Neurocomputing 275 (2018) 2355–2366

Contents lists available at ScienceDirect

Neurocomputing

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

Efficient distributed clustering using boundary information

Qiuhui Tong, Xiu Li, Bo Yuan*

Intelligent Computing Lab, Division of Informatics, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China

ARTICLE INFO

Article history:
Received 18 August 2016
Revised 24 February 2017
Accepted 6 November 2017
Available online 14 November 2017

Keyword:
Distributed clustering
DBSCAN
Cluster boundary
Density gradient

ABSTRACT

In the era of big data, it is increasingly common that large amount of data is generated across multiple distributed sites and cannot be gathered into a centralized site for further analysis, which invalidates the assumption of traditional clustering techniques based on centralized models. The major challenge is that these distributed datasets cannot be trivially merged due to issues such as privacy concerns, limited network bandwidth among sites and limited computational capacity of a single site. To tackle this challenge, we propose an efficient distributed clustering scheme using boundary information (DCUBI), which features good flexibility and scalability. The main procedure of DCUBI consists of three steps: local-global-local. Firstly, each local site extracts the boundary points from its own local data and applies traditional clustering on boundary points only. Secondly, labeled boundary points from each site are sent to the central site as local representatives where boundary and cluster fusion is conducted to form the global clustering model. Finally, the global boundary and cluster information is sent back to each local site for refined local clustering. To demonstrate the effectiveness of DCUBI, we plug the well-known DBSCAN algorithm into DCUBI and comprehensive experiments are conducted using datasets with different properties. Experiment results clearly verify the quality of clustering by DCUBI as well as its superior time efficiency when the volume of data in each site is large. Furthermore, other popular clustering techniques especially those with high time complexity such as spectral clustering and affinity propagation clustering are also plugged into DCUBI to demonstrate the flexibility of the proposed scheme.

1. Introduction

Traditional data mining techniques often require direct access to the entire dataset, which is assumed to be stored in the same site. Note that, in streaming data mining, although algorithms do not need to access all data points simultaneously, they still need to be processed in a single site. On distributed computing platforms such as Hadoop, although data can be assigned to a number of computing nodes, all data points are still regarded as a unity logically and different nodes can exchange data or other information frequently with little restriction. However, it is becoming more and more common that data is *intrinsically* generated in or gathered from geographically distributed sites. For example, Walmart, the world leading multinational retail corporation, runs 11,545 stores in 28 countries as of June 30, 2016 [1], featuring the largest civil database in the world [2]. Since each store collects a large amount of data on a daily basis, it is generally difficult to gather the data into a centralized site to do further analysis due to security concerns, limited network bandwidth and limited computational capacity of a single centralized site [3,4]. As a result, traditional data mining techniques relying on centralized models are no longer suitable for this class of distributed data mining tasks.

Clustering is one of the most widely studied data mining techniques in centralized data mining. Many centralized clustering algorithms have been introduced and analyzed in the literature [5]. Compared to the long history of research on centralized clustering techniques, the work on distributed clustering is relatively limited [2,3,6–10]. Furthermore, the results of distributed data mining may be varied due to varying conditions in data distributions, creating more challenges for distributed clustering [11].

To solve the task of distributed clustering in the above scenario, a small number of representatives from each local site can be selected and sent to a central site for global clustering. In this way, each site only needs to reveal a small amount of local information while the central site only needs to perform clustering on selected data points, instead of the entire dataset from each site, alleviating the data privacy and transmission problems properly. Under the assumption that each cluster can be represented by a solid object, the boundary points of a dataset can be adopted as the representatives of clusters. They are located at the margins of densely distributed data, which contain important characteristics of the distribution of data and can be used to decide whether a point belongs to a specific cluster [12].

* Corresponding author.
E-mail address: yuanb@sz.tsinghua.edu.cn (B. Yuan).

https://doi.org/10.1016/j.neucom.2017.11.014
0925-2312/

To achieve flexible and scalable clustering on distributed datasets, in this paper, we propose a general scheme for distributed clustering using boundary information (DCUBI), which employs a three-step clustering procedure: local–global–local. Firstly, each local site carries out boundary extraction independently. Then, clustering is executed on the boundary points instead of the original data and a local clustering model is determined consisting of labeled boundary information. Secondly, generated local models from all local sites are sent to a central site for global boundary and cluster fusion to obtain the global boundary information and the corresponding clustering results. Finally, the global information is sent back to local sites, which can be used to partition their local data to form the final clustering results. There are a number of issues that need to be properly addressed to ensure the effectiveness of the proposed scheme. For example, a data point regarded as a noise in a local site may turn out to be contained within a cluster from a global point of view. In addition, some boundary points of a local dataset may not qualify as boundary points when merged with boundary points from other sites.

A major benefit of DCUBI is that it strategically exploits boundary information of the original dataset, effectively reducing the volume of data to be clustered at each local site and achieving significant time savings for each local site, which is highly desirable in big data analytics. This is particularly important for traditional clustering methods featuring competent performance in tackling challenging clustering problems but with high time complexity. Many traditional clustering algorithms featuring high time complexity can be easily plugged into DCUBI to accomplish distributed clustering tasks in a time efficient way. Since boundary points usually only account for no more than 10% of the original data, the pressure of data transmission and privacy is reduced significantly. In this paper, we mainly use DBSCAN (Density Based Spatial Clustering Applications with Noise) [13] as the base clustering method due to its popularity, effectiveness and high time complexity, which is $O(n^2)$ in the worst case and at least $O(n^{4/3})$ for dimensions higher than 3 [14]. The main objective is to demonstrate the good scalability of DCUBI with little negative impact on the quality of clustering. We also plug two other clustering techniques Affinity Propagation Clustering (APCLUSTER) [15] and Spectral Clustering (SC) into DCUBI to further demonstrate its flexibility. In summary, the main contributions of DCUBI are as follows:

- The concept of boundary points is introduced into distributed clustering and traditional clustering algorithms with high time complexity can be used to solve distributed clustering tasks in an efficient way, especially when each site owns a large amount of data.
- Efficient strategies are proposed to merge boundary points from different local sites to obtain global boundary points, which can be used to refine the clustering results in each local site.

The rest part of this paper is organized as follows. Section 2 gives some critical review of existing distributed clustering techniques and boundary extraction algorithms. The proposed distributed clustering scheme using boundary information is discussed in details in Section 3. Experiment results are presented in Section 4 and this paper is concluded in Section 5 with some directions for future work.

2. Related work

2.1. Distributed clustering

Distributed clustering aims to handle datasets inherently residing on distributed sites, which cannot be sent to a central site for clustering due to typical issues including privacy concerns and limited transmission bandwidth. Unlike centralized clustering, distributed clustering relies heavily on the network structure because different networks have their own properties and task objectives [4]. In general, distributed data mining can be categorized into two classes: facilitator–worker networks and peer-to-peer (P2P) networks [16]. The facilitator–worker paradigm demands a reliable facilitator to aggregate all representative information from distributed sites. By contrast, the P2P paradigm does not require a central server and all sites have limited view of the entire network and only exchange necessary information to perform their own local clustering tasks [4,16]. Implementing local clustering on each individual site is normally the first step in both paradigms and then a set of representatives is selected to communicate with other peer sites or the facilitator.

Based on the two types of networks, the goal of distributed clustering can be divided into two categories: globally optimized clustering and locally optimized clustering [16]. Globally optimized clustering focuses on the global grouping pattern of datasets from all local sites as if they are stored in a centralized site while locally optimized clustering creates a unique set of clusters at each site taking into account of the information from other sites.

Facilitator–worker networks are suitable for distributed clustering tasks whose objectives are to get the global view of the data rather than obtaining more precise clustering results for each individual site. There are two common data partition patterns among local sites: homogeneous and heterogeneous. Homogeneous partition means that each site contains different objects with the same set of features while heterogeneous partition means that each site contains the same objects with different feature subsets. In this paper, our focus is on the facilitator–worker network with homogeneous data, aiming to achieve globally optimized clustering results efficiently.

