Information Sciences 373 (2016) 200–218

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

DenPEHC: Density peak based efficient hierarchical clustering

Ji Xu[a,b,d], Guoyin Wang[c,*], Weihui Deng[b]

[a] School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China
[b] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, 400714, China
[c] Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[d] School of Information Engineering, Guizhou University of Engineering Science, Bijie 551700, China

ARTICLE INFO

Article history:
Received 17 March 2016
Revised 21 August 2016
Accepted 26 August 2016
Available online 31 August 2016

Keywords:
Hierarchical clustering
Density peaks
Grid granulation
Granular computing

ABSTRACT

Existing hierarchical clustering algorithms involve a flat clustering component and an additional agglomerative or divisive procedure. This paper presents a density peak based hierarchical clustering method (DenPEHC), which directly generates clusters on each possible clustering layer, and introduces a grid granulation framework to enable DenPEHC to cluster large-scale and high-dimensional (LSHD) datasets. This study consists of three parts: (1) utilizing the distribution of the parameter γ, which is defined as the product of the local density ρ and the minimal distance to data points with higher density δ in "clustering by fast search and find of density peaks" (DPClust), and a linear fitting approach to select clustering centers with the clustering hierarchy decided by finding the "stairs" in the γ curve; (2) analyzing the leading tree (in which each node except the root is led by its parent to join the same cluster) as an intermediate result of DPClust, and constructing the clustering hierarchy efficiently based on the tree; and (3) designing a framework to enable DenPEHC to cluster LSHD datasets when a large number of attributes can be grouped by their semantics. The proposed method builds the clustering hierarchy by simply disconnecting the center points from their parents with a linear computational complexity $O(m)$, where m is the number of clusters. Experiments on synthetic and real datasets show that the proposed method has promising efficiency, accuracy and robustness compared to state-of-the-art methods.

1. Introduction

Data collection and generation are constantly occurring in our lives. Using particular assumptions to model the datasets under consideration to find the correlations, trends and rules in the data is the purpose of data mining, machine learning and intelligent information processing. Clustering is a general methodology and a rich conceptual and algorithmic framework for data analysis and interpretation that gathers objects into groups [26]. It simultaneously maximizes the similarity between the objects within a group and minimizes the similarity between the objects in different groups.

Clustering can be divided into two categories in terms of the result structure: flat clustering and hierarchical clustering. *Flat clustering* (also called *partition clustering* in some literature) only returns the clusters in one layer, which is efficient and conceptually simple but has some drawbacks. One example is that the number of clusters is too large to make good

* Corresponding author.
E-mail addresses: alanxuch@hotmail.com (J. Xu), wanggy@ieee.org (G. Wang), dengweihui@cigit.ac.cn (W. Deng).

http://dx.doi.org/10.1016/j.ins.2016.08.086
0020-0255/

sense from the perspective of human cognition. According to Miller's "7 ± 2" theory, only 7 objects exist in the span of attention, and the 7 digits in the span of immediate memory [22]. Excessive numbers of clusters do not offer people a good understanding of the data. Another drawback is that some real-world datasets are hierarchically structured in nature, but flat clustering fails to reflect the truth. These limitations can be overcome by *hierarchical clustering*, which outputs a hierarchy and is more informative than flat clustering [21]. Hierarchical clustering conventionally involves two major facets: clustering on one layer and building a hierarchy agglomeratively or divisively. However, our approach belongs to neither of these fashions, instead first selecting the centers of each layer, and then constructing each layer of the hierarchy directly by disconnecting the centers of a layer from their parents in a tree structure.

Hierarchical clustering can be regarded as a method to construct multi-granular information granules for the original finest-grained data from the perspective of granular computing (GrC). *GrC has emerged as one of the fastest growing information-processing paradigms in the domain of computational intelligence and human-centric systems* [40]. *Granular computing is often regarded as an umbrella term to cover theories, methodologies, techniques, and tools that make use of granules in complex problem solving* [41]. One of the basic ingredients of granular computing is granules. Merriam-Webster's Dictionary defines a granule as "a small particle; especially, one of numerous particles forming a larger unit".[1] Examples of a granule include a subset of a set, an equivalent class of a universe, a section of an article, an interval of a universe, and a module of a system [40]. Therefore, an intelligent system can obtain an abstract of or simplified knowledge on massive and complex data by hierarchical clustering, which is significantly beneficial for further processing or problem solving. When plain DenPEHC fails to cluster LSHD datasets, we design a framework by using *grid granulation* (granulating the dataset by first grouping its attributes vertically and then randomly sampling the subsets horizontally into moderately sized grids) to address this issue.

Many works of hierarchical clustering have been published in recent years, which will be further discussed in Section 2.2. This research can be categorized into two classes: one type presents a new method or methodology, and the other type combines or integrates multiple clustering methods. Both classes exhibit some limitations: some are not robust to the shape of the clusters, while others are not efficient enough at forming the resulting hierarchy. Meanwhile, the efficiency and robustness of mining algorithm have been very important recently because of the variety and velocity characteristics of big data.

Recently, Rodriguez and Laio proposed DPClust, which can efficiently and accurately cluster datasets of any shape [32]. This method picks out centers by defining two simple measures: the local density and the distance to the nearest neighbor of higher density. The process of assigning noncenter data points to centers is very efficient (with a time complexity of $O(n)$, where n is the number of objects). DPClust is a flat clustering method that returns one partition on the dataset per run. Additionally, this method interactively selects centers before assigning the objects to clusters, which enables the users to embrace human intuition. However, this method is inconvenient, which makes iteratively calling DPClust prohibitive. Furthermore, this method allows wrong choices of centers.

We improve on DPClust by leveraging the parameter γ (originally proposed to differentiate meaningful datasets from randomly generated datasets) to develop an efficient and robust hierarchical clustering method. This process leads to an approach called density peak-based efficient hierarchical clustering (DenPEHC). This method automatically detects all possible centers and builds a hierarchy presentation for the dataset if it is hierarchically structured in nature. Additionally, the tree structure of the intermediate result in DPClust is revealed, which forms a solid foundation for DenPEHC.

DenPEHC cannot be directly used to process large-scale datasets with high dimensionality, because the $n \times (n-1)/2$-sized distance matrix would be beyond the memory capacity of ordinary personal computers (e.g., a dataset that consists of 100,000 objects requires over 74GB of memory space if a tuple (object1, object2, distance) takes 16 Bytes), and high dimensionality would create a distance concentration problem [1]. We propose a general framework that is based on grid granulation to enable ordinary clustering methods to handle large-scale and high-dimensional datasets.

Finally, we demonstrate the effectiveness and high efficiency of our method on synthetic datasets and real world datasets.

The rest of the paper is organized as follows: Section 2 briefly discusses related works. In Section 3, we describe the proposed algorithm DenPEHC, including some related discussion. Section 4 presents a general framework that uses grid granulation to enable DenPEHC (or other distance-based clustering methods) to cluster big data. Section 5 describes the datasets and the experimental settings, shows the experimental results, and presents some analysis and discussion on the results. Conclusions are provided in Section 6.

2. Related works

2.1. Clustering with density peaks

Our approach is mainly based on [32], so we provide a brief introduction to this paper's idea and algorithm here. The authors first made a sound intuitive assumption that centers are always surrounded by non-center data points with lower density, no matter the shape of the clusters, and that the distance between two centers is relatively long. Then, two simple measures, namely, the local density (denoted as ρ) and the minimal distance to data points with higher density (denoted as δ), are employed to accomplish the clustering task.

[1] http://www.merriam-webster.com/.

Table 1
Notations in DPClust and DenPEHC.

Symbol	Meaning
$X = (x_1, \ldots, x_N)$	The dataset with x_i as its ith data point
$D = \{d_{i,j}\}$	The distances of the pairs of data points in X, where $1 \le i < j \le N$
$\mathbf{I} = (1, \ldots, N)$	The set of the indices of data points in the dataset
$\boldsymbol{C} = (C_1, \ldots, C_K)$	K centers of the clustering result
$\boldsymbol{\rho} = (\rho_1, \ldots \rho_N)$	The local density of X
$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_N)$	The distance to nearest points of higher local density
$\boldsymbol{Q} = (q_1, \ldots, q_N)$	The sorted indices of $\boldsymbol{\rho}$ in descending order i.e., $\rho_{q_1} \ge \rho_{q_2} \ge \cdots \ge \rho_{q_N}$
$\boldsymbol{Nn} = (Nn_1, \ldots, Nn_N)$	The indices of the nearest neighbor with larger ρ for data points series(x_1,..., x_N)
$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_N)$	The elementwise product of $\boldsymbol{\rho}$ and $\boldsymbol{\delta}$
$\boldsymbol{Cl} = (Cl_1, \ldots, Cl_N)$	The cluster labels of X

The notations that are used to describe this algorithm and our method are listed in Table 1.
The algorithm of DPClust takes the distance matrix of a given dataset as input, and performs the following steps:

(1) Compute $\{\rho_1, \rho_2, \ldots, \rho_N\}$ via a cut-off kernel:

$$\rho_i = \sum_{j \in I \setminus \{i\}} \chi(d_{i,j} - d_c), \ \textit{where} \ \chi(x) = \begin{cases} 1, & x < 0; \\ 0, & x \ge 0. \end{cases} \tag{1}$$

where d_c is the cutoff distance; or via a Gaussian kernel:

$$\rho_i = \sum_{j \in I \setminus \{i\}} e^{-\left(\frac{d_{i,j}}{d_c}\right)^2}. \tag{2}$$

Eq. (2) is used in the authors' implementation;
(2) Sort $\{\rho_1, \rho_2, \ldots, \rho_N\}$ in descending order to obtain $\boldsymbol{\rho}$;
(3) Compute Δ via

$$\delta_{q_i} = \begin{cases} \min\limits_{j<i} \{d_{q_i,q_j}\}, & i \ge 2; \\ \max\limits_{j \ge 2} \{d_{q_i,q_j}\}, & i = 1. \end{cases} \tag{3}$$

and write the index of the nearest neighbor with larger ρ in the vector $\boldsymbol{Nn}$:

$$Nn_i = \begin{cases} 0, \ if \ i = q_1 \\ j \ such \ that \ \delta_i = d_{i,j}, \ otherwise \end{cases} \tag{4}$$

(4) Interactively choose the points with "anomalously large" ρ and δ as centers;
(5) Assign each data point to the same cluster as its nearest neighbor with larger ρ. Initially, the centers are assigned to their corresponding cluster label, and then each non-center object x_i is assigned to the same cluster as Nn_i. Formally, this approach is written as follows:

$$Cl_i = \begin{cases} k, \ if \ i = C_k, k \in \{1, \ldots, K\} \\ Cl_{Nn_i}, \ otherwise \end{cases} \tag{5}$$

The authors of DPCLUST stated in their paper "*for large data sets, the results of the analysis are robust with respect to the choice of* d_c" [32]. This statement is true, especially when a Gaussian kernel is used. However, if the datasets are small or moderately sized, the choice of d_c must adapt to the distribution of the data to achieve good clustering accuracy. The heuristic rules of assigning d_c and the sensitivities regarding d_c and other parameters will be discussed in Section 5.2.5.

