An Enhanced and Efficient Clustering Algorithm for Large Data Using MapReduce

Hongbiao Li, Ruiying Liu, Jingdong Wang and Qilong Wu

***Abstract*—With the rapid growth of data size, there are great challenges of clustering algorithms in terms of efficiency, reliability and scalability. Recently, many parallel algorithms using MapReduce framework have been proposed to address the scalability problem caused by the size of the data increases. When the massive data is clustered by KMeans algorithm in parallel, it will be read repeatedly in each iterative process, which increases both I/O and network costs significantly. In this paper, we propose a new sampling-based KMeans clustering algorithm, named SKMeans, which decreases the data size effectively, while improves the clustering accuracy by representative verification. Secondly, a parallelized SKMeans using MapReduce, named MR-SKMeans, is implemented on a Hadoop cluster and further explore the effect of MR-SKMeans. The empirical performance of MR-SKMeans is compared to parallel KMeans and other algorithms applying statistical sampling techniques. Our experimental results indicate that MR-SKMeans perform better in terms of efficiency, scalability and accuracy.**

***Index Terms*—sampling, representative verification, KMeans distributed computing**

I. Introduction

CLUSTERING is a fundamental problem and used in a variety of areas of computer science and related fields for data analysis [1], [2], such as pattern recognition, artificial intelligence, image segmentation, text analysis, etc. There is a growing tendency that clustering algorithm is expected to deal with large data. Google's MapReduce [3]–[5] or its open-source equivalent Hadoop [6] is a powerful tool for building such applications. MapReduce has become more and more popular for its simplicity, flexibility, fault-tolerance and scalability. It is appreciable that some researchers use MapReduce for large data clustering.

KMeans algorithm is a typical clustering algorithm based on partition, which can cluster the large data sets efficiently. It specifies an initial number of groups, and reallocates objects iteratively among groups to convergence. When the large-scale data is clustered by KMeans algorithm in parallel using MapReduce [7], [8], restarting jobs, inputting the initial data and shuffling between networks in the parallel framework have been caused by many iterations before convergence, which increase both I/O and network costs significantly and affect the clustering performance. In this paper, we initially present a new sampling-based KMeans clustering algorithm, named SKMeans, which can reduce the data size effectively, and improve the clustering accuracy with representative verification. Moreover, we develop MR-SKMeans, which is a parallelized SKMeans using MapReduce. This algorithm selected a sample set from the data set in parallel firstly. Then the sample set is used as input data for parallel KMeans. Representative verification algorithm has been put to use for each data point in order to assign the cluster center. The experimental results on various datasets show that MR-SKMeans is efficient and scalable. The traditional sampling algorithm has many shortcomings [9], [10], such as many points cannot be represented by sample set, and low-density clusters and noise points will be lost with high probability, etc. However, missing clusters can be avoided in the sampling by using MR-SKMeans, and the accuracy of clustering results can be improved when representative verification algorithm is utilized in the clustering process.

The rest of this paper is organized as follows. Related work is discussed in Section II. Formal definitions of sampling algorithm and representative verification algorithm are described in Section III. Section IV describes the parallel version of SKMeans using MapReduce, named MR-SKMeans. The experiments are analyzed in Section V. And we summarize our paper in Section VI.

II. Related Work

The traditional clustering algorithm performs efficiently and achieves satisfied results when the size of data is small. However, it performs poorly on large data due to some factors, such as data quantity, data dimensions, computing power and memory.

In order to improve efficiency and accuracy of clustering results under the vast amounts of data environment, various methods have been proposed and divided into two categories. One aims to improve the execution of algorithm and optimize parameter setting, such as GCHL [11], ISB-DBSCAN [12], MCluStream [13], DisAP [14], DBCURE-MR [15], etc. The other is an attempt to apply statistical sampling techniques on clustering algorithms, such as Iterative-Sample-kMedian,

Manuscript received April 04, 2018; revised August 15, 2018. This work was supported by National Social Science Foundation of China under Grant no. 16BJY028.

Hongbiao Li is with College of Information Engineering, Northeast Electric Power University, Jilin, Jilin, China (e-mail: lihongbiao@126.com).

Ruiying Liu is with State Grid Hebei Electric Power Company Co., Ltd Information & Telecommunication Branch, Shijiazhuang, Hebei, China (e-mail: neduqlw@foxmail.com).

Jingdong Wang is with College of Information Engineering, Northeast Electric Power University, Jilin, Jilin, China (e-mail: wjd_nedu@126.com).

Qilong Wu is with State Grid Hebei Electric Power Company Co., Ltd Information & Telecommunication Branch, Shijiazhuang, Hebei, China (e-mail: 863276967@qq.com).

Iterative-Sample-kCenter [16], DMC, WMC [17], etc. MR-SKMeans algorithm, which is proposed in this paper, applies the statistical sampling techniques on clustering algorithms, and belongs to the latter.