In theory, many traditional centralized clustering algorithms can be adopted in the local clustering process but the clustering results may not be easily described or represented by a simple prototype [17]. In density based distributed clustering (DBDC) [6], which adopts the facilitator–worker network, each local site implements DBSCAN on its own data and selects a specific set of core points to represent local clusters. Central site receives all local representatives and applies DBSCAN to form the global clustering model. Finally, central site sends the global clustering model back to each local site to update their local clustering models. However, the time complexity of DBDC is high, which is unacceptable if the local site has a large quantity of data and the number of sites is large. The total time complexity of DBDC is $O(n^2) + O((n_s \cdot r_{avg})^2) + O(n \log(n_s \cdot r_{avg}))$with kd-tree for neighborhood query [18], where n is the average amount of data in each site, r_{avg} is the average amount of local representatives sent to the central site and n_s is the number of sites. Note that two sets of parameters are required for local DBSCAN and global DBSCAN, respectively, creating challenging parameter tuning issues in the application of the algorithm.

Scalable density based distributed clustering (SDBDC) [2] follows the similar procedure of DBDC. The main difference is in the representative selection process. SDBDC selects a set of data points based on their representation quality from local sites directly without clustering. All representatives selected are located in the relatively central part of clusters and if the percentage of representatives is small, all representatives tend to come from dense clusters and sparse clusters may be neglected. In this case, a large number of representatives are needed to ensure a proper local clustering model.

In summary, DBDC directly applies DBSCAN on the original dataset in each local site, potentially resulting in a time consuming process for large datasets due to the high time complexity of DBSCAN. Although SDBDC avoids conducting clustering using DBSCAN

on local sites, the representatives selected are not always appropriate for specifying clustering patterns. In our work, we propose to use boundary points as the representatives to not only lower the computational cost by reducing the number of data points to be clustered in each local site but also generate more principled prototypes for the original data as far as clustering is concerned. Furthermore, with boundary points as the representatives, many existing clustering techniques can be readily incorporated as the definition of boundary points is independent of the specific clustering algorithms in use. As a result, DCUBI is expected to be both efficient in handling large datasets at local sites and flexible in working with various clustering algorithms.

2.2. Boundary points

Existing works on boundary point detection can be generally categorized into three themes: density-based [19–21], concave theory based [22,23] and graph based [24–26].

Concave theory based methods mainly look for the envelope of the dataset, which can be viewed as the boundary. They adopt the convex theory combined with the k-nearest neighbors (kNN) approach to figure out every boundary point one by one. These methods can extract a series of refined boundary points regardless of the distribution of the data. However, they are not robust against noises as there is a need to iteratively check whether all data points can be encompassed by the current boundary points found in previous steps. Furthermore, they cannot be easily extended to dimensions higher than 3 [23].

Graph based detection methods consider all data points in the dataset as a whole. Normally, a Delaunay diagram is built where each point corresponds to a node in the graph and each edge represents the distance between two points. The core idea is that the variation of distances between a boundary point and its neighbors is greater than that of interior points. However, building the Delaunay diagram for high dimensional datasets is time consuming. As a result, existing research on the boundary points based on Delaunay diagram is often limited to low dimensions (less than 3) [24,25].

In density-based boundary detection methods, a point is regarded as a boundary point if most of its neighbors lie on the same side of the tangent plane passing through it. The tangent plane can be viewed as a plane that divides the space into two separate regions: one with high density of points and the other is relatively sparse. Fig. 1 shows three examples with different curvature shapes where the red lines represent the desired tangent planes.

Note that locating the tangent planes is difficult in practice. BORDER [12] takes advantage of the reverse kNN to avoid explicitly determining the plane. It assumes that the reverse kNN value of a boundary point is much smaller than that of interior points. The definition of the reverse kNN is [27]:

$$rkNN(x_i) = \left\{ x_j \middle| \forall x_j, x_i \in kNN(x_j) \right\} \tag{2.1}$$

BORDER can deal with datasets with simple shapes and without noises effectively. However, when the interior of a cluster has lower density compared to the boundary region, it may incorrectly choose some interior points and miss out true boundary points. Also, the reverse kNN is an expensive query: the time complexity of the original reverse kNN is $O(n^3)$, although it can be reduced to $O(n^2 \log n)$.

Alternatively, the normal vector of the tangent plane instead of the plane itself can be calculated [19–21]. In practice, the normal vector of the tangent plane is also the direction of density gradient. In [20], the attractor of a point x is defined as the neighbor of x sharing the highest number of ε neighbors and the vector pointing from x to its attractor is regarded as the normal vector. In [19,21], according to Theorem 1, the normal vector is defined as the vector pointing from x to the average position of the neighbors of x, which is also called mean shift (Theorem 1).

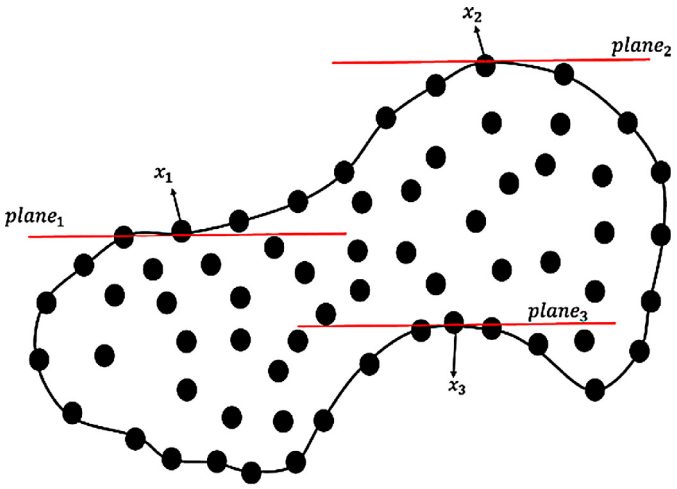


Fig. 1. Examples of boundary points. The three red lines represent the desired tangent planes of three different boundary points x_1,x_2 and x_3, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Theorem 1. *The normal vector (density gradient) of an object x is as follows*:

$$NV(x) = \rho \cdot \frac{1}{k} \sum_{i=1}^{k} u_i$$
$$u_i(x) = x_i - x \tag{2.2}$$

where $x_i \in kNN(x)$.

Next, the relative location (the dense side or sparse side of the tangent plane) of the neighbor of the object x can be readily decided. Note that, for a boundary point, its normal vector must point to the dense side of the tangent plane. So, for neighbors located at the same subspace (the tangent plane divides the whole space into two subspaces) as the normal vector, the angles between each $u_i(x)$ and the normal vector should be within $[0, \frac{\pi}{2}]$. As a result, if for most $u_i(x)$ of an object x, the angles between $u_i(x)$ and the normal vector are within $[0, \frac{\pi}{2}]$, x is considered as a boundary point. Various methods have been used in quantitative evaluation by counting the number of neighbors with angles between $[0, \frac{\pi}{2}]$ with the normal vector or adding up the cosine values of the angles between each u_i and the normal vector.

3. DCUBI

The proposed DCUBI employs the distributed facilitator–worker network: each local site extracts the local boundary points and implements clustering on the local boundary points only; the global/central site collects the boundary and cluster information from all local sites and carries out boundary and cluster fusion to obtain the labeled global boundary; local sites partition their local data based on the labeled global boundary points received from the global site. The general structure of DCUBI is shown in Fig 2.

