Journal of Network Intelligence ©2025 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **10**, Number **2**, May **2025**

Data Mining Based on Improved Density Peak Clustering in Parallelised Teaching Platforms

Song Zhang*
College of Architecture and Environmental Engineering
Zhengzhou Technical College, Zhengzhou 450121, P. R. China
zs19871224@163.com

Yu Chang
Department of Pharmaceutical and Chemical Engineering
Zhengzhou Business Technician Institute, Zhengzhou 450121, P. R. China
changyu1987007@163.com

Jun-Feng Cai
Baekseok University, Seoul 06695, South Korea
xe4909@163.com

*Corresponding author: Song Zhang
Received March 27, 2024, revised July 12, 2024, accepted November 6, 2024.

Abstract. Data mining in large-scale instructional platforms may suffer from high data dimensionality and complex data formats. Density Peaking Clustering (DPC), as a clustering algorithm, does not rely on the assumption of data distribution and is able to discover clusters of arbitrary shapes, which is useful for dealing with complex or irregular datasets. However, the DPC algorithm also has its limitations, such as the difficulties that may be encountered when dealing with clusters with different densities and the relatively high computational complexity of the algorithm. Therefore, in order to perform data mining effectively in Spark teaching platform, this paper proposes an automatic determination of centre DPC (NPADPC) based on normalisation process and gives the corresponding parallelisation design method. First, the principle of DPC is analysed. Then, a DPC based on normalisation processing is provided, the basic idea of which is to normalise the local density (ρ) and distance (δ) in the decision function to ensure that these two parameters are uniformly distributed across different data sets. For each data point, a Gaussian kernel function is used to compute the local density and distance to ensure that these parameters are comparable across different datasets and different feature spaces. Then, the dataset to be clustered is partitioned into intervals such that the data in each interval is traversed in this interval. The experimental results show that the average accuracy of NPADPC ranges from 0.9287 to 0.9458 and is generally higher than the mean value of DPC for data on large online learning platforms. The running time of the parallel NPADPC algorithm decreases significantly as the data samples rise.
Keywords: Data mining; Clustering algorithm; Peak density; Parallel computing; Gaussian kernel function; KNN

1. Introduction.

Data mining techniques have emerged as a way to make use of data and to obtain knowledge and information from data. However, traditional data mining techniques are still somewhat inadequate to deal with modern massive data although they can still be of significant effectiveness in most cases [1, 2]. Therefore, while researching new data mining algorithms, it has become a hot research topic to study how to apply these data mining algorithms to contemporary massive data and improve the operation efficiency of the algorithms [3, 4, 5].

Traditional data mining algorithms applied directly to distributed databases are not effective as it implies a significant communication overhead, and the computational environment thus becomes an important change in exploiting the scalability of the system for high performance data mining implemented in a distributed manner [6]. In traditional data mining techniques, centralised approaches are largely unsuitable due to the sheer volume of data, the infeasibility of storing data centrally at multiple sites, bandwidth constraints, energy constraints and privacy concerns [7]. Therefore, it is important to develop a more adaptable and flexible mining framework to discover hidden, useful and meaningful patterns and it is particularly important to obtain information from distributed and complex databases rather than centralised databases. Big data mining research focuses on data mining in distributed environments [8].

Clustering-based data mining in Spark's parallelised teaching platform demonstrates high research value, mainly in its ability to efficiently process large-scale teaching data [9, 10]. Using Spark's distributed computing framework, multi-dimensional data such as students' learning behaviours, grades, and interactions can be quickly clustered and analysed. This kind of analysis helps educators to identify students' learning patterns and group characteristics, which can lead to more accurate allocation of teaching resources and formulation of personalised teaching strategies. For example, clustering can identify students' common problems with specific course content. In addition, clustering analysis can provide data support for course design and help educators optimise course structure and content to better suit the learning needs of different students [11]. Therefore, clustering data mining on the Spark platform can not only improve the quality of teaching and students' learning experience, but also promote the innovation of educational research and practice, which has far-reaching educational significance and application prospects. Data mining in large-scale teaching platforms may have the problems of high data dimensionality and complex data format. Density Peaking Clustering (DPC), as a clustering algorithm [12, 13], does not rely on the assumption of data distribution and is able to discover clusters of arbitrary shapes, which is very useful for dealing with complex or irregular datasets. Therefore, the aim of this study is to apply DPC to a Spark-based teaching platform and improve it in order to achieve accurate and efficient data mining tasks.

1.1. **Related work.**

In the field of data mining and analysis, clustering is one of the important algorithms, the purpose of which is to be able to divide the data with high correlation in the same dataset together, thus forming a cluster. Density-based clustering is different from other clustering methods in that it does not need to point out the number of clusters to be clustered before clustering, and density-based algorithms are insensitive to noise points, and are able to get rid of the disadvantage of distance-based clusters that can only find circular clusters, and identify clusters of arbitrary shapes and sizes, and thus have a wide range of applications in the field of data mining.

Kerm [14] proposed a new local density estimation method, given the number of clusters, which can automatically perform clustering and improve the clustering without human intervention. Li et al. [15] proposed a fuzzy density clustering method for effective selection of clustering centres, which can effectively and adaptively select the clustering centres and obtain better clustering results. Das et al. [16] proposed a method with low computational complexity for automatic selection of clustering centres and used the Canopy algorithm to automatically decide the threshold value of clustering centres. Hulme et al. [17] proposed a selection mechanism for automatically determining the number of cluster centres based on the B-DPC algorithm for cluster boundary partitioning, which improves the basic density peak clustering and also reduces the computational complexity by downgrading the dimensionality. Li et al. [18] improved the DPC algorithm by maximising the average profile index to develop an automatic clustering centre-of-mass selection method that utilises the density estimation entropy approach to select and optimise the DPC parameters. He and Tan [19] used an automatic cluster centroid selection strategy for the purpose of automatic cluster centre selection and adopted a new method to calculate the relative density, i.e., nearest neighbour information. The proposed algorithm based on this is able to cluster any dataset quickly and accurately using decision diagrams. Seyedi et al. [20] proposed a new density peak clustering method called IDPC which not only uses local density to identify cluster centres but also proposes a new label propagation method to form clusters.