In [7], [8], parallel KMeans algorithm using MapReduce could speed up the efficiency and enhance the processing capacity. However, multi-iterations must be operated before convergence. And the computing jobs will be restarted and the initial data will be read and shuffled for each iteration, which causes huge cost.

In [16], a clustering algorithm based on Iterative-Sample was presented using MapReduce by Alina Ene, named Iterative-Sample-kMedian and Iterative-Sample-kCenter. These methods used sampling method to decrease the data size and ran a time consuming kCenter and kMedian algorithm. They also used the sampling ideas proposed by M Thorup as a subroutine for Iterative-Sample [18]. The role of Iterative-Sample performed that it added a small sample of points to the final sample in each iteration, and discarded most points which are close to the current sample. Once they got a good sample, kCenter and kMedian just ran on the sampled points. However, data points must be read repeatedly and eliminated continuously, which led to low sampling efficiency and non-uniform sample set. The points in unsampled set, which cannot be well represented by the sampled set, will be lost when kCenter and kMedian ran just on the sampled points, and the accuracy of clustering will be decreased.

Distribution-based merge clustering (DMC) and Weight-based merge clustering (WMC) were proposed by Xiaoli Cui [17], et al. WMC and DMC were sampled on the original large-scale dataset for $2k$ times to get $2k$ sample sets with probability $p_x = 1/(\varepsilon^2 N)$, where $\varepsilon \in (0,1)$ and controls the size of the sample, N is the number of points and k is the number of clusters. $2k$ small-scale samples were merged into k cluster centers, and then all points were assigned to appropriate centers. It is effective to achieve preferable performance on accuracy of clustering. However, the distribution of sample set which is brought by random sampling is non-uniform, and may lead to miss the points with high probability which cannot be well represented by the sample.

For the purpose of getting satisfied sample set, improving the efficiency of sampling, and ensuring the consistency of the sample, we propose a fast sampling-based KMeans clustering algorithm, named SKMeans, and then develop MR-SKMeans, which is a parallelized SKMeans using MapReduce.

III. SKMeans: KMeans Algorithm Based on Sampling

In this section, an improved version of the sampling algorithm and representative verification will be discussed. As the amount of data increases, the time cost of the algorithm will be enormous. Sampling algorithm aims to get a smaller subset of points substantially that represents all of the points well and decrease the data size. Once we get the sample set, KMeans with representative verification will be run on the sample points. Sampling algorithm can decrease the data size and speed up the clustering algorithm. Representative verification algorithm can achieve high performance in terms of accuracy.

A. Definition

Let $D = \{p_1, \ldots, p_n\}$ denote a set of n points. Each point $p_i \in D$ is a d-dimensional vector $<p_i(1), p_i(2),\ldots, p_i(d)>$ where $p_i(j)$ represents the j-th dimension of the point and n is the total number of points in D.

Definition 1: (grid cell) Given a constant parameter ε. According to the ε, each dimension of the data space is divided into n' spaces, then the whole data space will be partitioned into grids and the length of each grid is $\boldsymbol{\varepsilon}$. We denote the grid cell as *Cell*, which is defined as $Cell = \{c_1, c_2, \ldots, c_{n'}\}$, $c_j = \{|l_{jj'}, h_{jj'}|\}$, $1 \le j \le n'$ and $h_{jj'} - l_{jj'} = \boldsymbol{\varepsilon}$.

Where j represents the index of dimension, j' represents the spatial ordinal, c represents a vector and all elements are in $[l, h)$ for each dimension, l and h are respectively the maximum and minimum values of c.

Definition 2: (sampling domain) Let r denote the sampling radius and point p is referred to as a center. And then, the neighborhood is the sampling domain of p, denoted by $N(p)$, where $N(p)=\{q \in D \mid dist(p, q) \le r\}$, and $dist(p, q)$ is the distance between p and q.

Definition 3: (core sampling point) Let $|N(p)|$ denote the number of points in $N(p)$. We call p a core sampling point if $|N(p)| \ge Minpts$, where $Minpts$ is the minimum of sampling points.

Definition 4: (rectangular domain) Given *Cell* and r, the 2^d vertices of the *Cell* form the sampling domains. And then the *Cell* are referred to as the center and extended K cells along the positive and opposite directions of each dimension, which could cover the 2^d vertex sampling domains. Then the area named rectangular domain formed by the grid set. In order to calculate the rectangular domain, the length of *Cell*, which is ε, is defined as $\varepsilon = r/K$. In experiment, we set $K = 1$ and $\varepsilon = r$.

Example 1: Considering the 2-dimension data points in Fig. 1. Suppose that we would find the rectangular domain of *Cell* with $K = 1$ and $\varepsilon = r$, where *Cell* is a grid and r is the sampling radius. *Cell*, which is considered as the center, has 4 vertices *A*, *B*, *C*, *D* which are utilized to discover the sampling domains. And then K cells are extended both vertically and horizontally. A new rectangular domain which could cover the 4 vertex sampling domains has been formed.

