IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 12, 2021, accepted August 27, 2021, date of publication September 9, 2021, date of current version September 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111156

An Adaptive Density-Sensitive Similarity Measure Based Spectral Clustering Algorithm and Its Parallelization

GEN ZHANG[1,2], LANJUN WAN[1,2], KUN GONG[1,2], CHANGYUN LI[1,2], AND MANSHENG XIAO[1]

[1]School of Computer Science, Hunan University of Technology, Zhuzhou 412007, China
[2]Hunan Key Laboratory of Intelligent Information Perception and Processing Technology, Hunan University of Technology, Zhuzhou 412007, China

Corresponding author: Lanjun Wan (wanlanjun@hut.edu.cn)

This work was supported in part by the National Natural Science Foundation for Young Scientists of China under Grant 61702177; in part by the Natural Science Foundation of Hunan Province, China, under Grant 2019JJ60048; in part by the National Key Research and Development Project under Grant 2018YFB1700204, Grant 2018YFB1003401, and Grant 2019YFD1101305; and in part by the Key Research and Development Project of Hunan Province under Grant 2019GK2133.

ABSTRACT The clustering effect of the spectral clustering algorithm depends on the calculation of the similarity between samples. Although a better clustering effect of the spectral clustering algorithm can be obtained using the Gaussian kernel function to calculate the similarity between samples, it relies on the setting of the kernel parameter. Therefore, an adaptive density-sensitive similarity measure based spectral clustering (DSSC) algorithm is proposed for improving the clustering effect. Specifically, firstly, the Euclidean distances between samples are calculated to get the nearest neighbors of each sample. Secondly, the standard deviation of distances between each sample and its nearest neighbors is calculated as the density parameter. Thirdly, the density-sensitive distances between each sample and its nearest neighbors are calculated. Finally, the similarities between each sample and its nearest neighbors are calculated to construct a similarity matrix. In addition, the proposed DSSC algorithm is parallelized on Dask distributed parallel computing platform with CPU+GPU, which can improve the computational efficiency of the DSSC algorithm by taking full advantage of the CPU and GPU resources. A series of experiments are conducted to verify the effectiveness of the proposed DSSC algorithm on several synthetic datasets and UCI datasets, and the results show that the DSSC algorithm not only achieves satisfactory clustering results, but also obtains better efficiency of performing large-scale clustering analysis.

INDEX TERMS Dask, density-sensitive, parallelization, similarity, spectral clustering.

I. INTRODUCTION

The clustering algorithm [1] is one of the unsupervised learning algorithms commonly used for data mining, and its purpose is to divide the samples of the same class into the same cluster as many as possible. The clustering algorithms can be divided into density-based clustering, partition clustering, hierarchical clustering, grid-based clustering, and graph-based clustering. The spectral clustering algorithm [2] belongs to graph-based clustering, which has a better clustering effect. It has been successfully applied to fault diagnosis [3], image segmentation [4], text classification [5], healthcare [6], and other fields.

In recent years, the spectral clustering algorithm is still a popular research object in the field of data mining [7]. Xie *et al*. [8] proposed a local standard deviation spectral clustering algorithm, which uses the standard deviations of distances between samples and their nearest neighbors as the kernel parameter of the Gaussian kernel function. Du *et al*. [9] used the local covariance to construct an adjacency matrix to improve the traditional spectral clustering, which ensures that the adjacency matrix is not affected by intersection points. Ye and Sakurai [10] adopted the probability neighborhood measure to calculate the similarities between samples, which can improve the adaptability to the complex data and

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

the clustering accuracy of the spectral clustering algorithm. Park and Zhao [11] used the symmetric double stochastic similarity matrix to construct a Laplacian matrix, which is applied to the single-cell RNA-sequencing and good results are achieved. Zhu *et al*. [12] employed a local representation method to remove the interference of some outliers and sparsification to remove redundant features, and a robust similarity matrix is constructed. Yuan and Zhu [13] developed a spectral clustering algorithm based on fast search of natural neighbors, which can adaptively determine the number of natural neighbors of each sample and the number of clusters. The above research has improved the clustering effect of the spectral clustering algorithm, but the Euclidean distance measure is utilized to calculate the similarities between samples, which cannot well reflect the distribution of samples.

Recently, the density-sensitive similarity measure is often used to improve the clustering effect of the spectral clustering algorithm. Zhang *et al*. [14] proposed a spectral clustering algorithm based on local density adaptive similarity, which adopts the common near neighbor measure method to construct a similarity matrix. Yang *et al*. [15] proposed a spectral clustering algorithm based on density sensitive similarity, which uses an adjustable line segment length measure method to calculate the distances between samples to construct a similarity matrix, and a random matrix is constructed based on the Markov chain. Yan *et al*. [16] improved the spectral clustering algorithm by using the density sensitive similarity measure and optimizing the selection of initial clustering centers of K-Means clustering algorithm. Wang *et al*. [17] adopted the density sensitive similarity measure to construct a similarity matrix and the affinity propagation algorithm to replace K-Means clustering algorithm for the final clustering. The above research has achieved good clustering results, but the parameter values are generally determined by experience, and the constructed similarity matrix is a dense matrix.

Although the spectral clustering algorithm can provide a high clustering accuracy, it usually needs to solve the eigenvalues and eigenvectors of the Laplacian matrix. This process is not only time-consuming but also takes up a lot of memory resources, which is not conducive to perform large-scale clustering analysis. To reduce the running time of the spectral clustering algorithm for large-scale data, the Nyström method is adopted in [18], [19]. Some distributed parallel computing frameworks are used to parallelize the spectral clustering algorithm recently. In [20], [21], Hadoop MapReduce is used to parallelize the spectral clustering algorithm. Taloba *et al*. [22] designed an efficient spectral clustering algorithm based on Spark for large-scale graph processing. Huo *et al*. [23] proposed an efficient parallel spectral clustering algorithm based on Julia. Although the above research can effectively reduce the running time of the spectral clustering algorithm, how to fully utilize all available computing resources of a cluster to improve the efficiency of performing large-scale clustering analysis is still a challenge.

In this paper, an adaptive density-sensitive similarity measure based spectral clustering algorithm is proposed, which can better calculate the similarities between samples and their nearest neighbors to improve the clustering effect to a certain extent. Aiming at the problems of long running time and high memory occupancy for the spectral clustering algorithm, the proposed DSSC algorithm is parallelized on Dask distributed parallel computing platform, which can improve the efficiency of the DSSC algorithm for performing large-scale clustering analysis.

The main contributions of the paper are as follows.

- An adaptive density-sensitive similarity measure method for the spectral clustering algorithm is proposed to improve the clustering effect. At first the nearest neighbors of each sample are determined according to the Euclidean distances between samples, then the density-sensitive distances between each sample and its nearest neighbors are adaptively calculated, and finally the similarities between each sample and its nearest neighbors are calculated to construct a similarity matrix.
- The proposed DSSC algorithm is parallelized on Dask distributed parallel computing platform with CPU+GPU, which can improve the computational efficiency of the DSSC algorithm by taking full advantage of the CPU and GPU resources.
- A series of experiments are conducted to verify the effectiveness of the proposed DSSC algorithm on several synthetic datasets and UCI datasets, and the results show that the DSSC algorithm not only achieves satisfactory clustering results, but also obtains better efficiency of performing large-scale clustering analysis.

The rest of the paper is organized as follows. Section II outlines NJW algorithm and Dask. Section III describes the proposed DSSC algorithm and its parallelization. Section IV presents the experimental results and analysis. Section V gives the conclusion.

II. BACKGROUND

A. OVERVIEW OF NJW ALGORITHM

The spectral clustering originates from graph theory, which transforms a clustering problem into a graph cut problem in essence. It can be divided into 2-way spectral clustering and multi-way spectral clustering according to the graph cut criteria. The Ng-Jordan-Weiss (NJW) algorithm [2] which belongs to the multi-way spectral clustering is widely used for data mining. The NJW algorithm is described in Algorithm 1, which includes the following steps.

