After B new child nodes are created and trained for $\upsilon_{l,p,h}$, X_h is split into B subsets. As each new child node only represents a part of X_h, the current representation becomes more precise. Here, we also define the concept of a fine node to judge when to stop the growth of the topology.

Definition 3: Let $\upsilon_{l,p,h}$ be a child node with a s_h-point corresponding subset. Given the upper limitation U_L, the node $\upsilon_{l,p,h}$ is a fine node if and only if $s_h \leqslant U_L$.

When all the leaf nodes in the topology are judged as fine nodes, the growth is stopped. The entire GMTT algorithm is summarized in Algorithm 5. An example of the GMTT algorithm is illustrated in Fig. 1, where a three-layer topology is trained for a 20-point data set with $B = 3$ and $U_L = 4$. In the topology shown in Fig. 1, layer 1 contains only one top node $\upsilon_{1,0,1}$ with the corresponding subset X_1, which is also the entire data set X. Because $s_1 > U_L$, B child nodes are initialized and trained in the next layer using data points from X_1. A branch stops its growth with fine node $\upsilon_{2,1,3}$ in layer 2 because $s_3 \leqslant U_L$. Finally, the topology stops its growth in layer 3 because all of the leaf nodes are fine nodes, which means that the entire data set can be represented well. It can be seen from the figure that the union of all the leaf nodes' subsets X_3, X_5, X_6, X_7, X_8, and X_9 is the entire data set X.

Here, we also discuss why we design the GMTT algorithm but do not directly train sufficient seed points in one layer.

1) In GMTT, the number of corresponding data points of each leaf node will be smaller than U_L due to the GMTT. This guarantees that high-density regions have more nodes, and low-density regions have less. However, if we initialize a sufficiently large number of nodes once, some nodes will be trapped locally and will not represent the density distribution of the data well. Therefore, GMTT is more proper for hierarchical clustering. The results of GMTT and its one-layer version are compared in Fig. 2. It can be observed that the nodes trained through GMTT fit the density distribution better.
2) The structure of the topology trained through GMTT is consistent with the expected hierarchy whose nodes in deeper layers indicate a high-density distribution of data and vice versa. Moreover, links in the topology indicate the affiliation between subsets of nodes, which are similar to the links in the dendrogram. These properties make the topology suitable for hierarchical clustering. However, if all the seed points are trained in one layer, they cannot offer the desired information for hierarchical clustering.

Although the trained topology is similar to the desired dendrogram, they still have significant differences. First, each leaf node in the topology is the physical center of its subset but not an exact data point. Second, a link in the topology connecting two nodes only indicates their affiliation during the growth of the topology but not their detailed hierarchy relationship. Therefore, how to efficiently and effectively obtain the desired dendrogram through further processing of the topology will be discussed in Section III-B.

B. Fast Hierarchical Clustering Based on GMTT

From the perspective of hierarchical clustering, the constructed hierarchy should satisfy two properties: homogeneity and monotonicity [24]. Suppose we cut a dendrogram horizontally to produce a certain number of clusters; homogeneity is the property that the similarity between intracluster points is higher than that of the intercluster points. Monotonicity is the property that the clusters produced by cutting the dendrogram in a layer close to the bottom are more homogeneous than the clusters produced by cutting the dendrogram in a layer close to the top. In the topology obtained through GMTT, because the subset of each node is a local part of their parent node's subset, it roughly satisfies the property of homogeneity.

The monotonicity is also satisfied among the nodes that are lineal consanguinity of each other, where the concept of lineal consanguinity is defined in the following definition.

Definition 4: Let v_h be a node in T. If another node v_m in T can be found by searching T in a constant direction (bottom-up or top-down) from v_h, then v_h and v_m are said to be lineal consanguinity of each other.

For instance, $v_{3,2,5}$ and $v_{1,0,1}$ shown in Fig. 1 are lineal consanguinity of each other, but $v_{3,2,5}$ and $v_{2,1,3}$ are not. Even node $v_{2,1,3}$ is in layer 2, its homogeneity is not guaranteed to be lower than that of node $v_{3,2,5}$ in layer 3 because the subset of $v_{3,2,5}$ is not a local part of $v_{2,1,3}$.

To merge all of the data points according to the topology, data points inside leaf nodes' subsets and the subsets themselves should be merged according to a certain linkage strategy. The merging procedures should also comply with the lineal consanguinity relationship between topology nodes to exploit the homogeneity and monotonicity of the topology. As discussed in Section I, potential-based methods have competitive performance because they consider both the local and global data distributions. The potential value can also be understood as an index indicating the density level of a data point. In other words, a very small potential value indicates that the data point is located in a very high-density region. Because we focus on the density distribution of data points in this paper, we define density as the negative of potential as defined in (1) and (2). However, computing the density value for each data point by considering all of the other data points is very time-consuming, especially for large-scale data sets. To accelerate the computation, we present our density measurement to compute the density value of data points and nodes. The density $\theta_{h,i}$ of a point $x_{h,i}$ inside a leaf node v_h's subset X_h is estimated according to both its neighbors inside X_h and the other leaf nodes of the topology, which can be written as

$$\theta_{h,i} = \frac{1}{n-1}\left(\sum_{j=1,j\neq i}^{s_h} \omega_{h,i,j} + \sum_{m=1,m\neq h}^{u} \Omega_{h,i,m}\right) \tag{6}$$

where n is the size of X and u is the total number of leaf nodes inside the topology. $\omega_{h,i,j}$ is the density value of $x_{h,j}$ received from another data point $x_{h,j}$ in X_h. $\Omega_{h,i,m}$ is the density value of $x_{h,j}$ received from another leaf node v_m. Here, $\omega_{h,i,j}$ and $\Omega_{h,i,m}$ are defined as

$$\omega_{h,i,j} = \frac{1}{||x_{h,i} - x_{h,j}||_2} \tag{7}$$

and

$$\Omega_{h,i,m} = s_m \cdot \frac{1}{||x_{h,i} - v_m||_2} \tag{8}$$

respectively.

The density ϵ_h of a leaf node v_h can be written as

$$\epsilon_h = \frac{1}{n - s_h} \sum_{m=1,m\neq h}^{u} s_m \cdot \frac{1}{||v_h - v_m||_2}. \tag{9}$$

Accordingly, the density of a nonleaf node can be estimated in the same manner according to all of the other leaf nodes that are not lineal consanguinity of itself.

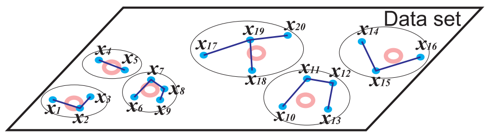


Fig. 3. Data points in the subsets are linked to form sub-MSTs.

```
Algorithm 6 GMTT Hierarchical Clustering Framework
Input: Data set $X$, learning rate $\eta$, upper limitation $U_L$ and
    branching factor $B$
Output: MST $M$
1: train a topology $T$ through Algorithm 5;
2: measure the density for the new child nodes according to
       Eq.(9) and their corresponding data points according to
       Eq.(6)-(8);
3: form sub-MSTs for the new child nodes and their corre-
       sponding data points according to Eq.(10);
```

Based on the above-mentioned density estimation, we also present our DL linkage to comply with the topology as follows. For a data point $x_{h,i}$ inside a leaf node v_h's subset X_h, a set of data points with higher density values than $x_{h,i}$ is selected out from X_h as $X_h^{\succeq\theta_{h,i}} = \{x_{h,j}|\theta_{h,j} \geq \theta_{h,i}, j = 1, 2, \ldots, s_h, j \neq i\}$. In $X_h^{\succeq\theta_{h,i}}$, the winner $x_{h,w}$ with the shortest distance to $x_{h,i}$ is selected out by

$$w = \operatorname{argmin}_{\{j|x_{h,j}\in X_h^{\succeq\theta_{h,i}}\}} ||x_{h,i} - x_{h,j}||_2 \tag{10}$$

and linked with $x_{h,i}$ through an edge with length $||x_{h,i} - x_{h,w}||_2$. When all of the data points in X_h are linked with their winner points, a sub-MST has been formed for X_h. In Fig. 3, we take the same data set and topology shown in Fig. 1 as an example to show the sub-MSTs formed through DL. According to the sub-MSTs, subsets should be linked to form a complete MST. Therefore, nodes in the same layer sharing the same parent node are also linked to form sub-MSTs according to their density values. It is commonly recognized that the hierarchical clustering result can be expressed in the form of an MST instead of a dendrogram because they contain the same information and can be converted to each other easily [16], [17], [24]. Therefore, the hierarchical clustering task can also be converted to form an MST for our GMTT framework with DL linkage. When sub-MSTs are formed for all of the leaf nodes' subsets and all of the nodes sharing the same parent, a complete MST linking all of the data points has been formed. The entire GMTT hierarchical clustering framework is summarized in Algorithm 6.