Mehmood et al. [21] proposed a Sheather-Jones based adaptive estimation method to determine the truncation distance for clustering algorithms. Cui et al. [22] proposed an overdiffusion-based method, which is a nonparametric method for estimating the probability distribution of a given dataset, that takes into account both the choice of the truncation distance and the boundary correction of the kernel density estimation. Based on thermal diffusion in an infinite domain, Nutz et al. [23] proposed a new method for automatically extracting the threshold of the original dataset using the potential entropy of the data field, which enables a clustering process for adaptive density peak detection.

1.2. Motivation and contribution.

The algorithms presented above are all further improvements to the DPC algorithm for cluster centre selection. Although most of the studies have analysed the potential of each data sample point as a cluster centre, no in-depth research has been done on how to correctly select the cluster centres. In summary, the research of DPC algorithm on automatic determination of cluster centres has also achieved some results, but the problem of how to select the number of cluster centres efficiently and reasonably is still an urgent problem. Therefore, in order to solve the above problem, an automatic determination of centre DPC based on normalization process (NPADPC) is proposed. The main innovations and contributions of this work include: (1) Aiming at the problem of the traditional DPC method, which needs to rely on experience to select the clustering centre, a method based on the normalisation process to determine the clustering centre automatically is provided. In addition, a strategy is developed to dynamically adjust the truncation distance so that it can be automatically adjusted according to the local density distribution of the data set. For each data point, a Gaussian kernel function is used to compute the local densities and distances, ensuring that these parameters are comparable across datasets and different feature spaces. (2) Aiming at the problem that the proposed NPADPC algorithm has high computational complexity and low running efficiency in Spark teaching platform, a parallel NPADPC algorithm based on Spark is proposed.

2. Principles of peak density clustering.

2.1. Dataset segmentation.

Density Peaking Clustering (DPC) is a density-based clustering algorithm proposed by Alex Rodriguez and Alessandro Laio in 2014. The core idea of the algorithm is to find peaks of density in the data space, which are considered to be the core of the clustering [24, 25]. The algorithm does not require a pre-specified number of clusters, is able to recognise clusters of arbitrary shape and is robust to noisy points.

Let N datasets X be classified into k classes, which are mathematically represented as $C = \{C_1, C_2, \ldots, C_k\}$, where $k \leq N$, $X = C_1 \cup C_2 \cdots \cup C_k$, and $C_i \cap C_j = \emptyset (i \neq j)$.

The distance from point x_i to point x_j is r_{ij}, then the distance from x_i to x_j is shown as follows:

$$r_{ij} = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{in} - x_{jn}|^2} \tag{1}$$

2.2. **Local density and distance.** For each point in the dataset, the number of points in its neighbourhood is calculated as the local density of that point. The local density ρ_i of the data point x_i is shown below:

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

where r_c is the truncation distance and χ denotes the density kernel function.

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

Let the kernel function be a Gaussian function, then ρ_i can be expressed as:

$$\rho_i = \sum_j \exp\Big(-\frac{r_{ij}^2}{2\, r_c^2}\Big) \tag{4}$$

For each point, find the nearest point in its neighbourhood that has a local density greater than that point and calculate the distance between them. The minimum distance corresponding to a point x_i is denoted as the distance to the nearest point to x_i with a density value greater than ρ_i:

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

Determine the point i as a centroid of the cluster according to Equation (4) and Equation (5).

In order to prevent the situation where only one of the values of ρ_i and δ_i is larger and the other smaller, Equation (6) is generally used for the calculation:

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

2.3. **Decision function.** Combining local densities and distances, a decision function is defined for each point, which is used to evaluate whether the point is a clustering centre or not. The cluster centre points are selected for clustering based on a combination of ρ_i, δ_i and γ_i values. Figure 1 expresses the selection process of cluster centroids more intuitively.

It can be seen that the samples are initially distributed into 3 categories and the respective distances are calculated based on the distribution of features of each sample point according to Equation (1).

2.4. **Cluster centre selection and assignment.** Based on the value of the decision function, the clustering centres are selected, usually those points with high decision function values. The point with the highest decision function value is selected as the clustering centre and then this process is iterated until the desired number of clusters is reached or the stopping condition is satisfied [26]. Finally, the other points are assigned to the nearest cluster centres to form clusters. The density ρ and distance value δ are calculated

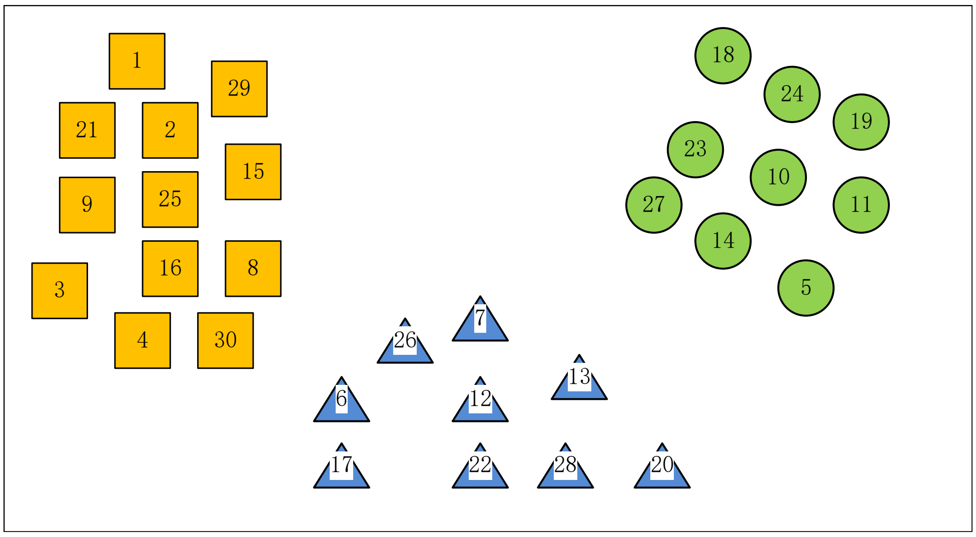

Figure 1. Distribution of data points

according to Equation (4) and Equation (5), respectively, and their generated 2D decision distribution graphs are shown in Figure 2.