Step 1: Construct a similarity matrix S. Assuming that a sample set $X = (x_1, x_2, \ldots, x_n)$ is given, the similarity $s_{i,j}$ between the sample x_i and the sample x_j can be calculated by

$$s_{i,j} = \begin{cases} \exp\left(\dfrac{-\|x_i - x_j\|^2}{2\sigma^2}\right), & \text{if } i \neq j; \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where σ is a scale parameter.

Step 2: Construct a degree matrix D. The degree matrix is a diagonal matrix, so the value of the i-th element on the

Algorithm 1 NJW Algorithm

Input: The scale parameter σ, the number of clusters c

Output: The clustering results

1: Construct a similarity matrix S by (1);
2: Construct a degree matrix D by (2);
3: Construct a Laplacian matrix: $L = D - S$;
4: Normalize L by (3) and obtain a normalized Laplacian matrix L_{sym};
5: Perform eigen-decomposition for L_{sym} and select eigenvectors corresponding to the first c maximum eigenvalues to construct a new matrix V;
6: Standardize V by (4) to obtain a new matrix Y;
7: Perform clustering analysis for Y using K-Means clustering algorithm;

diagonal can be calculated by

$$d_{i,i} = \sum_j s_{i,j}. \tag{2}$$

Step 3: Construct a normalized Laplacian matrix L_{sym}. At first the Laplacian matrix L can be calculated as follows: $L = D - S$, and then the normalized Laplacian matrix can be constructed by

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}. \tag{3}$$

Step 4: Perform eigen-decomposition for the normalized Laplacian matrix L_{sym}, and the eigenvectors corresponding to the first c maximum eigenvalues are selected to construct a new matrix $V = (v_1, v_2, \ldots, v_c)^T$, where V is a $n \times c$ matrix. A new matrix Y can be obtained by standardizing the matrix V as follows:

$$y_{i,j} = \frac{v_{i,j}}{\sqrt{\sum_j v_{i,j}^2}}. \tag{4}$$

Step 5: The K-Means clustering algorithm is used to perform clustering analysis for the matrix Y.

B. OVERVIEW OF DASK

Dask [24] is a lightweight distributed parallel computing platform based on Python, which provides two types of APIs: the low-level APIs and the high-level APIs. The low-level APIs include Delayed and Future, and the high-level APIs include Array, Dataframe, and Bag. Users can construct a task graph using these APIs. Furthermore, Dask also provides three kinds of schedulers: multithreading, multiprocessing, and distributed. The multithreading or multiprocessing scheduler can only be used on a single node via threads or processes. The distributed scheduler can be used on a cluster with multiple workers. Fig. 1 depicts the flow of processing a task on a Dask cluster. Firstly, the client creates a task using APIs provided by Dask and submits the task to the distributed scheduler. Secondly, the distributed scheduler divides the task into several subtasks. Thirdly, these subtasks are assigned to several worker nodes. Each worker node processes the subtasks assigned to it independently, and the obtained subresults are returned to the distributed scheduler. Finally, the obtained results are returned to the client.

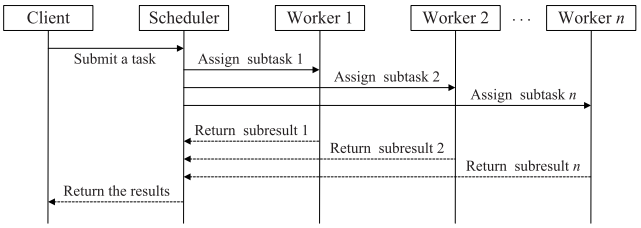


FIGURE 1. Diagram of processing a task on a Dask cluster.

III. THE PROPOSED DSSC ALGORITHM AND ITS PARALLELIZATION

A. THE PROPOSED DSSC ALGORITHM

The key of the spectral clustering algorithm is the calculation of the similarity between samples, and a good similarity measure method can improve the clustering effect of the spectral clustering algorithm. Therefore, an adaptive density-sensitive similarity measure method is proposed to optimize the construction of the similarity matrix for the spectral clustering algorithm. Assuming that the number of the nearest neighbors of each sample is k, the sample x_i has k nearest neighbors: $x_{i.1}, x_{i.2}, \ldots, x_{i.k}$, and the density-sensitive distance between the sample x_i and its l-th nearest neighbor can be calculated by

$$d(x_i, x_{i.l}) = \sum_{t=1}^{k-1} als\,(p_t, p_{t+1})\,. \tag{5}$$

In (5), p_t represents a path point between the sample x_i and its l-th nearest neighbor, where $1 \le t \le k$. Specifically, the path point p_1 is one of among remaining $k-1$ nearest neighbors excluding the l-th nearest neighbor, which has the shortest Euclidean distance from the sample x_i. The path point p_t is one of among remaining $k-t$ nearest neighbors excluding the path points $p_1, p_2, \ldots, p_{t-1}$ and the l-th nearest neighbor, which has the shortest Euclidean distance from the path point p_{t-1}, where $2 \le t \le k-1$. The path point p_k is the l-th nearest neighbor of the sample x_i. $als\,(p_t, p_{t+1})$ is the adjustable line segment length between the path point p_t and the path point p_{t+1}, which can be calculated as follows:

$$als\,(p_t, p_{t+1}) = \left(\rho_i^{dist(p_t, p_{t+1})} - 1\right)^{\delta_i}. \tag{6}$$

In (6), $dist\,(p_t, p_{t+1})$ is the Euclidean distance between the path point p_t and the path point p_{t+1}, the standard deviation of distances between the sample x_i and its k nearest neighbors is $\delta_i = \frac{1}{\sqrt{k}}\sqrt{\sum_{t=1}^{k} (dist\,(x_i, x_{i.t}) - \mu_i)^2}$, where $\mu_i = \frac{1}{k}\sum_{t=1}^{k} dist\,(x_i, x_{i.t})$, and $\rho_i = \exp\left(\delta_i^{-1}\right)$ is an adaptive density parameter. The density parameter ρ_i of the sample x_i is determined according to the standard deviation of distances between the sample x_i and its k nearest neighbors, which

overcomes the weakness of manually setting the density parameter. Therefore, the similarity between the sample x_i and its l-th nearest neighbor can be calculated by

$$s_{i,l} = \frac{1}{d(x_i, x_{i,l}) + 1}. \tag{7}$$

The proposed adaptive density-sensitive similarity measure based spectral clustering algorithm is described in Algorithm 2, which includes the following steps.

Step 1: Construct a similarity matrix. Supposing that there is a sample set $X = (x_1, x_2, \ldots, x_n)$, the calculation of the similarities between each sample and its nearest neighbors can be described as follows. Firstly, the Euclidean distance $dist(x_i, x_j)$ between the sample x_i and the sample x_j is calculated as follows: $dist(x_i, x_j) = \|x_i - x_j\|_2$. Secondly, k nearest neighbors of each sample are determined according to the shortest Euclidean distance. Thirdly, $k-1$ path points of the path from the sample x_i to its l-th nearest neighbor are determined according to the shortest Euclidean distance. Fourthly, the density-sensitive distance $d(x_i, x_{i,l})$ between the sample x_i and its l-th nearest neighbor is calculated by (5), where $1 \le l \le k$. Fifthly, the similarity $s_{i,l}$ between the sample x_i and its l-th nearest neighbor is calculated by (7), where $1 \le l \le k$, and the similarities between the sample x_i and its non-nearest neighbors are set to 0. Finally, the similarity matrix $S_{n\times n} = (s_1, s_2, \ldots, s_i, \ldots, s_n)$ is obtained, where $s_i = \left(s_{i,1}, s_{i,2}, \ldots, s_{i,n}\right)$.

Step 2: Construct a degree matrix. The value of the i-th element on the diagonal is calculated by (2), and the degree matrix $D_{n\times n}$ is obtained, where $1 \le i \le n$.