Here, we also introduce how to transform the MST into a dendrogram according to the corresponding topology and MST in three stages.

Stage 1: Only data points belonging to leaf nodes' subsets are considered for merging. Specifically, for a leaf node v_h, all the pairs of linked data points in its subset X_h are stacked together according to the ascending order of their edge lengths. The stacked pairs form a local merging queue (LMQ) q_h. After the LMQs: $\{q_1, q_2, \ldots, q_{u_l}\}$, are formed

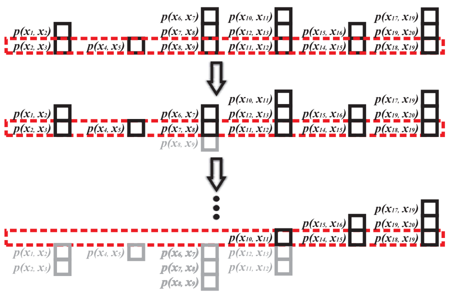


Fig. 4. Six LMQs: $\{q_1, q_2, \ldots, q_6\}$, are formed according to the corresponding sub-MSTs in Fig. 3. At the beginning, $q_3(1) = p(x_8, x_9)$ is the most similar pair among the candidates C (dashed frame). Therefore, x_8 and x_9 are merged first and $p(x_8, x_9)$ is removed from q_3.

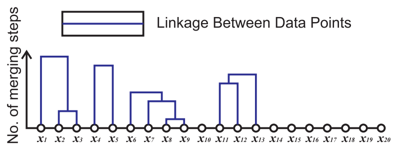


Fig. 5. Dendrogram at the end of Stage 1.

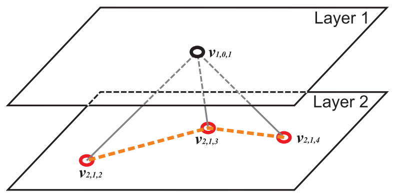


Fig. 6. Topology at the end of Stage 2.

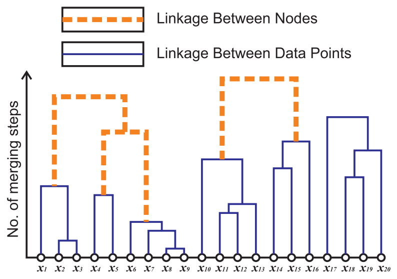


Fig. 7. Dendrogram at the end of Stage 2.

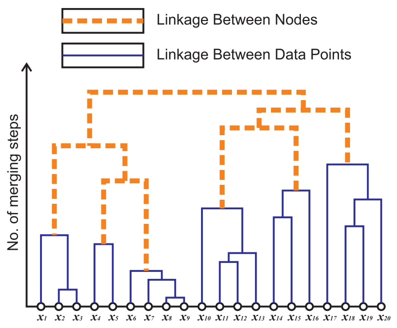


Fig. 8. Dendrogram at the end of Stage 3.

for all the u_l leaf nodes, a candidate set $C = \{q_1(1), q_2(1), \ldots, q_{u_l}(1)\}$ containing the pairs with the shortest edges in each LMQ is formed. A set of lengths of the edges $D = \{d_1(1), d_2(1), \ldots, d_{u_l}(1)\}$ of the candidates is also formed. Then, the most similar pair $q_g(1)$ that should be merged is found by

$$g = \operatorname*{argmin}_{1 \le i \le u_l} D(i). \tag{11}$$

After the merging, $q_g(1)$ is removed from both set C and q_g. Subsequently, the current most similar pair in q_g is popped up into C. The above-mentioned operations are iteratively performed until an all-leaf-parent (ALP) node in T becomes an all-candidate-parent (ACP) node. For a nonleaf node, if all its child nodes are leaf nodes, it is an ALP. When all the data points belonging to the subsets of ALP's child nodes are merged together within their subsets, the ALP becomes an ACP. Fig. 4 illustrates the merging procedure of Stage 1 using the 20-point data set from Figs. 3, and 5 shows the corresponding dendrogram. After merging the data points according to Fig. 5, an ALP $v_{2,1,2}$ becomes an ACP. Therefore, both the child nodes of ACP and data points belong to the subsets of all the other leaf nodes should be considered for merging in Stage 2.

Stage 2: Because the topology only guarantees the monotonicity of nodes that are lineal consanguinity of each other, lengths of edges between ACP's leaf nodes are not guaranteed to be larger than the edges linking unmerged data points belonging to the subsets of all the leaf nodes. Therefore, pairs of ACPs' leaf nodes are viewed as merging candidates and should be considered together with the data point candidates for merging. Suppose that v_h is the only ACP at the beginning of Stage 2, pairs of its leaf nodes should also be stacked to form an LMQ Q_h for node v_h according to their edge lengths. Afterward, the closest pair of nodes $Q_h(1)$ is put into the candidate set C. Because data points belonging to the subsets of ALP's child nodes continue to be merged, more ALPs will become ACPs in Stage 2. Because ACPs' child nodes also continue to be merged in Stage 2, when all the child nodes of an ACP are merged together, the ACP becomes a leaf node. In Stage 2, the merging of leaf nodes and data points is performed repeatedly until all of the ALPs become ACPs. Fig. 6 demonstrates the topology at the end of Stage 2. The corresponding dendrogram is presented in Fig. 7.

Stage 3. After Stage 2, the candidate set C only contains pairs of nodes in this stage. These nodes are finally merged according to their edge lengths until all of the candidates are merged together. The final dendrogram of the 20-point data example formed after Stage 3 is shown in Fig. 8.

The transformation algorithm is summarized in Algorithm 7.

Algorithm 7 MST-Dendrogram Transformation Algorithm

```
Input: MST $M$, topology $T$ trained through Algorithm 5
Output: Dendrogram $D$
1: generate LMQs for the subset of each leaf node;
2: generate merging candidates $C$;
3: while $C$ is not empty do
4:     if new ACP occurs then
5:         generate LMQ for the ACP;
6:         move the closest pair from the LMQ to $C$;
7:     end if
8:     merge the most closest pair in $C$;
9:     remove the merged pair from $C$;
10:    if the merged pair's LMQ is not empty then
11:        move the present closest pair from the LMQ to $C$;
12:    end if
13: end while
```

Traditional linkages, i.e., SL, AL, and CL, can also be applied to the GMTT framework, and their results can be transformed into dendrograms easily. Here, we offer guidance regarding how to apply them to the GMTT framework.

SL: Similar to the proposed DL, SL can also form an MST for hierarchical clustering tasks. Therefore, SL can be applied by taking the place of DL in the GMTT framework. The MST produced by it can be transformed into a dendrogram using Algorithm 7.

AL: Differing from DL and SL, AL merges data points according to the average distance between the present members of clusters and does not produce an MST. Therefore, we apply it to directly produce the candidate set C without forming LMQs. Whenever a pair of objects (data points or nodes) is selected from C for merging, AL will produce a new candidate among the objects with the same parent node as the merged one.

CL: CL can be applied in the same manner as AL.

When applying the three traditional linkages, nodes are viewed as data objects and processed according to their real values.

