Journal of Computers Vol. 33 No. 1, February 2022, pp. 101-114
doi: 10.53106/199115992022023301010

MR-DBIFOA: a parallel Density-based Clustering Algorithm by Using Improve Fruit Fly Optimization

Wei Liu[1], Jiaxin Wang[2], Xiaopan Su[3], Yimin Mao[2*]

[1] Gannan University of Science and Technology, GanZhou 341000, China
liusu8152203@sina.com
[2] Institute of information engineering, Jiangxi university of science and technology, GanZhou 341000, China
wjxbic@163.com, mymlyc@163.com
[3] Ganzhou851, GanZhou 341000, China

Received 1 April 2021; Revised 17 August 2021; Accepted 8 September 2021

Abstract. Clustering is an important technique for data analysis and knowledge discovery. In the context of big data, the density-based clustering algorithm faces three challenging problems: unreasonable division of data gridding, poor parameter optimization ability and low efficiency of parallelization. In this study, a density-based clustering algorithm by using improve fruit fly optimization based on MapReduce (MR-DBIFOA) is proposed to tackle these three problems. Firstly, based on KD-Tree, a division strategy (KDG) is proposed to divide the cell of grid adaptively. Secondly, an improve fruit fly optimization algorithm (IFOA) which use the step strategy based on knowledge learn (KLSS) and the clustering criterion function (CFF) is designed. In addition, based on IFOA algorithm, the optimal parameters of local clustering are dynamically selected, which can improve the clustering effect of local clustering. Meanwhile, in order to improve the parallel efficiency, the density-based clustering algorithm using IFOA (MR-QRMEC) are proposed to parallel compute the local clusters of clustering algorithm. Finally, based on QR-Tree and MapReduce, a cluster merging algorithm (MR-QRMEC) is proposed to get the result of clustering algorithm more quickly, which improve the core clusters merging efficiency of density-based clustering algorithm. The experimental results show that the MR-DBIFOA algorithm has better clustering results and performs better parallelization in big data.

Keywords: density-based clustering algorithm, KD-Tree, MR-DBIFOA, fruit fly optimization

1. Introduction

The fast development of information technology leads to the generation of massive amounts of data from several emerging applications. For example, according to the September 2017 statistics, the customers of Wal-Mart provided approximately 2.5 petabytes of data per hour. These huge and unbounded data are referred to as big data. Big data has four special characteristics (i.e., volume, velocity, value, and variety) that bring many challenges in formulating data mining algorithms for extracting hidden knowledge. In order to discover knowledge from big data sets, data mining techniques are proposed [1], which involves various stages such as preprocessing of data and exploring interesting patterns in the data [2]. These data mining techniques are also known as Knowledge Discovery in Database. The common ways used in mining data includes clustering [3], regression, classification, and association rule mining algorithms, among which, density-based clustering algorithm [4] as a branch of clustering is widely used in the geography, medicine, finance, and image analysis fields [5] because of its capability to discover clusters_of arbitrary shapes in dense areas and handle noise or outliers effectively. Unfortunately, the huge volume of data generated in the traditional density-based clustering algorithms, can be too massive for a single machine to handle in a reasonable amount of time. Moreover, the data itself that may be too large to be stored on a single computer.

For reducing the computational complexity, the engineers at Google created the MapReduce software framework [6], a programming abstraction which automatically handles running failures, hides the underlying complexity and makes it easier to develop program [7]. Because of the characteristics of the MapReduce framework mentioned above, many algorithms based on the MapReduce framework have been proposed. Li et al proposed a hierarchical-based DBSCAN algorithm [8] (named HDBSCAN) by improving the existing density-based clustering algorithm DBSCAN. However, the algorithm does not mention a specific data division method.

Comparing the HDBSCAN, an efficient and distributed DBSCAN algorithm using MapReduce MR-DBSCAN is designed by He [9], which splits the data by dividing the grid to obtain clustering results, but the algorithm needs a lot of time to obtain the global clusters. For reducing the cost of obtaining global clusters, Wang et al [10] proposed an algorithm named IP-DBSCAN which adopts R*-tree index structure to improve the time efficiency

* Corresponding Author

of clustering. Meanwhile, Cheng et al presented the DBSCAN-PSM algorithm [11] combined with the KD-tree structure and the neighbor query method to reduce the number of access to the data set. Although the IP-DBSCAN algorithm and the DBSCAN-PSM algorithm improve the efficiency, the algorithm is still sensitive to the selection of DBSCAN parameters. Therefore, how to improve the optimization ability of clustering parameters has become an urgent concern of the parallel density-based clustering algorithm [12].

Currently, the swarm intelligence optimization algorithm [13] is widely used in the parameter optimization problem of clustering algorithm, because it has the advantages of good global optimization effect [14] and easy realization [15]. Yu Qianqian [16] has discussed a Parallel density clustering algorithm to get the optimal initial clustering parameters by using particle swarm optimization algorithm. Hu et al. proposed a parallel DBSCAN [17] based on genetic algorithm, which uses genetic algorithm to calculate the optimal value of neighborhood and .Wang Jun developed a parameter adaptive density clustering algorithm based on MapReduce [18] that combines the PSO method [19] to optimize the parameters of the DBSCAN algorithm in local clustering. Deng [20] proposed a parallel density clustering algorithm DPDPSO based on particle swarm optimization [21] and k-dist graph, which uses particle swarm optimization algorithm to obtain the best initial clustering center for partitioning, and uses k-dist graph to find local clustering parameters after partitioning. However, these algorithms also have a limitation that it is easy to fall into local optimization in the process of parameter optimization, so the parameter optimization ability of the algorithm needs to be further improved. To assist readers, the state of this study with their techniques, advantages, and limitations is listed in Table 1.