It can be seen that the 3 clustering centroids are 10, 12 and 25. After selecting the centroids, all the sample points are classified according to the number of centroids. For the remaining points to be classified, the category closest to the centroid is selected as the clustering category for that point according to the density value.

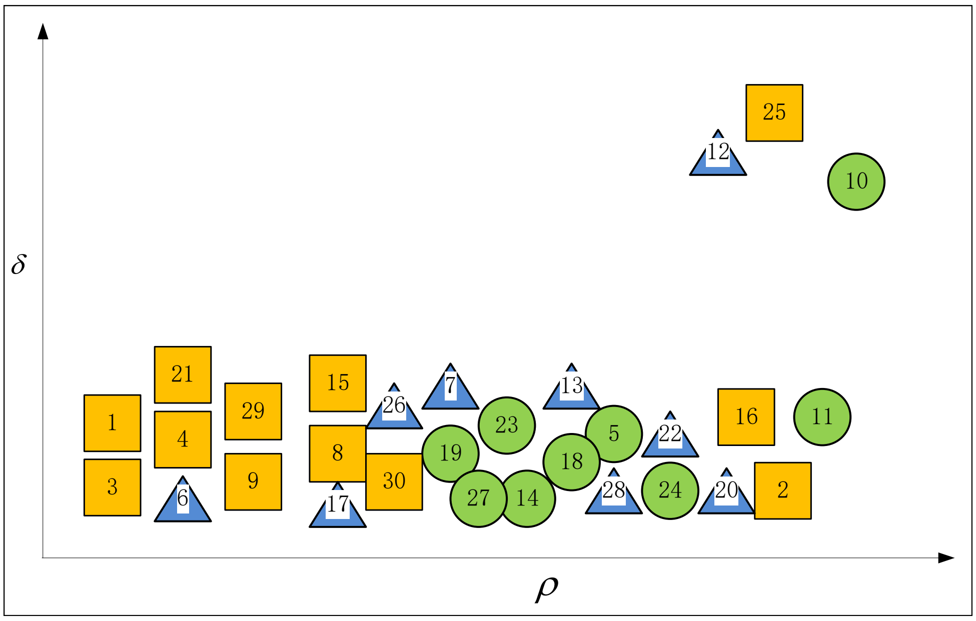

Figure 2. Decision chart

3. **Automatic determination of the centre DPC based on the normalization process.**

3.1. **Basic idea.** The traditional DPC method has the defect of needing to rely on experience to select the clustering centre, so this paper provides an automatic determination of centre DPC (NPADPC) based on the normalization process, whose basic idea is to

normalize the local density (ρ) and distance (δ) in the decision function to ensure that these two parameters are uniformly distributed across different datasets.

For each data point, a Gaussian kernel function is used to compute the local density and distance, ensuring that these parameters are comparable across datasets and different feature spaces [27]. Normalisation reduces the subjectivity of parameter selection and improves the stability and adaptability of the algorithm. Based on the normalised local density and distance, a decision function is constructed which is able to combine these two parameters to evaluate the potential of each data point as a clustering centre. In addition, a strategy is developed to dynamically adjust the truncation distance so that it can be automatically adjusted according to the local density distribution of the dataset. In this paper, a density-based method, such as KNN, is used to determine the appropriate truncation distance.

3.2. **Calculation of local densities and distances.** First, we compute the local density of each data point, i.e. the number of neighbouring points within a given truncation distance. Also, for each point, we find the nearest point in its neighbourhood whose local density is greater than that point and calculate the distance between them.

The local density ρ_i of each data point i is calculated, usually by determining a truncation distance δ to limit the size of the neighbourhood.

$$\rho_i = \sum_{j=1}^{n} K_\delta(d_{ij}) \tag{7}$$

where K_δ is the Gaussian kernel function, and d_{ij} is the distance between point i and point j.

For each data point i, find the nearest point j whose local density in its neighbourhood is greater than ρ_i and calculate the distance between i and j to obtain δ_i.

$$\delta_i = \min_{j\neq i,\ \rho_j>\rho_i} d_{ij} \tag{8}$$

3.3. **Normalisation.** Normalisation of local density and distance is used to avoid the problem of inconsistent clustering results in different datasets or feature spaces due to differences in magnitude or data distribution. This is usually achieved by dividing the local density and distance of each point by the maximum of all points.

Local densities were normalised to avoid the effect of magnitudes between different datasets.

$$\rho_i' = \frac{\rho_i}{\max(\rho_1, \rho_2, \ldots, \rho_n)} \tag{9}$$

Similarly, distances were normalised to maintain consistency across datasets.

$$\delta_i' = \frac{\delta_i}{\max(\delta_1, \delta_2, \ldots, \delta_n)} \tag{10}$$

3.4. **Construction of the decision function.** Combining the normalised local densities and distances, we construct a decision function that evaluates the potential of each point to become a clustering centre. The decision function used is a combination of local density and distance.