C. Incremental Hierarchical Clustering Based on GMTT

Streaming data processing is a significant challenge for hierarchical clustering approaches. To make the GMTT framework feasible for the processing of streaming data, we present its incremental version. We first train a coarse topology through GMTT using the former part of inputs. Then, for each new input, the coarse topology is dynamically updated through the incremental version of GMTT, which is abbreviated as IGMTT. Specifically, for a streaming data set X with n objects, the coarse topology is trained through the GMTT algorithm using the former r streaming inputs of X with the upper limitation U_L and branching factor B. Then, for each new input x_i, the closest leaf node υ_h of x_i is found by searching T from top to bottom according to the lineal consanguinity relationship. Subset X_h of υ_h incorporates the new input, and the size s_h of X_h is updated by $s_h^{(\mathrm{new})} = s_h^{(\mathrm{old})} + 1$. If υ_h is judged as a coarse node, the updating is triggered to update T by initializing and training B or B_s new child nodes for υ_h. To make the IGMTT algorithm more efficient, we choose a reasonable and efficient updating trigger condition. That is, the updating is only triggered when a full coarse node occurs. Otherwise, the algorithm will directly process the next input.

Algorithm 8 IGMTT Hierarchical Clustering Framework

```
Input: Streaming data set $X$
Output: MST $M$
1: train a coarse topology $T$ using the former $r$ inputs;
2: for $i = r + 1$ to $n$ do
3:     search to find the closest leaf node $\upsilon_h$ for $x_i$;
4:     $X_h = X_h \cup x_i$ and $s_h^{(new)} = s_h^{(old)} + 1$;
5:     if $\upsilon_h$ is a full-coarse node then
6:         generate $B$ new nodes through Algorithm 4;
7:         measure the density for the new child nodes and
8:         their corresponding data points;
9:         form sub-MSTs for the new child nodes and their
10:        corresponding data points;
11:    end if
12: end for
```

In our IGMTT framework, the MST connecting all the data points and nodes should also be updated dynamically according to each input. Specifically, for a new input x_i, the density values of existing data points and nodes are updated using the same trigger condition of the IGMTT algorithm for topology training. That is, if the subset X_h of node υ_h is judged as a full coarse node after accepting x_i, B new child nodes are initialized and trained for υ_h. The density values of the data points belonging to the subsets of the new child nodes and the child nodes themselves are calculated using (6) and (9). Then, sub-MSTs of each of the new child nodes' subsets and the new child nodes themselves are formed according to DL. The result of the IGMTT framework can also be transformed into a dendrogram according to Algorithm 7. To better explain the details of our IGMTT framework, we summarize it in Algorithm 8.

D. Discussion and Complexity Analysis

In this section, we further discuss and analyze the capabilities and potential limitations of the proposed GMTT framework in terms of distribution type and dimensionality of data sets. For the IGMTT framework, the relationship between clustering quality and the number of data points for training the coarse topology is also discussed.

1) *Distribution Type:* The proposed GMTT-DL approach is robust to different data distribution types, especially the overlapping type since the GMTT algorithm extracts the structural distribution of data and the DL linkage considers both the global and local distributions of data. For some special distribution types, i.e., chain-shaped, spherical-shaped, and ring-shaped distributions, its performance will not be very competitive compared to some traditional approaches that have an obvious bias for these distributions. However, these special distribution types will not occur individually in most of the real data sets.

By contrast, overlapping is very common in real data sets.

2) *Dimensionality:* The GMTT algorithm extracts the data distribution structure by gradually creating necessary nodes. New nodes gradually split the data space to detect and represent the data distribution. Due to the curse of dimensionality, the distribution of data points will be sparser for high-dimensional data. As a result, nodes trained through GMTT will be less representative, and the structural distribution information offered by the topology may have less contribution or even negative contribution to improve the clustering quality. However, the curse of dimensionality will also influence the other hierarchical clustering approaches since Euclidean distance is commonly utilized by the existing approaches.
3) *Coarse Topology:* The IGMTT algorithm trains a coarse topology using the former part of streaming data. Because it extracts the structural distribution of data and allows fine training for the coarse topology according to the following inputs, the size of the former part of streaming inputs for coarse topology training will not influence the clustering quality significantly if the distribution of streaming data does not change with time. The case in which the data distribution changes over time is another challenging problem for hierarchical clustering, which is not considered in this paper.

The above-mentioned discussion is further justified by the experimental results in Section IV.

We also prove that the time complexity of the GMTT and IGMTT frameworks can be optimized to $O(n^{1.5})$, which is lower than $O(n^2)$ of traditional approaches.

Theorem 1: The GMTT framework has time complexity $O(n^{1.5})$ if the upper limitation U_L is set at $\sqrt{n}$.

Proof: When the topology T trained through GMTT is a total imbalanced tree, we will have the worst case time complexity. In this case, the number of nonleaf nodes is $u_{nl} = (n - U_L/(B-1)U_L)$. From the top to the bottom of T, the numbers of data points for training the nonleaf nodes can be viewed as an arithmetic sequence $\{n, n-(B-1)U_L, n-2(B-1)U_L, \ldots, n-(u_{nl}-1)(B-1)U_L\}$. Therefore, total number of data points for training all the nonleaf nodes is $s_n = nu_{nl} - ((B-1)U_L(u_{nl}^2 + u_{nl})/2)$. For each of the data points, B nodes should be considered to find the winner node using (4). For each nonleaf node, the training will be repeated I times for convergence. Therefore, the time complexity for the topology training (Algorithm 6, line 1) is $O(s_n BI)$.

According to (6)–(8), $U_L - 1$ data points and $u_l - 1$ leaf nodes should be considered to measure the density for a data point, where $u_l = (n/U_L)$ stands for the number of leaf nodes in T. For n data points, the time complexity is $O(nU_L + nu_l)$. According to (9), at most $u_l - 1$ leaf nodes should be considered to measure the density for a node. For u_{nl} nonleaf nodes and u_l leaf nodes, the time complexity is $O(u_l(u_{nl}+u_l))$. Therefore, the time complexity for measuring the density for all the data points and nodes (Algorithm 6, line 2) is $O(nU_L + nu_l + u_l u_{nl} + u_l^2)$.

For each nonleaf node, a sub-MST should be constructed for its B child nodes. For u_{nl} nonleaf nodes in total, the time complexity is $O(u_{nl}B^2)$. For each of the leaf nodes, a sub-MST should be constructed for its corresponding U_L data points. For u_l leaf nodes in total, the time complexity is $O(u_l U_L^2)$. Therefore, the time complexity for constructing the MST (Algorithm 6, line 3) is $O(u_{nl}B^2 + u_l U_L^2)$.

The overall time complexity of the proposed GMTT framework is $O(s_n BI + nU_L + nu_l + u_l u_{nl} + u_l^2 + u_{nl}B^2 + u_l U_L^2)$. Here, I is a very small constant ranging from 2 to 10 according to the experiment. B is always set to a small positive integer, e.g., 2–4 in the experiments. When U_L is set at $\sqrt{n}$, the overall time complexity can be optimized to $O(n^{1.5})$. $\square$

With the same parameter setting, the complexity of applying SL, AL, and CL to the GMTT framework is also $O(n^{1.5})$.

Theorem 2: The IGMTT framework has time complexity $O(n^{1.5})$ if the upper limitation U_L is set at $\sqrt{n}$.

Proof: According to the proof of Theorem 1, the time complexity of the coarse topology training (Algorithm 8, line 1) is $O(r^{1.5})$, where r is the size of training set and $r \ll n$.

For n inputs, the time complexity for searching the closest leaf node (Algorithm 8, line 3) according to u_{nl} nonleaf nodes is $O(Bu_{nl}n)$.

According to Definition 1, lines 6–10 of Algorithm 8 will be performed once for every $U_L(B-1)$ new inputs. In other words, they will be triggered $n/U_L(B-1)$ times in total.

For each trigger, B new nodes should be trained by $U_L(B-1)$ data points and the training will be repeated I times for convergence (Algorithm 8, line 6). Therefore, the time complexity for $n/U_L(B-1)$ triggers is $O(nI)$.

