Journal of Network Intelligence ©2024 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **9**, Number **2**, May **2024**

Parallel Clustering Algorithm for High-dimensional Data Combining Improved Fruit Fly Optimization and Density Peak Clustering

Yufeng Zhao*

School of Electronics and IoT Engineering,
Chongqing Industry Polytechnic College
Chongqing, 401120, P.R. China
zhaoyf@cqipc.edu.cn

Jie He

Lee Kong Chian Faculty of Engineering and Science,
Tunku Abdul Rahman University
Kuala Lumpur, 43000, Malaysia
hejie@cqipc.edu.cn

*Corresponding author: Yufeng Zhao

Received June 28, 2023, revised September 17, 2023, accepted December 3, 2023.

Abstract. *To improve the clustering performance of complex high-dimensional data, an improved density peak clustering (DPC) method is used for cluster analysis. The density values are calculated based on the kernel function of the density peak clustering, and the distance and density values are sorted in descending order. The sample points corresponding to larger values are selected as the centers of several clusters. The sample category is determined by calculating the distance between each node and each cluster center point, according to the set distance threshold. To prevent the clustering results from being greatly disturbed by inappropriate distance threshold settings, the fruit fly optimization algorithm (FFO) is introduced to optimize and improve the distance threshold parameter. By continuously updating and optimizing the concentration of fruit fly populations, the optimal fitness individual is obtained. To address the high time complexity of the FFO algorithm, a parallel algorithm based on the Spark framework is proposed. Firstly, the data is partitioned, and then local clustering is performed in each data space, and finally, the local clustering results are aggregated for global clustering. Experimental results show that the proposed algorithm has the highest clustering accuracy, outperforming CNN and non-parametric density peak clustering algorithms. This proves that by setting the odor change rate range and other parameters of the fruit fly optimization algorithm reasonably, the clustering performance of different domain datasets can be effectively improved. In addition, the parallel algorithm under the Spark framework achieves better results in terms of computational efficiency compared to the original algorithm.*

Keywords: High-dimensional data; Parallel clustering; Fruit fly optimization; Density peak clustering; Distance threshold; Spark framework.

1. **Introduction.** With the development of the Internet, data has become increasingly complex and multidimensional. In practical applications, high-dimensional data is becoming more common, and identifying relationships between data in high-dimensional space is becoming increasingly important [1]. Cluster analysis is a commonly used unsupervised learning algorithm that divides data objects into different groups based on their

relationships and features. Data objects within the same group exhibit better correlation than those from different groups. By observing the characteristics of each subset through cluster analysis, relationships or patterns between different subsets can be discovered and analyzed [2].

In the absence of categorical information for samples, cluster analysis is based on the similarity of sample features to determine the data groups, with samples in the same group having high similarity and those from different groups having high dissimilarity [3]. Cluster analysis methods generally require determining the initial cluster centers and then iteratively optimizing to obtain the clustering results. Typical algorithms that randomly set the initial cluster centers include K-Means [4], which uses the principle of dividing samples based on the nearest distance in sample space, but is not suitable for clustering non-spherical data (such as manifold data with spatial folding structures) [5, 6]. For irregularly shaped datasets, clustering cluster centers are defined based on density center points, with typical algorithms such as DBSCAN defining clusters as the largest set of density-connected points, which can divide high-density areas into clusters and discover clusters of any shape in noise data [7]. Based on this, Density Peaks Clustering (DPC) algorithm [8] was proposed based on density peaks, which defines cluster centers as points with high local density and large distance from points with higher density, which can help to determine cluster centers to some extent. In addition, methods based on probability density distribution assume that different clusters within the data set follow different probability density distributions, and all sample points converge to the local maximum density, with points ultimately converging to the same local maximum density being judged as belonging to the same cluster. These algorithms are suitable for non-spherical data sets, but have high computational costs, with the Mean Shift algorithm being a representative algorithm [9, 10].

When using the same clustering algorithm on the same dataset with different initial parameters, there can be significant differences in the clustering results, and it is often difficult to accurately determine the number of clusters [11, 12]. These two issues make it challenging for users to select an appropriate clustering algorithm and find suitable parameters [13]. To address these problems and make clustering results more stable, researchers have proposed ensemble clustering algorithms [14]. Ensemble clustering involves using a series of basic clustering algorithms to learn from different parameter settings and data subsets, and then integrating the results to obtain more stable and superior clustering results than those obtained from a single algorithm [15]. Although ensemble clustering algorithms can improve clustering performance compared to single basic clustering algorithms, the computational complexity increases significantly, resulting in a significant increase in computation time. Especially when dealing with high-dimensional massive datasets, serial ensemble clustering algorithms are limited by factors such as the computing power, memory capacity, and disk speed of a single server [16]. Even algorithms such as FastESC [17], EulerSC [18], and U-SENC [19] that use methods such as approximate estimation to avoid constructing complete sparse similarity matrices to improve algorithm efficiency in a single-machine environment, their computation time is still affected by parameters such as the total amount of data, data dimensionality, and number of iterations. Distributed system technology uses the "divide and conquer" approach, with the final focus not on processing large amounts of data in batches, but on improving the algorithm to run on distributed computing platforms, thereby improving computation speed.