The normalised local density ρ_i' and the normalised distance δ_i' are combined to construct the decision function γ_i.

$$\gamma_i = \frac{\rho_i'}{\delta_i'} \tag{11}$$

3.5. **Selection of clustering centres.** By analysing the values of the decision function γ_i, the point with the highest γ value is selected as the clustering centre. The number of clustering centres is determined by setting a threshold method.

$$\gamma_i = \rho_i' \cdot (\delta_i')^{\beta} \tag{12}$$

where α and β are parameters used to adjust the relative importance of ρ_i' and δ_i'. Once the clustering centres are selected, we form clusters by assigning each data point to the nearest clustering centre. This process can be done by calculating the distance of each point to each cluster centre and assigning the point to the nearest centre. The clusters are formed by

$$C_k = \big\{ i \mid \arg\min_j \|x_i - \mu_j\|^2 \le \operatorname{dist}(x_i, C_k) \big\} \tag{13}$$

where C_k is the k-th cluster, μ_j is the mean of the j-th cluster centre, and $\|\cdot\|^2$ denotes the square of the Euclidean distance.

3.6. **KNN-based dynamic adjustment of truncation distance.** The selection of the truncation distance r_c is very important during the implementation of the DPC algorithm. r_c determines both the clustering accuracy and the execution efficiency of DPC. DPC is poor in dealing with the uneven distribution density of sample points due to the large influence of r_c, so this paper adopts the KNN algorithm [28,29] to improve DPC.

Let the entire set of samples to be clustered be X, where $x_i \in X$, $\operatorname{dist}(\cdot,\cdot)$ is the density distance function, and $\mathrm{NN}_k(x_i)$ denotes the set of points that are the kth closest to x_i. The KNN representation of x_i is shown below:

$$\mathrm{KNN}(x_i) = \{\, x_j \in X \mid \operatorname{dist}(x_i, x_j) \le \operatorname{dist}(x_i, \mathrm{NN}_k(x_i))\} \tag{14}$$

The density of KNN is shown below [30]:

$$\rho_i = \exp\Big(-\frac{1}{k} \sum_{x_j \in \mathrm{KNN}(x_i)} \operatorname{dist}(x_i, x_j)^2\Big) \tag{15}$$

$$\delta_i = \begin{cases} \min_j\big(\operatorname{dist}(x_i, x_j)\big), & \exists\, j \text{ s.t. } p_j > p_i, \\ \max_j\big(\operatorname{dist}(x_i, x_j)\big), & \text{otherwise}, \end{cases} \tag{16}$$

Next, a local density map is constructed which shows the local density of each point in the dataset. The local density can be estimated by calculating the average of the KNN distances for each point:

$$\rho_i = \frac{1}{K} \sum_{k=1}^{K} d_{i,k} \tag{17}$$

$$d_{i,k} = \min_{j \ne i} \|x_i - x_j\|, \quad \text{for } k = 1, \dots, K \tag{18}$$

where $d_{i,k}$ is the distance between the point x_i and its k nearest neighbour, and $\|\cdot\|$ denotes the Euclidean distance or other distance measure.

Using a local density map, a density threshold τ is determined. First, the local densities ρ_i of all the data points are ranked, from smallest to largest. This paper uses the Cumulative Density Distribution (CDF), which is the proportion of points less than or equal to each local density value. A local density value that makes the cumulative density reach a particular value is chosen as τ. The cumulative density distribution is calculated as:

$$F(\rho) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\rho_i \leq \rho) \tag{19}$$

where $\mathbb{I}$ is the indicator function that takes the value of 1 when $\rho_i \leq \rho$ and 0 otherwise.

Dynamically adjust the truncation distance δ according to the density threshold τ. The truncation distance δ should be greater than the minimum distance in the local density map and at the same time less than the density threshold τ.

$$\delta = \max\big(\min \ \text{distance in } \rho_i, \ \tau\big) \tag{20}$$

Based on the updated cluster centres, each data point is assigned to the nearest cluster centre to form clusters.

$$C_k = \left\{ x_i \mid \arg\min_j \|x_i - x_j\| \leq \delta \right\} \tag{21}$$

where C_k is the k-th cluster, and $\|x_i - x_j\|$ denotes the distance between the point x_i and the cluster centre x_j.

Through these steps, the truncation distance can be dynamically adjusted according to the local density distribution of the dataset, which improves the performance and adaptability of the density peak clustering algorithm. This method is especially suitable for the case of uneven data density, which can better identify and separate clusters. After optimising the DPC using KNN, the r_c parameter selection of the DPC is no longer necessary, but the k value of the nearest neighbours needs to be selected. After obtaining the samples to be clustered, the distance between two and two of the sample points to be clustered is calculated to generate a collection of distance matrices. Then calculate the density value and distance value of each sample point and select the sample point with the larger of the two as the centroid of the sample clustering. Calculate the distance value of the remaining nodes relative to each centre point, and select the category to which the closer centre point belongs as the category of each remaining node.

4. **Parallel NPADPC based on Spark teaching platform.**

4.1. **Pedagogical platform overview.**

Apache Spark is an open source distributed computing system designed for large-scale data processing. It originated from AMPLab at UC Berkeley and aims to address the shortcomings of the Hadoop MapReduce model in terms of iterative computation, real-time processing, and in-memory computation [31, 32]. Spark provides an easy-to-use programming model and a rich set of APIs with support for a variety of programming languages such as Scala, Java, Python, and R. It is designed to be used in a variety of applications such as data processing, data management, and data analytics.

Spark improves processing speed by executing computations in memory, enabling faster processing of large-scale datasets compared to Hadoop MapReduce.Spark provides a simple and intuitive programming model that makes the development of parallel applications easier through the RDD (Resilient Distributed Dataset) abstraction.Spark supports a variety of computational modes, including batch processing Spark supports a variety of computational modes, including batch processing, interactive querying, stream processing, and machine learning, making it a versatile platform for processing big data.Spark can run in small clusters up to large clusters of thousands of machines, providing excellent horizontal scalability.Spark is fault-tolerant through the lineage information of RDDs, which enables recomputation of lost data in the event of node failures.Spark Spark has a rich ecosystem of components, including Spark SQL (for structured data processing), Spark Streaming (for real-time data processing), MLlib (for machine learning), and GraphX (for graph computation). these features make Spark a popular choice for big data processing and analytics, especially for scenarios that require fast iteration and real-time processing. The structure of the Spark ecosystem is shown in Figure 3.