Step 3: Construct a normalized Laplacian matrix. At first the Laplacian matrix L is constructed as follows: $L = D_{n\times n} - S_{n\times n}$, and then the normalized Laplacian matrix L_{sym} is obtained by (3).

Step 4: Select the eigenvectors to construct a new matrix. Firstly, the eigen-decomposition is performed for the matrix L_{sym} to obtain z eigenvalues and z eigenvectors. Secondly, the eigenvectors corresponding to the first c minimum eigenvalues are selected from z eigenvectors to construct a new matrix $V_{n\times c} = (v_1, v_2, \ldots, v_c)^T$. Finally, the matrix $V_{n\times c}$ is standardized by (4) to get a new matrix $Y_{n\times c}$.

Step 5: Perform clustering analysis. The K-Means clustering algorithm is used to perform clustering analysis for the matrix $Y_{n\times c}$, and the clustering results are obtained.

B. PARALLELIZATION OF THE DSSC ALGORITHM

1) ANALYSIS OF PARALLELIZATION STRATEGY

Table 1 presents the running time of four main stages of the proposed DSSC algorithm obtained using one CPU core, eight CPU cores, and one GPU on a single worker node for a synthetic dataset, respectively. The four main stages include constructing a similarity matrix (stage 1), constructing a degree matrix and a normalized Laplacian matrix (stage 2), performing eigen-decomposition and selecting eigenvectors to construct a new matrix (stage 3), and performing clustering analysis using K-Means clustering algorithm (stage 4).

Algorithm 2 The Proposed DSSC Algorithm

Input: n samples, the number of nearest neighbors k, the number of clusters c
Output: The clustering results

1: **for** $i \leftarrow 1$ **to** n **do**
2: **for** $j \leftarrow 1$ **to** n **do**
3: Calculate the Euclidean distance between the sample x_i and the sample x_j;
4: **end for**
5: **end for**
6: Determine k nearest neighbors of each sample;
7: **for** $i \leftarrow 1$ **to** n **do**
8: **for** $l \leftarrow 1$ **to** k **do**
9: Determine the path point p_k, which is the l-th nearest neighbor of x_i;
10: Determine the path point p_1, which is one of among remaining $k-1$ nearest neighbors of x_i and has the shortest Euclidean distance from x_i;
11: **for** $t \leftarrow 2$ **to** $k-1$ **do**
12: Determine the path point p_t, which is one of among remaining $k-t$ nearest neighbors of x_i and has the shortest Euclidean distance from the path point p_{t-1};
13: **end for**
14: Calculate the density-sensitive distance $d(x_i, x_{i,l})$ by (5);
15: Calculate the similarity $s_{i,l}$ by (7);
16: **end for**
17: **end for**
18: Construct a degree matrix $D_{n\times n}$ by (2);
19: Construct a Laplacian matrix $L \leftarrow D_{n\times n} - S_{n\times n}$;
20: Get a normalized Laplacian matrix L_{sym} by (3);
21: Perform eigen-decomposition for L_{sym} and select the eigenvectors corresponding to the first c minimum eigenvalues to construct the matrix $V_{n\times c}$;
22: Standardize $V_{n\times c}$ by (4) to get the matrix $Y_{n\times c}$;
23: Perform clustering analysis for $Y_{n\times c}$ using K-Means clustering algorithm;

TABLE 1. Running time of each stage of the DSSC algorithm (seconds).

Platform	Stage 1	Stage 2	Stage 3	Stage 4
CPU (1 core)	45.595	47.196	391.054	0.034
CPU (8 cores)	14.528	15.216	40.813	0.028
GPU	18.396	0.285	27.428	0.018

It can be seen from Table 1 that the running time of these four stages obtained using eight CPU cores are decreased by 68.14%, 67.76%, 89.56%, and 17.65% compared with that obtained using one CPU core, respectively. The results show that it is necessary to parallelize the DSSC algorithm.

As shown in Table 1, the running time of stage 1 obtained using eight CPU cores is 21.03% less than that obtained using one GPU, this is because the 8-core CPU is more suitable to perform the complicated three-layer for-loop used for constructing a similarity matrix (see lines 7-17 in Algorithm 2)

than the GPU. The running time of stage 2, stage 3, and stage 4 obtained using one GPU are decreased by 99.81%, 32.80%, and 35.71% compared with that obtained using eight CPU cores, respectively, which shows that stage 2, stage 3, and stage 4 are more suitable to be executed on the GPU. Therefore, the parallelization strategy of the proposed DSSC algorithm is as follows: stage 1 can be parallelized on CPU, and stage 2, stage 3, and stage 4 can be parallelized on GPU.

2) IMPLEMENTATION OF THE PARALLEL DSSC ALGORITHM

According to the parallelization strategy discussed in Section III-B1, the DSSC algorithm is parallelized on Dask distributed parallel computing platform with CPU+GPU, as shown in Fig. 2. Firstly, the similarity matrix is constructed on CPU of each worker node in parallel, and the obtained similarity matrix is copied from CPU to GPU. Secondly, the degree matrix and the normalized Laplacian matrix are constructed on GPU of each worker node in parallel. Thirdly, the eigen-decomposition is performed and the eigenvectors are selected to construct a new matrix on GPU of each worker node in parallel. Fourthly, the K-Means clustering is performed on GPU of each worker node in parallel, and the clustering results are copied from GPU to CPU. Finally, the clustering results are gathered from each worker node to the master node.

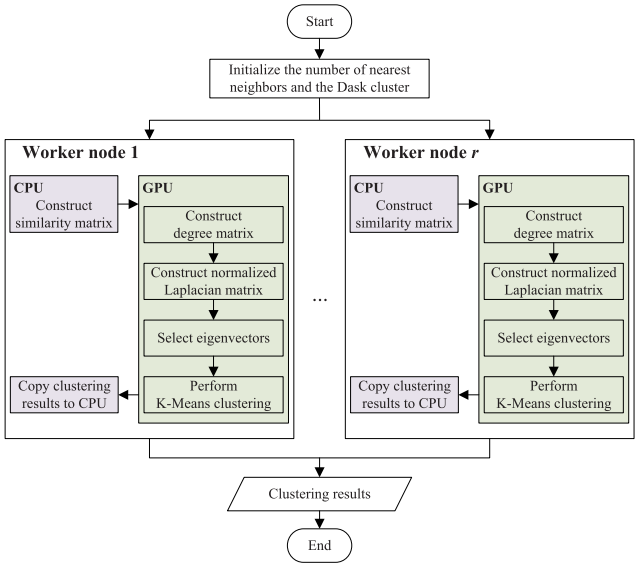


FIGURE 2. Flowchart of the parallel DSSC algorithm.

The parallel DSSC algorithm is described in Algorithm 3, which includes the following steps.

Step 1: Divide the sample set. Supposing that the number of samples is n and the block size is m, n samples are divided into n/m blocks, n/m blocks are distributed to r worker nodes evenly, and all CPU threads or GPU threads are used to process $(n/m)/r$ blocks in parallel on each worker node.