3.1. Local clustering and local model

In this paper, we use a modified boundary extraction method (Algorithm 1 [28]). In order to eliminate the impact of noises, the algorithm first calculates the average distance of x to its k-nearest neighbors. Intuitively, the neighbor points of a noise point are sparse and the average distance of a noise point to its k-nearest neighbors is

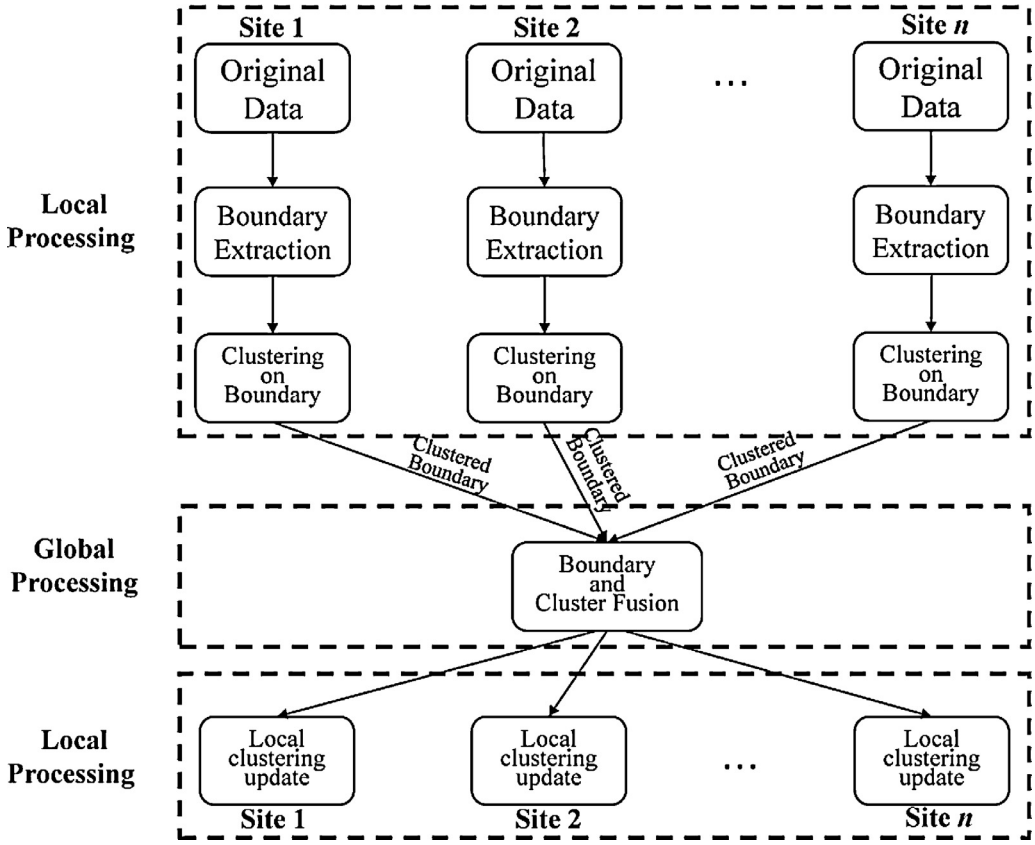


Fig. 2. The structure of DCUBI. Local sites extract the local boundary and implement clustering on local boundary; the global/central site implements boundary fusion and cluster fusion to form the global model and local sites then partition their own data according to the global model.

Algorithm 1
Boundary extraction.

Input: $\mathcal{D}, k, \alpha_{noise}, \alpha_{boundary}$
Output: *Boundary, NV*
1: $scores \leftarrow -\infty$
2: **For** each $x \in \mathcal{D}$ **do**
3: Find $kNN(x)$
4: $kNN_{average}(x) \leftarrow$ Average distance of $kNN(x)$ to x
5: **end for**
6: Sort kNN_{average} in descending order. Mark first α_{noise} as noises
7: **for each** $x \in \mathcal{D}$ **do**
8: **if** x is not the noise **then**
9: Exclude the noises in the $kNN(x)$
10: Find the $NV(x)$ of x
11: Calculate $cos(NV(x), u_i(x))$
12: $scores(x) \leftarrow \sum_{i=1} cos(NV(x), u_i(x))$
13: **end if**
14: **end for**
15: Sort *sores* in decending order. Select first $\alpha_{boundary}$ as *Boundary*
16: **return** *Boundary, NV* of boundary points

likely to be larger than those of non-noise points. A percentage value of noise points α_{noise} is used to identify noises. After the elimination of noise points, the normal vector of each non-noise point in the dataset can be calculated according to Theorem 1. Since the normal vector may be biased by the extremely large magnitude of $u_i(x)$, the calculation of normal vector is modified as follows:

$$NV(x) = \sum_{i=1}^{k} \frac{(x_i - x)}{|x_i - x|} = \sum_{i=1}^{k} \frac{u_i(x)}{|u_i(x)|} \tag{3.1}$$

With the normal vectors obtained, for each object x, the algorithm calculates the sum of the cosine values between $u_i(x)$ and $NV(x)$ as its score:

$$scores(x) = \sum_{i=1} \cos(u_i(x), NV(x)) \tag{3.2}$$

Boundary points typically feature larger scores than others as the $\cos(u_i(x), NV(x))$ values for x_i located on the same side of the tangent plane as the normal vector are greater than zero while others are negative. Finally, points with top $n \cdot \alpha_{boundary}$ scores are returned as boundary points along with their normal vectors where n is the size of original dataset and $\alpha_{boundary}$ is the percentage of points selected as boundary points.

Once the local boundary points are extracted using Algorithm 1, DBSCAN is performed on boundary points only, instead of the whole local dataset. Once the boundary points are grouped into suitable clusters, the boundary information of each local site, including normal vectors and cluster labels, is transmitted to the central site for further global analysis in the form of $< boundary, normalvector, localclusterid >$.

3.2. Global model

On the global/central site, the key task is boundary fusion and cluster fusion to form the global boundary and global cluster information.

The global/central site collects all the local boundary information from local sites in the form of $\bigcup_{i=1,2,\ldots,n_s} < boundary_i, normalvector_i, localclusterid_i >$ where n_s is the number of local sites. The overall relative relationships among cluster boundaries from different sites can be quite complicated. So, we start by considering circumstances involving only 2 sites and each site only features one cluster, which can be

Algorithms 2
Boundary combination.

Output: $boundary_{global}$, $clustersid_{global}$
$boundary_{global} \leftarrow \bigcup_{i=1}^{n_s} boundary_i$
  for each site $\in$ *siteid* **do**
    for each $cluster_{site}$ from *site* **do**
      for *each cluster* not from *site* **do**
        Determine the relationship between the boundary points in $cluster_{site}$ with boundary points of *cluster*
        if $points_{in} \cup points_{close}$ not empty **then**
          Combine $cluster_{site}$ and *cluster*
          Delete $points_{in}$ from $boundary_{global}$
        end if
      end for
    end for
end for

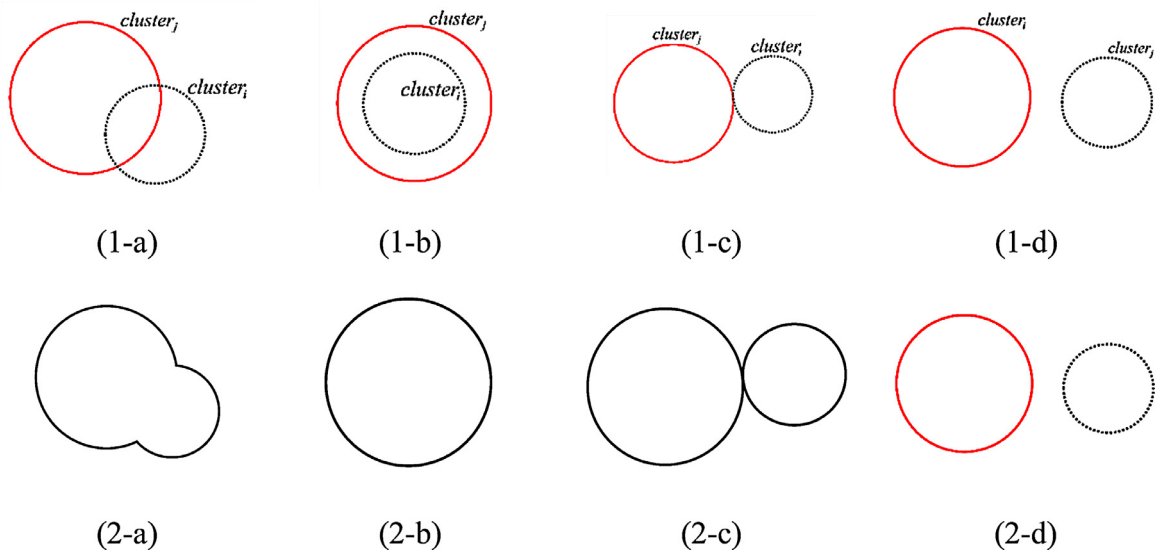


Fig. 3. The four different situations of boundary data fusion and the corresponding fusion results.