DPCLUST uses a parameter called *bord_rho* to distinguish *core* and *hallo* data points in a cluster. The hallo data points around a cluster are somehow similar to but are not actually outliers. Whether a point is a hallo point depends heavily on the choice of the d_c parameter. For simplicity, we omit discussion of hallo and core in this study.

2.2. Hierarchical clustering and multi-granular computing

Newly proposed hierarchical clustering methods can be categorized into two types. One type presents a new method or methodology. Bouguettaya et al. improved the efficiency of agglomerative hierarchical clustering by building a hierarchy that was based on a group of centroids rather than raw data points [4]. Frank de Morsier proposed a cluster validity measure (CVM) to describe the distribution of core points and outliers of clustering results that were obtained from different hierarchical clustering methods and used this metric to help build a proper clustering hierarchy [8]. Santos et al. presented a

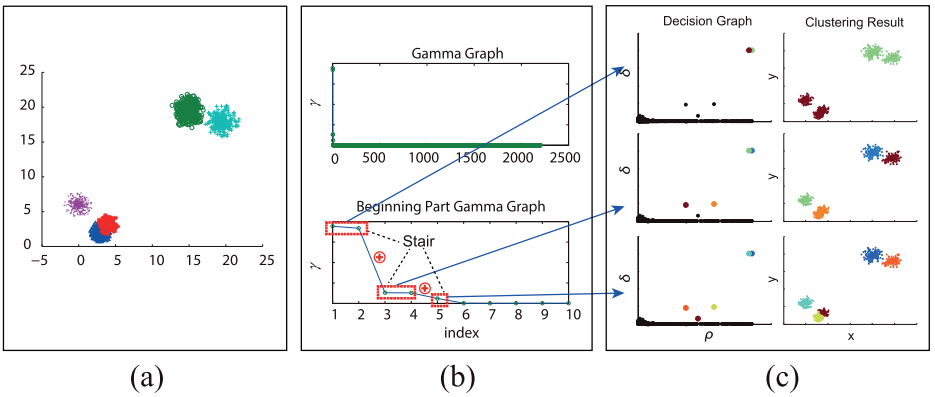

Fig. 1. (a) The dataset may be regarded as 5, 4, or 2 clusters. (b) γ curve of the dataset with the DPClust algorithm. The upper plane is the γ values (sorted in descending order) of the entire dataset and the lower plane is the 10 largest γ values. Five centers were selected, among which 2 were largest, 2 were relatively small, and one was smallest w.r.t their γ values. (c) The dataset is clustered with 3 sets of centers that correspond to the three levels of γ values of potential centers.

new clustering model by using subgraphs, which are constructed based on an entropic proximity matrix, to reflect the ideal local structure of the dataset. The overall approach to form a clustering hierarchy is agglomerative [34]. Shao et al. proposed a new clustering framework that explores the potential of the Extensive Kuramoto Model (a model that concisely describes the dynamical synchronization process), which can produce the final partition clustering result and record the intermediate results to build a hierarchical structure [35]. Qin et al. extended the gain ratio in the C4.5 algorithm [28] to a mean gain ratio to select attributes when clustering categorical data. In each round, the newly determined cluster is characterized by the lowest entropy, and the clustering hierarchy is constructed in a divisive manner as the iteration progresses [27]. Tang and Zhu addressed four important issues of hierarchical clustering based on fuzzy proximity on granular space and presented a strict mathematical analysis on their proposed algorithm's methods. The conclusions were mainly illustrated on numerical examples rather than real datasets [36]. The cohesion-based self-merging algorithm uses the metric *cohesion* to measure the distance between clusters. This method first forms small subclusters from a dataset with the standard K-means method and then iteratively merges the subclusters with highest cohesion [20]. Knowles and Ghahramani proposed Pitman Yor diffusion trees for Bayesian hierarchical clustering [17], which generalizes the Dirichlet Diffusion Tree [24].

The other type involves combinations and ensembles. Mirzaei et al. presented an algorithm called MATCH to combine multiple dendrograms (the presentation of a hierarchical clustering result) into one unit. This combination is performed by computing the min-transitive closure of the similarity matrices that correspond to the dendrograms to reach an aggregating matrix [23]. Rashedi and Mirzaei proposed a Bob-Hic algorithm to improve the performance of hierarchical clustering, whose key components involve computing the boosting values and updating the weights for objects [30].

A cluster can be taken as an information granule from the perspective of Granular Computing (GrC). In addition to hierarchical clustering, the multi-granular representation (MGrR) of a given dataset can be constructed by means of rough set, fuzzy set and quotient space, and so on [40]. Once an MGrR is built, many potential analyses and mining processes can be further conducted. For example, some complex problems in the context of big data can be solved on a coarser granular layer, thus offering an *effective solution* rather than a precise solution, which may violate time constraints [39]. Recently, GrC has been listed as one of the fundamental techniques and technologies to analyze big data [7].

3. DenPEHC method

We present a study of DenPEHC in this section. Section 3.1 presents the key observations in DPClust and an overview of our method. The method and corresponding algorithm of selecting centers and deciding the centers in each layer are described in Section 3.2. The tree structure of the intermediate result in DPClust is discussed in Section 3.3. We develop the DenPEHC algorithm in Section 3.4 based on the selection of centers, identification of stairs, and analysis of tree structure. Outlier detection with DenPEHC is addressed in Section 3.5, and the computational complexity is analyzed in Section 3.6.

3.1. Key observations in DPClust and an overview of our approach

When applying the DPClust algorithm to cluster datasets, the centers take γ value (defined as $\gamma_i = \rho_i \times \delta_i,\ 1 \le i \le N$ in [32]) on different levels if the dataset can be clustered hierarchically. Fig. 1 illustrates this key observation.

Note: *Potential centers are all the centers that correspond to the finest clustering. Some centers will be "degraded" as noncenter data points on the coarser layers, hence the name "potential centers".*

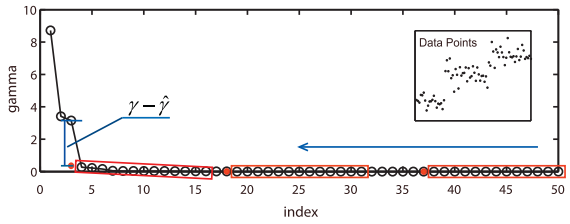

Fig. 2. Diagram to illustrate the idea of selecting centers with linear fitting. The black circles are the real γ values, and a red dot is the predicted γ value of the data point before the points that are currently being linearly fitted. If a significant "jump" occurs from the predicted γ value to the real γ value at position i, then the points whose γ values are larger than or equal to γ_i are chosen as cluster centers. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Based on this observation, our method first selects all the potential centers and then determines the stairs in the γ curves of the centers. Finally, the centers on each layer are used to transform the leading tree (see Section 3.3) into a forest, in which each subtree represents a cluster.

The following three problems must be solved to design the DenPEHC algorithm:

(i) Select all the potential centers and identify the stairs in the γ curve to decide layers in the hierarchy;
(ii) Analyze the underlying tree structure of the intermediate result $\boldsymbol{Nn}$;
(iii) Construct the resulting hierarchy by using the stairs and the tree.

3.2. Determining the centers and stairs

(i) Some works that automatically select cluster centers have been published, e.g., Hinneburg and Kim used a hill climbing approach to identify the local maxima of a density function that correspond to the cluster centers [13]. Hinneburg and Garbriel further progressed toward a faster DENCLUE (called DENCLUE 2.0) by reducing the hill climbing to a special case of an expectation maximization problem [12]. We use a linear fitting approach with DPClust to automatically select the centers because the cluster centers are characterized by their anomalously large γ values [32].
Let $[\boldsymbol{\gamma^s}, \boldsymbol{\gamma Ind}] = sortDescending(\boldsymbol{\gamma})$. For $i = N - l$ to 0 (from the end to beginning), we fit the γ points against their indices to a linear equation with length l:

$$\boldsymbol{\gamma}_i^s = a_i \boldsymbol{I}_i + b_i, \tag{6}$$

where $\boldsymbol{\gamma}_i = (\gamma_{i+1}, \gamma_{i+2}, \ldots, \gamma_{i+l})$, $\boldsymbol{I}_i = (I_{i+1}, I_{i+2}, \ldots, I_{i+l})$. The two variables a_i and b_i are solved to reach a minimal mean square error (MSE).
Then, we estimate $\hat{\gamma}_i = a_i I_i + b_i$ and denote $\Delta\gamma = \gamma_i - \hat{\gamma}_i$. When the first i satisfies

$$\Delta\gamma_i > LocalR * \mathrm{Max}(\boldsymbol{d\gamma}_i^s),$$

and

$$\Delta\gamma_i > GlobalR * \mathrm{Max}(\boldsymbol{d\gamma}^s),$$

the procedure of detecting centers terminates, where $\boldsymbol{d\gamma}_i^s$ and $\boldsymbol{d\gamma}^s$ are the difference vector of $\boldsymbol{\gamma}_i^s$ and $\boldsymbol{\gamma}^s$, respectively. *LocalR* and *GlobalR* are two parameters that control how significantly the value of a center should grow larger than the previous data points (with regard to the direction of linear fitting movement, "previous" is on the right side) and how large the ratio of $\Delta\gamma$ to $\mathrm{Max}(\boldsymbol{d\gamma}^s)$ should be to select the first center, respectively. *GlobalR* is meant to prevent a data point from being selected as a center when its $\Delta\gamma_i$ is very large relative to $\boldsymbol{d\gamma}_i^s$ but tiny relative to $\boldsymbol{d\gamma}^s$ (the maximal increment in the entire $\boldsymbol{\gamma}$).
When the linear fitting process terminates, the number of centers is i and the points with indices in $(\gamma Ind_1, \ldots, \gamma Ind_i)$ are the corresponding centers. This method is illustrated in Fig. 2 and described in detail as Algorithm 1.