Step 2: Parallelly construct n/m similarity matrices on r worker nodes with CPU. For the sample set of the ω-th block $X^{\omega} = \left(x_1^{\omega}, x_2^{\omega}, \ldots, x_m^{\omega}\right)$, firstly, the Euclidean distance

Algorithm 3 The Proposed Parallel DSSC Algorithm

Input: n samples, the number of nearest neighbors k, the number of clusters c, the block size m

Output: The clustering results

Divide n samples into n/m blocks;
Distribute n/m blocks to r worker nodes evenly;
for all r worker nodes with CPU **in parallel do**
  for $\omega \leftarrow 1$ **to** n/m **do**
    for $i \leftarrow 1$ **to** m **do**
      for $j \leftarrow 1$ **to** m **do**
        Calculate the distance between x_i^{ω} and x_j^{ω};
      end for
    end for
    Determine k nearest neighbors of each sample in the ω-th block;
    for $i \leftarrow 1$ **to** m **do**
      for $l \leftarrow 1$ **to** k **do**
        Determine the path point p_k, which is the l-th nearest neighbor of x_i^{ω};
        Determine the path point p_1, which is one of among remaining $k-1$ nearest neighbors of x_i^{ω} and has the shortest distance from x_i^{ω};
        for $t \leftarrow 2$ **to** $k-1$ **do**
          Determine the path point p_t, which is one of among remaining $k-t$ nearest neighbors of x_i^{ω} and has the shortest distance from p_{t-1};
        end for
        Calculate the density-sensitive distance $d(x_i^{\omega}, x_{i,l}^{\omega})$ by (5);
        Calculate the similarity $s_{i,j}^{\omega}$ by (7);
      end for
    end for
    Copy $S_{m\times m}^{\omega}$ from CPU to GPU;
  end for
end for
for all r worker nodes with GPU **in parallel do**
  for $\omega \leftarrow 1$ **to** n/m **do**
    Construct a degree matrix $D_{m\times m}^{\omega}$ by (2);
    Construct a Laplacian matrix $L^{\omega} \leftarrow D_{m\times m}^{\omega} - S_{m\times m}^{\omega}$;
    Get a normalized Laplacian matrix L_{sym}^{ω} by (3);
    Perform eigen-decomposition for L_{sym}^{ω} and select the eigenvectors corresponding to the first c minimum eigenvalues to construct the matrix $V_{m\times c}^{\omega}$;
    Standardize $V_{m\times c}^{\omega}$ by (4) to get the matrix $Y_{m\times c}^{\omega}$;
    Perform clustering analysis for $Y_{m\times c}^{\omega}$ using K-Means clustering algorithm;
    Copy the clustering results from GPU to CPU;
  end for
end for
Gather the clustering results from each worker node to the master node;

$dist(x_i^{\omega}, x_j^{\omega})$ between the sample x_i^{ω} and the sample x_j^{ω} is calculated as follows: $dist(x_i^{\omega}, x_j^{\omega}) = \|x_i^{\omega} - x_j^{\omega}\|_2$, where $1 \leq \omega \leq n/m$ and $1 \leq i, j \leq m$. Secondly, k nearest neighbors

$x_{i.1}^{\omega}, x_{i.2}^{\omega}, \ldots, x_{i.k}^{\omega}$ of the sample x_i^{ω} are determined according to the shortest Euclidean distance. Thirdly, $k-1$ path points of the path from the sample x_i^{ω} to its l-th nearest neighbor are determined according to the shortest Euclidean distance. Fourthly, the density-sensitive distance $d(x_i^{\omega}, x_{i.l}^{\omega})$ between the sample x_i^{ω} and its l-th nearest neighbor is calculated by (5), where $1 \leq l \leq k$. Fifthly, the similarity $s_{i,l}^{\omega}$ between the sample x_i^{ω} and its l-th nearest neighbor is calculated by (7), where $1 \leq l \leq k$, and the similarities between the sample x_i^{ω} and its non-nearest neighbors are set to 0. Finally, the ω-th similarity matrix $S_{m\times m}^{\omega}$ is obtained and copied from CPU to GPU.

Step 3: Parallelly construct n/m degree matrices on r worker nodes with GPU. The value of the i-th element on the diagonal is calculated by (2), and the ω-th degree matrix $D_{m\times m}^{\omega}$ is obtained, where $1 \leq i \leq m$ and $1 \leq \omega \leq n/m$.

Step 4: Parallelly construct n/m normalized Laplacian matrices on r worker nodes with GPU. At first the ω-th Laplacian matrix L^{ω} is constructed as follows: $L^{\omega} = D_{m\times m}^{\omega} - S_{m\times m}^{\omega}$, and then the ω-th normalized Laplacian matrix L_{sym}^{ω} is obtained by (3), where $1 \leq \omega \leq n/m$.

Step 5: Parallelly select eigenvectors to construct n/m new matrices on r worker nodes with GPU. Firstly, the eigen-decomposition is performed for the matrix L_{sym}^{ω} to obtain z eigenvalues and z eigenvectors, where $1 \leq \omega \leq n/m$. Secondly, the eigenvectors corresponding to the first c minimum eigenvalues are selected from z eigenvectors to construct a new matrix $V_{m\times c}^{\omega} = \left(v_1^{\omega}, v_2^{\omega}, \ldots, v_c^{\omega}\right)^T$. Finally, the matrix $V_{m\times c}^{\omega}$ is standardized by (4) to get a new matrix $Y_{m\times c}^{\omega}$.

Step 6: Parallelly perform clustering analysis on r worker nodes with GPU. The K-Means clustering algorithm is used to perform clustering analysis for the matrix $Y_{m\times c}^{\omega}$, where $1 \leq \omega \leq n/m$, and the clustering results are obtained and copied from GPU to CPU.

Step 7: Gather the clustering results from each worker node to the master node.

C. ANALYSIS OF TIME COMPLEXITY

The time complexities of the four main stages of the proposed DSSC algorithm are analyzed as follows.

In stage 1, the time complexity of calculating the Euclidean distances between samples is $O(n^2)$, the time complexity of determining k nearest neighbors of each one of n samples is $O(n^2 \log n)$, the time complexity of calculating the similarities between n samples and their own k nearest neighbors (see lines 7-17 in Algorithm 2) is $O(nk^2)$. Therefore, the time complexity of constructing the similarity matrix is $O\left(n^2 + n^2 \log n + nk^2\right)$, and the time complexity of parallelly constructing the similarity matrix on r worker nodes with θ available CPU threads each is $O\left((n^2 + n^2 \log n + nk^2)/(r\theta)\right)$.

In stage 2, the time complexity of constructing the normalized Laplacian matrix is $O(n^3)$, and the time complexity of parallelly constructing the normalized Laplacian matrix on r worker nodes with γ available GPU threads each is $O(n^3/(r\gamma))$.

In stage 3, the time complexity of performing eigen-decomposition for the normalized Laplacian matrix is $O(n^3)$, and the time complexity of selecting the eigenvectors corresponding to the first c minimum eigenvalues is $O(n \log n)$. Therefore, the time complexity of performing eigen-decomposition and selecting eigenvectors to construct a new matrix is $O(n^3 + n \log n)$, and the time complexity of parallelly performing eigen-decomposition and selecting eigenvectors to construct a new matrix on r worker nodes with γ available GPU threads each is $O\left((n^3 + n \log n)/(r\gamma)\right)$.

In stage 4, the time complexity of performing clustering analysis for a $n \times c$ matrix using K-Means clustering algorithm with c clustering centers is $O(nc^2\tau)$, where τ is the number of iterations of K-Means clustering algorithm. The time complexity of parallelly performing clustering analysis using K-Means clustering algorithm on r worker nodes with γ available GPU threads each is $O\left((nc^2\tau)/(r\gamma)\right)$.

IV. EXPERIMENT

A. EXPERIMENTAL SETUP

In order to verify the effectiveness of the proposed DSSC algorithm, K-Means clustering algorithm [25], DBSCAN clustering algorithm [26], NJW algorithm [2], DSC algorithm [15], and the DSSC algorithm are implemented on the six synthetic datasets and four UCI datasets [27], respectively. Furthermore, the computational efficiency of the parallel DSSC algorithm is evaluated on the Dask cluster with three synthetic datasets used for large-scale clustering analysis.

The hardware configurations and software configurations of the experimental platform are listed in Table 2 and Table 3, respectively. Note that the CPU and the GPU of each worker node contain 8 CPU cores and 2560 GPU cores, respectively.

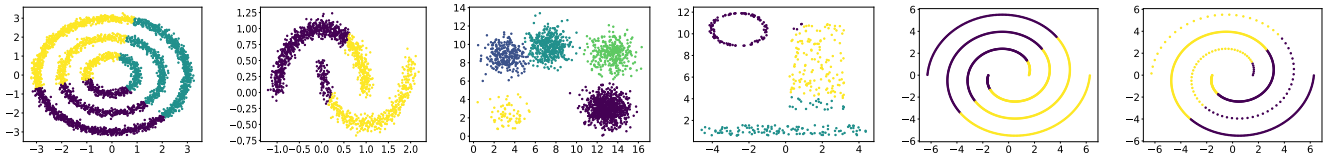
TABLE 2. Hardware configurations of the experimental platform.