In distributed computing platforms, the main focus is on MapReduce and Spark. Under the MapReduce computing model, Tripathi et al. [20] proposed the MR-EGWO (MapReduce-based Enhanced Grey Wolf Optimizer) algorithm, an enhanced grey wolf optimization algorithm based on MapReduce, which optimized the clustering process by introducing new variables and demonstrated the effectiveness of the MR-EGWO algorithm in processing large-scale datasets through testing on the UCI benchmark dataset. Kim et al. [21] proposed the density-based clustering algorithm DBCURE-MR for MapReduce framework, which parallelized the cluster search operation and demonstrated the effectiveness and scalability of the DBCURE-MR algorithm on different datasets. Sardar et al. [22] designed the parallel K-means algorithm under the MapReduce computing model and demonstrated its effectiveness through comparative experiments with existing serial K-means algorithms. In [23], Spark's advantages in memory parallel computing were utilized to accelerate the calculation speed of the HBMC (Basin-Hopping Monte Carlo) algorithm, and experimental results showed that the optimized HBMC algorithm based on Spark has a faster convergence speed. Hosseini et al. [24] implemented a density-based clustering algorithm based on Spark, which used the fuzzy weighted correlation coefficient as a similarity measurement method and solved the I/O load bottleneck problem in MapReduce clusters using RDD to store data. The superiority of the algorithm in accuracy and efficiency was demonstrated through the experiment comparison.

High-dimensional data has the characteristics of data sparsity and the curse of dimensionality. In the context of big data, traditional clustering algorithms face significant challenges in scalability, and also suffer from the following two problems: (1) Different clustering algorithms are better suited for different types of datasets and data structures (such as category shape or size). For example, the K-means clustering algorithm is more suitable for spherical data, while single-linkage hierarchical clustering is better at detecting linkage patterns. (2) When using the same clustering algorithm on the same dataset with different initial parameters, there can be significant differences in the clustering results, and it is often difficult to accurately determine the number of clusters. These two issues make it difficult for users to select an appropriate clustering algorithm and find suitable parameters. To address these problems, this paper proposes a parallel clustering algorithm for high-dimensional data that combines Fruit Fly Optimization (FFO) algorithm and improved DPC. An improved DPC algorithm is used to classify and organize high-dimensional data, and the FFO algorithm is used to improve the multi-dimensional feature clustering applicability of the traditional DPC algorithm. The core parameters of the DPC algorithm are optimized to solve the problem of low data clustering performance caused by randomly setting parameters. In addition, a strategy is proposed to implement parallel running of the proposed algorithm on the Spark parallel framework, reducing algorithm computation time and improving computational efficiency.

The rest of the text are organized as follows. In Chapter 2, background knowledge including the DPC algorithm and the FFO algorithm are introduced. Chapter 3 explains in detail the proposed parallel clustering algorithm, including the improved DPC algorithm combined with FFO and the parallel implementation of the improved algorithm on the Spark framework. Chapter 4 analyzes the performance of the proposed method through experiments, provides comparison and analysis of the results with other methods. Finally, Chapter 5 concludes the paper and points out the next research direction.

2. **Research Background.**

2.1. **DPC algorithm.**

DPC is a density-based clustering algorithm that determines initial cluster centers by calculating sample attribute densities and distance values, and then determines the category based on the distance between the sample point and the center point. This algorithm is suitable for situations where the local density of clustering center points is relatively high, and the distance between points with higher density than them is relatively far.

Suppose the sample set X consist of a total of k categories $C = \{C_1, C_2, \ldots, C_k\}$, satisfying the condition $k \leq N$, where N is the total number of samples, and $X = C_1 \cup C_2 \cup \ldots \cup C_k$, $C_i \cap C_j = \emptyset (i \neq j)$.

For any two sample points x_i and x_j, the distance r_{ij} between them can be expressed as [25]:

$$r_{ij} = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i1} - x_{j1}|^2 + \ldots + |x_{in} - x_{jn}|^2} \tag{1}$$

were n is the dimensionality, and the density ρ_i of a point x_i in a set of N points is given by:

$$\rho_i = \sum_j \chi(r_{ij} - r_c) \tag{2}$$

Where r_c represents the cutoff distance, which needs to be specified manually, and is usually chosen to be 1% to 2% of the entire dataset. The kernel function $\chi(x)$ is given by:

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{3}$$

Since the kernel function is not differentiable, a Gaussian function is often used as a replacement, and the calculation of ρ_i is converted to:

$$\rho_i = \sum_j e^{-\frac{r_{ij}^2}{2r_c^2}} \tag{4}$$

The minimum distance δ_i of the point x_i can be calculated as:

$$\delta_i = \begin{cases} \min_j(r_{ij}), & \text{if } \rho_i > \rho_j \\ \max_j(r_{ij}), & \text{otherwise} \end{cases} \tag{5}$$

After calculating the density ρ_i and distance δ_i for each of the N sample points, the resulting values are used to generate the decision graph, with density values on the y-axis and distance values on the x-axis. The multiplication of the density and distance values for each point is given by:

$$\gamma_i = \rho_i \cdot \delta_i \tag{6}$$

The density ρ_i, distance δ_i, and product γ_i are calculated for all sample points and then sorted in descending order. Points with high values for all three criteria are selected as cluster centers. Non-center points are assigned to the category of the closest center point based on its distance value. Figure 1 shows the process of generating the decision graph.

2.2. **FFO algorithm.** The FFO Algorithm selects a search direction and step size based on the concentration of food odor to obtain food during the search process. The FOA algorithm can be mathematically described as follows: Let the initial position of a fruit fly individual within its movement range be represented by X_{init} and Y_{init}, then the position update during its movement can be expressed as [26]:

$$\begin{cases} X_i = X_{\text{init}} + H_r \\ Y_i = Y_{\text{init}} + H_r \\ H_r = H \times [2 \times \text{rand}() - 1] \end{cases} \tag{7}$$

Where H represents the maximum step size for the individual's movement.