Table 1. Literature merits and demerits

Literature	merits	demerits
HDBSCAN	Split data to execute DBSCAN algorithm in parallel	Does not mention a specific data division method
MR-DBSCAN	Splits the data by dividing the grid to obtain clustering results	Needs a lot of time to obtain the global clusters
IP-DBSCAN	Adopts R*-tree index structure to improve the time efficiency of clustering	Sensitive to the selection of DBSCAN parameters
DBSCAN-PSM	Combined with the KD-tree structure and the neighbor query method to reduce the number of visits to the data set	Very sensitive to the threshold of the grid side length
PSO-DBSCAN	Combines the PSO method to optimize the parameters	Easy to fall into local optimization in the process of parameter optimization
DPDPSO	Uses particle swarm optimization and k-dist graph to obtain the best initial clustering center for partitioning	The parameter optimization ability of the algorithm is inefficient

In view of this, we present the design and implementation of the density-based clustering algorithm by using improve fruit fly optimization based on MapReduce (MR-DBIFOA). The contributions of this study are as follows:

•We propose a strategy named KDG (Divide Gird based on KD-tree) to divide the cell of the grid adaptively.

•We design a IFOA (Improve Fruit fly Optimization Algorithm) which uses the KLSS (Step Strategy based on Knowledge Learn) and CCF (Clustering Criterion Function) to adjust the optimal parameters in local clustering dynamically. Meanwhile, in order to improve the parallel efficiency, the DBIFOA (Density-Based clustering algorithm using IFOA) are proposed to parallel compute the local clusters of clustering algorithm.

•We present a MR-QRMC (Clusters Merging algorithm based on QR-Tree and MapReduce) to accelerate the convergence speed of parallel merged local clusters.

The remained of this paper is organized as follows: Section 2 presents background on KD-tree, fruit fly optimization algorithm, module similarity and QR-tree. Section 3 describes the specific design and the implementation of MR-DBIFOA. Section 4 shows the proposed MR-DBIFOA algorithm against that of the MR-DBSCAN, the IP-DBSCAN, the DBSCAN-PSM, and the DPDPSO algorithms. Finally, section 5 concludes the paper. It can be analyzed from the experimental results that MR-DBIFOA has made certain improvements in the clustering effect, optimization ability and robustness. With a larger number of data sets, the MR-DBIFOA algorithm can complete the clustering process more efficiently, with the least time overhead and the best execution efficiency.

2 Preliminary

2.1 KD-tree

KD-tree is defined as a binary tree of k-dimensional vectors at the node points, which is a data structure for separating k-dimensional data space. It uses cost estimation to determine the split plane position of the current node until a certain termination condition is reached and the node becomes a leaf node. The specific construction process of KD-tree is as follows:

Step1. Generate split plane candidates at some locations;
Step2. Evaluate cost function using SAH approximation at each location;
Step3. Pick the optimal candidate (with lowest cost) and perform split into two child nodes;
Step4. Pass over geometry to distribute it among children;
Step5. Repeat recursively;

2.2 Fruit Fly Optimization Algorithm

The Fruit Fly Optimization Algorithm (FOA) is a new interactive evolutionary method inspired by the knowledge from foraging behavior of fruit flies. There are two main foraging processes: smell the food source by osphresis organ and fly towards the corresponding location; and use the sensitive vision to find food and fly towards a better direction. The procedure of the FOA is summarized as follows:

Step1. Initialize parameters, including maximum number of generations and population size;
Step2. Initialize the fruit fly group;
Step3. Use osphresis for foraging: generate several fruit flies randomly around the fruit fly group to construct a population;
Step4. Evaluate the population to obtain the smell concentration values (fitness value) of each fruit fly;
Step5. Use vision for the foraging: find the best fruit fly with the maximum smell concentration value, and then let the fruit fly group fly towards the best one;
Step6. End the algorithm if the maximum number of generations is reached; otherwise, goback to Step 3.

2.3 Module Similarity

Given two independent modules and , their module similarity can be calculated by the degree of interaction between nodes and belong to the two modules, which defined as follows:

$$Sim(I_i, I_j) = \frac{\sum_{x\in I_i, y\in I_j} c(x,y)}{max(|I_i|,|I_j|)} \quad . \qquad \textbf{(1)}$$

Subject to

$$c(x,y) = \begin{cases} 1 & if \quad x = y \\ w(x,y) & if \quad x \neq y, and < x,y > \in E \\ 0 & otherwise \end{cases} \quad . \qquad \textbf{(2)}$$

2.4 QR-tree

QR-tree is a data structure composed of a Quad-tree *Qt* with depth *d* and *n R*-trees, which defined as follows:

$$QRt = \begin{cases} Qt_n \cup Rt_n \\ n = \sum_{i=0}^{d-1} (2^k)^i \end{cases} \quad . \qquad \textbf{(3)}$$

Where *d* is the depth of the Quad-tree *Qt* and *k* is the number of spatial dimensions.

3 MR-DBIFOA Algorithm

In this section, we provide a detailed introduction to the MR-DBIFOA algorithm. Our algorithm has three steps: data partitioning, local cluster formation and global merging. Initially, on the basis of KD trees and greedy algorithms, we propose the KDG (divide gird based on KD-tree) strategy for adapting to partition grid cells. Secondly, the IFOA algorithm based on KLSS strategy and cluster determination function CCF is proposed to dynamically adjust the optimal parameters in local clustering; meanwhile, a DBIFOA algorithm is developed in combination with MapReduce model to accelerate the formation of local clusters. Finally, a QR-Tree based local cluster merging algorithm QRMEC, is presented to accelerate the convergence of merging local clusters, and combined with MapReduce, the MR-QRMEC algorithm is developed to achieve parallelized merging of local clusters and improve the parallel efficiency of the algorithm.

3.1 Partitioning Data

In the current parallel density clustering algorithm in the big data environment, there are two problems when dividing the data grid: 1) The selection of grid threshold is sensitive, and the selection of the threshold for data grid partitioning is critical to the clustering results; 2) The data grid density is uneven, and the existing partitioning strategy is not adequately adapted to the data distribution of big data. To address the above deficiencies, this paper proposed the KDG (divide gird based on KD tree and greedy algorithm) strategy to automatically partition the data. The description of the KDG strategy is as follows:

Step 1. Construct a KD tree structure to index the entire data space. For the data space, the root node points to the whole data space and stores the number of all data *N*. Each leaf node records each grid cell after partition and the number of points in the grid cell *n*.