Algorithm 3
Position determination.

Input: *point*,*boundary*,*normvector*
$near\_boundary \leftarrow kNN(boundary, point)$
$y \leftarrow avg(near\_boundary)$
$u(y) \leftarrow point - y$
$NV \leftarrow avg(normvector(near\_boundary))$
if $cos(u(y), NV) \geq 0$ **then**
  point is inside *boundary*
else if $avg(distance(point, near\_boundary)) \leq threshold$ **then**
  point is close to *boundary*
end if
Output: The relationship between *point* and *boundary*

used as the foundation for handling more complex situations. Furthermore, the cluster boundaries from the same site are known to be separate as they correspond to different local clusters. Since we assume that each cluster is a solid object without holes, there are four possible relationships between two clusters from different sites, as shown in Fig. 3. In Fig. 3(1-a), boundary points of $cluster_i$ from site i and boundary points of $cluster_j$ from site j cross each other. In Fig. 3(1-b), boundary points of $cluster_i$ from site i are completely surrounded by the boundary points of$cluster_j$from site j. The two cluster boundaries may be also tangent or separate, as shown in Fig. 3(1-c) and Fig. 3(1-d), respectively.

The global boundaries for the four situations are shown in Fig. 3(2-a)–Fig. 3(2-d), respectively. It is clear that for the situation in Fig. 3(1-d), $cluster_i$ and $cluster_j$ will still be two separate clusters in the final global results and the boundary points of global results will be kept unchanged from local boundary points. The reason is that the two sets of boundary points from two clusters are rather far away from each other and consequently the global clustering results are as same as the local ones. For the other three cases, the two clusters will be merged as a new single cluster in the global clustering results and the boundary points corresponding to the new cluster will be updated as some old boundary points are no longer qualified as boundary points. In these three situations, points from the same global cluster are distributed to different local sites, updating local clustering results accordingly.

For cases in Fig. 3(1-a) and (1-b), some or all boundary points of one cluster are surrounded by the boundary of another cluster. These two situations can be identified by the fact that one or more boundary points of a cluster are surrounded by the boundary points of another cluster (the set $points_{\text{in}}$ is not empty). The case in Fig. 3(1-c) can be identified by checking the distance between the boundary points from two clusters and when the minimum distance is sufficiently small (the set $points_{\text{close}}$ is not empty), the two clusters can be merged. The details of cluster processing are presented in Algorithm 2.

The issue of determining whether a boundary point of a cluster is surrounded by the boundary points of another cluster can be reduced to the problem of determining the relationship between a single point and a series of points. For a point x to be surrounded by a set of points *boundary*, x must be located in the denser area of *boundary*, which means that x must be located at the same region as the normal vector of *boundary*, as shown in Theorem 2.

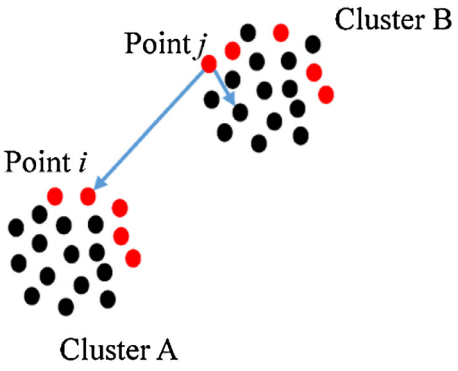


Fig. 4. Point j is the closest boundary point of cluster B for boundary point i of cluster A. Note that the cosine distance between the vector from point j to point i and the normal vector of point j is greater than 0 as the current set of boundary points is insufficient for specifying the shape of cluster B. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

Theorem 2. *A point x is surrounded by a set of boundary points boundary if*:

$$\cos(x - y, NV(y)) \geq 0 \tag{3.3}$$

where $y \in 1NN(boundary, x)$.

According to Theorem 2, the nearest neighbor y of x in *boundary* is identified first and the cosine distance between x–y and the normal vector of y is calculated. The result is then used to determine whether x is surrounded by *boundary*: if the cosine distance is smaller than 0, then x is regarded as not being surrounded by *boundary*; Otherwise, x is surrounded by *boundary*.

To make our method more robust, especially in the presence of isolated boundary points located inside the cluster (inner points misidentified as boundary points), we can use the m nearest neighbors in *boundary* of x instead of the nearest neighbor. We calculate the average position of the m neighbors denoted by y as well as the average vector of the normal vectors of the m neighbors denoted by NV. Then, the cosine distance between x–y and NV is obtained and if the value is greater or equal to 0, x is regarded as being surrounded by *boundary*. The implementation details are presented in Algorithm 3.

3.3. Local data partition

After obtaining the global boundary and cluster information, the global model is sent back to each local site where original data points are to be clustered. All former non-noise points still belong to a certain global cluster, which is the same cluster as their nearest global boundary points. Meanwhile, former local noise points may now belong to a cluster or still be regarded as noises. If a former noise point is surrounded by the boundary of a global cluster (which can be checked using Algorithm 3), it will be assigned to this cluster. Otherwise, it is still regarded as a noise point.

3.4. Discussion

The time complexity of DCUBI-DBSCAN mainly consists of three parts: local processing for local models, global data fusion for the global model and final local processing for final local data partition. Local processing consists of boundary detection and DBSCAN, which has the worst total time complexity of $O(n\log(n) + r^2)$when adopting kd-tree where n is the number of data points and r is the number of points selected as boundary points, which is much smaller than n. Since we select boundary points from the original dataset and traditional clustering algorithms are only applied to these boundary points, the execution time of local clustering can be reduced significantly, especially for clustering algorithms with high time complexity. Global processing has time complexity $O((c_{avg} \cdot n_s)^2 n_{avg}\log(n_{avg}))$where c_{avg} is the average cluster number of each site and n_{avg} is the average number of boundary points in each cluster. The final local partition has time complexity $O(n\log(n_b))$ where n_b is the number of global boundary points. Due to the flexibility of DCUBI, many traditional clustering algorithms can be adopted as required. In particular, centralized clustering algorithms with high time complexity are expected to receive more benefits in terms of time savings from our scheme.

The possible factors that may influence the clustering results of DCUBI-DBSCAN are analyzed as follows:

(a) In high dimensional spaces, the Euclidean distance cannot reliably characterize the relative distances among data points. Since DCUBI relies on the kNN search, the boundary points detected may not be accurate for high dimensional datasets. The proposed method can reliably deal with datasets less than 10 dimensions. The performance of other neighborhood queries in high dimensional spaces will be investigated in our future work.

(b) If the number of boundary points selected is insufficient, especially when the original dataset is small, the boundary points cannot describe the cluster shape properly as they cannot form a tightly enclosed shape.

In Fig. 4, an insufficient number of boundary points marked in red are selected for both cluster A and cluster B. For boundary point i of cluster A, its closest boundary point of cluster B is point j. Since the angle between the normal vector of point j and the vector from point j to point i is less than 90°, our algorithm will assume that point i is surrounded by the boundary points of cluster B and cluster A and cluster B will be merged incorrectly.

4. Experiments

We mainly evaluated the effectiveness and scalability of DCUBI with DBSCAN and other well-known centralized clustering algorithms (APCLUSTER and SC) were also plugged into DCUBI to demonstrate its flexibility. We conducted the experiments on a Linux Workstation with dual Intel Xeon 2.0 GHz CPUs and 128 GB RAM. The effectiveness of DCUBI was measured using four popular metrics: Adjusted Rand-Index (AR), Rand-Index (RI), Mirkin's-Index

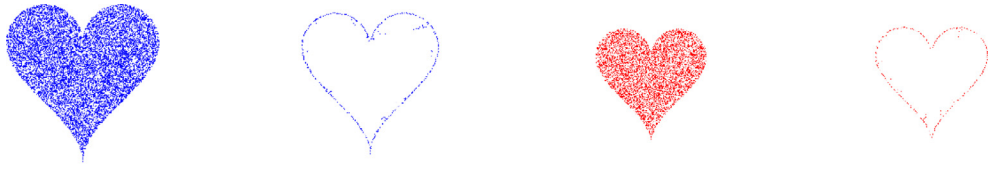


Fig. 5. The original datasets and the extracted boundaries in two local sites.