(ii) Identifying the stairs in the γ curve to decide layers in the hierarchy
Stairs are identified by using the ratio of the current γ value increment to the previous γ value increment. Formally, let R_i be the increment ratio at γ_i point, then

$$R_i = \gamma_i - \gamma_{i-1}, \quad 2 \le i \le K, \tag{7}$$

where K is the number of all selected centers.

Definition 1. The series of γ points $(\gamma_i, \ldots, \gamma_{i+m-1})$ that correspond to m centers is called a stair if it satisfies

$$\frac{R_{i+m}}{R_{i+m-1}} > StairThre \ \ and \ \ \frac{R_{i+l}}{R_{i+l-1}} \le StairThre \tag{8}$$

Algorithm 1: LF-ChooseLarge.

Input: a vector $\boldsymbol{V}$ sorted in descending order, *LocalR*, *GlobalR*
Output: *LN* (the number of anomalously large elements in $\boldsymbol{V}$)

Len=length($\boldsymbol{V}$);
$\boldsymbol{dV} = \{V_i - V_{i+1}\}_{i=1}^{Len-1}$;
AllMaxDiff=max($\boldsymbol{dV}$);
FitLen=20;
for *i=Len-FitLen; i≥2; i=i-1* **do**
  $\boldsymbol{CV} = \{V_j\}_{j=i+1}^{i+FitLen}$;
  $\boldsymbol{Ind} = \{i+1, \ldots, i+\text{FitLen}\}$;
  $[a, b] = \text{LinearFit}(\boldsymbol{Ind}, \boldsymbol{CV})$;
  $PredictV = a \times i + b$;
  $CDiff = V_i - PredictV$;
  $\boldsymbol{dCV} = \{CV_j - CV_{j+1}\}_{j=1}^{FitLen-1}$;
  $CMaxDiff = max(\boldsymbol{dCV})$;
  if $CDiff > LocalR \times CMaxDiff$ **AND** $CDiff > GlobalR \times AllMaxDiff$ **then**
    break;
  end
end
$LN = i$;
return *LN*;

for $1 \le l < m,\ 1 \le m \le K - i + 1$, where *StairThre* is a threshold value that is used to identify the "riser" of a stair. The concept of stairs begins to make sense only if at least three centers exist.

That is, a steep decrease must occur at the beginning of a stair, and the decrease in γ within the stair is gentle. This perspective matches common sense, in which the slope is gentle on the stair tread and abrupt at the riser. To highlight the stairs of γ values that are related to the centers, the first $2 \times k$ values of γ are picked out and shown in an additional figure, where k is the number of selected centers.

In our algorithm, the centers that correspond to the γ points in a given stair S and all stairs that are higher than S lead to the construction of the layer that corresponds to S (the meaning of the red symbol $\oplus$ in Figs. 1b and 4b). This definition avoids trivial clustering, that is, the largest γ point alone does not form a stair. However, the lowest γ point alone may form a stair (as in Fig. 4b).

The detailed steps of finding stairs with Definition 1 in the γ curve of the centers is described in Algorithm 4.

3.3. Leading tree in DPClust

As long as the centers and stairs are determined, the clustering hierarchy can be obtained in a straightforward manner, which is called straightforward Automatic Hierarchical clustering based on Density Peaks (AHDPClus_s). The dataset is first clustered with the centers that correspond to the highest stair $Stair_1$ to compute the highest layer of the hierarchy, and then the centers of $Stair_2$ are added to $Stair_1$ to compute the second layer of the hierarchy. This process continues until all the stairs are included to construct the hierarchy in a top-down fashion. The process in the opposite direction (bottom-up) can also produce the same result. However, this approach is apparently inefficient because the intermediate results are not fully exploited.

Careful investigation reveals that the intermediate result *Nn* in DPClust essentially represents a tree. In this tree, each node except the root is led by its parent to join the same cluster. Thus, we call this tree a *Leading Tree* (LT). With an LT, the process of assigning the non-center data points turns into disconnecting the centers from their parents. The resulting subtrees represent the clusters. Example 1 illustrates the idea.

Example 1. Clustering 13 2D points (named DS1, shown in Fig. 3a) with DPClust and LT representation.

The intermediate results *Nn*, *SortGamma* and *SortGammaInd* for DS1 and the final result *ClusterLabel* for DS1 are computed with DPClust, as shown in Table 2.

The LT of Example 1 is as Fig. 3b, and it is split into 3 subtrees after the three points with greatest γ values have been selected as centers (as shown in Fig. 3c). Each subtree is a cluster, which is corresponding to the CL row in Table 2.

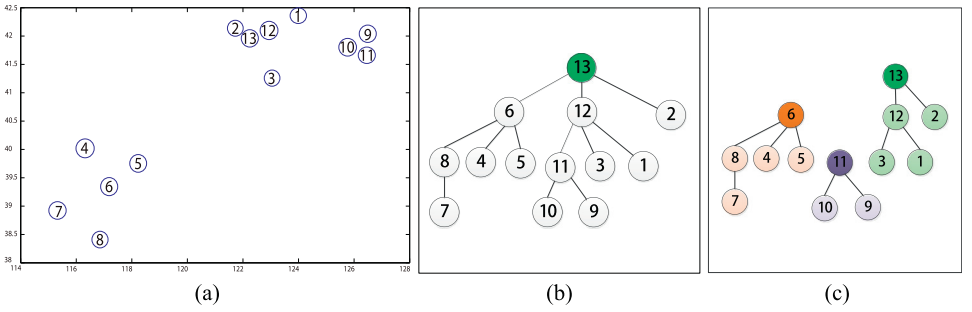

Fig. 3. Illustration of a Leading Tree: (a) DS1 dataset (b) logical LT structure, and (c) clustering result.

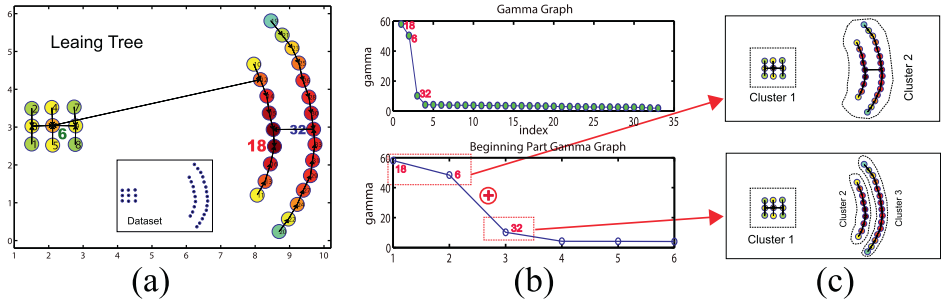

Fig. 4. Illustration of DenPEHC's main steps. The parameter configuration is (*percent* = 20, *LocalR* = 4.6, *GlobalR* = 0.01, *StairThre* = 1.8).

Table 2
(Intermediate) Results in Example 1.

Nn	12	13	12	6	6	13	8	6	11	11	12	13	0
SortGamma	5.89	2.26	1.35	0.56	0.43	0.42	0.39	0.30	0.29	0.22	0.16	0.12	0.05
SortGammaInd	13	6	11	3	12	1	8	4	2	7	10	5	9
ClusterLabel	1	1	1	2	2	2	2	2	3	3	3	1	1

3.4. DenPEHC algorithm

After determining the centers and stairs, one can easily build the hierarchy in divisive or agglomerative fashion. However, these approaches are not efficient enough because we must still determine the relationships between the centers on adjacent layers.

The process of building a clustering hierarchy becomes straightforward with the leading tree structure as discussed above. The steps of DenPEHC are described in Algorithm 2, which mainly include computing *Nn*, transforming *Nn* into an LT,

Algorithm 2: DenPEHC.

Input: the distance matrix D of the data points, d_c, *LocalR*, *GlobalR*, *StairThre*
Output: the clustering hierarchy in the form of l clustering solutions

1 Use D and d_c to compute $\boldsymbol{\rho}$ by Formula (2);
2 Compute $\boldsymbol{\delta}$ and ***Nn*** by Formula (3) and Formula (4) respectively;
3 LT=TransormLT(Nn); // Algorithm 3 $\boldsymbol{\gamma}$= elementwise product of $\boldsymbol{\rho}$ and $\boldsymbol{\delta}$;
4 [$\boldsymbol{\gamma Sort}$, $\boldsymbol{\gamma SortInd}$]=sortDescend($\boldsymbol{\gamma}$);
5 *centersCnt*=LF-ChooseLarge($\boldsymbol{\gamma Sort}$, *LocalR*, *GlobalR*); //Algorithm 1
Centers $= \{\gamma SortInd_i\}_{i=1}^{centersCnt}$;
6 ***HierarEnd***=DetectStairs($\{\gamma Sort_i\}_{i=1}^{centersCnt}$, *StairThre*); // Algorithm 4
l= length(***HierarEnd***);
7 **for** i=1 *to* l **do**
8 Stairs[i]= $\{Centers_i\}_{i=1}^{HierarEnd_i}$;
9 *ClusSolution*[i]=SplitLT(LT,Stairs[i]) // Algorithm 5
10 **end**
11 **return** *ClusSolution*;

figuring out the centers and stairs, and finally constructing each layer of the hierarchy by splitting the LT into a forest (a set of subtrees).