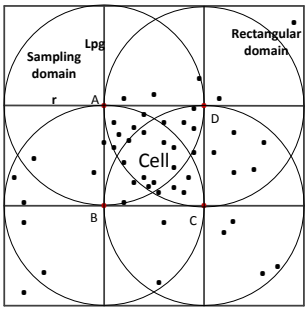

Fig. 1. The overview of rectangular domain

Definition 5: (sample point *SP*) Given $M \in D$, r and $Minpts$. Suppose that the number of points in the sampling domain of a data point M (i.e. $|N(M)|$) is greater than $Minpts$, so M is considered as the core sampling point. Next, we discover the sampling domain, in which M is referred to as the center and r is the radius. The center point of each quadrant of the

sampling domain boundary of M is considered as the boundary points (denoted by BP). Finding out the nearest points p_i (i = 1, ..., 2^d) among all points in sampling domain of M and BPs. Then p_i is the sample point if the distance $dist_1$ between p_i and the nearest BP of p_i is not more than the distance $dist_2$ between p_i and M.

Example 2: Considering the 2-dimension data points in Fig. 2. Suppose that we would find SPs of M with K = 1 and $\varepsilon = r$, where *Cell* is a grid and r is the sampling radius. The sampling domain is discovered with r, and BPs are the points of E, F, G, H. Then we will find out the nearest points e, f, g, h included in SPs.

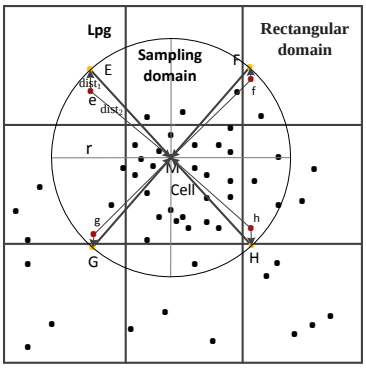

Fig. 2. The overview of rectangular sample points

B. Sampling Algorithm

When KMeans algorithm is performed, the number of iterations increased exponentially before the set of centers converges [19]. Restarting MapReduce jobs, inputting original data set and shuffling intermediate results will be executed repeatedly when the large-scale data is clustered by KMeans algorithm in parallel using MapReduce, which increases both I/O and network costs significantly. Accordingly, a new sampling algorithm is proposed in order to reduce the input data size. Simultaneously, the grid division method will be regarded as a complement to reduce the time cost of range query. The data set D is partitioned into grids, and the data points are mapped to the grid cells. The rectangular domain related to point p is computed by grid partition. If the number of points in the rectangular domain related to p is less than *Minpts*, then the number of points in sampling domain related to p must be less than *Minpts* (i.e. $|N(p)|$<*Minpts*), all the points in the rectangular domain are inserted into the sample set S. If the number of points in the rectangular domain related to p is more than *Minpts*, then we calculate the total number of points in sampling domain related to p (i.e. $|N(p)|$). If $|N(p)|$<*Minpts*, all the points in the sampling domain are added into the sample set S. If not, we call p the core sampling point, and find the SPs in the sampling domain related to p. Then p and its SPs are inserted into sample set S. Finally, SPs is regarded as the seed points to expand the sampling. The pseudo code of Algorithm 1 and Algorithm 2 (see Table I and Table II) explains how it works.

Sample set obtained by the algorithm can keep the characteristic distribution of the original data very well. The loss of low density clusters and noise points can be prevented effectively. The sample points obtained by the method we proposed are highly representative, and the method guarantees the clustering quality in the subsequent clustering.

TABLE I
PROCEDURE OF ALGORITHM 1

Algorithm1: Sampling Algorithm(*D, r, Minpts*)
1: Set $S \leftarrow \varphi$, $H \leftarrow \varphi$, $Queue \leftarrow \varphi$
2: divide data spatial into grids which size is ε, map all the points in D into the grid cells
3: mark all $p_i \in D$ as "*UNVISITED*"
4: **While** p_i is marked as "*UNVISITED*" **do**
5: mark p_i as "VISITED", calculate the number of points in rectangular domain of p_i as $|N_1(p_i)|$
6: **if** $|N_1(p_i)|$< *Minpts*, **then**
7: **for** each $p \in N_1(p_i)$ **do**
8: S.add(p), mark p as "VISITED"
9: **else**
10: calculate the number of points in sampling domain of p_i as $|N_2(p_i)|$
11: **if** $|N_2(p_i)|$< *Minpts*, **then**
12: **for** each $p \in N_2(p_i)$ **do**
13: S.add(p), mark p as "*VISITED*"
14: **else**
15: *Finding SPs*(p_i, $N_2(p_i)$)
16: **while** $Queue \neq \varphi$ **do**
17: select the head point p_1, and $p_i \leftarrow p_1$
18: **end while**
19: **end while**
20: calculate $\eta = |S| / |D|$
21: Output S, H,η