Step 2. Select the partition dimension *d* and the partition node *p* of the KD-tree. In order to reasonably divide the data space of the KD-Tree index according to the degree of dispersion in the spatial data, we propose the KD-Tree partition function f_{KD} to calculate the partition dimension *d* and the partition node *p* of the KD tree, which definition is as follows:

Definition 1 (KD-Tree partition function f_{KD}) Denote the mean variance and mean of the data under i-th dimension in the spatial data as S_i and μ respectively. The number of data points under the division dimension *d* in the KD tree is num. The KD-Tree partition function f_{KD} is defined as follows:

$$f_{KD} = \begin{cases} d = \{max(s_i/\mu) \mid i = 1,2,\ldots,k\} \\ P = \begin{cases} \frac{p_{num/2}+p_{num/2+1}}{2}, if \ \ num\ is\ odd \\ \frac{p_{num/2}}{2} \qquad\qquad\quad , else \end{cases} \end{cases} \cdot \tag{4}$$

Proof. (1) Known that $s_i = \sqrt{S_i{}^2} = \sqrt{\sum_{i=1}^{n}(n_i - \mu)^2}$, when the distribution of data in dimension *d* tends to be centralized, S_i tends to 0, so s_i/μ tends to 0. Therefore, the smaller the value of $s_i/\ \mu$, the more centralized the data distribution; the higher the value of s_i/μvalues, the more discrete the data distribution. (2) Denote the mean variance of d_1 and d_2 as S_1 and S_2, the mean of d_1 and d_2 as μ_1 and μ_2. When S_1 is equal to S_2, μ_1 tends to 0 and μ_2 tends to infinity, we have that S_1 tends to $\sqrt{\sum_{i=1}^{n}(n_i)^2}$ and S_2 tends to $\sqrt{\sum_{i=1}^{n}(n_i - \infty)^2}$. When the data n_i changes, it is obvious that $\Delta S_1 >> \Delta S_2$. Therefore, the larger the value of μ, the smaller the value of s_i/μ the disturbance of the data has little impact on the data, and the data distribution tends to be more stable; the larger the value of s_i/μ, the more unstable the data distribution. In summary, the size of s_i/μ can be used to measure the distribution of data in a certain dimension in space and the degree of change when disturbed. Therefore, the KD tree partition function f_{KD} can reasonably divide the data space of the KD-Tree index according to the degree of dispersion in the spatial data and is a criterion function for judging the degree of data dispersion.

- Divide the current data space using the KD tree from top to bottom. Specifically, if the data is smaller than the segmentation node *p* in the current segmentation dimension *d*, add the data to the left sub-tree grid set of the KD tree; otherwise, join the right sub-tree grid set of the KD tree.
- The condition of dividing the grid. If the number of data points *n* in the current grid is not greater than the maximum threshold *maxN* which is equal to the ratio of the number of all points *n* in the data space to the number of Map nodes, stop dividing the grid; otherwise, go back to the previous step.
- Reorganization of data grid partition. Combined with the greedy algorithm, the entire data grid partition is reorganized, and finally the data grid partitions with similar sample points are obtained, which would be used as the number of Map nodes.

3.2 Forming Local Clusters

In this stage, we briefly describe the main idea and steps of IFOA, which is presented to dynamically select the optimal parameters of local clustering to improve the clustering effect of local clustering. The objective of local cluster formation is to realize parallel computing of local clusters. The process proposes the IFOA algorithm based on KLSS and CCF strategy to improve the ability of optimizing parameters in density clustering. Then, combined with the MapReduce, the DBIFOA algorithm is presented to compute the local clusters parallelly.

3.2.1 KLSS Strategy and CCF Function

To solve the poor parameter optimization ability and easy to fall into local optimality problems of the parallel density clustering algorithm in the big data environment, we design the KLSS strategy and CCF function, and based on them, the IFOA algorithm is proposed to find the optimal parameters.

Definition 2 (the step strategy KLSS) Given that fruit fly individual *i* has a variation scale p_i and the initial coordinate position of the fruit fly individual is x_{ij}, the step strategy KLSS is as follows:

$$new\_x_{ij} = x_{ij} + (p_i + rand\ (\ )) \times r_{ij}\ . \qquad \textbf{(5)}$$

$$p_i = f\,(x_i - f_{max}) / \textstyle\sum_{i=1}^{n} (f(x_i) - f_{max})\ . \qquad \textbf{(6)}$$

Where *rand* () is a random number with normal distribution, r_{ij} is a random number on [0,1], $f\,(x_i - f_{max})$ is the degree of deviation between the individual fruit fly and the optimal fruit fly position, $\sum_{i=1}^{n} f\,(x_i - f_{max})$ is the total deviation distance of all fruit flies.

Proof. Given $f\,(x_i\text{-} f_{max})$ is the degree of deviation of the individual fruit fly *i* from the optimal fruit fly position, $\sum_{i=1}^{n} f\,(x_i\text{-} f_{max})$ is the total deviation distance of all fruit flies. When the fruit fly *i* from the optimal fruit fly position is greater, the sum of the other fruit flies is smaller, the p_i will increase at this time. Then, the degree of perturbation of the position of fruit fly *i* by formula (6) also increases, so the fruit *i* flycan improve the search ability and avoid falling into the local optimal solution.

Then, in order to further optimize the optimization process of the algorithm, the clustering decision function *CCF* is proposed, which can be introduced as follows:

Definition 3 (clustering decision function *CCF*) Given the number of clusters is *c*, the information entropy of cluster *i* is *H*(*i*), the probability of the existence of cluster *i* is $P_i = n_i/n$, the number of nodes in cluster *i* is n_i, and the total number of nodes is *n*, the *CCF* is defined as follows:

$$CCF = min(WIE - \textstyle\sum_{i=1}^{c}\sum_{j=1}^{c} Sim(I_i, I_j))\ . \qquad \textbf{(7)}$$

$$WIE = \tfrac{1}{c}\textstyle\sum_{i=1}^{c} H(i) = -\tfrac{1}{c}\sum_{i=1}^{c} p(i) \times lbp(i)\ . \qquad \textbf{(8)}$$

Where *Sim* (I_i, I_j) is the similarity between classes, *WIE* is the weighted information entropy.