The relative distance of an individual to the origin can be calculated based on its coordinates X_{init} and Y_{init}:

$$d = \sqrt{X_i^2 + Y_i^2}, \quad S_i = \frac{1}{d} \tag{8}$$

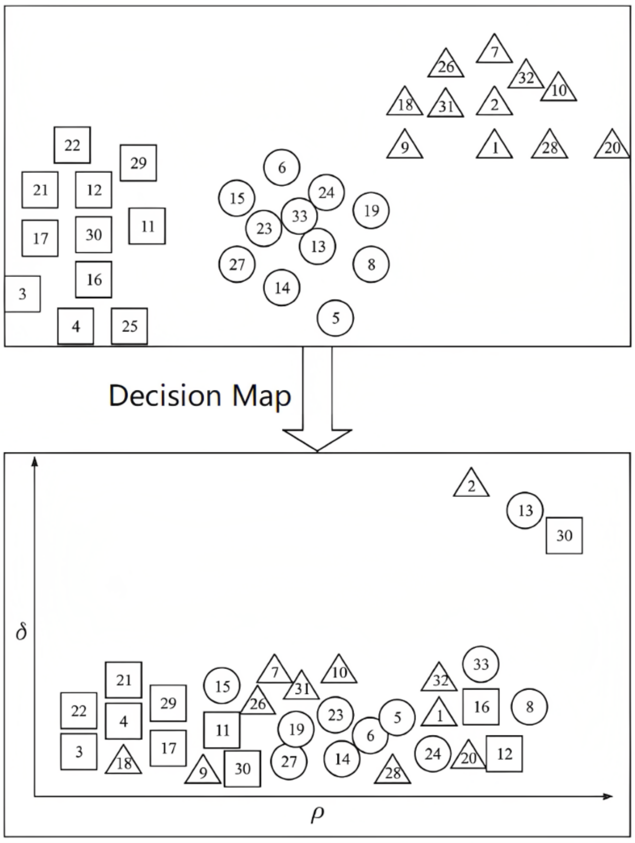


Figure 1. Example of the decision map generation process

Where i denotes the number of iterations. The odor concentration function is established based on S_i as:

$$\text{Smell}_i = \text{Function}(S_i) \tag{9}$$

where Function is the fitness function. Smell_i is updated through the movement of fruit fly individuals.

The maximum value of the Smell_i is denoted as BestSmell_i, and the corresponding fruit fly individual with the position X_{best} and Y_{best} is selected as the initial position of the new generation of fruit fly population.

The iterative process is repeated, and the maximum value of BestSmell_i from all generations is selected as the optimal individual.

3. **Improved Parallel DPC Algorithm.**

3.1. **Improved FFO algorithm.** The basic FFO algorithm has three problems: 1) variables take values in a continuous domain and cannot be directly applied to a discrete domain; 2) the information of fruit fly individuals is consistent, making it difficult to reflect the differences between different variables; 3) a fixed step size factor can easily lead to unstable optimization results. Therefore, the algorithm needs to be improved.

To overcome the three problems of the basic FFO algorithm, three improvement strategies are proposed: 1) The position parameters of fruit flies are discretized to ensure that each decision variable takes values in a discrete integer domain. 2) Fruit fly classification evolution is performed to distinguish between different types of decision variables and allow them to be adjusted within different ranges. 3) The flight ability of fruit flies is adjusted by introducing a step size factor β_j to improve the stability of optimization results.

The steps of the improved Fruit Fly Optimization (IFFO) algorithm are as follows:

1) Initialize hyperparameters, including the population size (popsize), the number of iterations (generation), and the flight range (dom).

2) Classify the decision variables into boolean variables x_b and discrete variables x_d.

3) Initialize the positions of the fruit fly population x_{pos} based on experience or randomly, and restrict the variable values according to Improvement Steps 1 and 2. That is, $x_{\text{pos},b,j}$ is the j-th boolean variable, and $x_{\text{pos},d,j}$ is the j-th discrete variable.

4) Limit the mobility of fruit flies based on their sense of smell and vision, and introduce a step size factor according to Improvement Step 3. The range of β_j is 0.2˜0.5, and $x_{i,j}$ is the j-th variable of the i-th fruit fly. The random value $R \in [-1, 0, 1]$ represents the flight direction of the fruit fly:

$$x_{i,j} = \begin{cases} \text{randbool(dom)}, & \text{if } x_{i,j} \in x_{\text{pos},b,j} \\ x_{\text{pos},d,j} + \beta_j \times |\text{dom}_{U,j} - \text{dom}_{I,j}| \times R, & \text{if } x_{i,j} \in x_{\text{pos},d,j} \end{cases} \tag{10}$$

5) Substitute $x_{i,j}$ into the objective function of minimum cost and solve for it, to obtain the odor concentration Smell_i at the current location of the fruit fly:

$$\text{Smell}_i = \text{Function}(x_{i,j}) \tag{11}$$

6) Identify the individual fruit fly in the population with the optimal odor concentration value.

7) Retain the optimal odor concentration value and have all fruit flies converge towards that position.

8) Determine whether the algorithm has reached the stopping criterion. If yes, terminate the algorithm. Otherwise, repeat steps 4∼6 and compare the current best fitness function value with the historical best fitness function value. If the current best fitness function value is better, update the historical best fitness function value and execute step 7.

9) Remove any redundant values of decision variables and output the optimal result.

3.2. **IFFO-based DPC.** In the implementation process of the DPC algorithm, the selection of parameter r_c is crucial as it determines the clustering accuracy and efficiency of DPC. Due to its strong dependence on r_c, DPC performs poorly in handling unevenly distributed sample points. Therefore, this paper proposes the improvement of DPC using the IFFO algorithm. After optimizing DPC, the parameter selection for r_c is no longer necessary. Instead, an adaptive step size is introduced to improve its applicability. After finding the optimal r_c using IFFO, the DPC algorithm utilizes the IFFO to perform data clustering with the obtained r_c by comparing distance and density values to determine the clustering center. Figure 2 presents the clustering process of the proposed method.

3.3. **Parallel algorithm under Spark framework.** In order to reduce the computation time and improve the computational efficiency of the algorithm, a strategy for parallel execution on the Spark parallel framework is proposed. The main idea is to divide the dataset into multiple data regions in space, with the partitioned dataset scattered among the computing nodes of the Spark cluster. Each node calculates the local data on the node through the parallel IDPC algorithm to obtain the local clustering center points, and finally, the locally clustered results are again globally clustered to complete the entire clustering process. Figure 3 shows the flowchart of the parallel algorithm.