Node Type	Number of Nodes	CPU Model	CPU RAM Size	GPU Model	GPU RAM Size
Master node	1	Intel Xeon E3-1225 v5	32 GB	–	–
Worker node	4	Intel Core i7-9700K	64 GB	RTX 2070 SUPER	8 GB

B. CLUSTERING ANALYSIS ON SYNTHETIC DATASETS

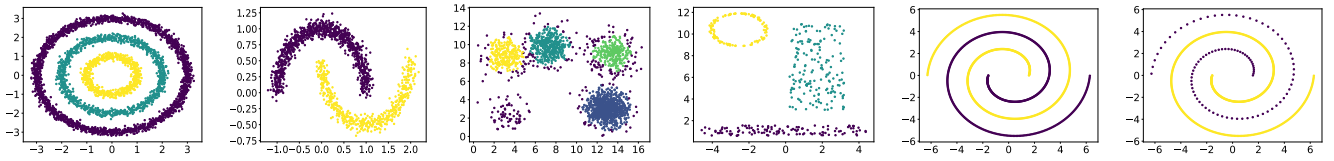
In order to more intuitively prove the superiority of the proposed DSSC algorithm, the experiments are conducted with K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, DSC algorithm, and the DSSC algorithm on six different synthetic datasets. The descriptions of six synthetic datasets are listed in Table 4, and the parameter settings of different clustering algorithms on six synthetic datasets are listed in Table 5.

As shown in Fig. 3, the K-Means clustering algorithm obtains good clustering effect on the fiveClusters dataset, but the clustering effects obtained on the other five non-convex



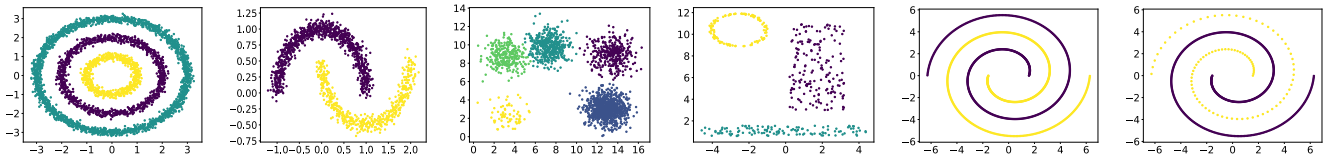
(a) threeCircles (b) twoMoons (c) fiveClusters (d) 3MC (e) twoSpirals (f) twoUnbalanceSpirals

FIGURE 3. Clustering effects of K-Means clustering algorithm obtained on six different datasets.



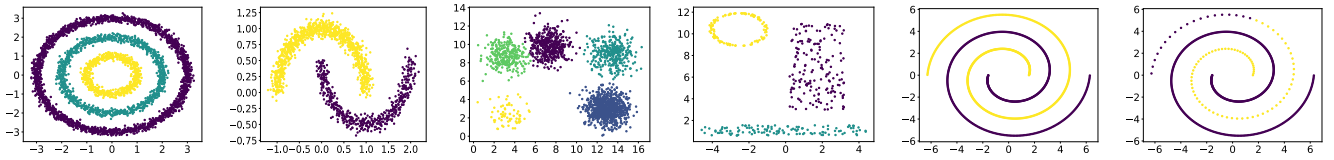
(a) threeCircles (b) twoMoons (c) fiveClusters (d) 3MC (e) twoSpirals (f) twoUnbalanceSpirals

FIGURE 4. Clustering effects of DBSCAN clustering algorithm obtained on six different datasets.



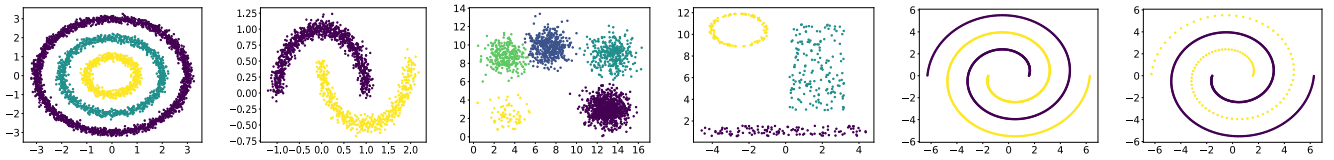
(a) threeCircles (b) twoMoons (c) fiveClusters (d) 3MC (e) twoSpirals (f) twoUnbalanceSpirals

FIGURE 5. Clustering effects of NJW algorithm obtained on six different datasets.



(a) threeCircles (b) twoMoons (c) fiveClusters (d) 3MC (e) twoSpirals (f) twoUnbalanceSpirals

FIGURE 6. Clustering effects of DSC algorithm obtained on six different datasets.



(a) threeCircles (b) twoMoons (c) fiveClusters (d) 3MC (e) twoSpirals (f) twoUnbalanceSpirals

FIGURE 7. Clustering effects of DSSC algorithm obtained on six different datasets.

TABLE 3. Software configurations of the experimental platform.

Software Name	Software Version
Operating system	CentOS 8.1
Python	Python 3.8
Dask	Dask 0.16
Cupy	Cupy 9.2.0

datasets are not good. This is because K-Means clustering algorithm uses the Euclidean distance measure to calculate the similarities between samples, it cannot effectively perform clustering analysis for the non-convex datasets. Besides, the clustering effect is also affected by the selection of initial clustering centers and the number of iterations.

As shown in Fig. 4, the DBSCAN clustering algorithm has a inferior clustering effect only on the fiveClusters dataset. This is because each kind of samples of the fiveClusters dataset has a different density, it is difficult to reasonably set the neighborhood distance threshold ε and the number of neighborhood samples *MinPts*.

As shown in Fig. 5, the NJW algorithm obtains satisfactory clustering effects on six different datasets, but this depends on finding an appropriate value of the scale parameter σ.

TABLE 4. Descriptions of six synthetic datasets.

Dataset	Number of Instances	Number of Attributes	Number of Clusters
threeCircles	3600	2	3
twoMoons	1600	2	2
fiveClusters	2000	2	5
3MC	400	2	3
twoSpirals	1000	2	2
twoUnbalanceSpirals	600	2	2

TABLE 5. Parameter settings of different clustering algorithms on six synthetic datasets.

Dataset	K-Means	DBSCAN		NJW	DSC	DSSC
	K	ε	$MinPts$	σ	ρ	k
threeCircles	3	0.20	5	0.04	3.00	15
twoMoons	2	0.25	10	0.04	3.00	15
fiveClusters	5	0.50	15	1.00	2.00	15
3MC	3	1.00	5	0.50	3.00	15
twoSpirals	2	0.20	3	0.04	2.00	5
twoUnbalanceSpirals	2	0.50	6	0.06	4.00	3

TABLE 6. Descriptions of four UCI datasets.

Dataset	Number of Instances	Number of Attributes	Number of Clusters
Iris	150	4	3
Seeds	210	7	3
Wine	178	13	3
Zoo	101	16	7

As shown in Fig. 6, the DSC algorithm achieves satisfactory clustering effects on the first five datasets, whereas the clustering effect obtained on the twoUnbalanceSpirals dataset is inferior. This is because the densities of the two kinds of samples of the twoUnbalanceSpirals dataset are quite different, it is difficult to find an appropriate value of the density parameter ρ to achieve a good clustering effect.

As shown in Fig. 7, the proposed DSSC algorithm achieves satisfactory clustering effects on six different datasets. This is because the DSSC algorithm can find an appropriate value of the density parameter ρ. Moreover, it can adaptively determine the value of ρ, thus it can obtain good clustering effects even on the complex datasets.