Proof. (1) Known $lb(P_i)$ is a monotonically increasing function, we can conclude $P_i \times lb(P_i)$ is a monotonically increasing function, so *WIE* is a monotonically decreasing function. And known $P_i = n_i / n$, when the number of nodes belonging to cluster *i* increases, P_i increases and *WIE* decreases. Therefore, the higher the stability of the cluster, the smaller the weighted information entropy. (2) According to formula (1) we can deduce that: when the similarity of the nodes *x* and *y* between two independent modules I_i and I_i is smaller, *c*(*x*,*y*) will decrease. Therefore, when the similarity between two modules I_i and I_j is lower, the $\sum_{i=1}^{c}\sum_{i=1}^{c} Sim(I_i, I_i)$ is smaller. In conclusion, the function *CCF* is a cluster fitness function that can determine the effect of density clustering.

3.2.2 IFOA Algorithm

Based on KLSS strategy and CCF function, we introduce the IFOA algorithm which is used to find the optimal parameters of clustering in different data partitions. The specific steps are as follows:

- **Setting parameters.** Set the value range of the parameters ε and *Minpts*, and initialize *k* fruit flies within the solution range. The value of parameter ε or *Minpts* represents the position of each fruit fly. The individual odor concentration value of each fruit fly at the current position is used as the fitness function value of the clustering result in the density clustering algorithm.
- **Searching food.** Calculate the odor concentration value according to the position of fruit fly and fitness function. Save the odor concentration value and the coordinate of the fruit fly with the best odor concentration. Then, other fruit flies began to update the current location to find food, and recalculate the odor concentration value by adopting the KLSS strategy. If the new solution is better than the old one, update the position from the best fruit fly; otherwise, if the number of solutions is reached the maximum number of iterations, the algorithm stops.
- **Returning parameters.** Return the location of the fruit fly with the best odor concentration as the optimal parameter of the density clustering algorithm in the data partition.

The density clustering algorithm and IFOA algorithm are implemented as follows (Table 2):

Table 2. The relation between IFOA and density-based clustering algorithm

IFOA	clustering
The position of the fruit fly moves	Locally find the optimal value of ε and *Minpts*
The location of the fruit fly	$x = \varepsilon$, *Minpts*
Objective function	*CCF*
Smell concentration of fruit fly	Evaluation value of clustering result
The location of the optimal smell concentration of fruit flies	Density clustering optimal clustering result

3.2.3 Local Clustering

After finding the optimal clustering parameters, the DBIFOA algorithm is proposed to implement the parallel computation of local clusters and thus complete the entire local clustering process. The algorithm includes two sub-algorithms, namely the parameter parallel optimization and the local cluster parallel computing.

In the parameter parallel optimization sub-algorithm, first, the data partition object *G* and the point set p_i in the partition are used as input; Secondly, call the map function in each node, calculate the clustering parameters ε and *Minpts* of the point set in each partition according to the fruit fly optimization algorithm, and output the values of key-value: $<g, \varepsilon>$ and $<g, minpts>$. Then use the reduce function to merge the key-value values, and finally output $<g, (\varepsilon, minpts)>$ to the next sub-algorithm. As shown in Fig. 1 (a).

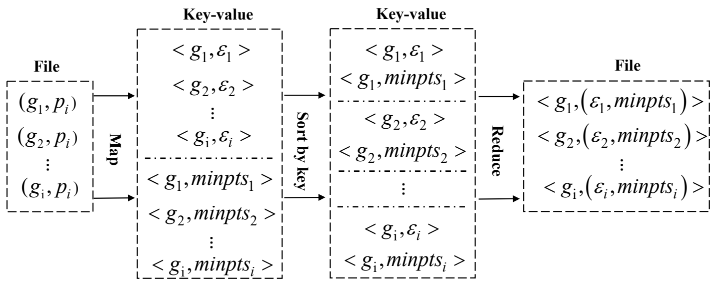
In the local cluster parallel computing sub-algorithm, the execution process is as follows:

Step 1. take the data partition object *G*, the point set p_i in the partition, and the key-value value <g, (ε, *minpts*) > calculated by the previous sub-algorithm as input.

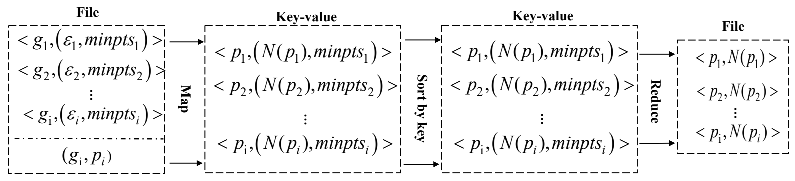
Step 2. Call the map function to calculate the point set $N(p_i)$ of the point p_i in the ε neighborhood, and output the key-value value $<p_i, N(p_i)>$.

Step 3. Execute the reduce function and calculate whether the current data object p_i is the core object according to the density clustering algorithm.

Step 4. If $pi \geq minpts$, then the current gird object p_i is the core gird, and the key-value value $<pi, N(p_i)>$ is output; If $pi < minpts$, no result is output. After the execution, the local cluster of the clustering algorithm is finally formed. As shown in Fig. 1(b).



(a) parameter parallel optimization sub-algorithm


File
Key-value
Key-value
File
Map
Sort by key
Reduce
$< g_1, (\varepsilon_1, minpts_1) >$
$< g_2, (\varepsilon_2, minpts_2) >$
$< g_i, (\varepsilon_i, minpts_i) >$
(g_i, p_i)
$< p_1, (N(p_1), minpts_1) >$
$< p_2, (N(p_2), minpts_2) >$
$< p_i, (N(p_i), minpts_i) >$
$< p_1, N(p_1) >$
$< p_2, N(p_2) >$
$< p_i, N(p_i) >$

(b) local cluster parallel computing sub-algorithm

Fig. 1. The process of DBIFOA algorithm

3.3 Merging Local Clusters

In the previous stage, the IFOA strategy was used to realize the parallel calculation of local clusters to complete the entire local clustering process. At this stage, for improving the efficiency of merging local clusters, we propose the MR-QRMC algorithm which is divided into two steps: firstly, a local cluster merging algorithm QRMC is designed to improve the local cluster merging efficiency; secondly, the MR-QRMC algorithm is presented based on MapReduce, which improves the parallel efficiency of the entire clustering process.