For each trigger, $U_L - 1$ data points and $u_l - 1$ leaf nodes should be considered to measure the density for each of the $U_L(B-1)$ data points; $u_l - 1$ leaf nodes should be considered to measure the density for each of the B new nodes (Algorithm 8, line 7). Therefore, the time complexity for $n/U_L(B-1)$ triggers is $O((U_L + u_l)n + (u_l n/U_L))$.

For each trigger, a sub-MST for the corresponding U_L data points of each of the new nodes should be formed. Therefore, the time complexity for B new nodes should be $O(BU_L^2)$; A sub-MST should also be formed for the B new nodes, which has time complexity $O(B^2)$. For $n/U_L(B-1)$ triggers, the time complexity for the MST construction part (Algorithm 8, line 9) is $O(U_L n + (Bn/U_L))$.

The overall time complexity of the IGMTT framework is $O(r^{1.5} + Bu_{nl}n + nI + (U_L + u_l)n + (u_l n/U_L) + U_L n + (Bn/U_L))$. Similar to the GMTT framework, the time complexity can be optimized to $O(n^{1.5})$ with $U_L = \sqrt{n}$. $\square$

The time complexity of the proposed MST-dendrogram transformation algorithm is analyzed as follows. For a leaf node, the time complexity for forming LMQ for the corresponding U_L data points is $O(U_L^2)$. For u_l leaf nodes, the time complexity is $O(U_L^2 u_l)$. In each merging step, the distance between the first pairs in u_l LMQs should be compared to find the smallest one. For $n-1$ merges, the time complexity is $O(u_l n)$. Therefore, the overall time complexity of the transformation algorithm is $O(U_L^2 u_l + u_l n)$. When we set U_L at $\sqrt{n}$, the time complexity can also be optimized to $O(n^{1.5})$.

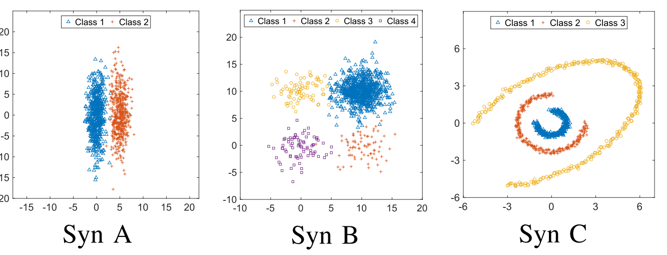


Fig. 9. Three synthetic data sets.

TABLE I

STATISTICS OF THE 10 DATA SETS

Data	#Ins.	#Att.	#Class	Data	#Ins.	#Att.	#Class
Seed	210	7	3	Mice	1080	77	8
Magic	19020	10	2	Urban	168	147	9
Occupy	20560	5	2	Syn A	1000	2	2
Frogs	7195	22	4	Syn B	1000	2	4
Protein	567	77	8	Syn C	903	2	3

IV. EXPERIMENTS

Experiments were conducted in three parts: 1) study of the parameters; 2) performance evaluation of the GMTT framework; and 3) performance evaluation of the IGMTT framework. All of the experiments were performed on both benchmark and synthetic data sets with different sizes, dimensions, and distribution types. All of the real data sets were collected from the UCI Machine Learning Repository[1] [13], and the synthetic data sets, Syn A–Syn C, are shown in Fig. 9. Statistics of the data sets is given in Table I. All of the feature values of the 10 data sets are normalized to the interval [0, 1] using the min-max normalization scheme for the experiments.

A. Evaluation Measures

The quality of the hierarchy produced by the proposed GMTT framework has been measured by two indices: hierarchy accuracy (H-Acc) [34] and the Fowlkes Mallows index (FM-index) [12].

The H-Acc is calculated by

$$\mathrm{Acc}_H = \frac{\sum_{c_i \in C} \|\varepsilon(c_i) \cap \varepsilon(h_i)\|}{n} \tag{12}$$

where n is the number of data points in the data set X and c_i stands for the ith class of the class set C. $\varepsilon(c_i)$ and $\varepsilon(h_i)$ denote the set of data points within class c_i in X and the set of data points under sub-hierarchy h_i, respectively. h_i stands for the subhierarchy in the hierarchy H, under which the points correspond to the points in the data set X with the class c_i. h_i can be defined as

$$h_i = \mathrm{argmax}_{h_j \in H} \frac{\|\varepsilon(c_i) \cap \varepsilon(h_j)\|}{\|\varepsilon(c_i) \cup \varepsilon(h_j)\|}. \tag{13}$$

To measure the FM-index, the constructed hierarchy structure should be cut horizontally to produce the same number of clusters as the number of classes of the original data. Suppose that R_1 is the classification result produced by the benchmark data set, and R_2 is the clustering result produced by horizontally cutting the hierarchy, the FM-index can be computed by

$$\mathrm{FM} = \sqrt{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \cdot \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}} \tag{14}$$

where TP is the total number of true positives, FP is the total number of false positives, and FN stands for a false negative. If two clustering results R_1 and R_2 match completely, the FM-index will take the maximum value 1, and vice versa.

To determine whether the performances of the proposed approaches are significantly better than those of their counterparts, we also use the Wilcoxon signed-rank test [35] to indicate the significance of improvements. In the experiments, we use "$-$" and "$+$" to express the acceptance and rejection of the null hypothesis, respectively. The acceptance and rejection of the null hypothesis indicate that the performance of our method is not significantly better and significantly better than that of the counterparts, respectively. For all the comparisons in the experiments, we use the commonly used significance level of $\alpha = 0.05$.

The experiments were conducted on a desktop computer with an Intel(R) Xeon(R) CPU with the main frequency of 3.30 GHz and 8 GB of DDR2-667 RAM.

B. Study of the GMTT Parameters

In the proposed GMTT framework, there are three parameters, i.e., the branching factor B, upper limitation U_L, and learning rate η. Each of them may influence the clustering performance in different ways. Here, we discuss them individually and also investigate the combination of any two of them by fixing the remaining one.

1) *Branching Factor B:* A too-large value of B may cause a flat topology, which makes the topology unable to distinguish between high-density and low-density distributions of data. Moreover, a too-flat topology cannot offer rich structural information for hierarchical clustering. Therefore, a too-large value of B may influence the quality of the hierarchy. By contrast, a too-small B may make the topology too deep. Too many layers will lead to a high computational cost for topology training. In addition, a too-small B will split data points into large subsets in the topology, which may incorrectly split real clusters and yield poor clustering accuracy. Therefore, a too-small B may influence the run time and quality of the GMTT framework.
2) *Upper Limitation U_L:* A too-large U_L may make the subsets too large to distinguish different clusters. However, it will accelerate the run time of GMTT framework. A too-small U_L may lead to large amounts of computation because it yields a deep topology.
3) *Learning Rate η:* Both a too-large and a too-small η will lead to a high computation cost and will influence the clustering quality. When η is too large, the training is very unstable after each adjustment and hard to converge. When η is set too small, training needs many