TABLE II
PROCEDURE OF ALGORITHM 2

Algorithm2: Finding *SP*s(p_i, $N_2(p_i)$)
1: identify boundary data object BPs of p_i
2: calculate the distance dist(p_i, BP) between BPs and all the points in sampling domain of p_i, and select the closest points p from BPs
3: **for** each p **do**
4: **if** p is "*UNVISITED*" **then**
5: **if** $dist(p, BP) < dist(p, p_i)$ **then**
6: S.add(p), $Queue$.add(p)
7: mark each $p_m \in N_2(p_i)$ as "VISITED"
8: **for** each $p_m \in N_2(p_i)$ and $p_m \neq p$ **do**
9: p_m add tags of its core sampling point p_i
10: H.add(p_m)
11: **end if**
12: **end if**
13: return S, H, $Queue$

C. Representative Verification

In order to improve the quality of clustering and avoid losing data points that cannot be represented by S, Representative verification is proposed in KMeans. In this paper, the distance of a point p_i to a set *Cluster* means the shortest distance between p_i and any point in *Cluster*. If $dist(p^*, C) \leq dist(p, C)$, then point p is satisfied by S with respect to C, where C is the set of cluster centers and p^* is the core sampling point of p. If p is unsatisfied by S, then it cannot be well represented by S and needs to be inserted into S. Suppose that itr is the convergent boundary, k is the number of clusters, t is the total number of iterations and b is the b-th cluster where $b \in [1, k]$. The loop of KMeans will be continued until there are no differences of the centers between (t-1)-th iteration and t-th iteration. In the following experiments, itr is equal to 10^{-6}. The pseudo code of Algorithm 3 (see Table III) and Algorithm 4 (see Table IV) is described as below.

TABLE III
PROCEDURE OF ALGORITHM 3

Algorithm3: KMeans algorithm with representative verification(*S*, *H*, *k*)

1: Let $a = Float.MAXVALUE$; $t = 1$
2: Choose k centers from S, let $C^{(0)}=C_1^{(t)},C_2^{(t)},\ldots,C_k^{(t)}$
3: **while** $a > itr$ **do**
4: form k clusters by assigning $p_i \in S$ to its nearest center and get the nearest distance $dist_{min}$
5: *Representative Verification*(*S*, *H*)
6: find new centers of the k clusters $C_1^{(++t)}$, $C_2^{(++t)}$,..., $C_k^{(++t)}$
7: $a \leftarrow \sum_{m=0}^{k} \left\| c_m^t - c_m^{t-1} \right\|^2$
8: Output $C^{(t)}$

TABLE IV
PROCEDURE OF ALGORITHM 4

Algorithm4: Representative Verification(*S*, *H*)

1: **for** each $p_m \in H$ **do**
2: get p_m* which is added as tags of core sampling point
3: **if** $dist_{min}(p_m, C) < dist(p_m^*, C)$ **then**
4: p_i is unsatisfied by S with respect to C
5: S.add(p_m), H.delete(p_m)
6: return *S*, *H*

IV. PARALLEL SKMEANS ALGORITHM USING MAPREDUCE

In this section, a distributed algorithm MR-SKMeans using MapReduce is proposed on the basis of SKMeans algorithm.

MR-SKmeans algorithm is divided into two chief phases as shown in Fig. 3. The sampling algorithm runs in parallel by Mapper1 and Reducer1 in the first phase. KMeans with representative verification algorithm is carried out in parallel by Mapper2 and Reducer2 in the second phase.

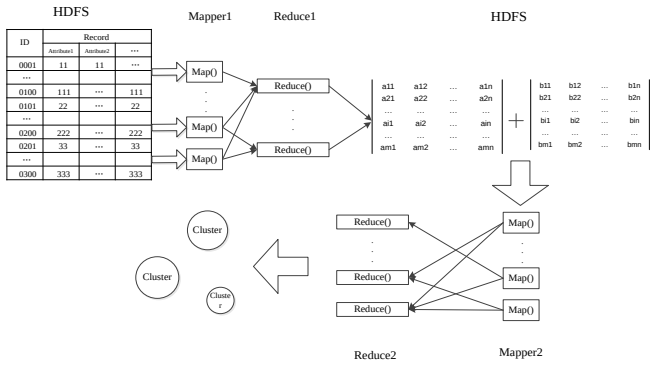

Fig. 3. MR-SKMeans algorithm in MapReduce

A. Sampling Algorithm in MapReduce

According to the size of processed file size in the distributed file system, data block is stored in different nodes whose storage capacity is 64M, and each data block is allocated to a Map task to complete the calculation. In Mapper1, k_1 denotes the processed file *id* and v_1 indicates the data records, each Map task output S_j and H_j, where j is the j-th Map task. In Reducer1, the inputs are (1, S_j) and (2, H_j). Each Reduce task outputs are *S* and *H*. The pseudo code of Algorithm 5 (see Table V) is detailed as below.