3.3.1 QEMC Algorithm

The QEMC algorithm uses the *unchecked*, *uncore*, *core*, and *outlier* to mark the changes in the data state. The initial state of all data points is *unchecked*, and the input table R composed of local cluster objects, each of which consists of the core object p_i and $N(p_i)$ the point setin theneighborhood. Then, the specific process of the QEMC algorithm can be described as follows:

- Obtain the $<p_i, N(p_i)>$ value of the core objectin sequence, and construct the QR-Tree to each local cluster to speed up the indexing of the data. Then, change the state of the core object p from *unchecked* to *core*.
- Check the state of the data object $N(p)$ in the neighborhood of the core object p in turn:

(1) If the state of a data object pi in $N(p)$ is *unchecked*, it is added to the local cluster with p as the core, and the state of p_i is changed to *uncore*.

(2) If the state of a data object p_i in $N(p)$ is *uncore*, which means the current data object p_i has been allocated to other clusters, so the state of the data object p_i unchanged.

(3) If the state of a data object p_i in $N(p)$ is *core*, the local cluster with p_i as the core will be merged into the local cluster of p.

Finally, the clustering global cluster is formed, and the state of the data object which still is *unchecked* will be changed to *outlier*.

3.3.2 MR-QEMC Algorithm

In order to solve the problem that the current parallel density clustering algorithm does not adopt parallelization when merging local clusters, and further improve the efficiency of obtaining global clusters. Based on the QEMC algorithm, the MR-QEMC algorithm is proposed which takes the data object set, the data setand the tablethat composed of the local clusters output by the DBIFOA algorithm as input. The specific process of the MR-QEMC algorithm can be described as follows:

- According to the number P of parallel nodes in the MapReduce frame, the data set P and table R will be divided into $P_1, P_2, ..., P_K$ and $R_1, R_2, ..., R_K$.

- Run the map function in each node to obtain the key-value value of the core object $<p_i, N(p_i)>$ in turn, and index in $P_1, P_2, ..., P_K$ according to the key value P to get the corresponding k value. Then, add the core gird object to the corresponding R_k, and output the key-value value $<M_i, (p_i, N(p_i))>$ and pass it to the Reduce function.
- Execute the Reduce function, distributed operation of QMEC algorithm for each M_i , repeat the loop and finally obtain the clustering result.

The MR-QMEC algorithm is implemented as follows:

Algorithm 1. MR-QMEC algorithm

```
Input: Data object setafter data partition, tablecomposed of local clusters, data set
Output: Clustering global clusters
Function MECORE-MR($P$,$R$,$D$)
Begin
1.  $k$=Count(Machine);
2. $P_1, P_2, ..., P_k$=Partial($P$,$k$);
2. $Result$ =RunMapReduce($P,R,D,P_1,P_2,...,P_k$) {
3.   For each $< p_i, N(p_i)> \in R$ Do {
4.     MapReduce.Map($P, P_1, P_2, ..., P_k$,){
5.     $i$ = Partial_Id($P, P_1, P_2, ..., P_k$);
6.      $R_1, R_2, ..., R_k$ =Partial($R$,$i$);
7.     $M_i = R_i$;
8.      emit($<Mi, (p_i, N(p_i))>$);
9.     }end map
10.   }end for
11.   For each  $M_i \in < M_i, (p_i, N(p_i))>$ do {
12.     MapReduce.Reduce($P_i$,($p_i$, $N(p_i)$)){
13.      $M_i$ = MECORE($P_i$,($p_i$, $N(p_i)$));
14.     }end reduce
15.     $M_i$ = MECORE($M_i$,$M_{i+1}$);
16.   }end for
17.   $Result$ =Point($M$,$<p, p_i>$);
18.   Return($Result$);
19. }end RunMapReduce
End
```

3.4 MR-DBIFOA Algorithm

The MR-DBIFOA algorithm uses the KDG strategy to automatically divide the data into data grid partitions with relatively consistent density, while ensuring that the number of data partitions is the same as the number of nodes in the map phase. Reasonable data division also helps the algorithm to balance the load of each computing node. The algorithm reduces the degree of fluctuation in the process of clustering. At the same time, by using the DBIFOA algorithm, the density clustering parameters can be dynamically optimized in the process of local clustering, and the search strategy KLSS and the clustering criterion function are adopted to avoid falling into the local optimization, thus further improving the clustering effect.

But when the amount of data is small, the data scale is much smaller than the data scale required by the cluster, and when the data is distributed to the computing nodes, the MapReduce architecture starts the cluster. It takes a certain amount of time to allocate tasks and store data. In the case of a small amount of data, it does not obviously contribute to the calculation speed of the algorithm, but it increases the calculation time of the algorithm, so the algorithm has almost no parallel performance at this time. The MR-DBIFOA algorithm is implemented as follows:

Algorithm 2. MR-DBIFOA algorithm

```
Input: Point set $D$ of data space, dimension $d$ of data space
Output: Global cluster $Result$
Initialization parameters: Number of Map nodes, maximum number of iterations, individual fruit fly $k$
Begin
1. For each $p_i \in D$ Do {
2.   Call the KDG strategy to divide the data and get the grid data partition set $G$ with the same number of Map nodes
3. }end for
4. For each $g_i \in G$ and $p_i \in D$ Do {
5.   Execute local cluster algorithm DBIFOA, output key-value value $<p_i, N(p_i)>$
6.   The key-value value $<p_i, N(p_i)>$sequence forms a local cluster table
7.   Return($R$)
8. }end for
9. Call the parallelized merge local cluster algorithm MR-QRMEC ($P$,$R$,$D$) to get the clustered global cluster
10. Return($Result$)
    End
```

4. Evaluation

In this section, we describe various actual experiments and the performance evaluation of MR-DBIFOA.

4.1 Experiment Settings

We perform the experiments in a Hadoop 3.2 cluster of four nodes (1 Master node, 3 Slaver nodes), where each node contains an Intel(R)Core(TM)i5-9400H CPU @ 2.9GHz processor, 32G memory. All the experiments are implemented in python 3.5.2 and Ubuntu 16.04. The Specific configuration of each node is shown in Table 3.