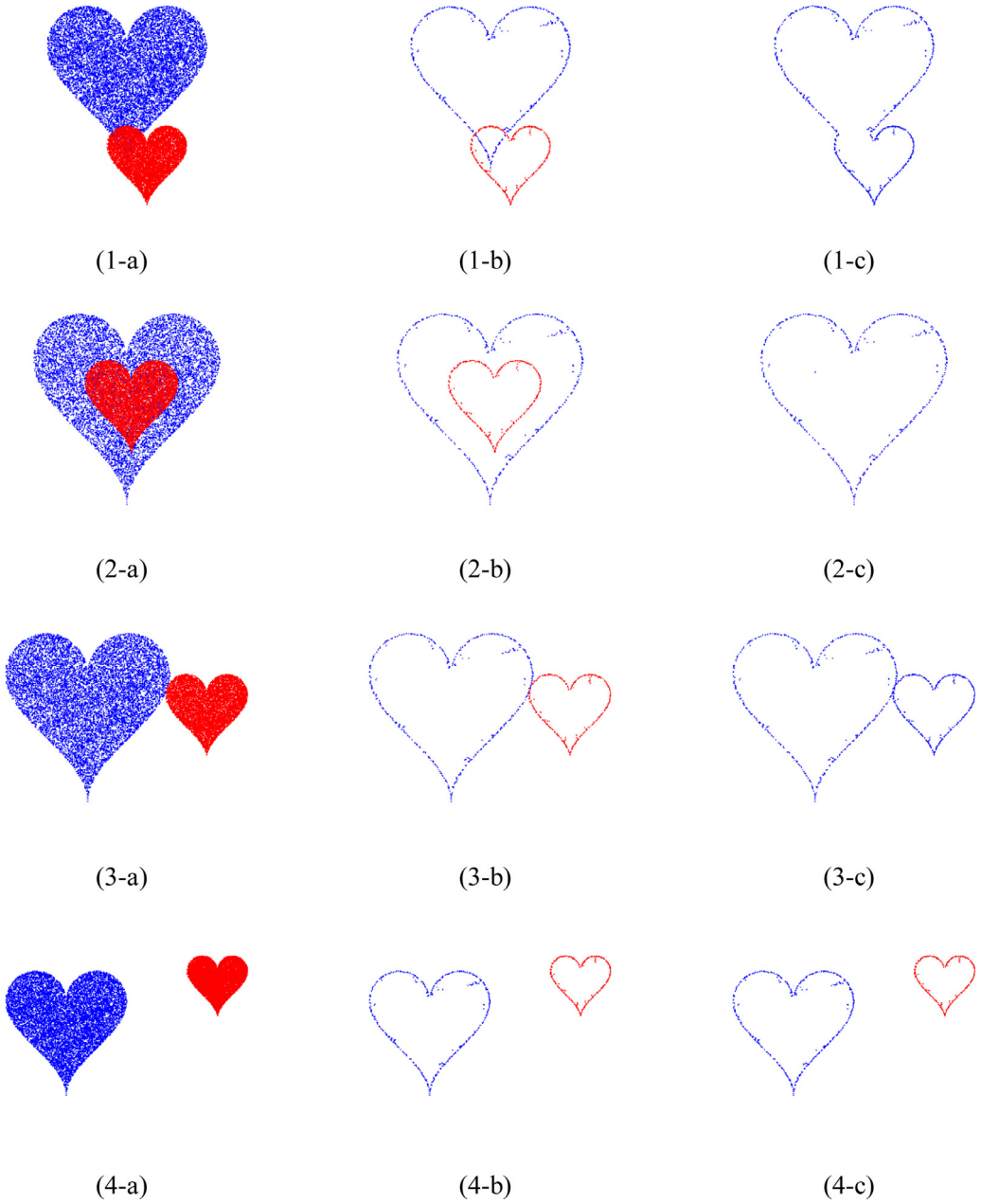


Fig. 6. Four different relationships between the two clusters from two local sites. The three columns (from left to right) show the original data, the boundaries extracted separately and the global boundary after fusion.

(MI) and Hubert's-Index (HI) [29]. All metric values have maximum values 1 and a large value indicates that the two partitions are similar for AR, RI and HI while the reverse is the case for MI. The efficiency of DCUBI was measured by the total time consumed locally and globally.

4.1. Quality

4.1.1. Basic fusion

Firstly, we evaluated the effectiveness of boundary and cluster fusion based on the different situations discussed in Section 3.2. Two local sites were set up with each site having only a single cluster. Site 1 had a heart shape 2D data with 10,456 points and site 2 contained a smaller heart shape dataset with 8358 points. In this experiment, 5% of data points were selected as boundary points, as shown in Fig. 5. The four different relationships between the two clusters are shown in Fig. 6 in which the three columns represent the original data, the original boundaries extracted and the global boundary. The metric values were 1 for AR/RI/HI and 0 for MI, which indicates that the clustering results of DCUBI were the same as the centralized clustering (ground truth).

According to Fig. 6, data in the first three cases were merged into a new single cluster while in the last case the two clusters from the two sites were kept unchanged. Note that when two clusters are intersected with each other or one cluster is surrounded by another one, some boundary points will become the inner points of the new cluster and will be removed from the global boundary point set.

4.1.2. Multiple fusion

In the second experiment, there were six local sites with each site containing different numbers of clusters of varying sizes,