Algorithm 5 shows the sampling process using MapReduce. Each mapper does sampling and then shuffle these samples S_j and H_j to the corresponding reducers. In the end, samples *S* and *H* can be obtained, where $1 \le j \le Mp$ and Mp is the number of Map tasks.

TABLE V
PROCEDURE OF ALGORITHM 5

Algorithm5: MapReduce-Sampling Algorithm(*D*, *r*, *Minpts*)

1: Mapper1(k_1, v_1)	5: Reducer1((1, S_j),(2, H_j))
2: **for** each mapper **do**	6: **for** each $p_i \in S_j$, $p_m \in H_j$ **do**
3: do sampling by Sampling Algorithm(*D*, *r*, *Minpts*)	7: S.add(p_i), H.add(p_m)
4: Output((1, S_j), (2, H_j))	8: Output(*S*, *H*);

B. Representative Verification in MapReduce

In the distributed file system, KMeans with representative verification algorithm is carried out with *S* and *H* which is the input data set. In Mapper2, the inputs for each Map task are *S*, *H* and the cluster centers of the last round of iteration (or the initial cluster centers). And the outputs are cluster ID denoted by *Cluster* and points denoted by p_b where $b \in [1, k]$. In Reducer2, each Reduce task is to calculate the new cluster centers with b and C_b as the input, and output final centers. The pseudo code of Algorithm 6 (see Table VI) is shown as below.

TABLE VI
PROCEDURE OF ALGORITHM 6

Algorithm6: MapReduce-KMeans with Representative Verification(*S*,*H*, *k*)

1: Mapper2(k_1, v_1)	5: Reducer2(b, C_b)
2: form k clusters by assigning each $p_i \in S$ to its nearest center	6: calculate new centers of the k clusters
3: *Representative Verification*(*S*, *H*)	7: repeat 1,2,3,4,5,6 until satisfied the convergence condition
4: Output((1,C_1), (2,C_2),…, (k,C_k))	8: Output final centers

C. Complexity Analysis

Algorithm 1 and Algorithm 2 consist of three steps. Firstly, the data spatial is divided into grids. The maximum and minimum of each dimensionality in space need to be calculated, thus the time complexity is $O(n*d)$, where d is the total dimensionality of data space. Secondly, all the points are mapped to the grid cells. Suppose that n is the number of points in D which must be traversed when all the points are mapped to the grid cells, so the time complexity is $O(n)$. At last, the sample points *SP*s are selected and extended as the seed points, so the time complexity is $O(n*d+2^d)$. In summary, the time complexity of Algorithm 1 and Algorithm 2 is $O(n*d+2^d)$.

According to the Algorithm 3 and Algorithm 4, some points in *H* cannot be well represented by *S* and need to be inserted into *S* in each iteration. Suppose that n_m is the number of points which need to be inserted into *S* in the m-th iteration, where $m \in [1, t]$, and t is the total number of iterations. Thus, the time complexity of Algorithm 3 and Algorithm 4 is $O((s*t+\sum_{m=1}^{t} n_m)*k)$, where s is the total number of points in *S*, k is the total number of clustering centers.

When MR-SKMeans is executed, Mp_1 means the average number of map tasks of Mapper1 on one node. N_1 is the total number of nodes in the Hadoop cluster used to perform the tasks, then the time complexity of Algorithm 5 is $O((n*d+2^d)/(Mp_1*N_1))$. Meanwhile, the time complexity of algorithm 6 is $O((s*t+\sum_{m=1}^{t} n_m)*k/(Mp_2*N_2))$, where Mp_2 is the average number of map tasks of Mapper2 on one node, N_2 is

the number of nodes in the Hadoop cluster to perform the tasks.

In conclusion, the total time complexity of MR-SKMeans is $O((n*d+2^d)/(Mp_1*N_1)+(s*t+\sum_{m=1}^{t} n_m)*k/(Mp_2*N_2))$.

V. Experimental Setup

A. Datasets

In our experiments, five datasets are used to evaluate the performance of MR-SKMeans. The type of all data sets is numerical. And the clustering results of the data sets must be convex shape. In other words, the clusters must be convex.

The first dataset, S-2 Data Set [20], [21], consists of 5,000 points in 2 dimensions and has 15 clusters. To evaluate MR-SKMeans with large data sets, S-2 Data Set is extended, and has been named S-2*, by duplicating its original records 500 times and the number of data points is 2,500,000. In order to generate data points which are different from data points included in S-2 Data Set, a small random noise with a variation of +/- 5 is adapted for each coordinate value of the data points.