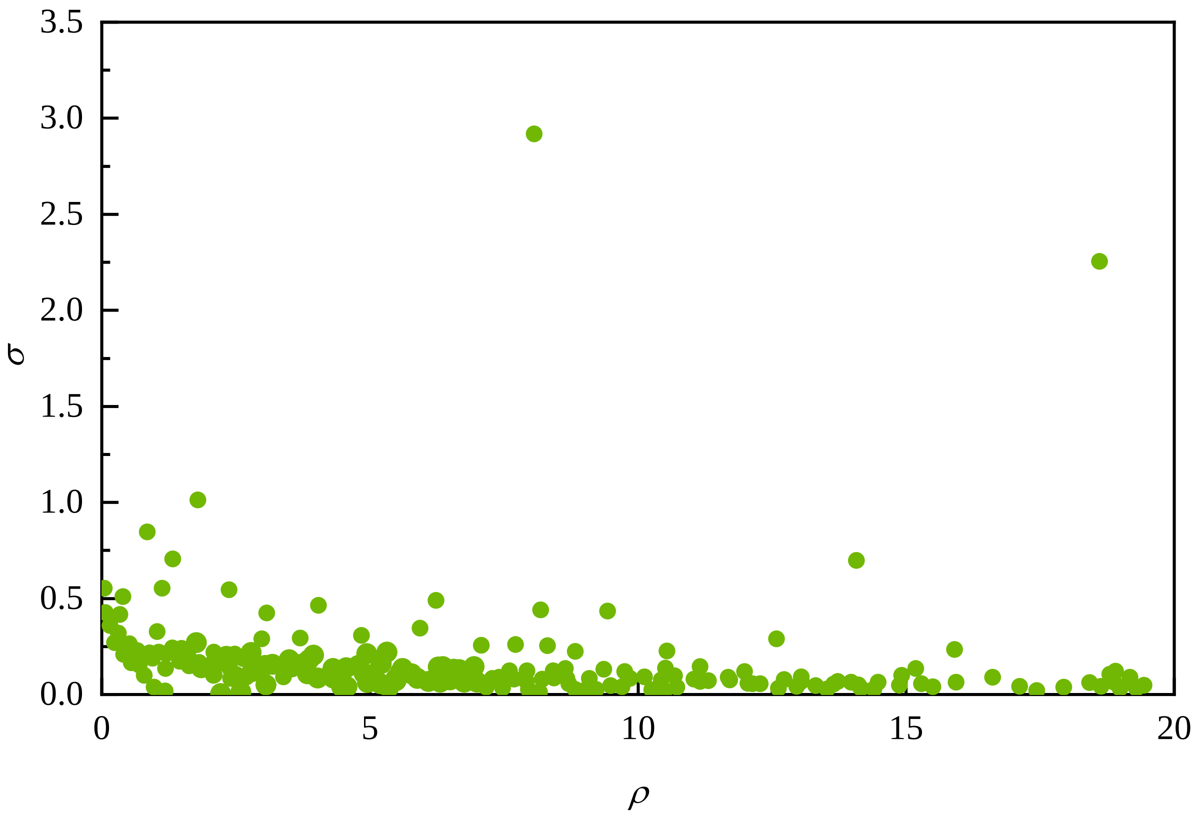

Figure 3. Spark ecosystem

The Spark-based Parallelisation Teaching Platform is a teaching platform based on the Apache Spark distributed computing framework, which aims to provide efficient parallel computing capabilities and rich teaching resources to help teachers and students better understand and apply parallel computing technologies.

Through the Spark-based parallelisation teaching platform, teachers can create and publish parallel computing courses and assignments, including theoretical knowledge lectures, practical case studies and programming practice. Students can complete course learning, assignment submission and experimental practice on the platform, and with the computing resources and tools provided by the platform, they can deeply understand the principles and technologies of parallel computing and improve their parallel programming ability. The features of the platform include:

(1) Distributed computing support: Spark-based parallelisation teaching platform provides distributed computing resources and tools to support theoretical learning and practical operation of parallel computing.

(2) Rich teaching resources: The platform provides rich teaching resources, including course handouts, experimental cases, programming examples, etc., to help students better learn and apply parallel computing technology.

(3) Real-time interaction and feedback: The platform supports real-time classroom interaction and online feedback, so teachers and students can communicate and solve problems in a timely manner to improve teaching effectiveness and learning experience.

(4) Automated assessment and feedback: The platform provides an automated assignment assessment and feedback system to help teachers efficiently assess students' learning outcomes and provide personalised learning advice and guidance to students.

The Spark-based parallelisation teaching platform can effectively promote the teaching and learning of parallel computing technology, cultivate students' parallel programming ability, and problem solving ability, and promote teachers' and students' in-depth exploration and application in the field of parallel computing.

4.2. **Design of the parallel NPADPC algorithm.** Although the proposed NPADPC algorithm can discover arbitrary shape clusters and automatically determine the number of clustering centres, its calculation of local density and distance in its decision function requires traversing the entire dataset, resulting in a relatively low running efficiency in the Spark teaching platform. In order to solve the problem of high computational complexity and low running efficiency of NPADPC algorithm, this paper proposes a parallel NPADPC algorithm based on Spark.

Firstly, parallel computing frameworks such as Apache Spark are utilised to speed up the clustering process. The overall running time of the algorithm can be significantly reduced by partitioning the dataset into multiple small chunks and performing local clustering on multiple compute nodes simultaneously. The entire dataset is then divided into multiple data partitions, each containing an approximately equal number of data points. This ensures that each compute node handles roughly the same amount of data, thereby achieving load balancing. Next, the NPADPC algorithm is applied independently on each data partition for local clustering. This means that each partition determines its clustering centre and clustering result independently. Finally, after the local clustering is completed, the clustering results of all partitions are combined for global clustering. This step involves the merging of clustering centres across partitions and the redistribution of data points to ensure that the final clustering results reflect the global structure of the entire dataset. The steps of the parallel NPADPC algorithm are as follows:

Step 1: Partition the large dataset into multiple smaller data partitions (or chunks) for parallel processing on different compute nodes.