C. CLUSTERING ANALYSIS ON UCI DATASETS

In order to further verify the effectiveness of the proposed DSSC algorithm, in terms of four different performance evaluation indexes, the DSSC algorithm are compared with K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm on four different UCI datasets. These four performance evaluation indexes include the clustering accuracy, adjusted rand index (ARI) [28], Fowlkes-Mallows index (FMI) [29], and normalized mutual information (NMI) [30]. The descriptions of four UCI datasets are listed in Table 6, and the parameter settings of different clustering algorithms on these four UCI datasets are listed in Table 7.

TABLE 7. Parameter settings of different clustering algorithms on four UCI datasets.

Dataset	K-Means	DBSCAN		NJW	DSC	DSSC
	K	ε	$MinPts$	σ	ρ	k
Iris	3	1.00	5	0.50	2.00	15
Seeds	3	0.90	15	2.50	3.00	21
Wine	3	25.00	5	9.00	4.00	20
Zoo	7	1.00	5	1.50	4.00	21

TABLE 8. Clustering accuracies of different clustering algorithms obtained on different datasets.

Dataset	K-Means	DBSCAN	NJW	DSC	DSSC
Iris	0.8933	0.6667	0.9067	0.9067	**0.9533**
Seeds	0.8928	0.5619	0.8945	0.8810	**0.8952**
Wine	0.6960	0.4708	0.5687	0.6846	**0.7278**
Zoo	0.7366	0.6402	0.7366	0.6481	**0.7884**
Average	0.8047	0.5849	0.7766	0.7801	**0.8412**

TABLE 9. ARI of different clustering algorithms obtained on different datasets.

Dataset	K-Means	DBSCAN	NJW	DSC	DSSC
Iris	0.7302	0.5681	0.7583	0.7726	**0.8680**
Seeds	0.7166	0.3197	0.7166	0.6810	**0.7226**
Wine	0.3711	0.2956	0.2926	0.3765	**0.4111**
Zoo	0.6981	0.6050	**0.7544**	0.5165	0.6700
Average	0.6290	0.4471	0.6305	0.5867	**0.6679**

TABLE 10. FMI of different clustering algorithms obtained on different datasets.

Dataset	K-Means	DBSCAN	NJW	DSC	DSSC
Iris	0.8208	0.7715	0.8395	0.8487	**0.9114**
Seeds	0.8106	0.5372	0.8106	0.7874	**0.8161**
Wine	0.5835	0.5651	0.6192	0.5883	**0.6285**
Zoo	0.7653	0.6985	**0.8113**	0.6190	0.7429
Average	0.7451	0.6431	0.7702	0.7109	**0.7747**

Table 8 shows the clustering accuracies of different clustering algorithms obtained on different datasets. It can be seen from Table 8 that the clustering accuracies of five different clustering algorithms obtained on the Iris dataset are higher than that obtained on the other three datasets. Especially, the clustering accuracy of the DSSC algorithm obtained on the Iris dataset is 5.81%, 22.55%, and 16.49% higher than that obtained on the Seeds, Wine, and Zoo datasets, respectively. It also can be seen from Table 8 that the average clustering accuracy of the DSSC algorithm obtained on four datasets is 3.65%, 25.63%, 6.46%, and 6.11% higher than that of K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm obtained on four datasets, respectively. The results indicate that the proposed DSSC algorithm can offer a satisfactory clustering accuracy to a certain extent.

It can be seen from Table 9 that the average value of ARI of the DSSC algorithm obtained on four UCI datasets is 3.89%, 22.08%, 3.75%, and 8.13% higher than that of K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm obtained on these datasets,

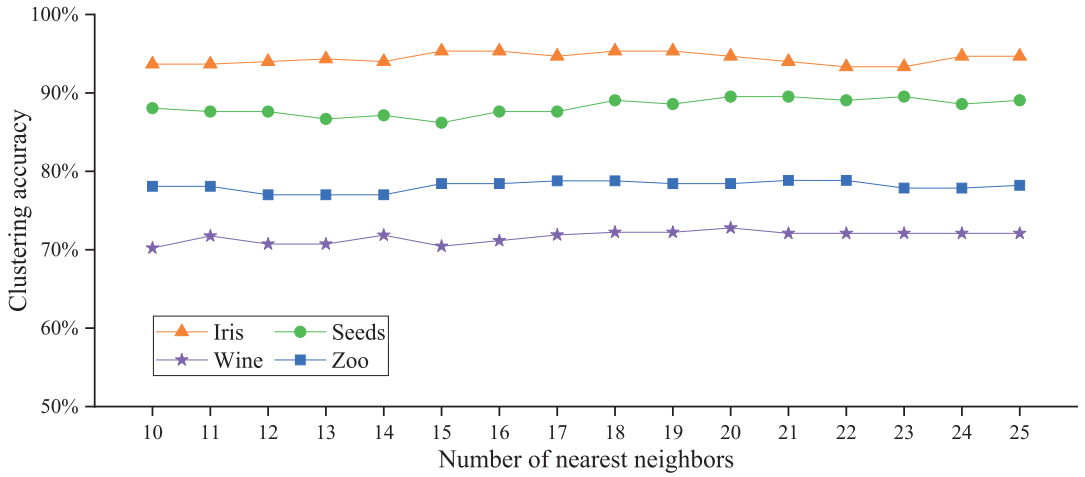


FIGURE 8. Clustering accuracies of the proposed DSSC algorithm obtained with different number of nearest neighbors and different datasets.

TABLE 11. NMI of different clustering algorithms obtained on different datasets.

Dataset	K-Means	DBSCAN	NJW	DSC	DSSC
Iris	0.7558	0.7337	0.7857	0.7941	**0.8465**
Seeds	0.6949	0.4741	0.6949	0.6803	**0.7197**
Wine	0.4286	0.3933	0.4049	0.4307	**0.4417**
Zoo	0.7836	0.7635	**0.8447**	0.7057	0.7752
Average	0.6657	0.5912	0.6826	0.6527	**0.6958**

respectively. It can be seen from Table 10 that the average value of FMI of the DSSC algorithm obtained on four UCI datasets is 2.97%, 13.17%, 0.46%, and 6.39% higher than that of K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm obtained on these datasets, respectively. It also can be seen from Table 11 that the average value of NMI of the DSSC algorithm obtained on four UCI datasets is 3.01%, 10.46%, 1.32%, and 4.31% higher than that of K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm obtained on these datasets, respectively. The results further prove that the proposed DSSC algorithm can achieve satisfactory clustering effects on different datasets. This is because the DSSC algorithm can adaptively determine the value of the density parameter to construct a more robust similarity matrix.

D. ANALYSIS OF PARAMETER SENSITIVITY

In the proposed DSSC algorithm, the number of nearest neighbors needs to be set manually. In order to explore whether the number of nearest neighbors has an impact on the clustering accuracy of the DSSC algorithm, the comparative experiments are carried out with different number of nearest neighbors on four different UCI datasets.

As shown in Fig. 8, with the increase of the number of nearest neighbors, the clustering accuracies of the DSSC algorithm obtained on four different datasets have a slight change. The maximum clustering accuracies of the DSSC algorithm obtained with different number of nearest neighbors on the Iris, Seeds, Wine, and Zoo datasets are 95.33%, 89.52%, 72.78%, and 78.84%, respectively. The minimum clustering accuracies of the DSSC algorithm obtained with different number of nearest neighbors on the Iris, Seeds, Wine, and Zoo datasets are 93.33%, 86.19%, 70.21%, and 77.00%, respectively. The ranges of clustering accuracies of the DSSC algorithm obtained with different number of nearest neighbors on the Iris, Seeds, Wine, and Zoo datasets are 2.00%, 3.33%, 2.57% and 1.84%, respectively. The results show that the number of nearest neighbors has a little impact on the clustering accuracy of the proposed DSSC algorithm.

TABLE 12. Descriptions of three synthetic datasets used for large-scale clustering analysis.