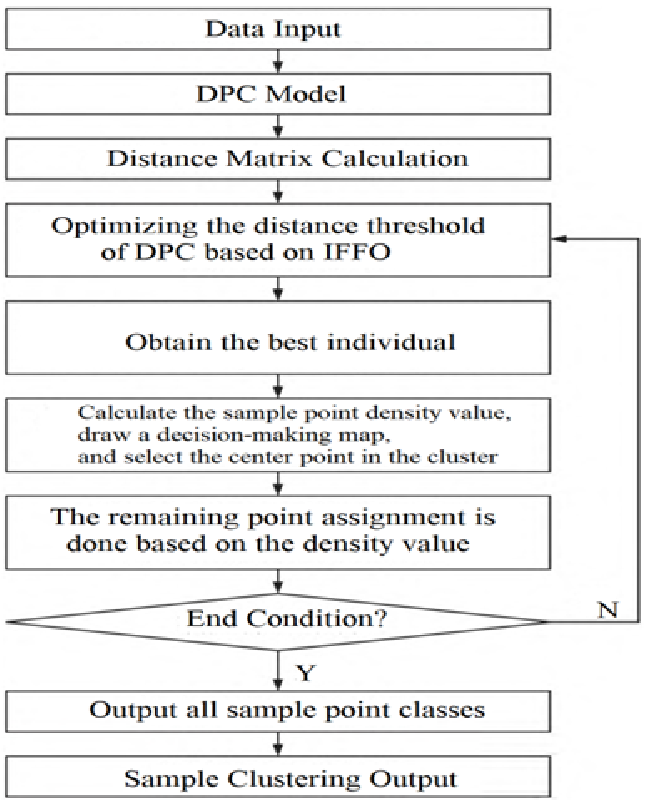

Data Input
DPC Model
Distance Matrix Calculation
Optimizing the distance threshold of DPC based on IFFO
Obtain the best individual
Calculate the sample point density value, draw a decision-making map, and select the center point in the cluster
The remaining point assignment is done based on the density value
End Condition?
N
Y
Output all sample point classes
Sample Clustering Output

Figure 2. Clustering flowchart of improved DPC

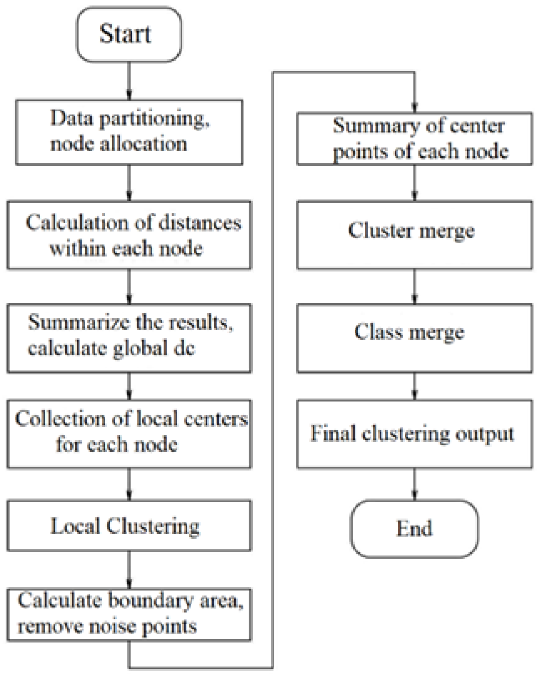

Start
Data partitioning, node allocation
Calculation of distances within each node
Summarize the results, calculate global dc
Collection of local centers for each node
Local Clustering
Calculate boundary area, remove noise points
Summary of center points of each node
Cluster merge
Class merge
Final clustering output
End

Figure 3. Parallel algorithm based on Spark framework

Step 1. Uniformly partition the data and slice it into spatially divided data regions according to the nodes, so that the data is divided into small, essentially consistent unit data, and each node is responsible for calculating a block of data. It should be noted that the local density calculation after partitioning is independent of other spatial data outside this node, and is only related to the local data center of the current node's data space. However, the truncation distance is calculated by summarizing the distances of data points calculated by each node after data partitioning to calculate the global truncation distance d_c.

Step 2. Calculate the center points of each node separately and find the clustering center points of the data block of that node.

Step 3. Merge local clusters globally. When dividing the data into grids for inter-data exchange, the dataset is partitioned into partially overlapping different data grid units to enable each sub-node to perform independent local data clustering. These overlapping partial regions contain some common data objects, which can be simply understood as a part of a cluster after global clustering. In the stage of merging local clusters globally in the third part, the algorithm can evaluate these common data objects, namely the features of these edge critical points, to determine the local clusters that need to be merged.

1) Input the local clustering results D of each node, where the data format of D is expressed in the form of Key-Value required by the Spark framework. The value carried by a data point q as the Key includes: the ID of the data unit where the data point q is located, the ID of the local cluster to which the data point q belongs within the data unit, and whether the data point is a core member. The final form can be represented as $(q, (q_{id}, q_{cl}, q_{core}))$

2) Group the clustering results D by $(q, (q_{id}, q_c, q_{core}))$, and judge the data groups to determine whether the number of data points is greater than 1, that is, whether the point is an edge critical point. Data points with a number of data points greater than 1 in the data group are listed as a data set M.

3) Traverse the data set M and judge whether the data points in the data set are core members based on q_{core}. Determine the important merging points in this way and list them as a new data set M_s.