In Algorithm 3, the children of each non-leaf node are added in descending γ value order for the purposes of quick

Algorithm 3: TransormLT.

Input: *Nn* and *SortedGammaInds*
Output: An LT in the form of adjacent list *LT_AL*

```
Initialize the adjacent List for each object;
for i = 2 to N do
    ChildID = SortedGammaInds[i];
    ParentID = Nn[ChildID];
    LT_AL[ParentID].add(ChildID);
end
return LT_AL;
```

Algorithm 4: DetectStairs.

Input: $\{\gamma Sort_i\}_{i=1}^{centersCnt}$, *StairThre*
Output: ***HierarEnd*** stores every center of minimal γ value on each layer.

```
LayerCnt=0;
PrevSlope = γSort_2 − γSort_1;
for i=2:centersCnt-1 do
    slope = γSort_{i+1} − γSort_i;
    if slope/PrevSlope > StairThre then
        LayerCnt++;
        HierarEnd[LayerCnt] = i;
    end
    PrevSlope = slope;
end
return HierarEnd ;
```

disconnection operation in Algorithm 5.

Algorithm 5: SplitLT.

Input: *LT_AL*, *Nn*, *Centers*[*m*] sorted by γ value in descending order
Output: A forest to represent the clustering result

```
for i=2 to m do
    root = Centers[i] ;
    parentID = Nn[ root];
    LT_AL [parentID].RemoveFirst();
end
return LT_AL;
```

With an LT constructed, the process of assigning non-center points to m centers becomes disconnecting the $m-1$ links from each center to its parent (the center with the largest γ value is the root of the entire LT and thus does not require disconnection). See Algorithm 5.

To make the DenPEHC algorithm easier to follow, we illustrated its main steps on a toy 2D dataset in Fig. 4. As in Fig. 4a, the LT of the 2D dataset is constructed after the vectors $\boldsymbol{\rho}$ and ***Nn*** are computed. The colors of the nodes represent the local densities (the warmer the color, the larger the local density). Subsequently, the centers are selected via a linear fitting approach, and the stairs are detected (Fig. 4b). Intuitively, we can see a "jump" in the γ value at the data point x_{32}. Therefore, x_{32} and the points with even greater γ (x_{18} and x_6) are chosen as centers. Among the 3 centers, x_{18} and x_6 form a "stair" with higher γ value. Point x_{32} forms the second stair. In Fig. 4c, the constructed LT and the stair in the γ curve of the centers are used to directly obtain the result of hierarchical clustering. The first stair has two centers: x_{18} and x_6. x_{18} is the root of the entire LT and thus has no parent to disconnect from. Thus, the top layer of the clustering is achieved by only disconnecting point x_6 from its parent x_{12}. For the second stair, the corresponding set of centers is x_{18}, x_6, x_{32}, so the clusters are determined by disconnecting the elements (except the root)from their parents. That is, the clustering on each layer can be independently obtained, which is different from traditional divisive or agglomerative approaches.

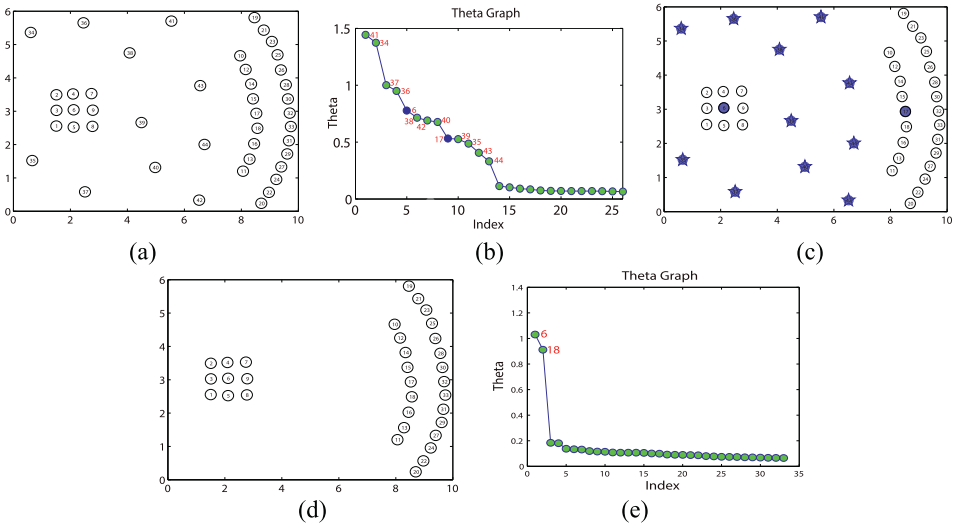

Fig. 5. Outlier detection with the parameter θ (parameters' configuration: *percent* = 20, *LocalR* = 4.6, *GlobalR* = 0.01). (a) Dataset of 33 normal objects and 11 outliers. (b) Beginning part of the theta graph, with which the linear fitting approach selected 13 objects of anomalously large θ. (c) The objects that were selected in (b) is marked by colored faces, and the ultimate outliers are marked by pentagrams after x_6 and x_{17} were excluded for their relatively large ρ. (d) Dataset without noise. (e) Theta graph of the dataset in (d). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.5. Outlier detection

Outlier detection has a long history because data are collected or generated from a variety of sources. This process distinguishes and removes anomalous samples from datasets [14]. Recently, Milos et al. defined *antihubs* based on reverse nearest neighbors and *hubs* to detect outliers, especially in high-dimensional settings [29]. Huang et al. combined the idea of nearest neighbors and reverse nearest neighbors to propose a concept of *natural neighbors*, in which *natural values* and *natural outlier factors* are computed to select outliers [15].

With DPClust, outliers are characterized by large δ and small ρ [32] and thus can be detected by the ratio of δ to ρ. Similar to the mechanism of selecting centers, we define a parameter $\theta = \delta/\rho$ to indicate how likely a data point would be a potential outlier. This approach can be easily implemented in the context of DenPEHC because the parameters ρ and δ are readily available for every object and the routine of selecting the anomalously large value is ready for calling (Algorithm 1). However, detecting noises has one difference from selecting centers, that is, the data points with very large θ might have an extraordinarily large δ and a relatively large ρ. A point with large ρ is surrounded by a group of objects and hence is by no means an outlier. After the first step of choosing data points with anomalously large θ, a second step is needed to filter points with relatively large ρ. After these two steps, the remaining points are chosen as outliers and should be removed before the DenPEHC algorithm begins running.

This outlier detection is demonstrated with two examples. First, 33.3% noise points are added to the dataset in Fig. 4a, and we show that our method can accurately identify the outliers. This result is illustrated in Fig. 5a–c.

Second, we test the outlier detection method on the original dataset without noise checking whether a false alarm would be triggered. As in Fig. 5e, the linear fitting approach finds two points (x_6 and x_{18}) of very large θ values with a theta graph of the dataset in Fig. 5d. However, both the two are excluded from the set of potential outliers in the second step of filtering data points with relative large ρ, because of their high local density. Thus, this method finds no noise when no noise actually exists.

3.6. Complexity analysis and comparative discussion

The merits of DenPEHC include its strong adaptability in terms of the shape of data, competitive accuracy, and low computational complexity. The adaptability and accuracy are inherited from DPClust, while the low computational complexity is partly attributed to the efficient hierarchical clustering algorithm. Algorithm 2 states that the additional complexity for constructing the entire hierarchy based on the LT is actually $O(m)$ and is significantly lower than that of state-of-the-art methods. For example, this additional time complexity for the CSM algorithm is $O(m^2 \log m)$, where m is the number of subclusters [20], and the complexity of the *hSync* algorithm is $O(L \cdot T \cdot N \log N \cdot d)$, where L is the number of different clustering models, T is the time evolution, N is the number of objects, and d is the dimensionality [35]. We will compare DenPEHC to other methods with the running time in the experimental evaluation section.

The space complexity of DenPEHC is the same as DPClust in magnitude. This method only requires $3 \times N$ additional units to store the LT and $2 \times N$ units to store the sorted γ values and the indices of the objects in γ-descending order.

DenPEHC is more efficient at finding hierarchical clustering than many other competing methods, but the underlying reason is that the model is heuristic and ad-hoc, which avoids iteratively finding the "best" clustering to optimize a given criterion. However, this heuristic and ad-hoc nature also makes the performance lack rigorous mathematical guarantees. By contrast, many of the existing hierarchical clustering methods have a solid theoretical foundation to ensure the clustering meets the corresponding criteria (although at the cost of higher time complexity). For example, *sync* optimizes the Minimal Description Length (MDL) [9] to achieve a meaningful hierarchy of the clustering; entropy-based clustering [19], from which LEGClust partially originated, minimizes the entropic summation of all clusters on a layer to guarantee a satisfactory result; Bayesian hierarchical clustering [11] merges two subsets, which are most likely generated by the same probabilistic distribution, thus maximizing the corresponding conditional probability.

Another drawback of DenPEHC that was inherited from DPClust is that this method may have different similarity matrices for the same dataset depending on the distance metric that is chosen, and the clustering result is directly induced by the similarity matrix. However, the entropy-based clustering and Bayesian clustering do not rely on distance but on the probabilistic distribution of the objects instead. Thus, these methods entirely avoid the issue of choosing a distance metric.

4. Enabling DenPEHC to cluster large-scale and high-dimensional datasets

4.1. Clustering LSHD data

As mentioned before, DenPEHC cannot be directly used to cluster large-scale and/or high-dimensional dataset because of the tremendous distance matrix and/or the distance concentration effect. Usually, these two problems can be dealt with independently.