Table 3. The foundation configuration of each node in the experimental environment

Node type	Host name	IP	Role
Master	Master	192.168.1.109	Master/JobTracker/NameNode
Slaver	Slaver_1	192.168.1.110	Slaver/TaskTracker/DateNode
Slaver	Slaver_2	192.168.1.111	Slaver/TaskTracker/DateNode
Slaver	Slaver_3	192.168.1.112	Slaver/TaskTracker/DateNode

4.2 Experimental Data Set

Data sets on our experiments are from three real data sources:

- Iris is the most famous data set with 150 pieces of data in the area of pattern literature recognition.
- Impeel is a data set with 141712 pieces of data collected from the audit system of *ServiceNowTM* platform instances used by Internet company.
- Susy records the use of particle accelerators particles which consists of 5000000 records and each record includes 190 items.

The specific information of the data set is shown in Table 4:

Table 4. Data sets used in the experiments

	Iris	Impeel	Susy
Number of records	150	141712	5000000
Number of items	4	5	190
Data size	0.86	163	321

4.3 Evaluating Indicator

In order to evaluate the MR-DBIFOA algorithm, this paper uses the value of F-Measure, which is composed of the precision and recall, to evaluate the clustering results. The higher the value of F-Measure, meaning that the better the clustering result and the more reasonable the clustering algorithm. The F-Measure can be defined as follows:

$$F - measure = \frac{(\lambda^2+1)precision\times recall}{\lambda^2 precision+recall} \quad . \tag{9}$$

Where the value of the parameter λ is 1 as usual.

4.4 Performance of IFOA

In this stage, experiments were carried out to compare the accuracy and optimization ability between the IFOA algorithm and the standard FOA algorithm under four classical test functions. The test functions is shown in Table 5:

Table 5. The test functions

Function name	Test function	Search scope
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-50,50]
Schwefel's	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} x_i$	[-20,20]
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i)^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-500,500]
Rastrigin	$f_4(x) = \sum_{i=1}^{n}[x_i^2 - 10\,cos(2\pi_i) + 10]$	[-5.12,5.12]

Meanwhile, we set,during the experiments processing. Then, the FOA and IFOA algorithms will execute 60 times on the four functions independently, taking the average and variance of each function to evaluate the performance of the algorithms, the results is shown in Table 6:

Table 6. The validity analysis of algorithm

Function	Algorithm	Average value	Variance
$f_1(x) = \sum_{i=1}^{n} x_i^2$	FOA	1.498E-5	8.367E-7
	IFOA	2.164E-15	1.298E-12
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} x_i$	FOA	8.013E-11	2.763E-10
	IFOA	6.326E-15	1.216E-15
$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i)^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	FOA	0.018	0.022
	IFOA	2.581E-16	2.887E-16
$f_4(x) = \sum_{i=1}^{n}[x_i^2 - 10\,cos(2\pi_i) + 10]$	FOA	7.452	2.637
	IFOA	0.023E-16	0.436E-16

It can be seen from Table 6 that compared to the FOA algorithm, the average and variance of the IFOA algorithm running on the function are reduced by 1.868 and 0.665 respectively, indicating that the IFOA algorithm is obviously better than the FOA algorithm in convergence accuracy and stability. Because the IFOA algorithm adopts the KLSS strategy to mutate individual fruit fly with a certain probability and dynamically adjust the search space and the search step, which reduces the number if useless searches and strengthens the local search capabilities. Especially in the Rastrigin function, the accuracy of the IFOA algorithm is close to the theoretical optimal value of 0 which is sufficient to prove that IFOA has a strong ability to jump out of local extreme values and a good global search ability. Therefore, the experiment shows that the IFOA algorithm performs better than FOA algorithm in global optimization capability and stability.

4.5 Comparison of Clustering Results

In order to evaluate the clustering effect and performance of MR-DBIFOA, this paper uses Flame, Compound, Aggregation and D31 data to make a comprehensive comparison with MR-DBSCAN and PSO-DBSCAN algorithm, runs 50 times to get the clustering results, and takes the optimal value, accurate value, variance and time cost of the algorithm as the evaluation criteria. The Flame dataset has 240 data points, and its density cluster is composed of a concave dataset and a convex dataset; Compound is a dataset composed of 399 data points; Aggregation forms 7 different morphological non-Gaussian distribution density clusters; The D31data set contains 31 Gaussian data clusters. The optimization ability of the clustering algorithm can be evaluated by the optimal value, and the better the optimization ability of the algorithm to the local cluster is, the smaller the value is; the clustering effect can be evaluated directly by the accuracy; the stability of the algorithm can be evaluated by the variance. the better the stability of the algorithm, the smaller the value, and the time required to implement the clustering algorithm can be evaluated by time cost. The experimental results are shown in Table 7.

Table 7. Comparison of results in different clustering algorithms

Data set	Algorithm	Optimal value	Accuracy/%	Variance	Running time/s
	MR-DBSCAN	2.385E+7	88.2	1.793E-7	3.56
Flame	PSO-DBSCAN	1.945E+7	93.8	3.856E-9	4.68
	MR-DBIFOA	1.779E+7	96.7	4.478E-10	2.59
	MR-DBSCAN	6.685E+5	94.6	2.378E-7	5.73
Compound	PSO-DBSCAN	6.221E+5	96.1	1.198E-7	6.71
	MR-DBIFOA	6.083E+5	98.4	1.271E-8	5.75
	MR-DBSCAN	4.548E+6	92.8	2.954E-7	6.12
Aggregation	PSO-DBSCAN	3.759E+5	95.7	1.728E-8	6.47
	MR-DBIFOA	1.156E+5	98.5	0.834E-9	5.01
	MR-DBSCAN	3.548E+6	92.4	2.954E-7	6.34
D31	PSO-DBSCAN	2.359E+5	94.9	3.453E-8	6.98
	MR-DBIFOA	1.178E+6	96.8	2.651E-9	5.09

The data in Table 7 show that MR-DBSCAN algorithm performs worst in terms of clustering effect because its optimal value is the largest and its accuracy value is the lowest. Compared with the MR-DBSCAN algorithm, the accuracy of the PSO-DBSCAN algorithm is improved by 5.6% and 1.5% respectively in the Flame and Compound datasets. Because the PSO-DBSCAN algorithm uses the particle swarm optimization algorithm (PSO) to optimize the density clustering parameters in the process of local clustering, which solves the sensitive problem of selecting MR-DBSCAN parameters, so the clustering effect of PSO-DBSCAN has been improved to a certain extent. However, when dividing the data, it is not divided according to the distribution of the data under big data, so the stability of the PSO-DBSCAN algorithm is poor.