4) Build a relationship connectivity graph based on q_c in M_s to obtain the relationship set T. Only a portion of the data information in M_s needs to be extracted in set T, and the data representation of set T is $((q_{1_{id}}, q_{1_{cl}}), (q_{2_{id}}, q_{2_{cl}}))$, where $q_{1_{id}}$ and $q_{2_{id}}$ represent two adjacent grid units.

5) Initialize the relationship connectivity graph $q_{2_{id}}$, with all local clusters as the vertices of $q_{2_{id}}$, and each vertex will have an edge pointing to it.

6) Traverse the relationship set T and use $((q_{1_{id}}, q_{1_{cl}}), (q_{2_{id}}, q_{2_{cl}}))$ as edges to fill the relationship connectivity graph G.

7) Obtain the relationship mapping between local clusters and global clusters through the complete relationship connectivity graph G.

8) Update the cluster labels.

4. **Experiments and Analysis.**

4.1. **The experimental environment and dataset configuration.**

The Spark environment is deployed in a virtual cluster environment using VMware Workstation Pro 15. The host server for the simulated cluster environment is PowerEdge R740, and the operating system for the virtual environment is Ubuntu 18.04.1. The virtual cluster configuration and other detailed information are shown in Table 1.

To experimentally measure the accuracy, computational efficiency, and other indicators of the proposed algorithm, this paper selects 6 unsupervised datasets from the UCI database and text datasets [27]. The 6 datasets used for unsupervised clustering algorithms are: 20 Newsgroup, Glass, Iris, Letters, Pendigits, and Sonar. Among these 6 datasets, the

Table 1. Experiment Environment.

Item	**Description**
Operating System	Windows 10
CPU	I5-12400F 2.5 GHz
RAM	64 GB
Hard Disk Drive	960 GB NVME SSD
Virtual Software	VMware Workstation Pro 15
Virtual Machine Operating System	Ubuntu 18.04.1
Virtual Machine Configuration	2 processor + 2GB RAM + 20GB disk
Spark Version	12.0.1

text datasets use the most commonly used 20 Newsgroup and Letters for text clustering experiments, with 20 Newsgroup including 20 news groups and nearly 20,000 messages. The remaining datasets are from the UCI database, and the detailed information of the unsupervised datasets is shown in Table 2.

Table 2. Dataset Statistics.

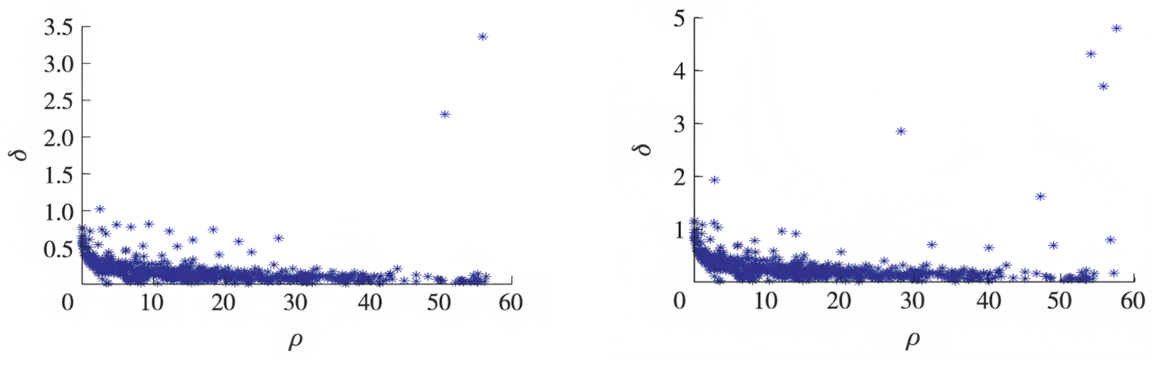
Dataset	**Class Number**	**Feature Dimensions**	**Data Volume**
20 Newsgroup	26214	20	18864
Glass	6	10	214
Iris	3	4	150
Letters	26	16	20000
Pendigits	10	16	10992
Sonar	2	60	208

The experimental dataset uses the Standard dataset. However, since the initial dataset only has a data size of 600×60 and a file size of 81KB, the original dataset is too small to meet the data volume requirements for performance testing. Therefore, in the experiment, the data in Standard is replicated and extended (to avoid completely duplicate samples with the original sample, the data of the replicated samples is the source data plus a random value in the range of [-0.1, 0.1]), and the original dataset is expanded by 10 times, 50 times, 100 times, 500 times, 1000 times, and 5000 times, respectively. The data volume and size are shown in Table 3.

Table 3. Expanded Dataset

Dataset	**Expansion Factor**	**Dataset Scale**	**File Size**
Standard	1	600 × 60	81 KB
Standard × 10	10	6000 × 60	816 KB
Standard × 50	50	30000 × 60	4082 KB
Standard × 100	100	60000 × 60	8161 KB
Standard × 500	500	300000 × 60	401807 KB
Standard × 1000	1000	600000× 60	81615 KB
Standard × 5000	5000	3000000 × 60	408078 KB

4.2. **The performance simulation of the decision graph with different sample sizes.** In the IDPC clustering process, the core content is to obtain an accurate and effective decision graph, which is used to select appropriate clustering center points, and then obtain the sample category based on the distance between the sample points and each center point. 1000 samples are selected from the 20 Newsgroup and Letters datasets for clustering, and the decision graphs obtained by the proposed algorithm for the 2 datasets are shown in Figure 4. As shown in Figure 4, different clustering decision graphs are obtained by selecting different data dimensions. When the data dimension is 16 for the Letters dataset, most of the sample points are distributed in the range of $\delta < 1$, and the sample point distribution range on the ρ axis is relatively wide. When the data dimension increases to 20 for the 20 Newsgroup dataset, the number of points located at $\delta > 2$ and $\rho > 50$ increases significantly. According to the center point selection strategy of the IDPC algorithm, the number of candidate center points increases, making it more difficult to select appropriate center points. Overall, as the sample data dimension increases, the maximum value of the local density ρ increases. This is because when the sample data dimension increases, the number of nodes within the clustering center threshold range increases, leading to an increase in the density value.