[1]http://archive.ics.uci.edu/ml/

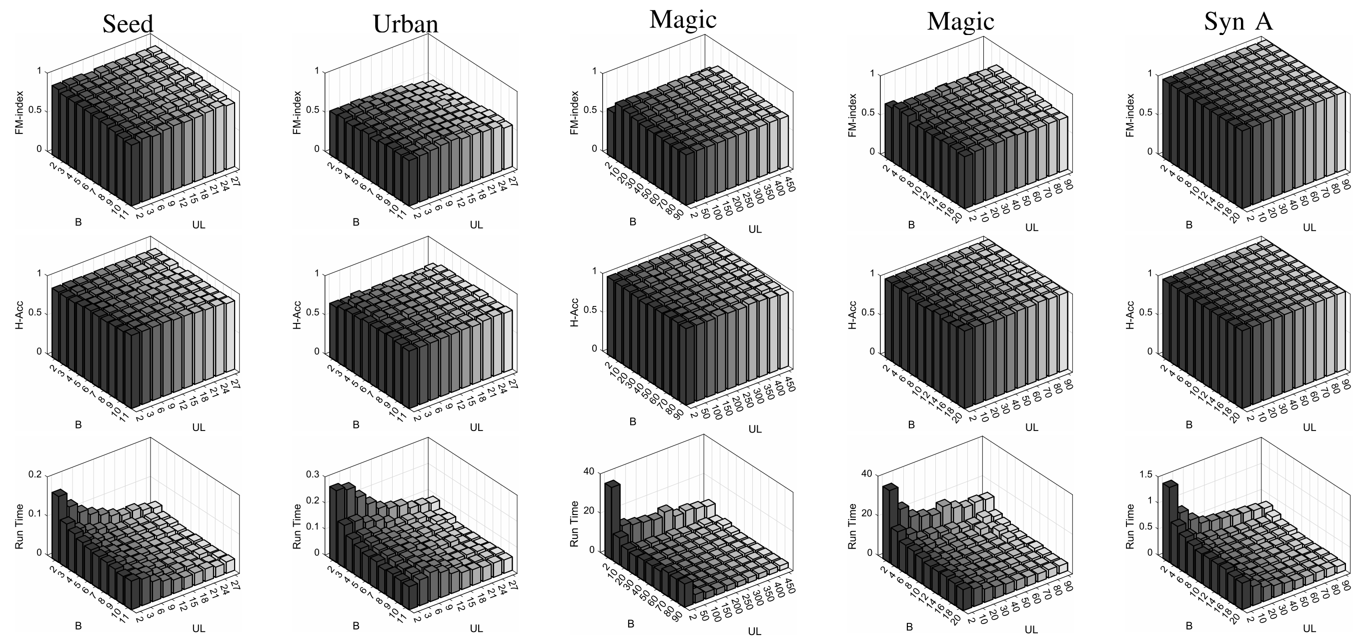


Fig. 10. Performance of GMTT-DL with different B-U_L value combinations on four data sets.

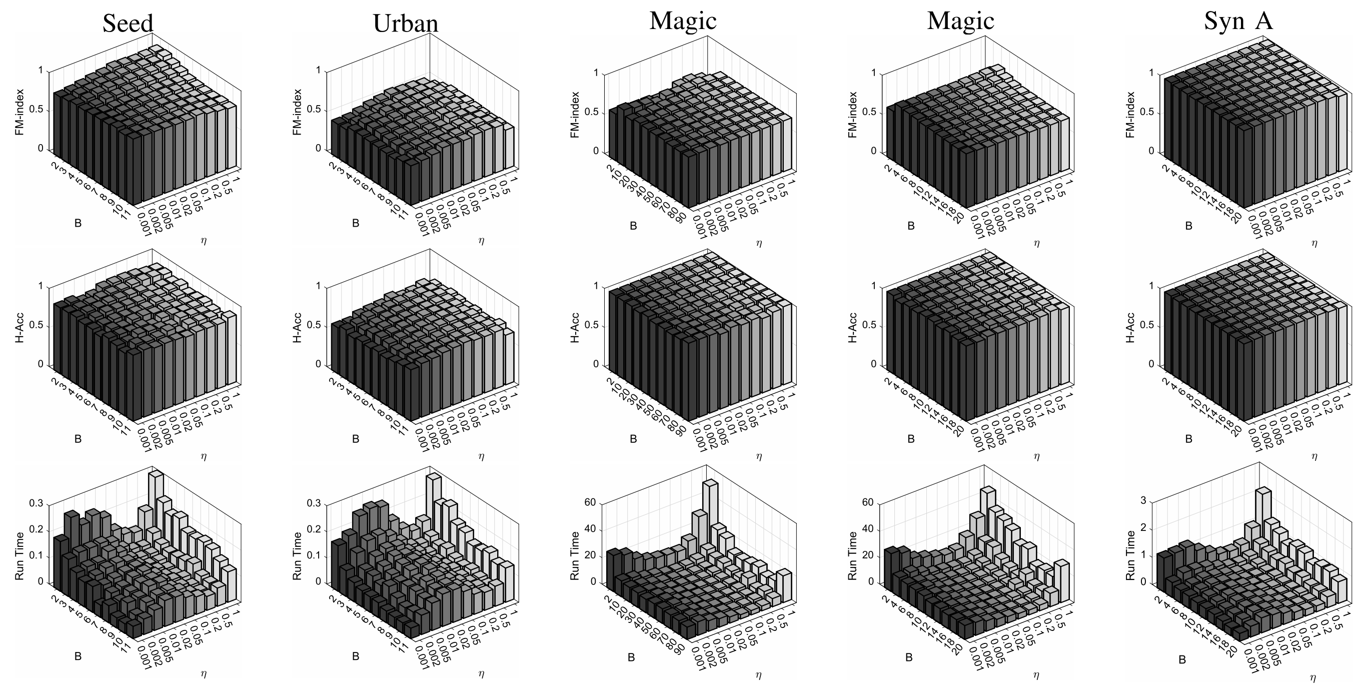


Fig. 11. Performance of GMTT-DL with different B-η value combinations on four data sets.

iterations to converge and will become trapped in local optima.

To experimentally investigate the impact of any pair of the parameters, the proposed GMTT-DL has been performed 10 times for different value combinations of each pair of the parameters on four typical data sets: Seed, which is a real and small data set; Urban, which is a real and high-dimensional data set; Magic, which is a real and large-size data set; and Syn B, which is a synthetic data set with overlapped clusters. For each pair of parameters, the remaining one is fixed. As a rule of thumb, the value of η was set at 0.1 when investigating the impact of the B–U_L relationship. B was set at 4 when evaluating the U_L–η relationship. According to the analysis in Section III-D, U_L was set at $\sqrt{n}$ when studying the B–η relationship. For all of the experiments, $B = 1$ and $U_L = 1$ are not evaluated because they make the GMTT algorithm meaningless. Because the size of the Magic data set is large, the parameters U_L and B are evaluated with large and small spacing steps to better indicate the relationships between parameters. The experimental results of B–U_L, B–η, and U_L–η are presented in Figs. 10–12, respectively. It can be observed that our discussion regarding the

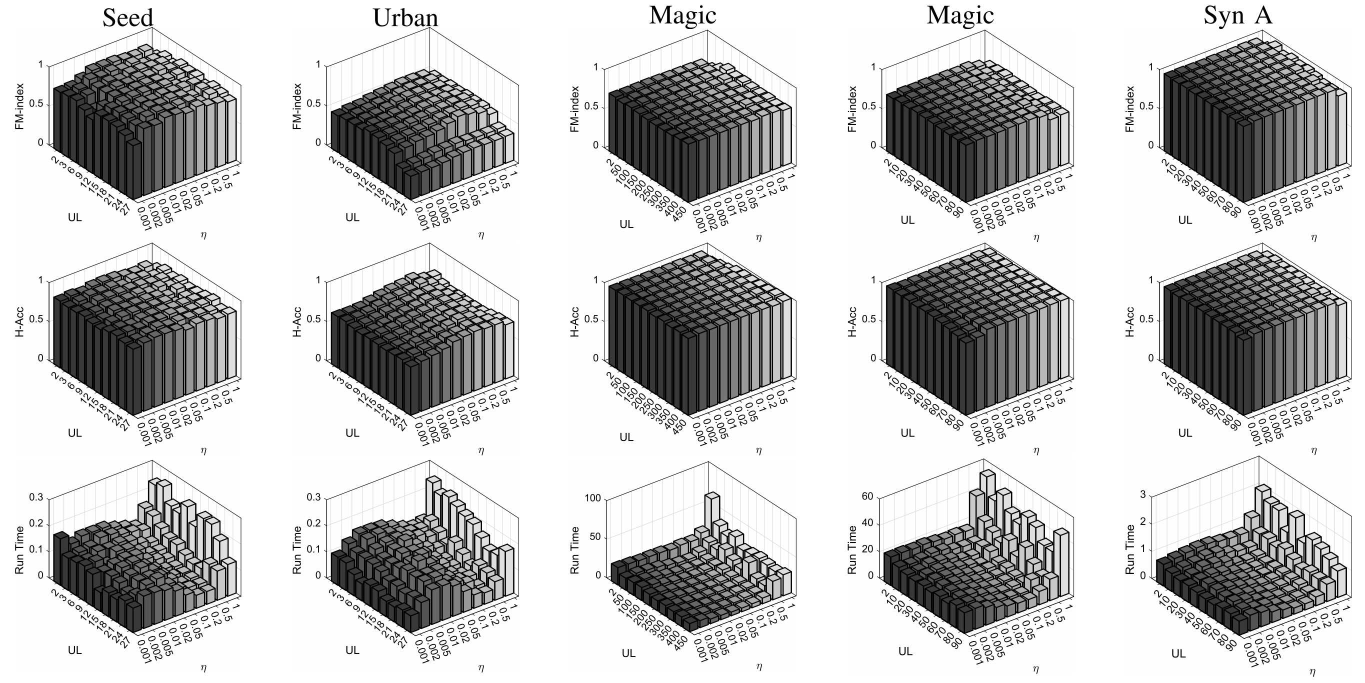


Fig. 12. Performance of GMTT-DL with different U_L-η value combinations on four data sets.