When clustering datasets with high dimensionality, one can directly adopt subspace clustering methods to find an appropriate subspace, which is oriented either parallel to the axis or arbitrarily, and then cluster the objects that were mapped to this subspace [25] or first perform dimensionality reduction (e.g., by locally linear embedding [33] or by Laplacian eigenmaps [2]) and then cluster the derived dataset with lower dimensionality. Using existing dimensionality reduction methods to preprocess the data or integrating this subspace clustering with DenPEHC would diverge from the major objective of this research. Therefore, we instead employ a semantical grouping on the selected attributes (the irrelevant attributes are removed) and then iteratively call Auto-DPClust to map each group of numerical attributes to one categorical attribute (for more details, the readers are referred to Algorithm 6).

Algorithm 6: DenPEHC-LSHD algorithm.

Input: LSHD Dataset X, Group Configuration $\{G_1, \ldots, G_R\}$, SN_A, SN_T
Output: The clustering result Cl for X

1 Split X vertically into R subsets $\{X^{(1)}, \ldots, X^{(R)}\}$, attributes of each set is G_i;
2 **for** $i = 1$ *to* R **do**
3 horizontally split $X^{(i)}$ into $X_1^{(i)}, \ldots, X_{N_{SA}}^{(i)}$;
4 **for** $j = 1$ *to* SN_A **do**
5 $[Centers_j^{(i)}, Clusters_j^{(i)}]$ = DenPEHC($X_j^{(i)}$);
6 **end**
7 $Centers^{(i)}$ = MergeAll($Centers_j^{(i)}$);
8 $Clusters^{(i)}$ = MergeAll($Clusters_j^{(i)}$);
9 $AttrCenters^{(i)}$ =DenPEHC ($Centers^{(i)}$);
10 Replace the cluster Id in $X^{(i)}$ with $AttrCenters^{(i)}$;
11 $CDM^{(i)} \leftarrow$ compute the distance matrix for $AttrCenters^{(i)}$;
12 **end**
13 Y = replace the attribute values $\{G_i\}_{i=1}^{R}$ in X with R categorical values;
14 Apply the same steps of Line 3–10 on Y as on $X^{(i)}$ (except the distances between objects are computed based on CDMs and the sample size is SN_T) to get Cl;
15 **return** Cl;

Tong and Kang categorized three basic ways to handle large-scale problems [1]. The second method, which involves "reducing the iteration number", is of special interest in our study. Our DenPEHC-LSHD algorithm borrows the idea of "random sampling", "clustering the partitions", and "clustering the preclustered data" from CURE [10], which is an example of the second method.

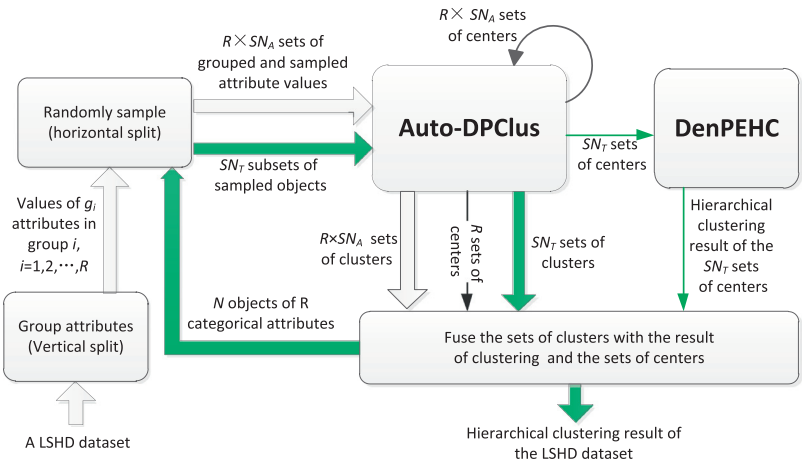

Fig. 6. Framework that enables DenPEHC to cluster large-scale and high-dimensionality datasets, where N is the number of objects in the dataset; SN_A and SN_T are the numbers of subsets during the 1st and 2nd round of horizontal splitting, respectively; $D = g_1 + g_2 + \cdots + g_R$ is the dimensionality of the dataset; g_i is the number of attributes in attribute group $i (i = 1, 2, \ldots, R)$; and R is the number of attribute groups. The processing starts from the bottom left and ends at the bottom right. The black and white arrows depict the data in the first round of "merging the attributes", and the green arrows depict the data in the second round after the attributes have been merged into R latent concepts. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. DenPEHC-LSHD algorithm

If the dimensionality of the dataset D is large (e.g., $D > 20$), we attempt to granulate the attributes into R groups by their semantic relationships from the domain knowledge (this process can also be referred to as *vertical granulation*). This idea is intuitively meaningful because each group of attributes actually describes a latent concept, each of which collectively compose the characteristics of the objects in the dataset. Thus, the values of all the attributes will be replaced by R categorical values after vertical granulation. The distance between two points as described by these R categorical values will be discussed in Section 4.3.

After attribute projection (first round) or vertical granulation (second round), a dataset is randomly split into SN subsets (this process can also be referred to as *horizontal granulation*). Thus, a large-scale dataset is transformed into many moderate-sized isomorphic sub-datasets, for which we run Auto-DPClust and subsequently fuse the clustering results of these SN sub-datasets. The idea of horizontal granulation can be found in [38]:

More specifically, the global mining can be featured with a two-step (local mining and global correlation) process, at data, model, and at knowledge levels.

If a dataset has large scale and high dimensionality, then we design a framework (Fig. 6) to use both horizontal and vertical granulation (called *grid granulation*). This framework reflects the ideas of two-step mining [38], cluster merging [4], and vertical granulation [39].

In addition to the framework diagram in Fig. 6, we describe the detailed steps in applying DenPEHC to the LSHD dataset as Algorithm 6.

4.3. Distance measurement with the categorical attribute value

If we define the distance between object i and object j based on L_p-norms as in Eq. (9), the distance measurement under vertical granulation will have the characteristic of *form invariance* because we can compute the distance between two states of a latent concept by using the distance between their corresponding centers, as shown in Eq. (10):

$$d_{ij} = \sqrt[p]{\sum_{k=1}^{R} (D_{ij}(A_k))^p}, \tag{9}$$

where $D_{ij}(A_k)$ is the distance between object i and object j in the attribute group (or latent concept) A_k:

$$D_{ij}(A_k) = \sqrt[p]{\sum_{m=1}^{g_k} (x_{i,(a_{k,m})} - x_{j,(a_{k,m})})^p}, \tag{10}$$

where $a_{k,\,m}$ is the mth attribute in the attribute group A_k.

Therefore, we obtain Eq. (11) by substituting $D_{ij}(A_k)$ into Eq. (9). Here, $D = \sum_{k=1}^{R} g_k$, so we can conclude that the distance with vertical granulation has the same form as that without vertical granulation.

$$d_{ij} = \sqrt[p]{\sum_{k=1}^{R}\sum_{m=1}^{g_k}(x_{i,(a_{k,m})} - x_{j,(a_{k,m})})^p} \tag{11}$$

This semantic regrouping is a *locality-preserving projection* [1] because the pair-wise distance for any two data points remains approximately unchanged before and after the projection. The distance as defined in Eq. (11) produces much better accuracy than directly taking the categorical number as the numerical value of the grouped attributes.

The property of form invariance no longer holds for another frequently used measure: *cosine similarity*. Therefore, if the distance metric that is used in vertical granulation is cosine similarity (although not strictly a distance), then the distance in the new dataset Y cannot be computed with L_p-norms. We employ a "multidimensional scale" [2] method to transform the categorical-valued distance matrices (CDMs) into 2-D points. Thus, Y with R categorical attributes becomes a new dataset Y' with $2 \times R$ numerical attributes, and cosine similarity is now applicable.

4.4. Determining the size of subsets

Guha et al. used *Chernoff bounds* to derive Eq. (12) to estimate the minimum sample size when clustering large-scale data [10].

$$s_{\min} = k\beta\left(\xi + \ln(1/\alpha) + \sqrt{(ln(1/\alpha))^2 + 2\xi ln(1/\alpha)}\right), \tag{12}$$

where k is the number of clusters; α is the upper bound of the probability that the sample contains fewer than $f|u|$ points that belong to cluster u ($f = \xi/|u_{min}|$ is the proportion of the size of the data points that represent the geometry of the smallest cluster to the size of the smallest cluster); $\beta > 1$ is used to decide the ratio of the smallest cluster to the average cluster in size, i.e., $|u_{min}| = N/(k\beta)$; and ξ is a constant number of data points that can adequately represent the geometry of the smallest cluster.

The minimum computed sample size with Eq. (12) is independent of the overall dataset size N. When empirically evaluating the DenPEHC-LSHD algorithm, we will use this formula to compute the minimum size of the randomly sampled data and then relax this value to a larger number if possible to exploit the computational capacity.

4.5. Accelerating by horizontal granulation

Grid granulation can substantially accelerate the clustering procedure because almost all clustering algorithms (including DenPEHC) have a time complexity of $O(n^2)$. According to the work flow in Fig. 6, we must perform Auto-DPClust $R \times (SN + 1)$ times, where SN denotes the number of subsets in the horizontal granulation, so the time complexity of the grid granulation approach changes into $R \times (SN + 1) \times O((N/SN)^2)$. The complexity ratio of the grid granulation approach to plain DenPEHC is

$$\frac{R \times (SN+1) \times O((N/SN)^2)}{O(N^2)} \approx R \times (SN+1) \times \frac{1}{SN^2} \approx \frac{R}{SN} \tag{13}$$

Usually, we have $R \ll SN$, so the grid granulation approach can accelerate the clustering process in addition to tackling large matrices and distance concentration issues.

5. Experiments

5.1. Datasets and settings

The experiments are conducted on a personal computer with an Intel i5-2430M CPU, 8G RAM, Windows 7 64bit OS, and MATLAB 2014 programming environment. We test our method on 8 datasets: two of them are synthetic and six are real-world datasets from the UCI Machine Learning Repository. Brief information on the 8 datasets is tabulated in Table 3. We choose datasets No. 3 to No. 6 for two reasons: the datasets are tested with two major comparison methods from [35] and [34], and the size of the datasets is moderate, allowing them to be directly clustered by DenPEHC. Datasets No.7 and No.8 are LSHD and are used to demonstrate the effectiveness of the proposed framework in Fig. 6.