Dataset	Number of Instances	Number of Attributes	Number of Clusters
DB-1	1,000,000	4	4
DB-2	10,000,000	4	4
DB-3	20,000,000	4	4

TABLE 13. Running time of the parallel DSSC algorithm obtained with different number of worker nodes (seconds).

Dataset	1 Worker Node	2 Worker Nodes	3 Worker Nodes	4 Worker Nodes
DB-1	1498.80	793.89	547.85	425.34
DB-2	15054.86	7857.62	5127.20	4032.86
DB-3	34404.52	16880.56	11893.19	8918.23

E. ANALYSIS OF COMPUTATIONAL EFFICIENCY

To evaluate the computational efficiency of the parallel DSSC algorithm, three different size of synthetic datasets used for large-scale clustering analysis are adopted to carry out experiments on the Dask cluster with CPU+GPU. The descriptions of these three synthetic datasets are shown in Table 12.

Table 13 presents the running time of the parallel DSSC algorithm obtained with different number of worker nodes. For the three different size of datasets, with the increase of the number of worker nodes, the running time of the parallel

DSSC algorithm gradually decrease. For the DB-1, DB-2, and DB-3 datasets, the running time of the parallel DSSC algorithm obtained with four worker nodes are 71.62%, 73.21%, and 74.08% less than that obtained with one worker node, respectively. Compared with one worker node, the running time of the parallel DSSC algorithm obtained with 2, 3, and 4 worker nodes on the three different size of datasets are decreased by 48.59%, 64.94%, and 72.97% on average, respectively. The results show that the computational efficiency of the parallel DSSC algorithm can be improved for large-scale clustering analysis to a certain extent by properly increasing the number of worker nodes.

The speedup is often used to evaluate the computational efficiency of a parallel algorithm. In this experiment, the absolute speedup is used to evaluate the computational efficiency of the parallel DSSC algorithm, and it can be calculated by

$$Speedup = \frac{T_s}{T_p}, \tag{8}$$

where T_s is the running time of the serial DSSC algorithm and T_p is the running time of the parallel DSSC algorithm obtained on the Dask cluster with p worker nodes.

Fig. 9 shows the speedups of the parallel DSSC algorithm obtained with different number of worker nodes on different size of datasets. As illustrated in Fig. 9, with the increase of the number of worker nodes, the speedups of the parallel DSSC algorithm obtained on the three different size of datasets gradually increase. For example, the speedup of the parallel DSSC algorithm obtained on the DB-3 dataset increases from 1.99× to 7.69× when the number of worker nodes increases from 1 to 4, which shows that the parallel DSSC algorithm has good parallelism. It also can be seen from Fig. 9 that the speedups of the parallel DSSC algorithm obtained with four worker nodes are 5.64×, 6.34×, and 7.69× for the DB-1, DB-2, and DB-3 datasets, respectively. The results demonstrate that the parallel DSSC algorithm achieves a high speedup in performing large-scale clustering analysis. This is because the parallel DSSC algorithm can make full use of the computing resources of a Dask cluster with CPU+GPU to improve the efficiency of performing large-scale clustering analysis in parallel.

F. EVALUATION ON DIFFERENT DASK CLUSTERS

In order to better evaluate the computational efficiency of the parallel DSSC algorithm, for the three different size of datasets listed in Table 12, the experiments are conducted on the Dask cluster with CPU and Dask cluster with CPU+GPU, respectively. It is worth noting that both of the Dask clusters contain four worker nodes.

As depicted in Fig. 10, with the increase of dataset size, the running time of the parallel DSSC algorithm obtained on both Dask clusters increase gradually. For the three different size of datasets, the running time of the parallel DSSC algorithm obtained on the Dask cluster with CPU are longer than that obtained on the Dask cluster with CPU+GPU. For

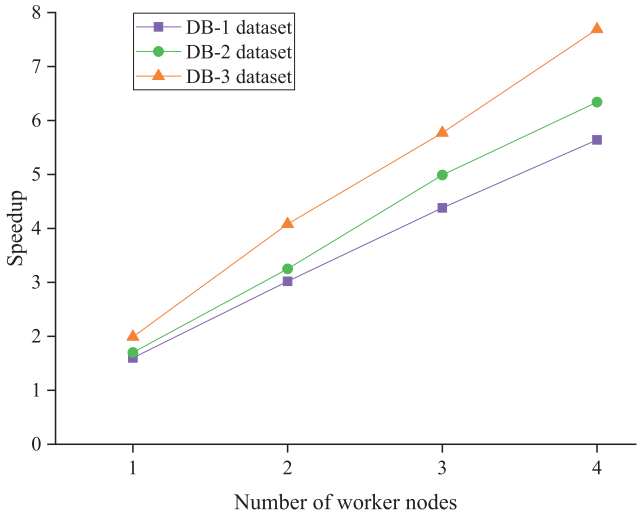


FIGURE 9. Speedups of the parallel DSSC algorithm obtained with different number of worker nodes on different size of datasets.

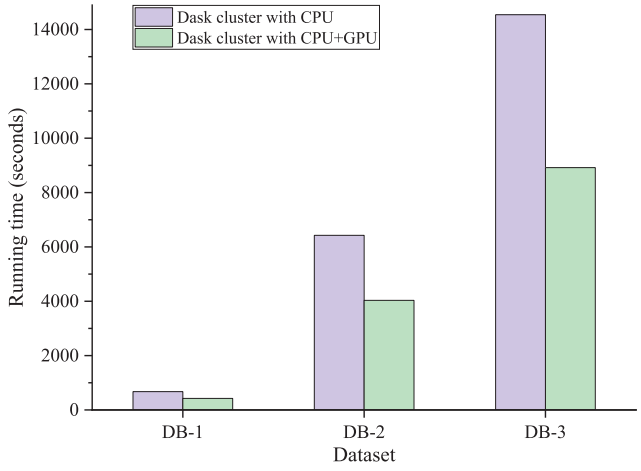


FIGURE 10. Running time of the parallel DSSC algorithm obtained on different Dask clusters.

the DB-1, DB-2, and DB-3 datasets, the running time of the parallel DSSC algorithm obtained on the Dask cluster with CPU+GPU are 36.78%, 37.23%, and 38.68% shorter than that obtained on the Dask cluster with CPU, respectively. The results demonstrate that the parallel DSSC algorithm can fully utilize all available CPU and GPU resources of a Dask cluster to improve the computational efficiency.

V. CONCLUSION

In the paper, an adaptive density-sensitive similarity measure based spectral clustering algorithm is proposed to improve the clustering effect of the spectral clustering algorithm. First of all the Euclidean distances between samples are calculated to determine the nearest neighbors of samples, then the density-sensitive distances between samples and their nearest neighbors are adaptively calculated, and finally the similarities between samples and their nearest neighbors are calculated to construct a similarity matrix. In order to improve the efficiency of the DSSC algorithm for performing

clustering analysis, the DSSC algorithm is parallelized on Dask distributed parallel computing platform with CPU+GPU. Experiments on six synthetic datasets and four UCI datasets show that the proposed DSSC algorithm obtains more satisfactory clustering results compared with K-Means clustering algorithm, DBSCAN clustering algorithm, NJW algorithm, and DSC algorithm. For example, the DSSC algorithm obtains a clustering accuracy of 95.33% on the Iris dataset. In addition, the parallel DSSC algorithm obtains better efficiency of performing large-scale clustering analysis. For example, the running time of the DSSC algorithm obtained with four worker nodes on the DB-3 dataset is reduced by 87.01% and 74.08% than that obtained with one CPU core and that obtained with one worker node, respectively.

Compared with some existing clustering algorithms, the proposed DSSC algorithm has the following advantages: 1) it can improve the clustering effect to a certain extent by adaptively calculating the similarities between each sample and its nearest neighbors; 2) it is more suitable for performing large-scale clustering analysis through its parallelization on Dask distributed parallel computing platform with CPU+GPU. However, the parallelization of the DSSC algorithm on GPU has the following limitations: 1) the parallel efficiency of constructing a similarity matrix on GPU is low, because the logic of constructing a similarity matrix is complex; 2) the size of each block is limited to the GPU RAM size when the dataset is divided into multiple blocks.