(a) The relationship between δ and ρ in letters dataset.

(b) The relationship between δ and ρ in 20 Newsgroup.

Figure 4. Examples of the decision graphs.

4.3. **The clustering performance with different R value ranges.** Firstly, in the improved FFO algorithm, the odor concentration change rate range is set to [0.7, 1.4], and different ranges $[R_{\min}, R_{\max}]$ are used for the proposed algorithm's clustering performance simulation on 6 sample sets, as shown in Table 4, and the optimal accuracy values are highlighted in bold. From Table 4, it can be seen that the upper and lower limits of R have a significant impact on the clustering algorithm's performance for the same dataset. This may be because the range of R affects the search step size and is sensitive to oscillations when optimizing r_c in DPC, making it difficult to obtain the r_c value that optimizes clustering performance. In the clustering process of the 6 different datasets, oscillations in accuracy due to changes in the R range were observed. Therefore, the R range of FOA has a significant impact on clustering performance for sample data. Considering the significant differences in the clustering performance of the 6 different datasets, in the subsequent experiments using the improved FOA+DPC clustering simulation, the R value range with the highest accuracy was selected for simulation.

4.4. **Comparison and analysis.** In order to verify the improvement performance of the proposed method on data clustering, DPC, FOA+DPC, and proposed IFOA+DPC algorithms were used to simulate clustering on the experimental dataset, and the comparison results are shown in the Figure 5. It can be seen that in terms of clustering accuracy performance, the difference between the DPC algorithm and the FOA-improved

Table 4. The impact of $R_{\max}$ and $R_{\min}$ on the accuracy of the proposed method (%).

Dataset	R_{min}	R_{max}			
		1.1	1.2	1.3	1.4
20 Newsgroup	0.7	36.17	36.52	36.40	35.85
	0.8	36.22	36.73	36.33	35.73
	0.9	36.34	36.60	35.98	35.40
	1.0	36.09	36.18	35.77	35.28
Glass	0.7	57.51	57.34	57.72	57.26
	0.8	57.67	57.75	57.12	57.36
	0.9	57.21	57.85	57.44	57.08
	1.0	57.69	57.28	57.19	57.23
Iris	0.7	91.36	91.56	91.44	91.80
	0.8	91.54	91.26	91.88	92.01
	0.9	91.59	92.15	91.37	91.84
	1.0	91.59	91.29	91.42	91.31
Letters	0.7	37.10	36.81	37.28	36.95
	0.8	37.03	37.45	36.51	37.08
	0.9	36.75	36.47	36.89	37.09
	1.0	36.55	36.63	36.60	36.63
Pendigits	0.7	64.31	65.18	65.00	64.98
	0.8	64.79	65.00	65.23	64.80
	0.9	64.77	65.80	64.97	64.44
	1.0	64.71	65.04	64.42	63.99
Sonar	0.7	59.51	61.16	61.18	60.97
	0.8	61.22	61.85	62.88	61.78
	0.9	61.82	62.48	62.55	62.01
	1.0	61.77	62.56	62.28	61.80

DPC algorithm is significant. This indicates that using only the DPC algorithm for data clustering is not very applicable, and randomly setting the rc value of DPC reduces its performance. Therefore, it is necessary to optimize the rc value of DPC using intelligent algorithms. The Silhouette values of FOA+DPC and prposed IFOA+DPC algorithms are close, but the performance of FOA is further enhanced after improvement. The proposed algorithm has the best clustering performance, followed by FOA+DPC, and DPC has the worst performance. This shows that after optimizing the rc of the DPC clustering algorithm using IFOA, the clustering stability is enhanced. This is mainly because the randomly set rc value interferes with the clustering stability during DPC operation, and FOA optimization obtains rc value that is more suitable for DPC clustering, reducing the clustering performance oscillations. In terms of iteration times, the DPC algorithm after IFOA optimization has fewer iterations, because in the proposed algorithm, the selection of DPC clustering centers is more appropriate, and fewer iterations are required to achieve the set clustering accuracy threshold.

4.5. **Clustering accuracy comparison.** The algorithms were compared and tested in a single-machine environment. Different algorithms were applied to the datasets, 5 independent runs were performed, and the average of the clustering results was selected for comparison. DBSCAN [7], Mean-shift [9], K-means [4] clustering, and the proposed algorithm were used to simulate the performance of the samples in Table 5, where in the

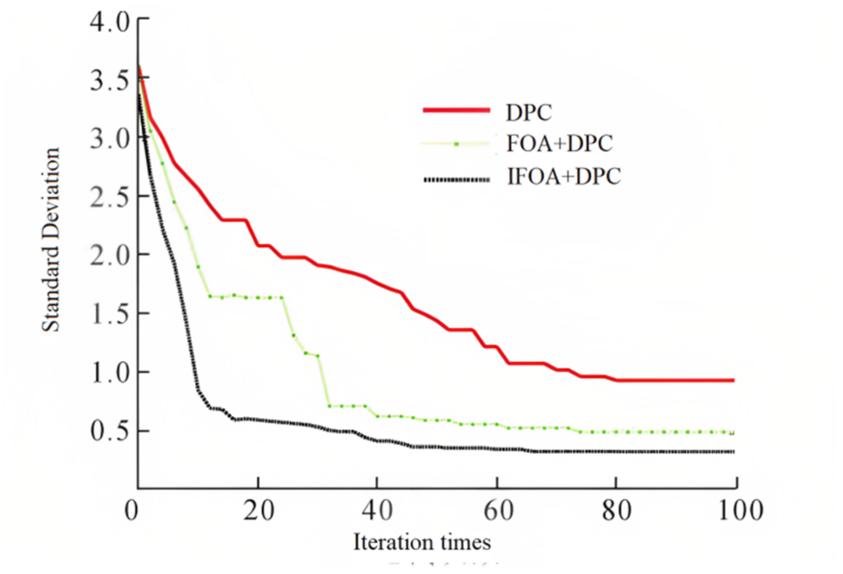


Figure 5. Standard deviation of the clustering accuracy for different methods.

proposed algorithm, the R value range was set to the optimal range. From the results, it can be seen that the proposed algorithm has the highest clustering accuracy, followed by the K-means algorithm, and the Mean-shift clustering has the lowest accuracy. This fully demonstrates the clustering performance and stability of the proposed algorithm.