The other three datasets are from real-world settings and can be publicly available from the UC Irvine Machine Learning repository [22]. The 3D Road Network Data Set consists of 434,874 points in 4 dimensions. And 3D Road Network Data Set is also increased by duplicating its records 10 times, named Road*, and the number of data points is 4,348,740. And a small random noise with a variation of +/- 1 is also adapted. The Individual Household Electric Power Consumption Data Set, named House, consists of 2,075,260 points in 9 dimensions. Bag of Words Data Set, named BoW, consists of 483,450,157 points in 3 dimensions. S-2 dataset is tested in a single machine and the other four datasets are verified with the parallel implementation in the Hadoop framework.

In our experiments, there are five datasets (S-2, S-2*, Road*, House and BoW) used in parallel KMeans [7] based on MapReduce, Iterative-Sampling-KMeans (IS-KMeans) [16], WMC, DMC [17] and MR-SKMeans. Moreover, Euclidean distance was used as similarity measure. In this paper, the Davies-Bouldins index [23] (DBI) used in our proposed clustering algorithm for examining the soundness of our clusters will be discussed. Davies-Bouldins index is a function of the ratio of the sum of within-cluster distribution to between-cluster separation.

B. Parameter Setting Comparison

For the MR-SKMeans algorithm, the sampling results and clustering accuracy are determined by the sampling radius *r* and minimum of sampling points *Minpts*. In this subsection, the effects bought by *r* and *Minpts* on sampling results and clustering accuracy are explored for different parameter settings. MR-SKMeans algorithm and KMeans algorithm are carried out with S-2 dataset. When *Minpts*=50, the sampling rate and DBI index of different *r* are shown in Fig. 4. When $r \in [1750, 2250]$, we can get a lower DBI index which is close to 2.27. When *r*=2000, the sampling rate and DBI index of different *Minpts* are shown in Fig. 5. When $Minpts \in [45, 50]$, we can get a lower DBI index which is close to 2.48.

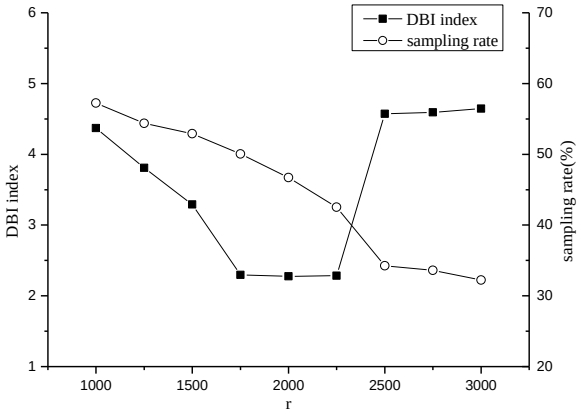

Fig. 4. Effects of varying *r* on DBI index and sampling rate

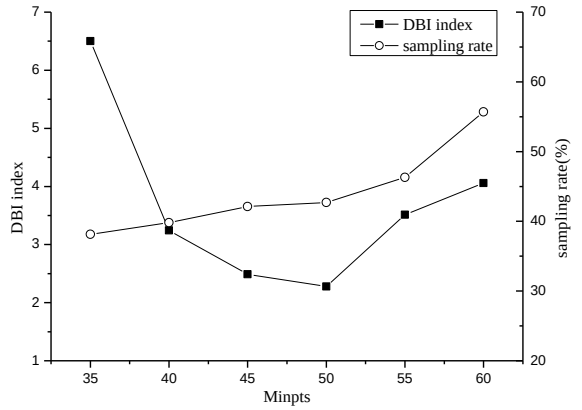

Fig. 5. Effects of varying *Minpts* on DBI index and sampling rate

DBI index is close to 4.07 when KMeans is executed with S-2 dataset. In Fig. 4 and Fig. 5, the sampling rate decreases with the increasement of *r*, moreover, increases with the increment of *Minpts*. However, when $r \in [1750, 2250]$, $Minpts \in [45, 50]$, DBI index of MR-SKMeans is less than 4.07. Therefore, MR-SKMeans can get a lower DBI index and achieve better clustering results than KMeans by means of *r* and *Minpts*.

C. Performance Evaluation

S-2*, Road*, House and BoW datasets were used in parallel KMeans, MR-SKMeans, IS-KMeans, WMC, and DMC in this subsection. Fig. 6 shows the performance of these algorithms when *k*=15 for S-2* and Road*, *k*=50 for House and BoW. For S-2*, Road*, House and BoW is utilized, Table VII indicates the DBI index of these algorithms. And KMeans is carried out in parallel based on MapReduce. The same cluster validity index is used to select the clustering. These algorithms are executed repeatedly for 20 times, the average of execution time is described in Fig. 6 and the average DBI indexes are showed in Table VII.

As shown in Fig. 6, MR-SKMeans is far better than the parallel KMeans. Moreover, it is better than that of IS-KMeans, DMC and WMC algorithms. As shown in Table VII, DBI index of MR-SKMeans is slightly larger than KMeans, WMC and DMC in terms of Road* dataset, however, it is inferior to IS-KMeans. In terms of S-2*, House and BoW datasets, DBI index of MR-SKMeans is significantly lower than that of other algorithms. Experimental results indicate that the average clustering time has been significantly reduced.

