CAAI Transactions on Intelligence Technology

Research Article

Efficient algorithm for big data clustering on single machine

ISSN 2468-2322
Received on 27th June 2019
Revised on 8th October 2019
Accepted on 5th November 2019
doi: 10.1049/trit.2019.0048
www.ietdl.org

Rasim M. Alguliyev, Ramiz M. Aliguliyev ✉, Lyudmila V. Sukhostat

Institute of Information Technology, Azerbaijan National Academy of Sciences, 9A, B. Vahabzade Street, AZ1141 Baku, Azerbaijan
✉ *E-mail: r.aliguliyev@gmail.com*

Abstract: Big data analysis requires the presence of large computing powers, which is not always feasible. And so, it became necessary to develop new clustering algorithms capable of such data processing. This study proposes a new parallel clustering algorithm based on the k-means algorithm. It significantly reduces the exponential growth of computations. The proposed algorithm splits a dataset into batches while preserving the characteristics of the initial dataset and increasing the clustering speed. The idea is to define cluster centroids, which are also clustered, for each batch. According to the obtained centroids, the data points belong to the cluster with the nearest centroid. Real large datasets are used to conduct the experiments to evaluate the effectiveness of the proposed approach. The proposed approach is compared with k-means and its modification. The experiments show that the proposed algorithm is a promising tool for clustering large datasets in comparison with the k-means algorithm.

1 Introduction

Clustering algorithms are widely applied to big data analysis to show the internal relationship between the data [1–8]. The most popular among them is k-means. It is fast enough and does not require large computational resources for small datasets. However, the method shows unsatisfactory results on big data. Analysis of large-scale datasets requires the presence of large computing powers, which is not always feasible. As well as the result of k-means clustering depends on the position of the initial centroids of the clusters, in this regard, it can easily fall into the local optimum [9, 10].

An effective way to process big data is to split data into small blocks (batches) [11–17]. At the same time, the result of each batch should be reliable and demonstrate a good result for the entire dataset. It is difficult to choose the optimal batch size [18].

In this paper, we propose an approach based on parallel processing of batches using the k-means algorithm. In this case, the dataset is split into several batches to satisfy the limitations of the k-means algorithm [19]. The resulting clusters are created in parallel without full memory loading, which significantly speeds up the clustering.

The relevance of the work is that the use of small batches reduces computational cost and increases the convergence speed of the clustering algorithm. Real large datasets are used to evaluate the performance of the proposed approach. The proposed approach is compared with k-means and Mini Batch k-means algorithms.

The rest of this paper is organised as follows: a literature review is given in Section 2. Section 3 describes the proposed clustering algorithm for big data on a single machine. The experimental results are given in Section 4, followed by conclusion.

2 Related work

A large number of researches have been devoted to big data analysis [3, 4, 6, 7]. The issue with big data clustering is that a lot of memory is required. Researchers offer new methods and extend existing clustering algorithms to solve this issue.

In recent years, the k-means algorithm and its modifications have been the subject of research on the analysis of large volumes of data [16, 20–23].

However, the k-means algorithm is quite sensitive to initialisation issues. So, an approach for the k-means algorithm initialisation was proposed in [24]. It is quite simple to implement, not trivial and converges quickly enough in a small number of iterations. However, this approach requires a large computational cost. It is necessary to use parallel technology to perform clustering on high-dimensional data.

In connection with this, there is a growing need to parallelise big data clustering while preserving the main clustering structure and reducing computational costs. So, k-means based parallel algorithms are used to cluster data in various applications [20, 25, 26].

A multi-core parallelisation of the k-means/k-modes algorithm for biological data clustering that provides complex cluster number estimations for big data on a single computer was proposed [21]. However, this approach requires additional effort and equipment (specialised hardware for fast communication between computers, multiple software installations in heterogeneous environments).

A parallel implementation of the k-means clustering algorithm on a cluster of personal computers (PCs) was described in [27]. The proposed algorithm is parallelised based on the inherent data-parallelism especially in the distance calculation and centroid update operations for DNA dataset. The time complexity of this method is highly dependable on the number of iterations.

An efficient method for topological data clustering and discovering clusters of arbitrary shape was proposed [25]. Experiments on real and synthetic datasets showed the efficiency of the proposed method. However, the speedup of the algorithm was not evaluated.

Summarising the analysis of the state of research in parallel big data processing using k-means, we can draw the following conclusions. Firstly, not all papers based on k-means initialisation issues consider parallel data processing. Secondly, works aimed at improving the quality of clustering often require large computational resources. Besides, parallel clustering is quite a demanded area of research due to the constant growth of data volumes. This confirms the relevance of our research.

A new clustering method based on parallel batch clustering on a single machine using the k-means algorithm is proposed. The basic idea is that the dataset is divided into several parts of equal dimensions, which are then parallel clustered to detect the centroids of the clusters and overcome the curse of dimensionality

inherent in k-means for big data clustering. Then k-means applies again to the resulting array of centroids from all batches. It leads to a reduction in computational costs comparing to the classic k-means. Experiments on datasets of medium and high dimension show that the proposed approach based on parallel clustering of batches significantly improves the clustering time for all datasets.