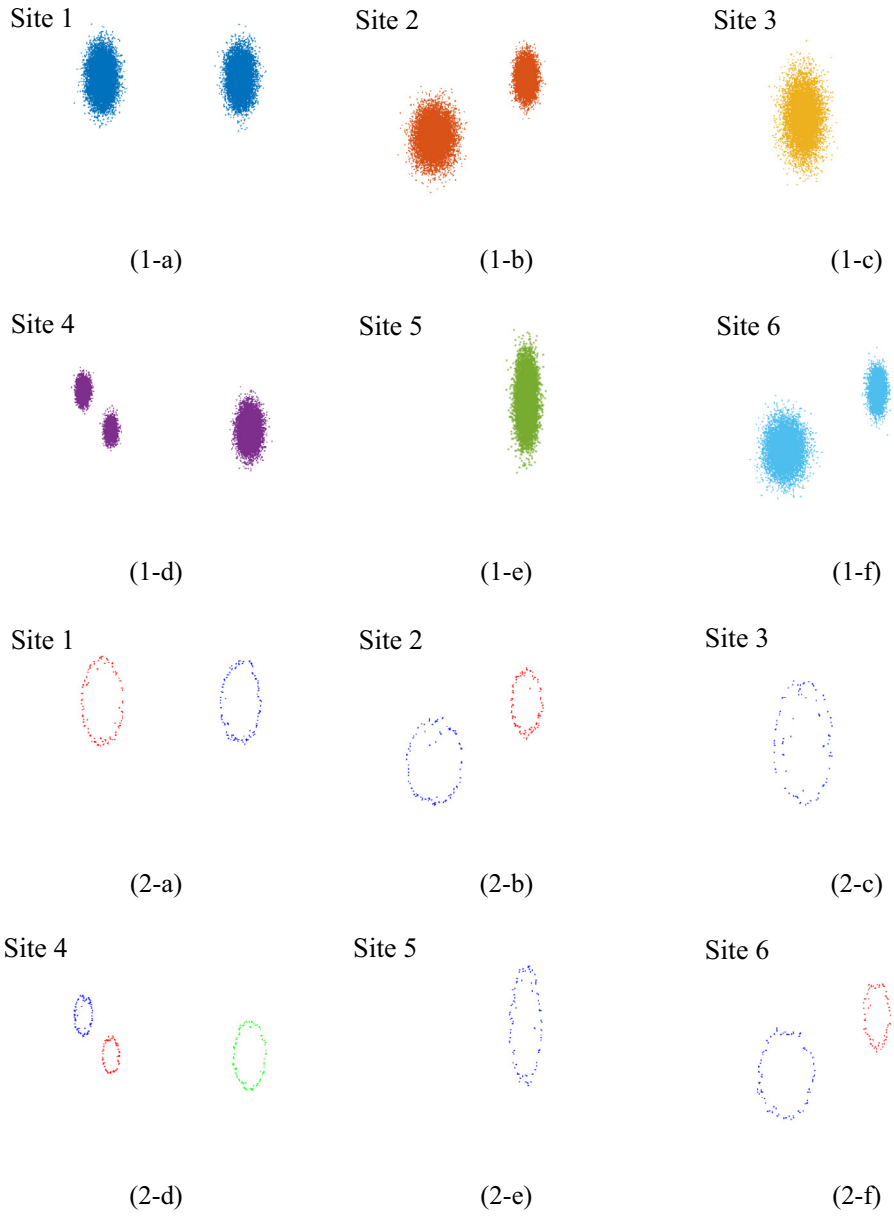


Fig. 7. The original datasets in six local sites (1-a–1-f) and their corresponding boundaries (2-a–2-f).

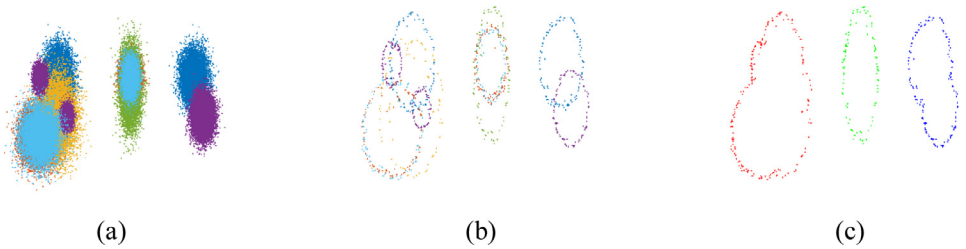


Fig. 8. The effect of boundary fusion: the combination of the original data from each local site (a); the original boundaries from each local site (b) and the resulting global boundaries (c).

as shown in Fig. 7(1-a)–Fig. 7(1-f). The data sizes were 24,300, 20,000, 9000, 25,000, 9000 and 15,000, respectively, and 1% of data points were selected as the boundary points. The corresponding local models (clusters and boundaries) are presented in Fig. 7(2-a)–(2-f). The combination of all original data is presented in Fig. 8(a) while Fig. 8(b) shows all original boundaries. After data fusion, the resulting global boundaries are shown in Fig. 8(c), representing three global clusters. The final data partition pattern of each site is presented in Fig. 9.

In Fig. 8, we can see that the situation was quite complex with multiple sites and clusters. However, after boundary fusion implemented in the global/central site, global boundaries can represent

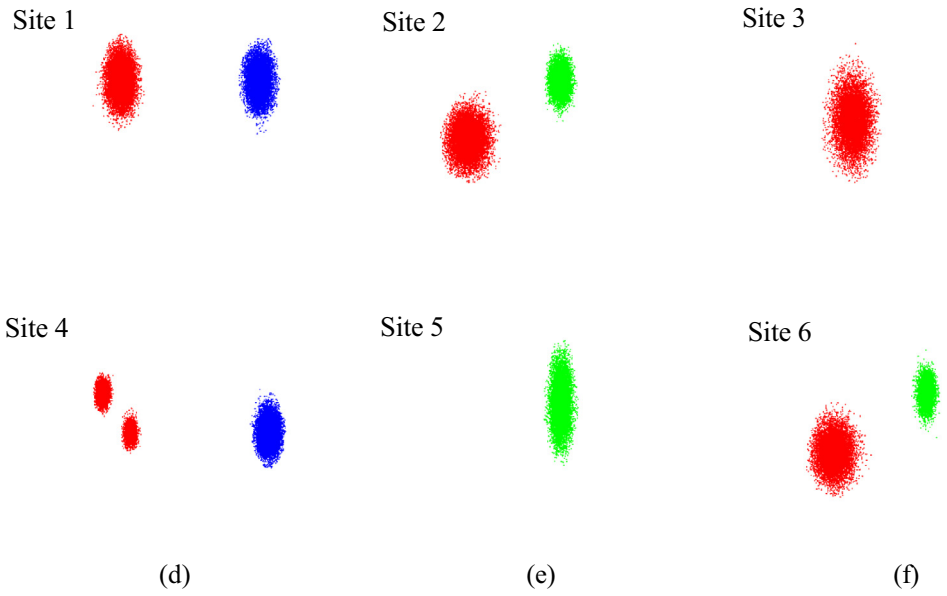


Fig. 9. The final data partition pattern in each local site. Data points belonging to the same global cluster are marked in the same color. Note the difference in grouping pattern compared to Fig. 7. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 10. The original data with three clusters and 34,500 points.

Table 1
The clustering quality with different site numbers using uniform partition.

Site number	2	4	8	16	32	64
AR	0.9789	0.9786	0.9786	0.9618	0.9687	0.5871
RI	0.9897	0.9895	0.9895	0.9813	0.9847	0.7867
MI	0.0103	0.0105	0.0105	0.0187	0.0153	0.2133
HI	0.9793	0.9790	0.9790	0.9626	0.9693	0.5734

the shape of global clusters properly, which can be exploited for refining the local data partition in each site. For example, in site 4, there were three clusters in the original local model while two of them were merged together once the global information was introduced, as these two clusters were contained in the same global cluster.

4.1.3. #Site vs. effectiveness

In the third experiment, we investigated the impact of site number on clustering quality. The original dataset had 34,500 points consisting of 3 clusters, as shown in Fig. 10. Furthermore, we tested both uniform partition (same number of data points for each local site) as well as random partition (different number of data points for each local site) to provide more insights into the performance of distributed clustering.

We first divided the dataset uniformly among sites with the number of sites varied from 2 to 64 and extracted 10% data from each local site as boundary points. The AR/RI/MI/HI values for different site numbers are presented in the Table 1.

Table 2
The clustering quality with different site numbers using random partition.

Site number	2	4	8	16	32	64
AR	0.9791	0.9786	0.9783	0.6757	0.5352	0.5046
RI	0.9897	0.9895	0.9894	0.8312	0.7484	0.7320
MI	0.0103	0.0105	0.0106	0.1688	0.2512	0.2680
HI	0.9794	0.9790	0.9787	0.6624	0.4981	0.4640

Next, we randomly divided the dataset among sites. To account for the randomness, the AR/RI/MI/HI values in Table 2 were averaged over 5 trials.

According to Tables 1 and 2, the general trend is that when the number of sites became large, the quality of distributed clustering dropped. This phenomenon was due to the fact that the data size of a single site could be quite small with a large number of sites and 10% of data points were not sufficient to represent the shape of clusters properly, leading to the issue described in Section 3.4 (Fig. 4) where two separate clusters are merged into a single cluster by mistake.

4.2. Time efficiency

4.2.1. #Site vs. running time

The running time of DCUBI consists of three parts: local clustering (including boundary extraction), global cluster fusion and updating of all local clustering results. For local clustering in the first and last steps, since all sites can process their data in parallel, the longest processing time was regarded as the running time of local clustering.

In the experiment, the original dataset contained 300,000 data points of 8 dimensions, which was composed of three Gaussian clusters located relatively distant from each other. We randomly divided the 300,000 points equally among different sites while the number of sites varied from 2 to 20 and 1% of data points were selected as the boundary points. The time consumed by each part for each site number is shown in Fig. 11. Note that, the final clustering results were the same as the ground truth as the gaps among these clusters were large.