TABLE VII
COMPARISON OF DBI INDEX BY VARIOUS DATASETS

Data Set	K	KMeans	IS-KMeans	WMC	DMC	MR-SKMeans
S-2*	50	3.7594	2.7613	3.5106	3.2907	2.4597
Road*	50	0.5901	0.6720	0.5847	0.5879	0.6257
House	50	2.0664	2.0789	2.2056	2.2081	1.7516
BoW	50	3.0449	2.0892	2.2789	1.9461	1.2451

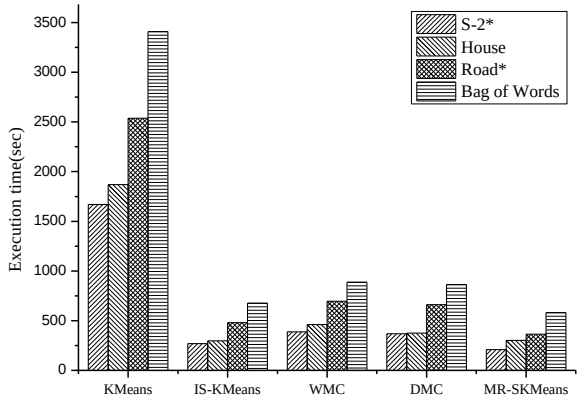

Fig. 6. Comparison of time measured by various methods

Meanwhile, the efficiency of MR-SKMeans is explored on varying numbers of machines from 3 to 9. Fig. 7 illustrates the results of clustering on S-2*, Road*, House and BoW datasets, when k=15 for S-2* and Road*, k=50 for House and BoW. The result shows that MR-SKMeans is scalable.

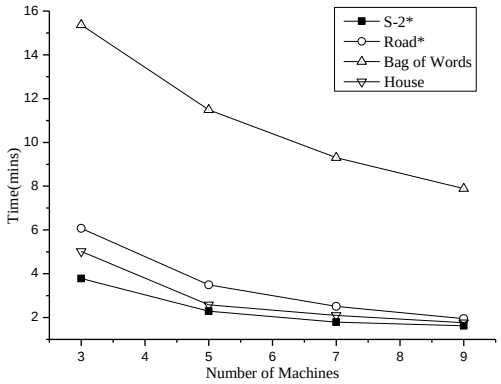

Fig. 7. Running time with varying number of machines

D. Discussions

The efficiency and accuracy of the proposed algorithms are evaluated by varying in parameters, sizes of datasets and numbers of machines respectively. Fig. 4 and Fig. 5 show that the parameters of r and *Minpts* have great influence on sampling results and clustering accuracy, which can be inferred from the sampling rate and DBI index. When $r \in$ [1750, 2250], *Minpts* $\in$[45, 50], the sampling rate is about 45%, DBI index of our algorithms is lower than that of KMeans. Conclusions can be made that the appropriate r and *Minpts* can be used to reduce the amount of data effectively, and get better clustering results.

Fig. 6 and Table VII indicate the efficiency and accuracy of various algorithms on different sizes of datasets. As shown in Fig. 6, MR-SKMeans spends less time than the other four algorithms. The reason for this phenomenon is that Algorithm 1 in Section III is used to obtain the points which can represent the unsampled points well. Moreover, the sample set, which is used for clustering, is far smaller than the initial data set. Thus, both I/O and network costs are decreased significantly. The method of grid division in Algorithm 1 is used to execute query in a limited extent related to each point, and it reduces the total computation cost greatly and saves a lot time. We can conclude from Table VII that MR-SKMeans gets lower DBI index than the other algorithms for S-2*, House and BoW datasets. The reason for the conclusion is that the distribution of sample set and the initial data set is consistent, and representative verification algorithm avoids missing data points which cannot be represented by the sample set. However, DBI index of MR-SKMeans is larger than the other algorithms for Road* set. The reason lies in the fact that there is a great difference between the shapes of clusters. Meanwhile in the Road* set, there exists a great deal of outliers included in the sampling set, which is sensitive to KMeans. Therefore, it exerts an influence on clustering results. Fig. 7 indicates that our proposed algorithm is scalable.

VI. CONCLUSIONS AND FUTURE WORKS

In order to overcome the shortcomings of traditional sampling algorithms when big data is dealt with the parallel KMeans. A new sampling algorithm is first proposed to obtain the sample set by making full advantages of the definitions of rectangular domain, sampling domain and sample point. And then, representative verification algorithm is proposed to avoid missing the points which cannot be well represented by the points in sample set. Based on the above, the parallel MR-SKMeans is proposed using MapReduce. Experimental results demonstrate that MR-SKMeans can find the appropriate parameters to get better clustering results. Compared to Iterative-Sampling-KMeans, WMC and DMC, MR-SKMeans performs better and could achieve higher accuracy when various large data sets are utilized. Furthermore, our proposed algorithm scales up well with MapReduce framework. Future work will be devoted to automatically identify the parameters, instead of specified by users.