In the future work, in order to make the DSSC algorithm more suitable for performing larger-scale clustering analysis, the DSSC algorithm with lower time complexity will be explored. Moreover, how to fuse multiple DSSCs into a better one through the ensemble clustering technique will be considered to further improve the clustering effect. It is also worth considering that the DSSC algorithm will be applied to mechanical fault diagnosis or other fields.

REFERENCES

[1] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017.

[2] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. 15th Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2002, pp. 849–856.

[3] W. Song, M. Lai, X. Li, Y. Song, and L. Gao, "A new spectral clustering based on particle swarm optimization for unsupervised fault diagnosis of bearings," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 386–391.

[4] X. Wang, C. Chang, and X. L. Wang, "A fast incremental spectral clustering algorithm for image segmentation," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2017, pp. 402–407.

[5] R. Janani and S. Vijayarani, "Text document clustering using spectral clustering algorithm with particle swarm optimization," *Expert Syst. Appl.*, vol. 134, pp. 192–200, Nov. 2019.

[6] M. Shi and G. Xu, "Spectral clustering using Nyström approximation for the accurate identification of cancer molecular subtypes," *Sci. Rep.*, vol. 7, p. 4896, Jul. 2017.

[7] H. Jia, S. Ding, X. Xu, and R. Nie, "The latest research progress on spectral clustering," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1477–1486, 2014.

[8] J. Xie, Y. Zhou, and L. Ding, "Local standard deviation spectral clustering," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 242–250.

[9] T. Du, G. Wen, Z. Cai, W. Zheng, M. Tan, and Y. Li, "Spectral clustering algorithm combining local covariance matrix with normalization," *Neural Comput. Appl.*, vol. 32, no. 11, pp. 6611–6618, Jun. 2020.

[10] X. Ye and T. Sakurai, "Spectral clustering with adaptive similarity measure in kernel space," *Intell. Data Anal.*, vol. 22, no. 4, pp. 751–765, Jun. 2018.

[11] S. Park and H. Zhao, "Spectral clustering based on learning similarity matrix," *Bioinformatics*, vol. 34, no. 12, pp. 2069–2076, Jun. 2018.

[12] X. Zhu, J. Gan, G. Lu, J. Li, and S. Zhang, "Spectral clustering via half-quadratic optimization," *World Wide Web*, vol. 23, pp. 1969–1988, Nov. 2020.

[13] M. Yuan and Q. Zhu, "Spectral clustering algorithm based on fast search of natural neighbors," *IEEE Access*, vol. 8, pp. 67277–67288, 2020.

[14] X. Zhang, J. Li, and H. Yu, "Local density adaptive similarity measurement for spectral clustering," *Pattern Recognit. Lett.*, vol. 32, no. 2, pp. 352–358, 2011.

[15] P. Yang, Q. Zhu, and B. Huang, "Spectral clustering with density sensitive similarity function," *Knowl.-Based Syst.*, vol. 24, no. 5, pp. 621–628, 2011.

[16] J. Yan, D. Cheng, M. Zong, and Z. Deng, "Improved spectral clustering algorithm based on similarity measure," in *Proc. Int. Conf. Advan. Data Mining Appl. (ADMA)*, Guilin, China, Dec. 2014, pp. 641–654.

[17] L. Wang, S. Ding, and H. Jia, "An improvement of spectral clustering via message passing and density sensitive similarity," *IEEE Access*, vol. 7, pp. 101054–101062, 2019.

[18] Q. Zhan and Y. Mao, "Improved spectral clustering based on Nyström method," *Multimedia Tools Appl.*, vol. 76, pp. 20149–20165, Oct. 2017.

[19] Y. Zhao, Y. Yuan, F. Nie, and Q. Wang, "Spectral clustering based on iterative optimization for large-scale and high-dimensional data," *Neurocomputing*, vol. 318, pp. 227–235, Nov. 2018.

[20] L. Zhang, L. Hou, and D. Lei, "Spectral clustering algorithm based on Hadoop cloud platform research and implementation," in *Proc. 5th Int. Conf. Adv. Mater. Comput. Sci.*, Mar. 2016, pp. 1–4.

[21] L. Ma, "Efficient parallel clustering spectral algorithm based on Hadoop," in *Proc. Int. Conf. Appl. Techn. Cyber Secur. Intell. (ATCI)*, Shanghai, China, Jun. 2018, pp. 935–941.

[22] A. I. Taloba, M. R. Riad, and T. H. A. Soliman, "Developing an efficient spectral clustering algorithm on large scale graphs in spark," in *Proc. 8th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2017, pp. 292–298.

[23] Z. Huo, G. Mei, G. Casolla, and F. Giampaolo, "Designing an efficient parallel spectral clustering algorithm on multi-core processors in Julia," *J. Parallel Distrib. Comput.*, vol. 138, pp. 211–221, Apr. 2020.

[24] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proc. 14th Python Sci. Conf.*, Jul. 2015, pp. 130–136.

[25] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient K-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

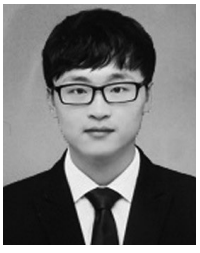
[26] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, Portland, OR, USA, Aug. 1996, pp. 226–231.

[27] D. Dua and C. Graff. *UCI Machine Learning Repository*. Accessed: May 3, 2021. [Online]. Available: http://archive.ics.uci.edu/ml

[28] D. Steinley, "Properties of the Hubert–Arable adjusted Rand index," *Psychol. Methods*, vol. 9, no. 3, pp. 386–396, 2004.

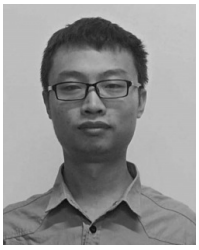
[29] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2, pp. 107–145, Dec. 2001.

[30] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Jan. 2010.



GEN ZHANG was born in Anhui, China, in 1995. He received the B.S. degree in network engineering from West Anhui University, Lu'an, China, in 2019. He is currently pursuing the M.S. degree in computer science and technology with Hunan University of Technology, Zhuzhou, China. His research interests include industrial big data analysis and industrial equipment fault diagnosis.

LANJUN WAN was born in Hunan, China, in 1982. He received the B.S. and M.S. degrees in computer science and technology from Hunan University of Technology, Zhuzhou, China, in 2005 and 2009, respectively, and the Ph.D. degree in circuits and systems from Hunan University, Changsha, China, in 2016. He is currently an Assistant Professor with the School of Computer Science, Hunan University of Technology. He has published many research articles in international conferences and journals, such as *JPDC*, *CCPE*, *ParCo*, and *Sensors*. His research interests include industrial big data analysis, industrial equipment fault diagnosis, high-performance computing, and parallel computing. He serves as a Reviewer for the *JPDC*, *CCPE*, *Sensors*, and IEEE Access.



KUN GONG was born in Hunan, China, in 1996. He received the B.S. degree in mechanical engineering from Hunan University of Technology, Zhuzhou, China, in 2019, where he is currently pursuing the M.S. degree in computer science and technology. His research interests include industrial big data analysis and industrial equipment fault diagnosis.

CHANGYUN LI was born in Hunan, China, in 1972. He received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2007. He is currently a Full Professor of computer science and the Dean of the School of Computer Science, Hunan University of Technology, Zhuzhou, China. His major research interests include industrial big data analysis, industrial equipment fault diagnosis, intelligent information perception and processing technology, the Internet of Things, and software methodology.

MANSHENG XIAO was born in Hunan, China, in 1968. He received the M.S. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2005. He is currently a Full Professor with the School of Computer Science, Hunan University of Technology, Zhuzhou, China. His major research interests include industrial big data analysis, intelligent information processing, pattern recognition, and image processing.