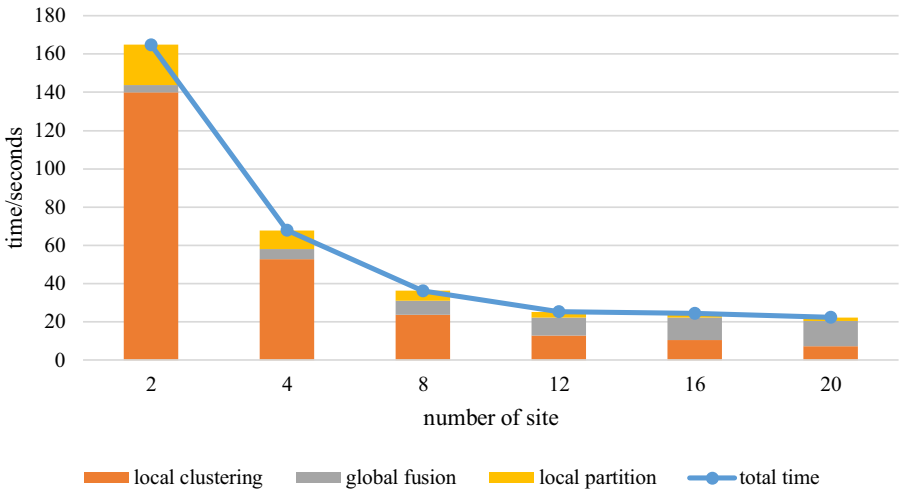


Fig. 11. The time consumed by each part of DCUBI with different numbers of sites.

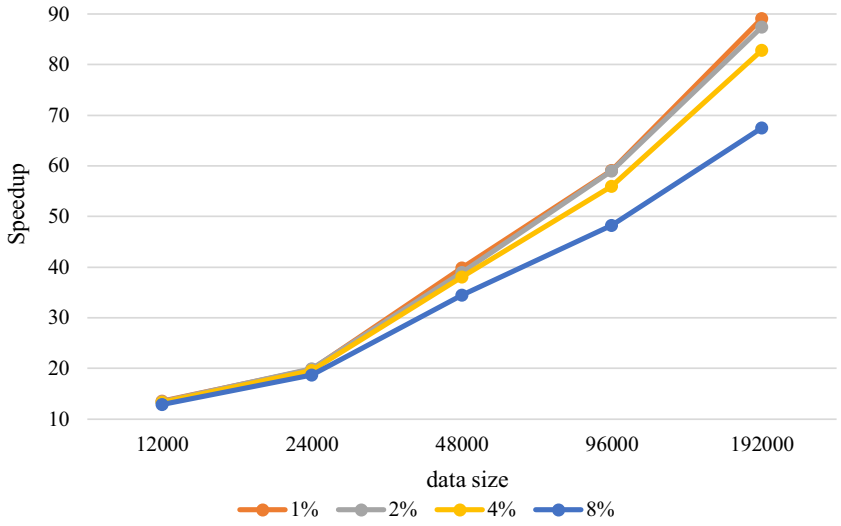


Fig. 12. The speedup curves with regard to the size of the Gaussian mixture datasets. It shows that the local clustering procedure of DCUBI features good scalability and is significantly more time efficient than performing traditional clustering on the original dataset.

In Fig. 11, when the site number increased from 2 to 20, the total running time dropped from 164.8 s to 22.3 s. The main source for time savings was from the local clustering, as each site had less amount of data to be processed by DBSCAN. By contrast, when the number of sites increased, the processing time on the global site went up as more sites were involved in the cluster fusion process. Note that as more and more sites are involved in the distributed clustering, the amount of time saved by local clustering may be less significant than the extra time required by global fusion and the overall running time will not keep reducing.

4.2.2. Scalability of local clustering

As shown in Fig. 11, if the local site has a large amount of data, the main time consumed in DCUBI is on the local clustering process. Compared to other distributed clustering methods, we first extract the boundary points and clustering is only implemented on the boundary points instead of performing clustering first and then the local model extraction. In this experiment, we compared the total time for boundary extraction and clustering on boundaries with the time required by DBSCAN on the original dataset. The datasets adopted followed the same distribution as the dataset in 4.2.1 but with different data sizes varied from 12,000 to 192,000. The two methods produced identical clustering results and the speedup curves for different $\alpha_{boundary}$ values are plotted in Fig. 12. The speedup was up to more than 80 times in our experiments with data size of 192,000, which implies significant time savings for local sites with large volume of data. Also, the less the number of boundary points extracted, the less the processing time required by the local clustering procedure of DCUBI.

4.3. Real dataset

We further tested DCUBI on a real dataset about the major felony incidents of New York City [30]. We chose the 2D crime location data related to Queens, Brooklyn and Staten Island with 22,693 incidents on distinct locations in total. The dataset was distributed equally to different numbers of sites varied from 2 to 10 and 10% of data points were selected as boundary points. Table 3 shows the comparison between our distributed clustering and centralized DBSCAN clustering (ground truth), showing extremely similar clustering results.

4.4. Flexibility

The efficiency and effectiveness of the proposed distributed clustering scheme have been demonstrated in the above

Fig. 13. The dataset in site 1 (left), the dataset in site 2 (middle) and the global dataset (right).

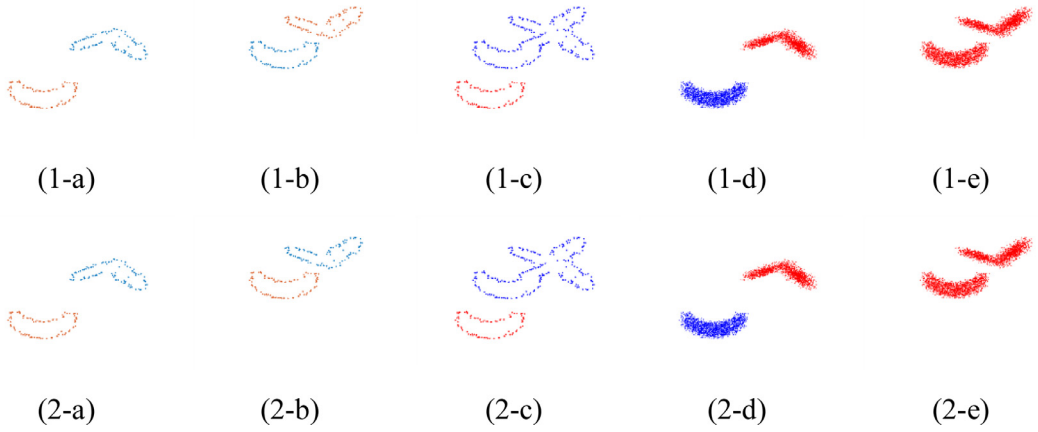


Fig. 14. The local models, global models and final clustering results of DCUBI-SC and DCUBI-APCLUSTER. The first row is the results of DCUBI-SC while the second row is the results of DCUBI-APCLUSTER. The first two columns represent the local models and the third column represents the global models. The fourth and fifth columns represent the final data partition of each site. Data points belonging to the same global cluster are marked in the same color in the last two columns. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Table 3
The clustering results of the felony incident dataset with different site numbers.

Site number	2	4	6	10
AR	0.9970	0.9951	0.9937	0.9352
RI	0.9991	0.9986	0.9981	0.9816
MI	0.0009	0.0014	0.0019	0.0184
HI	0.9982	0.9971	0.9963	0.9631

experiments. Since DCUBI is a general distributed clustering scheme, similar to DBSCAN, other clustering algorithms especially those with high time complexity can be also readily plugged into DCUBI. SC and APCLUSTER are two famous clustering algorithms with competent performance while both suffer from high time complexity: the time complexity of SC is $O(n^3)$and the time complexity of APCLUSTER is $O(n^2t)$ where t is the number of iterations. We applied these two clustering algorithms directly into DCUBI without any modification to demonstrate the flexibility of DCUBI. The dataset in each local site and the global dataset are shown in Fig. 13. The local models, global models as well as the final partitions of the two clustering methods are shown in Fig. 14. On this relatively simple task, both algorithms produced correct results. Note that, two local clusters were merged into a single cluster after taking the global model into consideration, which is demonstrated in Fig. 14. In fact, any clustering algorithm that can properly group boundary points into clusters can be incorporated into the framework of DCUBI.