Table 5. Clustering accuracy comparison (%).

Dataset	**DBSCAN**	**Mean-shift**	**K-means**	**Proposed Algorithm**
20 Newsgroup	26.17	30.16	35.08	36.73
Glass	43.35	52.85	56.27	57.85
Iris	78.92	85.40	90.98	92.15
Letters	29.85	35.19	37.33	37.45
Pendigits	49.93	61.28	64.77	65.80
Sonar	51.08	55.32	60.17	62.88

4.6. **Performance analysis of parallel algorithm.** In order to better verify the difference in computing time between the parallel algorithm and the original algorithm on the data volume, this simulation experiment verifies this performance characteristic by continuously increasing the data volume. All the experimental datasets are merged and divided, and 6 data sets of different sizes, 1 000, 5 000, 10 000, 15 000, 20 000, and 25 000, are generated for clustering analysis. The clustering execution time results are shown in Figure 6, with the unit of running time being ms. It can be seen that in the initial stage, when the data volume is small, there is not much difference in time consumption between the single-machine IFFO-DPC algorithm and the parallel algorithm based on Spark. However, with the increase of data volume, the parallel algorithm based on Spark gradually shows its computing advantage, and the algorithm parallelized by Spark is obviously much faster than single-machine computing. Therefore, the conclusion can be drawn that in large-scale data computing, the proposed algorithm based on parallel Spark is more efficient.

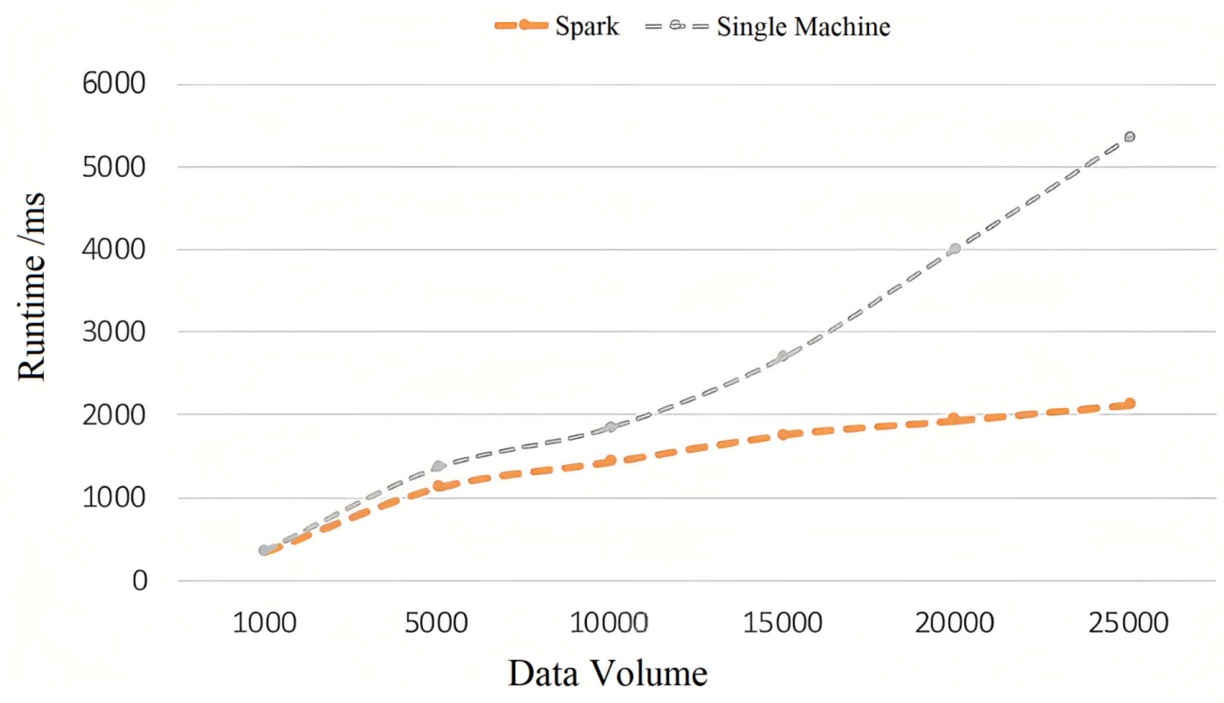


Figure 6. Comparison of running time between parallelization and stand-alone algorithms (ms).

5. **Conclusion.** Although clustering ensemble algorithms have improved clustering performance compared to single-base clustering algorithms, serial clustering ensemble algorithms are always limited by factors such as server computing power, memory capacity, and disk speed when facing high-dimensional massive datasets, and it is difficult to meet the speed requirements of the big data era. In order to improve the low computational efficiency of existing clustering ensemble algorithms in handling high-dimensional and large-scale data, a parallel clustering algorithm that can adapt to high-dimensional massive data is proposed. The experimental results show that by improving the DPC algorithm for high-dimensional data clustering, high clustering accuracy can be obtained. By optimizing the core parameter values ρ and δ of the DPC algorithm and selecting the point with a larger median value as the clustering center point for clustering, compared with commonly used high-dimensional data clustering algorithms, the proposed algorithm can achieve higher clustering accuracy performance and high clustering efficiency, and has high applicability in high-dimensional clustering.