TABLE II
H-ACC OF 8 COUNTERPARTS ON 10 DATA SETS

Approach	Seed	Magic	Occupy	Frogs	Protein	Mice	Urban	Syn A	Syn B	Syn C
GMTT-SL (+)	**0.8581**	**0.9848**	**0.8971**	**0.8071**	0.4977	**0.5051**	**0.7512**	**0.9915**	**0.9644**	0.7443
T-SL	0.6810	0.9788	0.8001	0.6505	**0.5062**	0.4194	0.5595	0.9690	0.9130	**1.0000**
GMTT-AL (-)	0.8471	0.9626	**0.8892**	**0.7781**	0.4684	**0.5305**	**0.7464**	0.9905	0.9682	0.7318
T-AL	**0.8667**	**0.9927**	0.8402	0.7001	**0.4868**	0.4250	0.7262	**0.9930**	**0.9820**	**0.8295**
GMTT-CL (+)	**0.8767**	0.8478	**0.8636**	**0.7685**	**0.4940**	**0.5069**	**0.7304**	**0.9924**	**0.9688**	**0.7441**
T-CL	0.8429	**0.9902**	0.8178	0.7122	0.3951	0.4398	0.5536	0.8970	0.9660	0.7287

TABLE III
FM-INDEX OF 8 COUNTERPARTS ON 10 DATA SETS

Approach	Seed	Magic	Occupy	Frogs	Protein	Mice	Urban	Syn A	Syn B	Syn C
GMTT-SL (+)	**0.8543**	**0.6753**	**0.7130**	**0.6397**	**0.3507**	**0.3270**	**0.5583**	**0.9915**	**0.9425**	0.5513
T-SL	0.3476	0.6487	0.7106	0.6146	0.1746	0.1981	0.2024	0.5010	0.7700	**1.0000**
GMTT-AL (+)	0.8638	**0.7033**	**0.7583**	0.6082	**0.3711**	**0.3255**	**0.5662**	**0.9905**	0.9717	**0.5381**
T-AL	**0.8952**	0.6492	0.7299	**0.6140**	0.1728	0.1722	0.5298	0.5010	**0.9820**	0.4961
GMTT-CL (+)	**0.8833**	**0.6555**	**0.7885**	0.6429	**0.3546**	0.3322	**0.5824**	**0.9924**	**0.8942**	**0.5497**
T-CL	0.8429	0.5294	0.7385	**0.7400**	0.1693	**0.3370**	0.5417	0.8970	0.8090	0.4651

three parameters is confirmed. Moreover, the clustering quality in terms of the H-Acc and FM-index of the GMTT framework is very robust to different parameter value combinations except for some extreme values, e.g., $B = 2$, $U_L = 2$, $\eta = 1$, and $\eta = 0.001$. From the run time results, it can be observed that the run time is the lowest when the value of U_L is approximately $\sqrt{n}$, which also confirms the time complexity analysis in Section III-D. According to the experimental results and the above-mentioned discussion, we set $B = 4$, $\eta = 0.1$, and $U_L = \sqrt{n}$ for all of the data sets in the following experiments.

C. Performance Evaluation of the GMTT Framework

To investigate the effectiveness of the GMTT framework, we have compared its performance with that of the traditional hierarchical clustering framework combined with traditional linkage strategies, i.e., SL, AL, and CL. For each data set, the H-Acc and FM-index were calculated to measure the performance of all the counterparts. Because there are randomization procedures in the GMTT framework, we perform it 10 times and take the average performance as the final result. The experimental results are given in Tables II and III. For each data set, the best result is highlighted via boldface. "+" and "−" beside the GMTT frameworks stand for the Wilcoxon test results. It can be observed that the GMTT framework obviously boosts the performance of SL, AL, and CL on most of the 10 data sets. T-SL outperforms its GMTT version on the Syn C data set because the data distribution type of Syn C is chain shaped, which is preferred by SL. The performance of T-AL is also obviously better than that

TABLE IV
H-ACC PERFORMANCE COMPARED WITH STATE-OF-THE-ART COUNTERPARTS ON 10 DATA SETS

Approach	Seed	Magic	Occupy	Frogs	Protein	Mice	Urban	Syn A	Syn B	Syn C
GMTT-DL	**0.886**±0.02	**0.993**±0.01	0.886±0.05	0.788±0.05	0.500±0.04	0.555±0.05	0.733±0.05	**0.995**±0.00	0.972±0.01	0.781±0.06
GMTT-SL	0.858±0.06	0.985±0.04	**0.897**±0.04	**0.807**±0.02	0.498±0.05	0.505±0.04	**0.751**±0.05	0.992±0.01	0.964±0.01	0.744±0.07
GMTT-AL	0.847±0.06	0.963±0.08	0.889±0.05	0.788±0.04	0.468±0.04	0.530±0.07	0.746±0.05	0.991±0.01	0.968±0.01	0.732±0.05
GMTT-CL	0.877±0.04	0.848±0.13	0.864±0.04	0.769±0.06	0.494±0.03	0.507±0.10	0.730±0.04	0.992±0.01	0.969±0.01	0.744±0.08
P-EL	0.876	0.987	0.810	0.739	0.404	**0.604**	0.637	**0.995**	0.898	0.698
RP-SL	0.681	0.979	0.800	0.650	**0.506**	0.419	0.560	0.969	0.913	**1.000**
RP-AL	0.867	**0.993**	0.840	0.700	0.487	0.425	0.726	0.993	**0.982**	0.829

TABLE V
FM-INDEX PERFORMANCE COMPARED WITH STATE-OF-THE-ART COUNTERPARTS ON 10 DATA SETS

Approach	Seed	Magic	Occupy	Frogs	Protein	Mice	Urban	Syn A	Syn B	Syn C
GMTT-DL	0.886±0.02	**0.732**±0.01	0.740±0.16	0.635±0.06	0.341±0.02	**0.333**±0.02	0.580±0.04	**0.995**±0.00	0.974±0.01	0.571±0.06
GMTT-SL	0.854±0.07	0.675±0.08	0.713±0.16	0.640±0.09	0.351±0.03	0.327±0.02	0.558±0.03	0.992±0.01	0.943±0.09	0.551±0.07
GMTT-AL	0.864±0.03	0.703±0.06	0.758±0.12	0.608±0.09	**0.371**±0.03	0.325±0.03	0.566±0.04	0.991±0.01	0.972±0.01	0.538±0.07
GMTT-CL	0.883±0.03	0.655±0.12	**0.789**±0.13	**0.643**±0.09	0.355±0.02	0.332±0.02	**0.582**±0.06	0.992±0.01	0.894±0.16	0.550±0.08
P-EL	0.890	0.728	0.703	0.615	0.160	0.197	0.202	**0.995**	0.843	0.401
RP-SL	0.348	0.649	0.711	0.615	0.175	0.198	0.202	0.501	0.770	**1.000**
RP-AL	**0.895**	0.649	0.730	0.614	0.173	0.172	0.530	0.501	**0.982**	0.496

of its GMTT version on the Syn B data set because the data distribution type of Syn B is spherical shaped, which is preferred by AL. We can also observe from the experimental results that the performances of different linkages with the GMTT framework are close to each other with competitive performance on most of the data sets. This indicates that the GMTT framework dominates the clustering performance and that different linkage strategies will not obviously influence the performance. In general, the GMTT framework is robust to different linkage strategies and outperforms the traditional one in terms of hierarchy quality.