Step 2: Calculate the local density ρ_i and distance δ_i on each data partition. The local density ρ_i is calculated as

$$\rho_i = \sum_{j \in N_\delta(i)} K_{r_c}\left(d_{ij}\right) \tag{22}$$

where $N_\delta(i)$ is the set of neighbouring points of point i within the truncation distance r_c, K_{r_c} is the Gaussian kernel function, and d_{ij} is the distance between point i and point j.

Step 3: Calculate the distance δ_i as

$$\delta_i = \min_{j \in N_\rho(i)} d_{ij} \tag{23}$$

where $N_\rho(i)$ is the set of neighbouring points whose local density ρ_j of point i is greater than ρ_i.

Step 4: Select local clustering centres independently on each data partition, usually choosing the point with the highest local density as the clustering centre.

Step 5: Assign each data point to its nearest local clustering centre to form local clusters.

Step 6: Bring together all the local clustering centres and determine the global clustering centre by global clustering method (K-Means).

Step 7: Re-allocate all data points using the global clustering centres to form the final clustering result.

5. **Experimental results and analyses.**

5.1. **Description of the experimental environment.** In order to validate the performance of the parallel NPADPC algorithm in big data clustering, a validation analysis is performed. The source of experimental data is an online large-scale learning platform, as shown in Table 1. Firstly, different sample sizes are clustered and simulated to generate the decision diagram of DPC and analyse its performance; then DPC and parallel NPADPC are used to perform clustering operations respectively.

Table 1. Sample set of online learning platforms

Data set number	Sample size	Number of attributes	Number of categories
1	2000	17	8
2	3000	20	8
3	4000	25	8
4	5000	35	8

The Spark cluster built for the experiment consists of a cloud platform, mainly two Dell PowerEdge R930 rackmount servers. A heterogeneous five-node distributed computing platform was built using these two servers as the basis. The parameters of the hardware of the server platform are shown in Table 2. The Spark platform requires many components of the Hadoop platform such as HDFS, YARN, etc. to run the data. The cluster uses virtualisation technology to virtualise each server into multiple logical hosts, i.e., each server is partitioned into multiple mutually independent and non-interfering virtual environments. One logical computer is used as the master node of the cluster and the remaining logical computers are used as data nodes. Logical computers can be specifically configured as well as added or deleted according to the requirements of the experiment. The operating system of each logical computer is Ubuntu 16.04 LTS, and the Spark version number is 2.4.0. The two servers used are heterogeneously constructed for cluster distribution, as shown in Table 3.

Table 2. Parameters of experimental platform hardware environment

Basic parameter	Instructions
Processing unit	2×Intel Xeon E7-4820 v4; 2.0 GHz 25 M cache; 6.4 GT/s OPII10C/20T
RAM	256 G
Hard drive	Mechanical HDD: 2×480 G, SSD: 2 T
CD or DVD Drive	DVD-ROM, SATA, Internal
Network card	Quad-Port Gigabit Ethernet Card

Each machine needs to deploy a Java environment, and the experimental uniform version number is 1.8. In addition, in order to facilitate the communication access between each cluster node, it is necessary to establish a master node to the slaver node confidential communication, that is, the public key of the master master node is added to the granting of each slaver node's information in the key pool. Subsequently, according to the Spark cluster configuration needs to modify the corresponding configuration files: `hdfs-site.xml`, `core-site.xml`, `mapred-site.xml`, `yarn-site.xml` and so on.

Table 3. Node cluster deployment

Node type	Hostname (of a networked computer)	IP
Master	Masrero1	192.168.24.10
Slaver	Slaver01	192.168.24.01
Slaver	Slaver02	192.168.24.02
Slaver	Slaver03	192.168.24.03
Slaver	Slaver04	192.168.24.04
Slaver	Slaver05	192.168.24.05

5.2. **Decision map performance.** In the DPC clustering process, the core element is to obtain a decision map that is effective in terms of accuracy. The decision map is used to select the appropriate centroids of the clustering categories and then the sample categories are obtained based on the distance of the sample points from each centroid. The decision map solved by the parallel NPADPC algorithm for dataset 1 is shown in Figure 4.

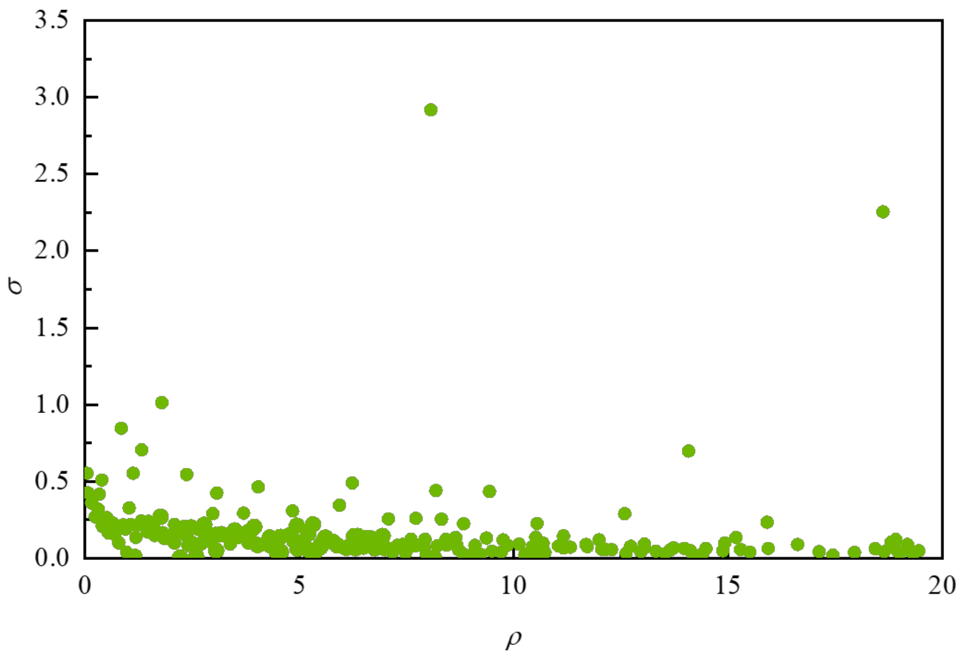

Figure 4. Decision diagram (dataset 1)