Compared with the above two algorithms, MR-DBIFOA algorithm has the lowest variance, indicating that it overcomes the shortcomings of the other two algorithms and performs better in terms of stability. In addition, in Flame data, compared with MR-DBSCAN and PSO-DBSCAN algorithm, the time overhead of MR-DBIFOA algorithm is reduced by 0.97s and 2.09s respectively. Because MR-DBIFOA algorithm uses local cluster parallel merging algorithm MR-QRMEC, to further improve the index speed and local cluster merging speed of local clusters on the basis of QR tree structure, the speed of MR-DBIFOA algorithm in clustering execution has been significantly improved.

4.6 Comparison of Running Time

The purpose of the experiment is to compare the time efficiency among MR-DBIFOA algorithm, MR-DBSCAN algorithm, IP-DBSCAN algorithm, DBSCAN-PSM algorithm and DPDPSO algorithm under same computing nodes and dataset. The running time of algorithms are shown in Fig. 2.

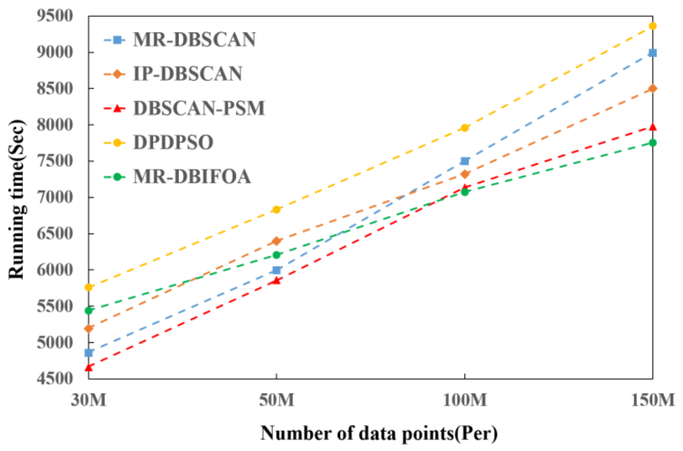


Figure 2. The running time of the algorithm in different data sets

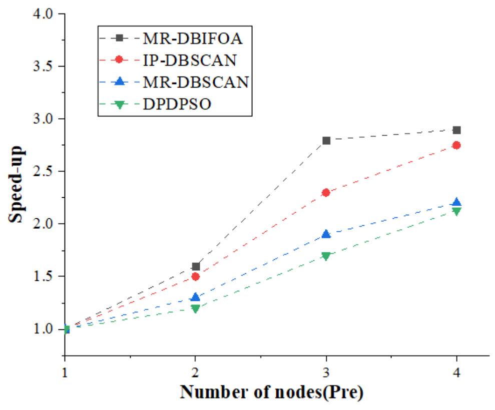
It can be seen from Fig. 2 that when the size of the data set is 30M, DBSCAN-PSM and MR-DBSCAN spend less time in the clustering process than IP-DBSCAN, DPDPSO and MR-DBIFOA algorithms. Because in small amount of data, the DPDPSO algorithm uses the particle swarm algorithm and k-dist graph to optimize the parameters, which increases the time cost of the algorithm; Similarly, the MR-DBIFOA algorithm adopts KDG and DBIFOA to adaptively segment the data space and optimization of density clustering parameters, increasing the time cost of the algorithm.

However, when the data size increases to 100M and 150M, compared with other algorithms, the running time of MR-DBIFOA algorithm is least. Because the MR-DBIFOA algorithm proposes a parallel local cluster merging algorithm, which uses QR-Tree index data to improve the indexing speed of the data, and the IFOA algorithm reduces the number of redundant iterations when optimizing the parameters.

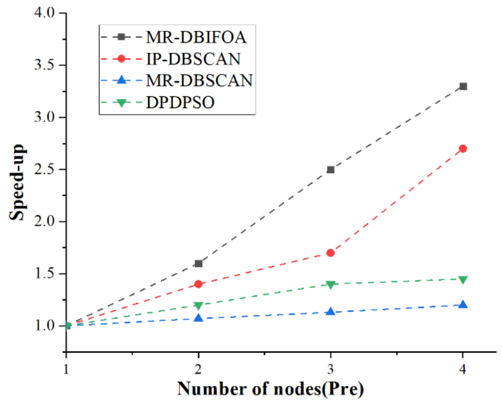
In general, the MR-DBIFOA algorithm can complete the clustering more efficiently under the large data sets.

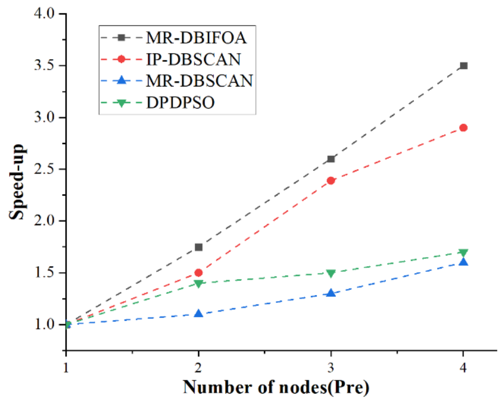
4.7 Comparison of Speed-up Ratio

We perform experiments based on the Iris, Impeel, Susy dataset to compare the MR-DBIFOA algorithm with the IP-DBSCAN, MR-DBSCAN and DPDPSO. The speed-up ratios are shown in the Fig. 3.