To verify the effectiveness of the proposed GMTT-DL approach, we have compared its performance with that of all the other linkage strategies with the GMTT framework, i.e., GMTT-SL, GMTT-AL, and GMTT-CL. Moreover, the state-of-the-art hierarchical clustering approaches, i.e., the potential-based framework with edge-weighted tree linkage (P-EL) [23] and the RP-based framework with SL (RP-SL) and AL (RP-AL), have also been compared. To make the comparison fair, we use the autoparameter-selection version of the RP-based approaches. The hierarchy quality of all the counterparts in terms of the FM-index and H-Acc are compared in Tables IV and V, respectively. Because there are randomization procedures in the GMTT framework, the standard deviations of all the linkages with the GMTT framework are also presented. The best and the second best results are highlighted by boldface and underlining, respectively. It can be observed from the experimental results that GMTT-DL outperforms the other counterparts on most of the data sets. Although its performance is not always the best one for all of the data sets, its performance is still competitive. Moreover, almost all of the winners on each data set are GMTT-based approaches, which indicate the effectiveness of the GMTT framework. It can also be observed from the results that GMTT-DL can effectively cope with the overlapping problem since most of the real data sets and the Syn A and Syn B

TABLE VI
WILCOXON SIGNED-RANK TEST BETWEEN GMTT-BASED APPROACHES AND THE OTHER THREE COUNTERPARTS

	H-Acc			FM-index		
Approach	P-EL	RP-SL	RP-AL	P-EL	RP-SL	RP-AL
GMTT-DL	+	+	+	+	+	+
GMTT-SL	+	+	+	+	+	+
GMTT-AL	+	+	-	+	+	+
GMTT-CL	-	+	-	+	+	+

TABLE VII
WILCOXON SIGNED-RANK TEST BETWEEN GMTT-DL AND THE OTHER GMTT-BASED APPROACHES IN TERMS OF H-ACC

Approach	GMTT-SL	GMTT-AL	GMTT-CL
GMTT-DL	+	+	+

TABLE VIII
WILCOXON SIGNED-RANK TEST BETWEEN GMTT-DL AND THE OTHER GMTT-BASED APPROACHES IN TERMS OF FM-INDEX

Approach	GMTT-SL	GMTT-AL	GMTT-CL
GMTT-DL	+	+	+

data sets have overlapped clusters. This is because the topology trained through GMTT extracted the structural distribution information of data sets and the DL considers the global and local distribution information together. In addition, the standard deviations indicate that all of the GMTT-based approaches have stable performance on different data sets. In Table VI, the experimental results of the GMTT-based approaches and the other three state-of-the-art counterparts, i.e., P-EL, RP-SL, and RP-AL, given in Tables IV and V are compared by the Wilcoxon test. From the test results, we can see that GMTT-DL and GMTT-SL are significantly better than all of the other counterparts in terms of H-Acc and FM-index. We also test the significance between GMTT-DL and all of the other GMTT-based approaches in Tables VII and VIII.

TABLE IX
H-ACC PERFORMANCE OF IGMTT-DL AND IHC ON 10 DATA SETS

Approach	Seed	Magic	Occupy	Frogs	Protein	Mice	Urban	Syn A	Syn B	Syn C
GMTT-DL (+)	**0.870**±0.05	**0.981**±0.03	**0.868**±0.05	**0.769**±0.06	0.466±0.05	**0.521**±0.05	**0.751**±0.03	**0.992**±0.01	**0.966**±0.01	0.778±0.04
IHC	0.774±0.11	0.955±0.05	0.862±0.10	0.763±0.10	**0.511**±0.16	0.418±0.09	0.592±0.10	0.905±0.11	0.926±0.02	**0.805**±0.13

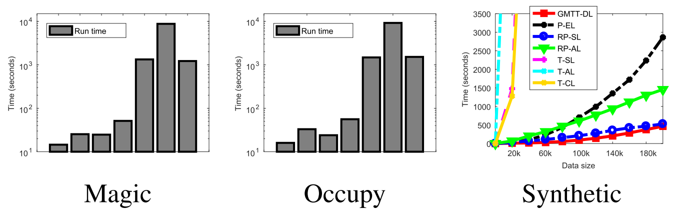


Fig. 13. Run time on the Magic, Occupy, and synthetic data sets. For the Magic and Occupy data sets, the bars from left to right stands for GMTT-DL, P-EL, RP-SL, RP-AL, T-SL, T-AL, and T-CL, respectively.

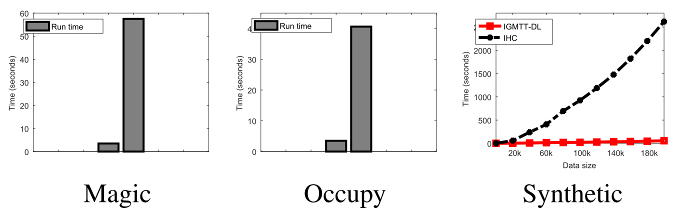


Fig. 14. Run time on the Magic, Occupy and synthetic data sets. For the Magic and Occupy data sets, the left and right bars stand for IGMTT-DL and IHC, respectively.

The results indicate that GMTT-DL significantly outperforms GMTT-SL, GMTT-AL, and GMTT-CL.

To verify the efficiency of GMTT-DL, the run times of all of the counterparts on the two large-scale data sets, Magic and Occupy, are compared in Fig. 13. The run times on each data set are recorded and visualized by histograms for comparison. To better observe the changing orientation of the run time for all of the approaches, we also run all of the counterparts on a synthetic data set with its size increased from 1000 to 200 000 by a step size of 20 000. From Fig. 13, we can observe from the run time of the Magic and Occupy data sets that the proposed approach takes much less time in comparison with all of the other counterparts. According to the run time of the synthetic data set with changing size, we can find that the run times of T-SL, T-AL, and T-CL increase dramatically with the size of the data set. Compared with them, the run times of the four fast hierarchical clustering approaches, i.e., GMTT-DL, P-EL, RP-SL, and RP-AL, increase obviously slower. Although the run time of GMTT-DL remains the smallest on the synthetic data set with sizes from 1000 to 200 000, RP-SL and RP-AL have lower growth rates than GMTT because their time complexity is lower. If the size of the synthetic data set continues increasing, the run times of the RP-based approaches will be smaller than that of the proposed GMTT-DL. However, the hierarchy quality of RP-SL and RP-AL is limited by T-SL and T-AL as discussed in Section I. Therefore, the performance of GMTT-DL is still competitive. Generally speaking, GMTT-DL is very competitive compared to the state-of-the-art counterparts when both the hierarchy quality and processing speed are considered in practical applications.

D. Performance Evaluation of IGMTT Framework

Furthermore, to verify the effectiveness and efficiency of the IGMTT framework, we compared it with another popular IHC method. The two online approaches were also performed on all 10 data sets. Because IHC does not form a binary hierarchy, its FM-index performance cannot be measured. Therefore, the two online approaches are only compared in terms of H-Acc. It can be observed from the experimental results shown in Table IX that IGMTT-DL evidently outperforms IHC on most of the data sets. Because IHC is an approximation of T-SL, it has higher accuracy on the Syn C data set, which is composed of chain-shaped clusters. As discussed in Section III-D, high-dimensional data will influence the performance of the GMTT framework. Therefore, the performance of IGMTT-DL is not better than that of IHC on the Protein data set, which has 77 attributes. According to the standard deviation recorded in Table IX, the performance of IGMTT-DL is obviously more stable than that of IHC on all of the data sets since the clustering procedure of IGMTT-DL is supervised by the topology, which reasonably represents the structural distribution of data sets. For IGMTT-DL and IHC, the significance level of the difference between their performances in terms of H-Acc is also tested through the Wilcoxon signed-rank test. "+" besides IGMTT-DL indicates that the H-Acc performance of IGMTT-DL is significantly better than that of IHC.