The first dataset (5Spherical) is generated by setting centers of five spheres and randomly sampling dots on the surface of the spheres and then projecting the dots to the plane (see Fig. 1a). 5Spherical is designed to show that the entire dataset

[2] https://en.wikipedia.org/wiki/Multidimensional_scaling.

Table 3
Datasets used to test DenPEHC.

No.	Dataset	# Objects	# Attributes	# classes	Real-world /Artificial
1	5Spherical	2200	2	{5, 4, 2}	Artificial
2	5Spiral	1060	2	{5, 3, 2}	Artificial
3	Glass	214	9	7(6 actually)	Real-world
4	Wisconsin	569	10	2	Real-world
5	Ecoli	336	7	{8, 4, 2}	Real-world
6	DHN	2000	6	10	Real-world
7	PAMAP(subject101)	376,417	54	{13, 4}	Real-world
8	Opportunity(S1ADL1)	51,116	250	4	Real-world

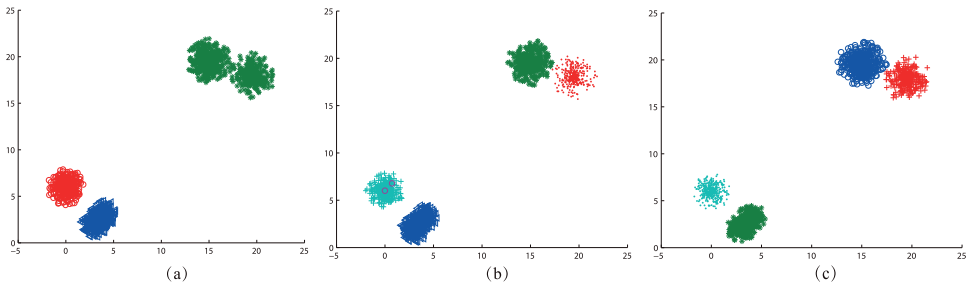

Fig. 7. Comparative experiments on the dataset "5Spherical": (a) LEGClust result of 3 clusters, (b) LEGClust result of 5 clusters, and (c) SynC clustering result.

maybe clustered as five, four or two groups. The second dataset (5Spiral) is 5 spiral curves that use Function (14):

$$\begin{cases} x = -t/8 \times \cos(t + \varphi); \\ y = -t/8 \times \sin(t + \varphi); \end{cases} \tag{14}$$

where $t \in (2, 4\pi)$ and φ is the parameter that controls the starting point of the spiral curves. Among the five spirals, two spirals in two pairs are arranged relatively close to each other and one spiral is separated; thus, these spirals can be clustered into five, three or two groups (see Fig. 8a). The two artificial datasets both have hierarchical structure, and are of spherical and non-spherical shape, with which the effectiveness and robustness of a hierarchical clustering method can be tested.

The full names of the datasets "Glass", "Wisconsin", and "DHN" are "Glass Identification Database", "Wisconsin Diagnostic Breast Cancer", and "The multi-feature digit dataset", respectively. DHN is a dataset of handwritten digits ('0'-'9'), whose group of features name "mfeat-mor" is used in our experiment. The Ecoli dataset has eight classes at the finest level, but can be merged into four or two classes at higher levels [35][3], possibly because of the semantics of the attributes in the domain. The last two datasets have large scale and high dimensionality, which are beyond the capacity of DenPEHC and require additional technology. Their full names are "PAMAP2 Physical Activity Monitoring" [31] and "OPPORTUNITY Activity Recognition" [6][5]. These two datasets are both from the domain of human activity monitoring and were chosen for two major reasons. First, their objects can be clustered hierarchically because human activities exhibit multiple layers; second, the attributes can be grouped with obvious semantics.

5.2. Results and evaluation

5.2.1. Synthetic data

LEGClust found a hierarchy for the 5Spherical data that contained two layers, which consisted of five and three clusters, respectively. One cluster with size of two was found, as shown in Fig. 7b. The *hSync* algorithm only detected the four clusters, as shown in Fig. 7c. DenPEHC recognized the hierarchical structure, completely matching human's intuition (see Fig. 1).

The clustering results for the 5Spiral data are depicted in Fig. 8. DenPEHC recognized a three-tiered structure. Each tier contained five, three, and two clusters. On the top layer, the spirals were divided into two groups because their densities were different. LEGClust produced a flat clustering result of 5 clusters (Fig. 9a), and *SynC* did not recognize the spiral structure. *SynC* finally clustered the entire dataset into one cluster, before which we could track down an intermediate result of 3 clusters (Fig. 9b).

5.2.2. Real-world data of moderate size and dimension

All the real-world datasets were normalized by using the $Z-score$ method, i.e., each attribute value was subtracted by the mean and divided by the standard deviation of the corresponding column. The distance metric that was employed was

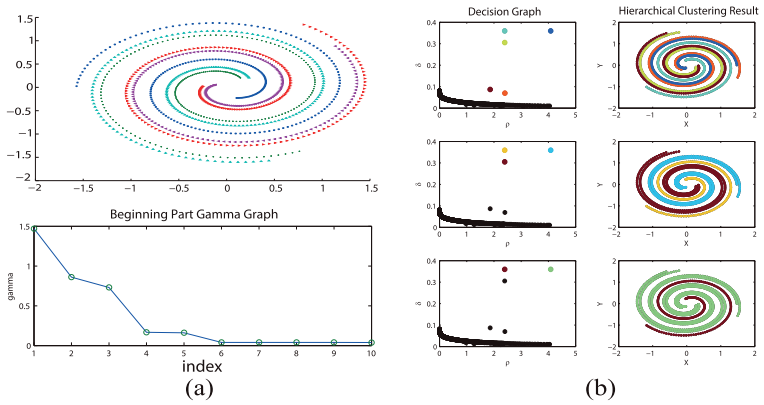

Fig. 8. DenPEHC experiment on the dataset "5Spiral": (a) original data and the beginning of gamma graph, and (b) DenPEHC result of the hierarchical clustering as 3 tiers, which contained 5, 3 and 2 clusters.

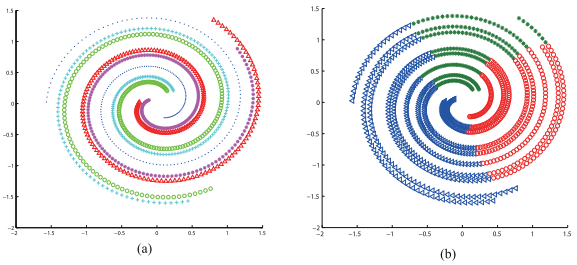

Fig. 9. Comparative experiments on the dataset "5Spiral": (a) LEGClust result of 5 clusters, and (b) *SynC* clustering's intermediate result of 3 clusters.

Table 4
Running time of the algorithms.

Algorithm	Running time (s)					
	5Spherical	5Spiral	Glass	Wisconsin	Ecoli	DHN
DPClust	9.125	1.126	0.041	51.079	0.094	21.176
DenPEHC	9.171	1.131	0.043	51.159	0.162	21.361
Sync	23.474	6.056	3.215	92.514	5.397	57.865
LEGClust	140.365	9.229	0.114	2.587	0.381	131.721

cosine distance rather than the Euclidean distance (which is used when clustering synthetic datasets). The results of the datasets Wisconsin and Ecoli are visualized in Fig. 10.

The distance matrix of the other two datasets DHN and Glass were not suitable to perform 2D multidimensional scaling, so we will not present the figures that visualize their clustering results. However, this limitation did not prevent the two datasets from being successfully clustered by DenPEHC, and the efficiency and effectiveness of DenPEHC are described in Section 5.2.3.

5.2.3. Quantitative evaluation of artificial and moderately sized data

We conventionally evaluated our approach by comparing it to state-of-the-art methods in terms of running time and clustering accuracy. The metrics NMI and ARI (introduced in [37] and [16], respectively) were employed to examine the clustering accuracy.

The running time of the algorithms under consideration on the six datasets are tabulated in Table 4. DenPEHC changed the original flat clustering result by DPClust into a hierarchical structure at the cost of only a small amount of additional time (approximately 0.5% of that for DPClust). This process was much more efficient than other methods at building a hierarchical structure for the original finest data.

As shown in Table 5, DenPEHC resulted in the best accuracy of the three competing methods on artificial datasets. If the clustering result contained several layers, we presented them in top-down order. *hSync* resulted in the lowest accuracy for the dataset 5Spiral, possibly because the "synchronization" scheme failed to detect a cluster of certain shapes. When tested on the real-world datasets, DenPEHC achieved varying accuracy between datasets, but remained comparative and acceptable overall. In the four real-world datasets, DenPEHC achieved the highest score for 10 times out of 14, comparing favorably with *hSync* and LEGClust (4 times and 0 times, respectively).

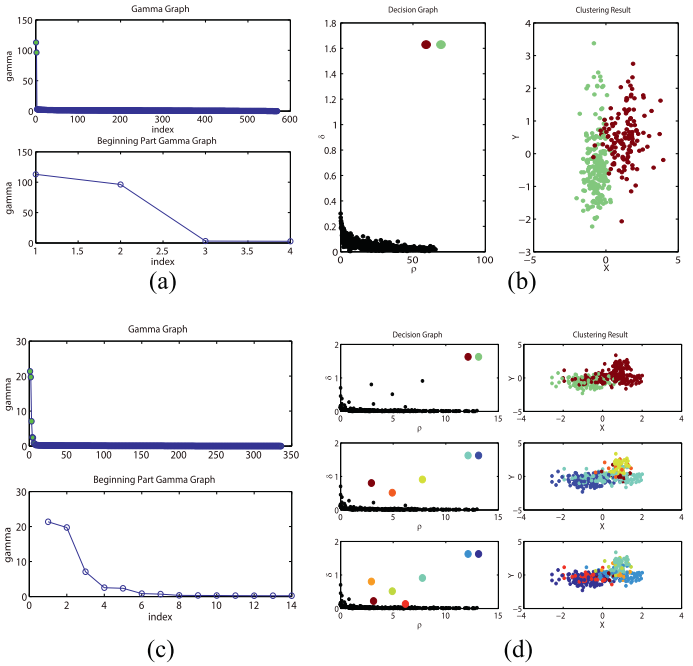

Fig. 10. DenPEHC experiments on the real-world datasets "Wisconsin" and "Ecoli": (a) gamma graph of the Wisconsin data, (b) clustering result of the Wisconsin data, (c) gamma graph of the Ecoli data, and (d) clustering result of the Ecoli data as 3 layers, each layer contains 2, 5, and 7 clusters.