(a) Speed-up ratios of each algorithm on dataset Iris





(b) Speed-up ratios of each algorithm on dataset Impeel (c) Speed-up ratios of each algorithm on dataset Susy

Figure 3. Speed-up ratios for each algorithm on three datasets

From Fig. 3, it can be seen that the speed-up ratio of each algorithm follows an increasing trend with the number of nodes. When the dataset is small, such as (a), the speed-up ratio of each algorithm gradually tends to be stable and no longer grows with the increase of the number of nodes. This is due to the data scale is far from the data scale required by the cluster and the algorithm spend most of the time for the start-up, data distribution, task distribution, etc. of the parallel framework. However, as the amount of data increases, we can see that the speed-up ratio of MR-DBIFOA algorithm has been improved, for instance, in the Susy dataset, it is higher than the IP-DBSCAN, MR-DBSCAN and DPDPSO algorithms by 0.6, 2.1 and 1.85 when at four nodes. This is because with the increase of data, the ability of the algorithm to shorten the running time far exceeds the cluster overhead of the MapReduce framework, so the speed-up ratio and the parallel efficiency of the algorithm has been increased. Therefore, the MR-DBIFOA algorithm has the ability to deal with big data, and has better parallel performance and efficiency in the case of a large number of nodes.

5. Conclusions

In this paper, we propose the design and implementation of MR-DBIFOA, a density-based clustering algorithm by using improve fruit fly optimization based on MapReduce. The algorithm first proposes the KDG strategy to adaptively divide the data; Secondly, an improve fruit fly optimization algorithm IFOA based on the KLSS strategy and the CCF function is proposed to dynamically seek the optimal parameters in local clustering; Then, combined with the MapReduce model, a local clustering algorithm DBIFOA is presented to compute the local clusters in parallel; Finally, we design a MR-QMEC algorithm based on QR-Tree, which can realize the parallel merge of local clusters and improve the overall parallel efficiency of the algorithm. We prove our algorithm by both theoretical analysis and practical experiments. Experiments are conducted on four data sets, which contain up to 46 data clusters. Compared with MR-DBSCAN algorithm, IP-DBSCAN algorithm, DBSCAN-PSM algorithm and DPDPSO algorithm, we can directly conclude that our algorithm has obvious advantages in clustering results, speed-up ratio and running time.

However, there are also some spaces for improving the MR-DBIFOA algorithm performance, like considering a strategy to solve the problem of fruit fly optimization algorithm parameters initialization. Moreover, further studies, could be done by using other methods to improve the parallel performance of the algorithm.

Acknowledgement

This study was supposed by the National Natural Science Foundation of China (41562019, 41530640), the National Key Research Development Projects of China (2018YFC1504705), and Jiangxi Provincial Technology of Education Department (GJJ161566, GJJ181504).

References

[1] H.A. Madni, Z. Anwar, M.A. Shah, Data mining techniques and applications — A decade review, in: Proc. 2017 23rd International Conference on Automation and Computing (ICAC). IEEE, 2017.

[2] A. Fahad, N. Alshatri, Z. Tari, et al.,A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. IEEE Transactions on Emerging Topics in Computing. 2(3)(2014) 267-279.
[3] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Algorithms, and Applications, Society for Industrial and Applied Mathematics, American Statistical Association, 2007.
[4] P. Bhattacharjee, P. Mitra, A survey of density based clustering algorithms, Frontiers of Computer Science (print) 15(1)(2020) 151308.
[5] M. Khader, G. Al-Naymat, Density-based Algorithms for Big Data Clustering Using MapReduce Framework, ACM Computing Surveys (CSUR), 2020.
[6] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51(1)(2008) 107-113.
[7] N. Maleki, A.M. Rahmani, M. Conti, MapReduce: an infrastructure review and research insights, The Journal of Supercomputing 75(10)(2019) 6934-7002.
[8] L. Li, Y. Xi, Research on Clustering Algorithm and Its Parallelization Strategy, IEEE Computer Society (2011) 325-328.
[9] Y. He, H. Tan, W. Luo, H. Mao, M. Di, S. Feng, et al., MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce, in: Proc. IEEE International Conference on Parallel & Distributed Systems. IEEE, 2012.
[10]X. Wang, Y. Wu, X. Jiang, L. Liao, Incremental Parallelization of Fast Clustering Based on DBSCAN Algorithm under Large-scale Data Set, Computer Applications and Software, 2018.
[11]G. Chen, Y. Cheng, W. Jing, DBSCAN-PSM: an improvement method of DBSCAN algorithm on Spark, International Journal of High Performance Computing and Networking 13(4)(2019) 417.
[12]Z. Benmounah, S. Meshoul, M. Batouche, P. Lio', Parallel swarm intelligence strategies for large-scale clustering based on mapreduce with application to epigenetics of aging, Applied Soft Computing 69(2018) 771-783.
[13]V. Ravuri, S. Vasundra, Moth-flame optimization-bat optimization: map-reduce framework for big data clustering using the moth-flame bat optimization and sparse fuzzy c-means, Big Data 8(3)(2020).
[14]N. Al-Madi, I. Aljarah, S.A. Ludwig, Parallel particle swarm optimization clustering algorithm based on MapReduce methodology, Nature & Biologically Inspired Computing. IEEE, 2013.
[15]D. Hussain, D. Surendran, Content based image retrieval using bees algorithm and simulated annealing approach in medical big data applications, Multimedia Tools and Applications 79(9)(2020).
[16]Q. Yu, Research and parallelization of DBSCAN algorithm[D]. Wuxi: Jiangnan University, 2013.
[17]Y. Hu, Research on clustering algorithms of location big data based on MapReduce[D]. Hangzhou: Zhejiang University of Technology, 2019.
[18]J. Wang, Research on DBSCAN clustering algorithm oriented big date[D]. Zhengzhou: Jiangnan University, 2017.
[19]R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization. Swarm Intelligence 1(1)(2007).
[20]D. Deng, Application of DBSCAN algorithm in data sampling, Journal of Physics Conference Series 1617(2020) 012088.
[21]W. Cao, K. Liu, M. Wu, S. Xu, J. Zhao, An improved current control strategy based on particle swarm optimization (pso) and steady state error correction for sapf, IEEE Transactions on Industry Applications (2019) 4268-4274.