As a whole, the local density ρ maximum increases when the sample size becomes larger. This is because an increase in sample size increases the number of nodes within the cluster centre threshold while the number of cluster centres remains the same, so the density value increases. The distance of the vertical axis from the point of local density maximum, on the other hand, does not change much, which indicates that the increase in the number of samples has less effect on δ, mainly because the number of cluster centre points does not change.

Parallel NPADPC clustering was performed for four datasets with different sample sizes and the statistical results are shown in Table 4.

It can be seen that for all datasets the average accuracy of NPADPC ranges from 0.9287 to 0.9458 and is generally higher than the mean value of DPC (from 0.8927 to 0.9098), which indicates that NPADPC also outperforms DPC in the average case. The minimum value of accuracy for NPADPC (0.9199) is higher than the DPC accuracy minimum (0.8839), which indicates that the performance of NPADPC in the worst case

Table 4. Clustering accuracy of parallel NPADPC and DPC

Data set	Sample size	NPADPC			DPC		
		Min.	Max.	Ave.	Min.	Max.	Ave.
Dataset 1	2000	0.9199	0.9322	0.9287	0.8839	0.8962	0.8927
Dataset 2	3000	0.9236	0.9417	0.9372	0.8876	0.9057	0.9012
Dataset 3	4000	0.9307	0.9443	0.9419	0.8947	0.9083	0.9059
Dataset 4	5000	0.9412	0.9534	0.9458	0.9052	0.9174	0.9098
Dataset 5	6000	0.9409	0.9533	0.9455	0.9049	0.9173	0.9095

is also better than the performance of DPC in the worst case. When the sample size is boosted from 2000 to 5000, the average clustering accuracy improves by 1.93%. This may be due to the fact that the increase in sample size enables more comprehensive access to the density values of the nodes when performing the normalised nearest neighbour density solution, which prompts the DPC to obtain better clustering results, thus improving the accuracy of big data clustering, and also suggests that parallel NPADPC clustering is particularly suitable for big data clustering.

5.3. **Running time comparison.** The experimental results of the performance comparison between the parallel NPADPC algorithm and the DPC algorithm executed on the above four test sets are shown in Figure 5, with the time unit of min.

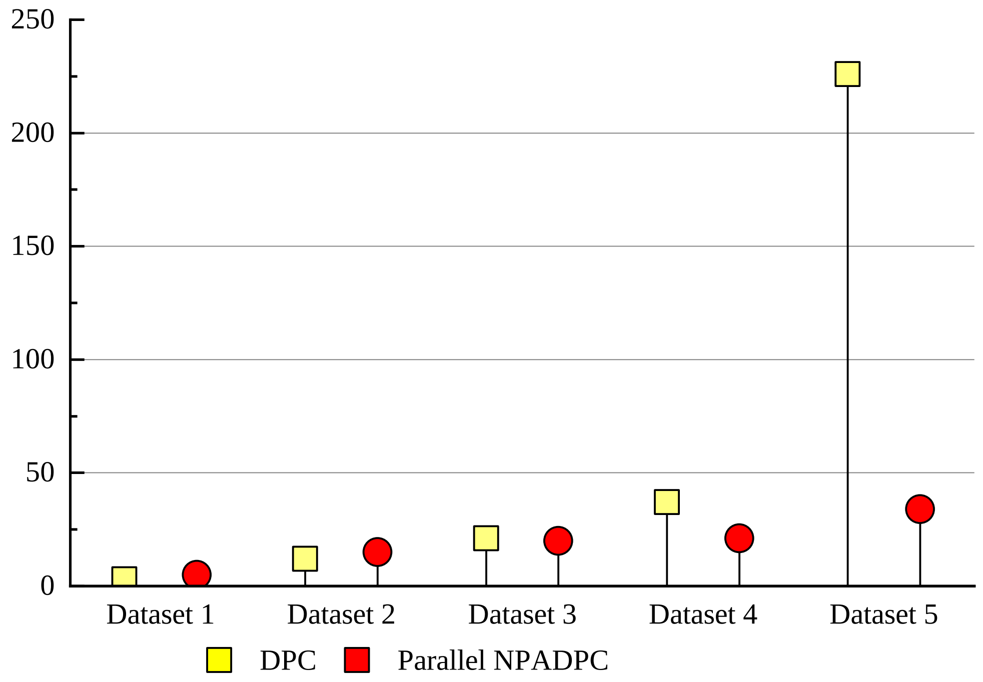

Figure 5. Comparison graph of running time

It can be seen that the DPC algorithm is slightly faster than the parallel NPADPC algorithm when the amount of data is small. However, as the number of data samples increases, the disadvantage of its computational efficiency slowly manifests itself. At Dataset 5, its computation time is much higher than that of the parallel NPADPC algorithm. When the amount of computation is small, the parallel NPADPC algorithm has a larger running expense when the number of data samples is small because it needs to partition the data, thus incurring some extra communication time. However, as the data samples rise, the advantages of the parallel NPADPC algorithm come out clearly.