Table 5
Accuracy of the algorithms.

Algorithm	Metrics	Datasets					
		5Sphe	5Spi	Glass	Wisc	Ecoli	DHN
DenPEHC	ARI	**0.9559** **0.9936** **1.0000**	**1.0000** **1.0000** **1.0000**	**0.4756** **0.4647**	**0.9669**	**0.6565** **0.6264** 0.3666	**0.6527** 0.5096 0.2511
	NMI	**0.9476** **0.9892** **1.0000**	**1.0000** **1.0000** **1.0000**	**0.3046** 0.2836	0.5620	**0.6281** **0.5851** 0.3385	**0.6873** 0.6392 0.6474
hSync	ARI	0.8530 **0.9936** 0.5140	0.0188 0.0398 0.0283	0.1955 0.1965	0.7366	0.4379 0.3901 **0.6359**	0.5218
	NMI	0.9179 **0.9892** 0.7104	0.0309 0.0372 0.0120	0.2930 **0.2939**	**0.6828**	0.5570 0.4632 **0.5610**	0.6551
LEGClust	ARI	0.8488 0.7697 0.7085	**1.0000** 0.6145 0.2099	0.1401 0.2376	0.0041	−0.0065 −0.0054 0.0024	0.2616
	NMI	0.9103 0.8764 0.8016	**1.0000** 0.8096 0.5576	0.3332 0.4118	0.0598	0.1116 0.0953 0.0835	0.4489

Even if the actual cluster number of a dataset is a fixed number or if the dataset is flatly clustered, constructing a hierarchical structure is always meaningful. Apart from the aforementioned perspective of Granular Computing, hierarchies can also help us cluster datasets with several possible numbers of clusters by running once, among which one number may lead to relatively higher accuracy.

5.2.4. LSHD data

(a) PAMAP dataset

The PAMAP dataset was generated by monitoring the activities of 9 subjects with 3 Colibri wireless IMUs (inertial measurement units) and a heart rate (HR) monitor. We chose the dataset of Subject101 because he performed the most types of activities. Before we began grid granulation, we preprocessed the dataset because of its many missing values.

Table 6
Vertical granulation result of PAMAP dataset.

Original Attributes	4–20	21–37	38–54
Latent Concept Name	Hand	Chest	Ankle
Number of Concept States	14	17	14

Table 7
Clustering accuracy on PAMAP dataset.

Method	Clusters Number	ARI	NMI
grid granulation	(13, 5)	(0.2913, 0.4509)	(0.4933, 0.5021)
total attributes	(12, 4)	(0.0721, 0.0839)	(0.3059, 0.1691)
random grouping	(12, 4)	(0.0266, 0.0398)	(0.0894, 0.0313)

Table 8
Vertical granulation result of OPPORTUNITY dataset.

Latent Concept Name	Back	HIP	LLA	LShoe	LUA	RKnee	RLA	RShoe	RUA
Number of Attributes	16	3	19	16	19	6	16	16	19
Number of Concept States	4	3	8	7	7	7	7	7	5

The missing values in the attribute HR were set to the same values as in its previous instance because these missing values were caused by the lower sampling frequency (9 Hz) than that of IMUs (100 Hz), while the instances contained missing values because the sensors or network failure were simply deleted. According to the README document, all instances with label 0 were excluded. After preprocessing, 247,206 instances were left for further processing with grid granulation and DenPEHC.

The 54 attributes were split into 3 groups (Hand, Chest, and Ankle), so we estimated the sampling size in the first round by using Eq. (12) with the parameters $k = 20,\ \alpha = 0.001,\ \beta = 2,$ and $\xi = 20$. Therefore, we obtained $20 \times 2 \times (20 + \ln 1000 + \sqrt{(\ln 1000)^2 + 2 \times 20 \times \ln 1000}) \approx 1796$. We took the value $s_{min} = 2000$ to randomly sample the vertically split datasets. The result of the grid granulation procedure is shown in Table 6.

After grid granulation, 3 groups of numerical attribute values in each object were transformed into 3 categorical values, and the number of unique records in the entire PAMAP dataset was reduced to 1686. We set the sample size as 2000 in the second round of horizontal granulation.

We finally obtained the hierarchical clustering result that is shown in Table 7. This table shows that the accuracy was greatly improved by grid granulation and that the clustering result can be even worse if randomly grouping the attributes without considering the domain knowledge.

(b) Opportunity dataset

"Opportunity" is a human activity dataset that was collected from a sensor-rich environment [6][5]. In addition to the sensors that were worn on the subjects' arms, hands, back, right knee, and shoes to monitor the acceleration and inertance of the body parts, other sensors were installed on various objects in the room, such as doors, fridge, drawers, etc. Seven tasks could be performed on this dataset, among which we picked the "locomotion recognition" task. The sensors installed on the objects were not necessary (although somehow helpful) for the task, so we excluded the corresponding columns.

We performed similar data preprocessing to that for the PAMAP dataset, including missing data completion and dirty data deletion. Consequently, the input dataset of the framework in Fig. 6 had 130 attribute and 37,507 instances. After the transformation from numerical attribute groups to categorical values, the 130 numerical values were vertically granulated as 9 latent concepts(as tabulated in Table 8), and the number of unique records was 2763. Therefore, the sample size in the second round of horizontal granulation was set to 3000.

Finally, we clustered the subset S1ADL1 from the Opportunity dataset into 4 categories, achieving an NMI of 0.4495 and ARI of 0.4432.

5.2.5. Parameter sensitivity analysis

DenPEHC involves 4 parameters: *Percent*(used to decide d_c by choosing the distance element at the first position $\lceil N * Percent\% \rceil$ in the ascendingly sorted distance array, i.e., $d_c = SortedDistance[\lceil N * Percent\% \rceil]$), *LocalR*, *GlobalR*, and *StairThre*. If we emphasize the local structure of a dataset to find more clusters, we must set a relatively smaller *Percent*. *LocalR* and *GlobalR* are set to smaller values to make points with large γ values more easily selected as centers. Additionally, smaller *StairThre* identifies more "stairs" on the centers' γ curve.

The parameter sensitivity was tested on two datasets, namely, 5Sphetical and Ecoli, which are synthetic and real, respectively. The parameters *LocalR* and *GlobalR* are closely related when selecting all potential centers, so these values were treated as one group. Consequently, the evaluation of the variations in the 4 parameters yielded 3 subtables for each dataset.

Table 9
Parameter variations on 5Spherical (Pct = 0.1, LR = 4.6, GR = 0.05, ST = 1.8).

Pct	#Clust	ARI	NMI	(LR,GR)	#Clust	ARI	NMI	ST	#Clust	ARI	NMI
0.01	5	0.9559	0.9476	1.2, 0.005	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000	1.2	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000
0.05	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000	1.8, 0.01	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000	5.4	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000
0.1	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000	4.6, 0.05	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000	16	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000
0.5	5, 2	0.9559 1.0000	0.9476 1.0000	5.4, 0.1	2	1.0000	1.0000	30	5, 4, 2	0.9559 0.9936 1.0000	0.9476 0.9892 1.0000
5	5, 2	0.9559 1.0000	0.9476 1.0000	6.8, 0.2	2	1.0000	1.0000	40	5, 4	0.9559 0.9936	0.9476 0.9892
10	4, 2	0.9936 1.0000	0.9892 1.0000	10, 0.3	2	1.0000	1.0000	50	5	0.9559	0.9476

Table 9 shows the sensitivities of the parameters for 5Sphetical. The left 4 columns show that the parameter *Percent* (abbreviated as Pct in the table) affected the choice of centers and shape of the gamma curve. When *Percent* = 0.01, the clustering achieved a flat result of the finest partition, dividing the points into 5 clusters. When *Percent* was 0.05 or 0.10, the results included 3 tires with cluster numbers of 5, 4, and 2 (from bottom to top), showing perfect consistency with human intuition. When *Percent* was 0.5 or 5, the results were also hierarchical but with only 2 layers, each consisting of 5 and 2 clusters. Assigning *Percent* = 10 also produced a hierarchical clustering of 2 layers, but the number of clusters on each layer was 4 and 2. Assigning *Percent* values within a significantly wide range changed the number of layers and the number of clusters on a layer, but the accuracy for a given cluster remained the same.

The parameter pair (*LocalR*, *GlobalR*), which are abbreviated as LR and GR respectively, was used to choose the centers for the clusters. Typically, we set the (*LocalR*, *GlobalR*) with 3 representative vectors (1.2, 0.005), (1.8, 0.01), or (5.4, 0.1), which stand for the difficulty level "easy", "medium" and "hard" for a data point to be chosen as a center, respectively. The clusters' changes with variations in (*LocalR*, *GlobalR*) are shown in the middle 4 columns of Table 9. The first 3 configurations all produced the best results, while the last 3 configurations ("hard" standards) chose the two most salient centers as one layer.

StairThre (abbreviated as ST) can determine how likely a change in the slope of the gamma curve will be regarded as a "stair" (corresponding to a layer of the clustering hierarchy). The smaller the value of *StairThre*, the easier that detecting a stair becomes, and vice versa. According to the right 4 columns of Table 9, when *StairThre* ranged from 1.2 to 30, DenPEHC correctly detected the 3 stairs. Even for the assignment of 40, this method still distinguished 2 layers of the target hierarchy. All the centers began to be regarded as in the same layer when *StairThre* reached an extraordinary value of 50.