To verify the efficiency of IGMTT-DL, its run time is also compared with that of IHC. The experimental settings are the same as for the efficiency verification experiment of GMTT-DL in Section IV-C. From Fig. 14, we can see that both the run time and growth rate of IGMTT-DL are remarkably lower than those of IHC.

In general, IGMTT-DL can incorporate new streaming inputs effectively and efficiently in hierarchical clustering tasks.

V. CONCLUSION

This paper has presented a topology training algorithm, GMTT, which can train a multilayer topological structure for a data set to fit its density distribution. Based on the GMTT algorithm, a hierarchical clustering framework has been designed, featuring lower time complexity and higher clustering quality compared to the existing approaches. The proposed framework can remarkably boost the performance of the existing traditional linkage strategies and has competitive performance when combined with the proposed DL linkage. We have analyzed that the GMTT framework improves the time complexity of hierarchical clustering to $O(n^{1.5})$ without sacrificing the hierarchy quality. Although three parameters should be set, its performance is robust to the parameter settings, which makes it easily utilized in different application domains. Furthermore, its incremental version, IGMTT, has also been proposed to expand its application domain. The IGMTT-based framework has the same time complexity as the GMTT framework but can dynamically update the topology and successively incorporate new inputs to update the corresponding hierarchy structure. Experiments have shown the promising results of the GMTT-DL and IGMTT-DL approaches in comparison with the existing counterparts.

REFERENCES

[1] H. F. Bassani and A. F. Araujo, "Dimension selective self-organizing maps with time-varying structure for subspace and projected clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 458–471, Mar. 2015.

[2] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander, "Data bubbles: Quality preserving performance boosting for hierarchical clustering," in *Proc. ACM SIGMOD Conf.*, 2001, pp. 79–90.

[3] M. M. Breunig, H.-P. Kriegel, and J. Sander, "Fast hierarchical clustering based on compressed data and optics," in *Proc. 4th Eur. Conf. Princ. Data Mining Knowl. Discovery*, 2000, pp. 232–242.

[4] I. Cattinelli, G. Valentini, E. Paulesu, and N. A. Borghese, "A novel approach to the problem of non-uniqueness of the solution in hierarchical clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 7, pp. 1166–1173, Jul. 2013.

[5] Y.-M. Cheung, "k*-means: A new generalized k-means clustering algorithm," *Pattern Recognit. Lett.*, vol. 24, no. 15, pp. 2883–2893, 2003.

[6] Y.-M. Cheung, "A competitive and cooperative learning approach to robust data clustering," in *Proc. IASTED Int. Conf. Neural Netw. Comput. Intell.*, 2004, pp. 131–136.

[7] Y.-M. Cheung, "A rival penalized em algorithm towards maximizing weighted likelihood for density mixture clustering with automatic model selection," in *Proc. 17th Int. Conf. Pattern Recognit.*, vol. 4, 2004, pp. 633–636.

[8] Y.-M. Cheung, "Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 750–761, Jun. 2005.

[9] Y.-M. Cheung, "On rival penalization controlled competitive learning for clustering with automatic cluster number selection," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1583–1588, Nov. 2005.

[10] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucl. Acids Res.*, vol. 16, no. 22, pp. 10881–10890, 1988.

[11] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann, "Time-hierarchical clustering and visualization of weather forecast ensembles," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 831–840, Jan. 2017.

[12] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Statist. Assoc.*, vol. 78, no. 383, pp. 553–569, 1983.

[13] A. Frank and A. Asuncion, "UCI machine learning repository," School Inform. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[14] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Netw.*, vol. 20, no. 8, pp. 893–903, Oct. 2007.

[15] S. Furao, A. Sudo, and O. Hasegawa, "An online incremental learning pattern-based reasoning system," *Neural Netw.*, vol. 23, no. 1, pp. 135–143, Jan. 2010.

[16] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Appl. Statist.*, vol. 18, no. 1, pp. 54–64, 1969.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[18] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[19] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.

[20] H. Koga, T. Ishibashi, and T. Watanabe, "Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing," *Knowl. Inf. Syst.*, vol. 12, no. 1, pp. 25–53, 2007.

[21] A.-A. Liu, Y.-T. Su, W.-Z. Nie, and M. Kankanhalli, "Hierarchical clustering multi-task learning for joint human action grouping and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 102–114, Jan. 2017.

[22] Y. Lu, X. Hou, and X. Chen, "A novel travel-time based similarity measure for hierarchical clustering," *Neurocomputing*, vol. 173, pp. 3–8, Jan. 2016.

[23] Y. Lu and Y. Wan, "PHA: A fast potential-based hierarchical agglomerative clustering method," *Pattern Recognit.*, vol. 46, no. 5, pp. 1227–1239, 2013.

[24] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *Comput. J.*, vol. 26, no. 4, pp. 354–359, 1983.

[25] S. Nassar, J. Sander, and C. Cheng, "Incremental and effective data summarization for dynamic hierarchical clustering," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 467–478.

[26] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, Dec. 2001, pp. 849–856.

[27] M. G. Omran, A. P. Engelbrecht, and A. Salman, "An overview of clustering methods," *Intell. Data Anal.*, vol. 11, no. 6, pp. 583–605, 2007.

[28] S. S. Ray, A. Ganivada, and S. K. Pal, "A granular self-organizing map for clustering and gene selection in microarray data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1890–1906, Sep. 2015.

[29] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, "Automatic extraction of clusters from hierarchical clustering representations," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, 2003, pp. 75–87.

[30] J. Schneider and M. Vlachos, "On randomly projected hierarchical clustering with guarantees," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 407–415.

[31] H. K. Seifoddini, "Single linkage versus average linkage clustering in machine cells formation applications," *Comput. Ind. Eng.*, vol. 16, no. 3, pp. 419–426, 1989.

[32] F. Shen and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," *Neural Netw.*, vol. 21, no. 10, pp. 1537–1547, Dec. 2008.

[33] S. Shuming, Y. Guangwen, W. Dingxing, and Z. Weimin, "Potential-based hierarchical clustering," in *Proc. 16th Int. Conf. Pattern Recognit.*, 2002, pp. 272–275.

[34] D. H. Widyantoro, T. R. Ioerger, and J. Yen, "An incremental approach to building a cluster hierarchy," in *Proc. IEEE Int. Conf. Data Mining*, 2002, pp. 705–708.

[35] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

[36] L. Xu, T. W. S. Chow, and E. W. M. Ma, "Topology-based clustering using polar self-organizing map," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 798–808, Apr. 2015.

[37] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[38] H. Zhang, X. Xiao, and O. Hasegawa, "A load-balancing self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1096–1105, Jun. 2014.

[39] Y. Zhang, Y.-M. Cheung, and Y. Liu, "Quality preserved data summarization for fast hierarchical clustering," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 4139–4146.

[40] Z. Zhang and Y.-M. Cheung, "On weight design of maximum weighted likelihood and an extended EM algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1429–1434, Oct. 2006.

[41] J. Zhou and J. Sander, "Data bubbles for non-vector data: Speeding-up hierarchical clustering in arbitrary metric spaces," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 452–463.

Yiu-ming Cheung (F'18) received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include machine learning, pattern recognition, visual computing, and optimization.

He is a Fellow of IET, BCS, and RSA, and a Distinguished Fellow of IETI. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Pattern Recognition*, and so on.

Yiqun Zhang received the B.Eng. degree from the School of Biology and Biological Engineering, South China University of Technology, Guangzhou, China, in 2013, and the M.Sc. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Computer Science.

His current research interests include machine learning, data mining, and pattern recognition.