6. **Conclusion.** In this work, an automatic determination of centre DPC (NPADPC) based on normalisation process is proposed. Firstly, an automatic determination of cluster centres based on normalisation processing is provided to address the problem of traditional DPC methods that need to rely on experience to select cluster centres. In addition, a strategy for dynamically adjusting the truncation distance is developed so that it can be automatically adjusted according to the local density distribution of the dataset. Secondly, a parallel NPADPC algorithm based on Spark is proposed to address the problem that the proposed NPADPC algorithm has high computational complexity and low running efficiency in the Spark teaching platform. The calculation of ρ (local density) and δ (distance) is completed in each interval independently distributed, after which each node is clustered using the NPADPC algorithm, and finally the results after clustering in each interval are re-clustered. Experiments show that as the data sample rises, the efficiency advantage of the parallel NPADPC algorithm comes out clearly. When the sample size is raised from 2000 to 5000, the average clustering accuracy is increased by 1.93%.

REFERENCES

[1] C. Romero and S. Ventura, "Data mining in education," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 12–27, 2013.

[2] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[3] R. Baker, "Data mining for education," *International Encyclopedia of Education*, vol. 7, no. 3, pp. 112–118, 2010.

[4] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2013.

[5] K. R. Koedinger, S. D'Mello, E. A. McLaughlin, Z. A. Pardos, and C. P. Rosé, "Data mining and education," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 6, no. 4, pp. 333–353, 2015.

[6] F. Coenen, "Data mining: past, present and future," *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 25–29, 2011.

[7] K. K. Hirji, "Exploring data mining implementation," *Communications of the ACM*, vol. 44, no. 7, pp. 87–93, 2001.

[8] R. Mikut and M. Reischl, "Data mining tools," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 5, pp. 431–443, 2011.

[9] S.-H. Liao, P.-H. Chu, and P.-Y. Hsiao, "Data mining techniques and applications–A decade review from 2000 to 2011," *Expert Systems with Applications*, vol. 39, no. 12, pp. 11303–11311, 2012.

[10] L.-D. Chen, T. Sakaguchi, and M.-N. Frolick, "Data mining methods, applications, and tools," *Information Systems Management*, vol. 17, no. 1, pp. 67–68, 2000.

[11] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 19–46, 2024.

[12] T.-Y. Wu, J. C.-W. Lin, U. Yun, C.-H. Chen, G. Srivastava, and X. Lv, "An efficient algorithm for fuzzy frequent itemset mining," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 5787–5797, 2020.

[13] T.-Y. Wu, J. Lin, Y. Zhang, and C.-H. Chen, "A Grid-Based Swarm Intelligence Algorithm for Privacy-Preserving Data Mining," *Applied Sciences*, vol. 9, no. 4, 774, 2019.

[14] P. Van Kerm, "Adaptive kernel density estimation," *The Stata Journal*, vol. 3, no. 2, pp. 148–156, 2003.

[15] M. J. Li, M. K. Ng, Y.-M. Cheung, and J. Z. Huang, "Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 11, pp. 1519–1534, 2008.

[16] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2007.

[17] O. J. Hulme, L. Whiteley, and S. Shipp, "Spatially distributed encoding of covert attentional shifts in human thalamus," *Journal of Neurophysiology*, vol. 104, no. 6, pp. 3644–3656, 2010.

[18] B. N. Li, C. K. Chui, S. Chang, and S. H. Ong, “Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation,” *Computers in Biology and Medicine*, vol. 41, no. 1, pp. 1–10, 2011.

[19] H. He and Y. Tan, “A two-stage genetic algorithm for automatic clustering,” *Neurocomputing*, vol. 81, pp. 49–59, 2012.

[20] S. A. Seyedi, A. Lotfi, P. Moradi, and N. N. Qader, “Dynamic graph-based label propagation for density peaks clustering,” *Expert Systems with Applications*, vol. 115, pp. 314–328, 2019.

[21] R. Mehmood, G. Zhang, R. Bie, H. Dawood, and H. Ahmad, “Clustering by fast search and find of density peaks via heat diffusion,” *Neurocomputing*, vol. 208, pp. 210–217, 2016.

[22] H. Cui, Y. Tan, R. Bai, Y. Li, L. Zhao, X. Zhuang, Y. Wang, Z. Chen, P. Li, and X. You, “Effect of melt superheat treatment on solidification behavior and microstructure of new Ni–Co based superalloy,” *Journal of Materials Research and Technology*, vol. 15, pp. 4970–4980, 2021.

[23] M. Nutz and J. Wiesel, “Entropic optimal transport: Convergence of potentials,” *Probability Theory and Related Fields*, vol. 184, no. 1, pp. 401–424, 2022.

[24] H. Yu, L. Chen, and J. Yao, “A three-way density peak clustering method based on evidence theory,” *Knowledge-Based Systems*, vol. 211, 106532, 2021.

[25] Y. Chen, X. Hu, W. Fan, L. Shen, Z. Zhang, X. Liu, J. Du, H. Li, Y. Chen, and H. Li, “Fast density peak clustering for large scale data based on kNN,” *Knowledge-Based Systems*, vol. 187, 104824, 2020.

[26] P. Bhattacharjee and P. Mitra, “A survey of density based clustering algorithms,” *Frontiers of Computer Science*, vol. 15, pp. 1–27, 2021.

[27] M. Sun, L. Xu, R. Luo, Y. Lu, and W. Jia, “Fast location and recognition of green apple based on RGB-D image,” *Frontiers in Plant Science*, vol. 13, 864458, 2022.

[28] S. Zhang, “Challenges in KNN classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4663–4675, 2021.

[29] S. Zhang and J. Li, “KNN classification with one-step computation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2711–2723, 2021.

[30] S. Zhang, “Cost-sensitive KNN classification,” *Neurocomputing*, vol. 391, pp. 234–242, 2020.

[31] V. Kocaman and D. Talby, “Spark NLP: natural language understanding at scale,” *Software Impacts*, vol. 8, 100058, 2021.

[32] N. Ahmed, A. L. Barczak, T. Susnjak, and M. A. Rashid, “A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench,” *Journal of Big Data*, vol. 7, no. 1, 110, 2020.