According to the above observations, small fluctuations in the best chosen parameters for the dataset 5Spherical did not change the clustering result. Thus, DenPEHC shows relative robustness, especially on the parameters (*LocalR*, *GlobalR*) and *StairThre*.

The parameter sensitivities for the Ecoli dataset are shown in Table 10, according to which, the robustness is also good. The exception was when (*LocalR*, *GlobalR*) was assigned as (13, 0.06), which is actually a very tough criterion. Here, DenPEHC yielded a trivial clustering of one cluster.

5.3. Analysis of the efficiency

Existing agglomerative hierarchical clustering methods are not very efficient, because they must decide which pair of clusters to merge or because they construct each layer as one intermediate result. For example, LEGClust calculates the connections between clusters to merge clusters [34], and Bouguettaya et al. proposed the algorithm *knA* to build a hierarchy on the centroids of middle-level clusters by using existing approaches such as SLINK and UPGMA [4]. A relatively new clustering framework, *hSync*, forms a clustering hierarchy through "synchronization" and establishes each layer by many rounds of "interactions" [35]. However, this method is also not efficient.

The time complexity for a family of Dirichlet Diffusion Trees that are used to perform hierarchical clustering by generative learning is empirically higher than DenPEHC, which can be found by referring to the running time on several small datasets [18].

DenPEHC is more efficient at constructing hierarchical structures based on leading tree structures as intermediate results because the steps of selecting centers, deciding stairs and splitting the leading tree into a forest are all highly efficient. If

Table 10
Parameter variations on Ecoli (Pct = 1.0, LR = 4.8, GR = 0.1, ST = 1.8).

Pct	#Clust	ARI	NMI	(LR,GR)	#Clust	ARI	NMI	ST	#Clust	ARI	NMI
1.0	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	1.2, 0.005	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	1.2	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385
1.3	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	1.8, 0.01	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	3.8	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385
1.8	5, 2	0.6505 0.3666	0.5849 0.3385	4.6, 0.02	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	7	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385
5.0	5, 2	0.6505 0.3666	0.5849 0.3385	12, 0.03	7, 5, 2	0.6565 0.6264 0.3666	0.6182 0.5851 0.3385	8.8	7, 5	0.6565 0.6264	0.6182 0.5851
10	4, 2	0.6260 0.5159	0.5600 0.4405	13, 0.06	1	0	NaN	10	7	0.6565	0.6182

the dataset has large scale and high dimensionality, we can use the grid granulation framework to make DenPEHC work. As an unexpected advantage, this framework can substantially accelerate the clustering process (see Section 4.5).

However, we must reaffirm an aforementioned point: DenPEHC is a heuristic algorithm of non-iterative fashion, hence the high efficiency, but lacks rigorous mathematical guarantees to achieve satisfactory results (although the empirical evaluations show good performance). Many existing hierarchical clustering methods are quite the opposite, i.e., they perform an iterative process to ensure an optimal criterion is met. Thus, our approach should be chosen conditionally, which serves as a good example of the "no free lunch" theorem.

6. Conclusions

This paper presented a hierarchical clustering method without agglomerative or divisive processes (DenPEHC) and introduced a grid granulation framework that enabled DenPEHC to cluster LSHD datasets. We used a linear fitting method to select all potential centers and mapped the "stairs" on a γ curve to the layers in the clustering hierarchy. The underlying leading tree structure in the intermediate result of DPClust was used to accelerate the process of assigning non-center data points to their centers. A framework that was based on vertical granulation and horizontal granulation (collectively called grid granulation) was designed to cluster LSHD datasets when the attributes in the dataset can be grouped according to their semantics. This grid granulation approach can solve problems that involve large distance matrices and distance concentrations and accelerate the entire clustering procedure. Our experiments have shown that our methods are robust, more efficient (increasing the acceleration by a factor of 1.8–33.2) and competitively accurate compared to competing algorithms.

Acknowledgments

This work has been supported by the National Key Research and Development Program of China under grant 2016YFB1000905, the National Natural Science Foundation of China under Grant numbers of 61272060 and 61572091. The authors thank Dr. Junming Shao for providing the *SynC* program and anonymous reviewers for their help in improving the manuscript.

References

[1] C.C. Aggarwal, C.K. Reddy, Data clustering: algorithms and applications, CRC Press, 2014.
[2] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[3] C. Böhm, C. Plant, Hissclu: a hierarchical density-based method for semi-supervised clustering, in: Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, ACM, 2008, pp. 440–451.
[4] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, A. Song, Efficient agglomerative hierarchical clustering, Expert Syst. Appl. 42 (5) (2015) 2785–2797.
[5] R. Chavarriaga, et al., Collecting complex activity datasets in highly rich networked sensor environments, in: 7th International Conference on Networked Sensing Systems, 2010, pp. 233–240.
[6] R. Chavarriaga, et al., The opportunity challenge: a benchmark database for on-body sensor-based activity recognition, Pattern Recognit. Lett. 34 (15) (2013) 2033–2042.
[7] C.P. Chen, C.Y. Zhang, Data-intensive applications, challenges, techniques and technologies: a survey on big data, Inf. Sci. 275 (2014) 314–347.
[8] F. de Morsier, D. Tuia, M. Borgeaud, V. Gass, J.P. Thiran, Cluster validity measure and merging system for hierarchical clustering considering outliers, Pattern Recognit. 48 (4) (2015) 1478–1489.
[9] P. Grünwald, A tutorial introduction to the minimum description length principle, in: Advances in Minimum Description Length: Theory and Applications, 2005, pp. 23–81.
[10] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, in: ACM SIGMOD Record, 27, ACM, 1998, pp. 73–84.
[11] K.A. Heller, Z. Ghahramani, Bayesian hierarchical clustering, in: Proceedings of the 22nd ICML, ACM, 2005, pp. 297–304.
[12] A. Hinneburg, H.H. Gabriel, Denclue 2.0: fast clustering based on kernel density estimation, in: Advances in Intelligent Data Analysis VII, Springer, 2007, pp. 70–80.

[13] A. Hinneburg, D.A. Keim, A general approach to clustering in large databases with noise, Knowl. Inf. Syst. 5 (4) (2003) 387–415.
[14] V.J. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2) (2004) 85–126.
[15] J. Huang, Q. Zhu, L. Yang, J. Feng, A non-parameter outlier detection algorithm based on natural neighbor, Knowl.-Based Syst. 92 (2016) 71–77.
[16] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[17] D. Knowles, Z. Ghahramani, Pitman Yor diffusion trees for Bayesian hierarchical clustering, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2) (2015) 271–289.
[18] D.A. Knowles, J. Van Gael, Z. Ghahramani, Message passing algorithms for the Dirichlet diffusion tree., in: ICML, 2011, pp. 721–728.
[19] H. Li, K. Zhang, T. Jiang, Minimum entropy clustering and applications to gene expression analysis, in: Computational Systems Bioinformatics Conference, IEEE, 2004, pp. 142–151.
[20] C.R. Lin, M.S. Chen, Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Trans. Knowl. Data Eng. 17 (2) (2005) 145–159.
[21] C.D. Manning, P. Raghavan, H. Sch¿ tze, Introduction to Information Retrieval, Cambridge University Press, 2009.
[22] G.A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information, Psychol. Rev. 63 (2) (1956) 81.
[23] A. Mirzaei, M. Rahmati, A novel hierarchical-clustering-combination scheme based on fuzzy-similarity relations, IEEE Trans. Fuzzy Syst. 18 (1) (2010) 27–39.
[24] R.M. Neal, Density modeling and clustering using Dirichlet diffusion trees, Bayes. Statist. 7 (2003) 619–629.
[25] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 90–105.
[26] W. Pedrycz, Knowledge-Based Clustering: From Data to Information Granules, John Wiley & Sons, 2005.
[27] H. Qin, X. Ma, T. Herawan, J.M. Zain, Mgr: an information theory based hierarchical divisive clustering algorithm for categorical data, Knowl.-Based Syst. 67 (2014) 401–411.
[28] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., 1993.
[29] M. Radovanovic, A. Nanopoulos, M. Ivanovic, Reverse nearest neighbors in unsupervised distance-based outlier detection, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1369–1382.
[30] E. Rashedi, A. Mirzaei, A hierarchical clusterer ensemble method based on boosting theory, Knowl.-Based Syst. 45 (2013) 83–93.
[31] A. Reiss, D. Stricker, Introducing a new benchmarked dataset for activity monitoring, in: 16th International Symposium Wearable Computers, IEEE, 2012, pp. 108–109.
[32] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[33] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5) (2000) 2323–2326.
[34] J.M. Santos, J.M. de Sá, L.A. Alexandre, Legclusta clustering algorithm based on layered entropic subgraphs, IEEE Trans. Pattern Anal. Mach. Intell. 30 (1) (2008) 62–75.
[35] J. Shao, X. He, C. Böhm, Q. Yang, C. Plant, Synchronization-inspired partitioning and hierarchical clustering, IEEE Trans. Knowl. Data Eng. 25 (4) (2013) 893–905.
[36] X. Tang, P. Zhu, Hierarchical clustering problems and analysis of fuzzy proximity relation on granular space, IEEE Trans. Fuzzy Syst. 21 (5) (2013) 814–824.
[37] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (2010) 2837–2854.
[38] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2014) 97–107.
[39] J. Xu, G. Wang, H. Yu, Review of big data processing based on granular computing, Chin. J. Comput. 38 (8) (2015) 1497–1517.
[40] J.T. Yao, A.V. Vasilakos, W. Pedrycz, Granular computing: perspectives and challenges, IEEE Trans. Cybern. 43 (6) (2013) 1977–1989.
[41] Y.Y. Yao, Perspectives of granular computing, in: Proceedings of IEEE International Conference Granular Computing Beijing, China, 1, 2005, pp. 85–99.