5. Conclusion

In this paper, we presented DCUBI, a flexible and scalable distributed clustering scheme, to deal with distributed clustering problems efficiently and effectively. The main idea is to perform standard clustering on selected boundary points first on each local site, to achieve efficient local clustering and obtain local models. Then, boundary points and their cluster labels are sent to a central site as the local representatives. The central site implements boundary as well as cluster fusion to form labeled global boundary points as the global model. Finally, local sites update their local clustering results with the help of the global model. By doing so, clustering can be done both effectively and efficiently without the need to put all the data into a centralized site.

In the experimental studies, we chose DBSCAN as the base clustering algorithm in the hope to overcome its high time complexity issue and extend it to the scenario of distributed clustering. Extensive experiments on both synthetic and real-world datasets confirmed that the proposed distributed clustering scheme can efficiently handle a variety of distributed datasets effectively, achieving almost identical clustering results as the original DBSCAN. We also demonstrated that other popular clustering algorithms with high time complexity such as APCLUSTER and SC can be directly plugged into the framework of DCUBI. Certainly, other clustering algorithms with low time complexity can be also employed in the local clustering process if the clustering tasks are simple.

As to future work, it is important to further reduce the time complexity of boundary point selection and improve the applicability of DCUBI to higher dimensional problems. To reduce the burden on communication between the central site and local sites, it is also desirable to minimize the number of boundary points needed while maintaining the clustering quality.

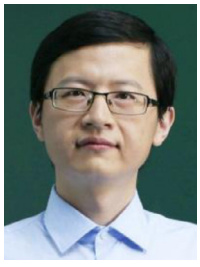
References

[1] Unit count information as of June 30, 2016. Available: http://stock.walmart.com/investors/financial-information/unit-counts-and-square-footage/default.aspx.
[2] E. Januzaj, H. Kriegel, M. Pfeifle, Scalable density-based distributed clustering, in: Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 2004, Pisa, Italy, 2004, pp. 231–244.
[3] E. Januzaj, H. Kriegel, M. Pfeifle, Towards effective and efficient distributed clustering, in: Proceedings of ICDM 2003 Workshop on Clustering Large Dataset, ICDM 2003, Florida, USA, 2003.

[4] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, P. Luo, Distributed data mining: a survey, Inf. Technol. Manag. 13 (2012) 403–409.
[5] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.
[6] E. Januzaj, H. Kriegel, M. Pfeifle, DBDC: density based distributed clustering, in: Proceedings of International Conference on Extending Database Technology, EDBT 2004, Crete, Greece, 2004, pp. 88–105.
[7] Y. Liu, Z. Liu, Scalable local density-based distributed clustering, Expert Syst. Appl. 38 (2011) 9491–9498.
[8] S. Datta, C. Giannella, H. Kargupta, Approximate distributed K-means clustering over a peer-to-peer network, IEEE Trans. Knowl. Data Eng. 21 (2009) 1372–1388.
[9] M. Klusch, S. Lodi, G. Moro, Distributed clustering based on sampling local density estimates, in: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, Acapulco, Mexico, 2003, pp. 485–490.
[10] P. Shen, C. Li, Distributed information theoretic clustering, IEEE Trans. Signal Process. 62 (2014) 3442–3453.
[11] H.A. Salam, K. Maly, R. Mukkamala, M. Zubair, D. Kaminsky, Scheduling-capable autonomic manager for policy based IT change management system, in: Proceedings of 12th International IEEE Enterprise Distributed Object Computing Conference, EDOC'08, Washington, DC, USA, 2008, pp. 386–392.
[12] C. Xia, W. Hsu, M.L. Lee, B.C. Ooi, BORDER: efficient computation of boundary points, Knowl. Data Eng. IEEE Trans. 18 (2006) 289–303.
[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 1996, Portland, USA, 1996, pp. 226–231.
[14] J. Gan, Y. Tao, DBSCAN revisited: mis-claim, un-fixability, and approximation, in: Proceedings of ACM SIGMOD International Conference on Management of Data. SIGMOD 2015, Melbourne, Australia, 2015, pp. 519–530.
[15] B.J. Frey and D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[16] K.M. Hammouda, M.S. Kamel, Models of distributed data clustering in peer–to-peer environments, Knowl. Inf. Syst. 38 (2014) 303–329.
[17] H.P. Kriegel, P. Kroger, A. Pryakhin, M. Schubert, Effective and efficient distributed model-based clustering, in: Proceedings of Fifth IEEE International Conference on Data Mining, ICDM'05, Houston, USA, 2005.
[18] R. Sproull, Refinements to nearest-neighbor searching ink-dimensional trees, Algorithmica 6 (1991) 579–589.
[19] F. Zhu, N. Ye, W. Yu, S. Xu, and G. Li, Boundary detection and sample reduction for one-class support vector machines, Neurocomputing, 123 (2014) 166–173.
[20] B.-Z. Qiu, F. Yue, J.-Y. Shen, BRIM: an efficient boundary points detecting algorithm, in: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2007, Nanjing, China, 2007, pp. 761–768.
[21] Y. Li, M. Liam, Selecting critical patterns based on local geometrical and statistical information, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 1189–1201.
[22] A. Moreira, M.Y. Santos, Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points, in: Proceedings of the 2007 International Conference on Computer Graphics Theory and Applications, GRAPP 2007, Barcelona, Spain, 2007, pp. 61–68.
[23] A. López Chau, X. Li, W. Yu, J. Cervantes, P. Mejía-Álvarez, Border samples detection for data mining applications using non convex hulls, in: Proceedings of the 10th Mexican International Conference on Artificial Intelligence, MICAI 2011, Cancún, Mexico, 2011, pp. 261–272.
[24] V. Estivill-Castro, I. Lee, Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets, in: Proceedings of the 5th International Conference on Geocomputation, GeoComputation 2000, Greenwich, UK, 2000.
[25] D. Liu, G.V Nosovskiy, O. Sourina, Effective clustering and boundary detection algorithm based on Delaunay triangulation, Pattern Recognit. Lett. 29 (2008) 1261–1273.
[26] J. Yang, V. Estivill-Castro, S.K. Chalup, Support vector clustering through proximity graph modelling, in: Proceedings of the 9th International Conference on Neural Information Processing, ICONIP 2002, Singapore, 2002, pp. 898–903.
[27] F. Korn, S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, SIGMOD Rec. 29 (2000) 201–212.
[28] Q. Tong, X. Li, B. Yuan, A highly scalable clustering scheme using boundary information, Pattern Recognit. Lett. 89 (2017) 1–7.
[29] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1985) 193–218.
[30] NYPD 7 Major Felony Incidents, 2017. Available: https://data.cityofnewyork.us/Public-Safety/NYPD-7-Major-Felony-Incidents/hyij-8hr7?category=Public-Safety&view_name=NYPD-7-Major-Felony-Incidents.

Qiuhui Tong receives her B.E. degree in Electrical Engineering from Beijing Jiaotong University in 2013. She is currently a student in the Master of Data Science program in Graduate School at Shenzhen, Tsinghua University. Her research interests include clustering and visualization.

Xiu Li received her Ph.D. degree in computer integrated manufacturing in 2000. Since then, she has been working in Tsinghua University. Her research interests include data mining, business intelligence systems, knowledge management systems and decision support systems.



Bo Yuan is mostly interested in Data Mining, Evolutionary Computation and Parallel Computing. He received the B.E. degree from Nanjing University of Science and Technology, China, in 1998, and the M.Sc. and Ph.D. degrees from The University of Queensland, Australia, in 2002 and 2006, respectively. From 2006 to 2007, he was a Research Officer on a project funded by the Australian Research Council at The University of Queensland. He is currently an Associate Professor in the Division of Informatics, Graduate School at Shenzhen, Tsinghua University. He is the author of more than 70 research papers and the inventor of 4 patents.