ACKNOWLEDGMENT

The authors would like to thank the Intelligent Information Processing Group for supporting the implementation of proposed algorithms and tools.

REFERENCES

[1] A. K. Jain and P. J. Flynn, "Image segmentation using clustering," *Journal of Global Optimization*, pp. 65–83, 1996.
[2] I. Cadez, P. Smyth and H. Mannila, "Probabilistic modeling of transaction data with applications to profiling, visualization, and prediction," in *Proc. 7th ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, San Francisco, 2001, pp. 37–46.
[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Proc. 6th conference on Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, 2004, pp.

10–10.
[4] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski and C. K, "Evaluating MapReduce for multi-core and multiprocessor systems," in *Proc. 13th IEEE International Symposium on High Performance Computer Architecture*, DC, 2007, pp. 10–14.
[5] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.
[6] Apache hadoop is available at http://hadoop.apache.org/.
[7] W. Zhao, H. Ma and Q. He, "Parallel k-means clustering based on mapreduce," Springer, Berlin Heidelberg.
[8] C. T. Chu, K. K. Sang, Y. A. Lin, Y. Y. Yu and G. Bradski, "Map-Reduce for Machine Learning on Multicore," *Advances in Neural Information Processing Systems*, vol. 19, pp. 281–288, 2006.
[9] Y. A. Cui, X. Li, Z. X. Wang and D. Y. Zhang, "A comparison on methodologies of sampling online social media," *Chinese Journal of Computers*, vol. 37, no. 8, pp. 1859–1876, Aug. 2014.
[10] M. K. Pakhira, "Fast Image Segmentation Using Modified CLARA Algorithm," in *Proc. International Conference on Information Technology*, Bhubaneswar, 2008, pp. 14–18.
[11] A. H. Pilevar and M. Sukumar, "GCHL: A grid-clustering algorithm for high-dimensional very large spatial data bases," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 999–1010, May. 2005.
[12] H. Lu, T. H. Ma, L. M. Tang, J. Cao, Y. Tian, A. D. Abdullah and A. R. Mznah, "An efficient and scalable density-based clustering algorithm for datasets with complex structures," *Neurocomputing*, vol. 171, no. C, pp. 9–22, Jan. 2015.
[13] Y. W. Yu, H. Wang, Q. Wang and J. D. Zhao, "Density-based cluster structure mining algorithm for high-volume data streams," *Journal of Software*, vol. 26, no. 5, pp. 1113–1128, May. 2015.
[14] W. M. Lu, C. Y. Du, B. G. Wei, C. H. Shen and Z. C. Ye, "Distributed Affinity Propagation Clustering Based on MapReduce," *Journal of Computer Research and Development*, vol. 49, no. 8, pp. 1762–1772, Aug. 2012.
[15] Y. Kim, K. Shim, M. S. Kim and J. S. Lee, "DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce," *Information Systems*, vol. 42, no. 2, pp. 15–35, Jun. 2014.
[16] A. Ene, S. Im and B. Moseley, "Fast clustering using MapReduce," in *Proc. the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, SanDiego, 2011, pp. 681–689.
[17] X. L. Cui. P. Zhu, X. Yang, K. Li and C. Ji, "Optimized Big Data K-means Clustering Using MapRedece," *Journal of Supercomputing*, vol. 70, no. 3, pp. 1249–1259, Dec. 2014.
[18] M. Thorup, P. Zhu, X. Yang, K. Li and C. Ji, "Quick k-Median, k-Center, and Facility Location for Sparse Graphs," *Siam Journal on Computing*, vol. 34, no. 2, pp. 249–260, Jul. 2000.
[19] A. Vattan, "K-means Requires Exponentially Many Iterations Even in the Plane," *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 596–616, Jan. 2006.
[20] P. Franti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761–775, May. 2006.
[21] Clustering datasets is available at http://cs.joensuu.fi/sipu/datasets/.
[22] UCI Machine Learning Repository is available at http://archive.ics.uci.edu/ml/.
[23] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, Apr. 1979

Hongbiao Li received his MSc degree in computer applied technology from Northeast Electric Power University, China in 2007. His research interests are in artificial intelligence and data mining.

Ruiying Liu received her MSc degree in computer applied technology from Northeast Electric Power University, China in 2017. Her research interests are in artificial intelligence and data mining.

Jingdong Wang received his MSc and PhD degree in computer science from Northeast Electric Power University, China in 2008 and University of Chinese Academy of Sciences, China in 2017, respectively. His research interests are in artificial intelligence and assessment of energy efficiency.

Qilong Wu received his MSc degree in computer applied technology from Northeast Electric Power University, China in 2017. His research interests are in data mining and pattern recognition.