REFERENCES

[1] Y. Zhou, Chalapathi N., Rathore A., Zhao Y., and B. Wang, "Mapper Interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data," in *2021 IEEE 14th Pacific Visualization Symposium (PVS 2021)*, IEEE, 2021, pp. 465-468.
[2] K. Djouzi, and B.-B. Kadda, "A review of clustering algorithms for big data," in *the 2019 International Conference on Networking and Advanced Systems (ICNAS 2019)*, IEEE, 2019, pp. 1-6.
[3] D. Granato, "Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective," *Trends in Food Science & Technology*, vol. 72, no.5, pp. 83-90, 2018.
[4] S.-D. Huang, Z. Kang, Z.-L. Xu, and W.-H. Liu, "Robust deep k-means: An effective and simple method for data clustering," *Pattern Recognition*, vol. 117, no. 11, pp. 107996, 2021.
[5] T.-Y. Wu, A. Shao, and J.-S. Pan, "CTOA: Toward a Chaotic-Based Tumbleweed Optimization Algorithm," *Mathematics*, vol. 11, no. 10, pp. 2339, 2023.

[6] T.-Y. Wu, H. Li, and S.-C. Chu, "CPPE: An Improved Phasmatodea Population Evolution Algorithm with Chaotic Maps," *Mathematics*, vol. 11, no. 9, pp. 1977, 2023.

[7] K.-M. Kumar, and A.-R.-M. Reddy, "A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method," *Pattern Recognition*, vol. 58, pp 39-48, 2016.

[8] Y. Li, L. Sun, Y. Tang, and W. You, "A review of related density peaks clustering approaches," in *2022 14th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC 2022)*, IEEE, pp. 145-149, 2022.

[9] L. You, H. Jiang, J. Hu, C.-H. Chang, L. Chen, X. Cui, and M. Zhao, "GPU-accelerated faster mean shift with euclidean distance metrics," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC 2022)*, IEEE, 2022, pp. 211-216.

[10] F. Zhang, T.-Y. Wu, and Y. Wang, "Application of quantum genetic optimization of LVQ neural network in smart city traffic network prediction," *IEEE Access*, vol. 8, pp. 104555-104564, 2022.

[11] A L H P Shaik, M K Manoharan, and A K Pani, "Gaussian mutation–spider monkey optimization (GM-SMO) model for remote sensing scene classification," *Remote Sensing*, vol. 14, no. 24, pp. 6279, 2022.

[12] L. Kang, R.-S. Chen, and N. Xiong, "Selecting hyper-parameters of Gaussian process regression based on non-inertial particle swarm optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504-59513, 2019.

[13] C.-M. Chen, S Lv, J. Ning, "A genetic algorithm for the Waitable time-varying multi-depot green vehicle routing problem," *Symmetry*, vol. 15, no. 1, pp.124, 2023.

[14] F.-J. Li, Y.-H. Qian, J.-T. Wang, C.-Y. Dang, and L.-P. Jing, "Clustering ensemble based on sample's stability," *Artificial Intelligence*, vol. 273, pp. 37-55, 2019.

[15] M. Zhang, "Weighted clustering ensemble: A review," *Pattern Recognition*, vol. 124, 108428, 2022.

[16] A. Nazari, A. Dehghan, S. Nejatian, V. Rezaie, and H. Parvin, "A comprehensive study of clustering ensemble weighting based on cluster quality and diversity," *Pattern Analysis and Applications*, vol. 22, pp. 133-145, 2019.

[17] L. He, N. Ray, Y.-S. Guan, and H. Zhang, "Fast large-scale spectral clustering via explicit feature mapping," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1058-1071, 2018.

[18] J.-S. Wu, W.-S. Zheng, J.-H. Lai, and C.-Y. Suen, "Euler clustering on large-scale dataset," *IEEE Transactions on Big Data*, vol. 4, no. 4, pp. 502-515, 2017.

[19] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol, 32, no. 6, pp. 1212-1226, 2019.

[20] T.-A. Kumar, K. Sharma, and M. Bala, "A novel clustering method using enhanced grey wolf optimizer and mapreduce," *Big Data Research* vol.14, pp.93-100, 2018.

[21] Y.-H. Kim, K. Shim, M.-S. Kim, and J.-S. Lee, "DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce," *Information Systems*, vol. 42, pp. 15-35, 2014.

[22] T.-H. Sardar, and Z. Ansari, "An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 200-209, 2018.

[23] Y.-H. Yang, J.-F. Zhang, J.-B. Wang, X.-B. Xu, G.-F. Shao, and Y.-H. Wen, "Spark-based improved basin-hopping monte carlo algorithm for structural optimization of alloy clusters," *Physics Letters A*, vol. 383, no.5, pp. 464-470, 2019.

[24] H. Behrooz, and K. Kiani, "A big data driven distributed density based hesitant fuzzy clustering using Apache spark with application to gene expression microarray," *Engineering Applications of Artificial Intelligence*, vol.79 pp. 100-113, 2019.

[25] J.-H. Cai, H.-L. Wei, H.-F. Yang, and X.-J. Zhao. "A novel clustering algorithm based on DPC and PSO," *IEEE Access*, vol. 8, pp. 88200-88214, 2020.

[26] T. Bezdan, C. Stoean, A.-A. Naamany, N. Bacanin, T.-A. Rashid, M. Zivkovic, and K. Venkatachalam, "Hybrid fruit-fly optimization algorithm with k-means for text document clustering," *Mathematics*, vol. 9, no.16, 1929, 2021.

[27] S. Aggarwal, A. Verma, and J. Singh, "Application based categorization of datasets for implementing data mining techniques," In *2nd Global Conference for Advancement in Technology (GCAT 2021)*, IEEE, 2021, pp. 1-7.