3 Proposed algorithm

In this paper, we propose an algorithm based on batches that are clustered in parallel. The proposed algorithm with a split dataset consists of several steps. The input dataset is divided into batches. Clustering is applied to each batch as a separate dataset. The initial centroids are selected randomly. Each batch is processed in parallel until the convergence condition is met. The algorithm minimises the sum of squared errors for all clusters. The resulting centroids of each batch form a new small dataset to which clustering is again applied to determine the centroids. Data partitioning into the batches before clustering and their parallel processing reduce the computation time.

Let us denote the following notations: $X = \{x_1, x_2, \ldots, x_n\}$ is the set of a finite number of points given in m-dimensional space, q is the batch size, the maximum value $q(q < n)$ is determined by the PC parameters, and is also processed within a reasonable time, $C = \{C_1, C_2, \ldots, C_k\}$ is a set of clusters, where $C_q(p = \overline{1,k})$ is the pth cluster and k is the number of clusters, O_p is the centroid of the pth cluster.

To determine the optimal batch size, we consider the method proposed in [28]

$$q = \frac{\nu(\alpha) \cdot k^2}{r^2}, \tag{1}$$

where α is the desired significance level, $\nu(\alpha)$ is the value obtained from the table in [29], k is the number of clusters, and r is the 'relative difference'. We assume that $\alpha = 0.05$ (with a 95% probability), $\nu(\alpha) = 1.27359$ and $r = 0.08$.

The objective function has the following form:

$$\text{minimise } f(x) = \sum_{p=1}^{k} \sum_{x_i \in C_p} \left\| x_i - O_p \right\|^2, \tag{2}$$

$$O_p = \frac{\sum_{x_i \in C_p} x_i}{\left| C_p \right|}, \quad p = \overline{1,k}, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathcal{R}^m$, $\left|C_p\right|$ is the number of data points in the cluster C_p.

The resulting centroid, obtained after applying k-means to the set of centroids of all batches, is denoted as $O_p^* \left(p = \overline{1,k}\right)$.

The task is to reduce the clustering time. Steps of the proposed algorithm are shown in Fig. 1.

If the dataset cannot be completely split into equal batches, the residual part of the data points is fed to step 5 of the algorithm, where the points are mapped to each cluster obtained in step 4.

4 Experimental results and discussion

Three large datasets are considered in the paper to evaluate the effectiveness of the proposed approach (Table 1). They were taken from the UCI machine learning repository [30].

In the experiments, these datasets are split into equal fragments (batches). Each batch has the same size (5000, 10,000, 15,000, and 20,000 samples). k samples are randomly selected from a dataset for centroids initialisation.

The algorithms were executed ten times with the number of clusters equal to 2, 3, 5, 10, and 15 to compare the efficiency of our algorithm with k-means and its modifications. The mean

Input: $X = \{x_1, x_2, \ldots, x_n\}$
q: batch size according to (1)
k: number of desired clusters
Output: A set of k clusters $C = \{C_1, C_2, \ldots, C_k\}$
Step 1. Split dataset of n elements into a set of batches with equal size q.
Step 2. Apply clustering to each batch:
Initialization of clusters' centroids
Repeat
Calculate the value of the function (2) for each data point x_i
Recalculate the new cluster centroids according to (3)
until the convergence condition is met
Step 3. Create a set of centroids obtained in Step 2.
Step 4. Repeat
Calculate the value of the function in (2) for each centroid from Step 3
Recalculate the new cluster centroids
until the convergence condition is met
Step 5. Mapping of data points to clusters.
End

Fig. 1 *Steps of the proposed algorithm*

Table 1 Summary of the datasets

Dataset	No. of instances	No. of attributes	References
US Census (1990)	2,458,285	68	[31–33]
YearPrediction MSD	515,345	91	[34]
Phone Accelerometer	1,048,575	6	[35]

values and standard deviations of the objective function (f) and execution time (T) have been recorded. All experiments were carried out to obtain objective results. The proposed algorithm and k-means algorithm were programmed in Matlab 2018a and Mini Batch k-means [11] were implemented in R 3.4.1. The algorithms were performed on Intel(R) Core(TM) i7-4170HQ CPU @ 2.50 GHz * 4, RAM 8 GB.

In general, the best performance according to the objective function is observed for the k-means algorithm.

However, compared to our approach, the computational efficiency of the k-means is worse in four times. The proposed approach can get results in a relatively short time, requiring less computational resources.

YearPredictionMSD and US Census (1990) datasets are very large, and therefore k-means cannot cluster them at $k = 10$ and $k = 15$. Due to this, only CPU execution time can show the superiority of the proposed approach.

The presented approach considerably reduces the computation time. We compared the efficiency of clustering and the runtime (the CPU time for the algorithm in one run) between the presented approach and the k-means algorithm on large datasets (Phone Accelerometer dataset, YearPredictionMSD dataset, and US Census (1990) dataset) (Table 1).

The experimental results on Phone Accelerometer, YearPredictionMSD, and US Census (1990) datasets using the proposed approach are presented in Tables 2–4. Standard deviations are shown in parentheses.

The tables show the relative improvement (in percentage) of the proposed approach compared to the k-means for the average values of the objective function and time.

The '+' sign indicates the improvement of clustering when applying the proposed approach, and the '−' sign indicates its deterioration compared to the k-means.

In Table 2, there is an increase in the performance of the proposed approach for different values of k and the size of the batch for the Phone Accelerometer dataset. With $k = 3$, the mean value of the

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 1, pp. 9–14

Table 2 Experimental results on Phone Accelerometer dataset

No. of clusters	k-means		Batch size	Mini Batch k-means		Proposed algorithm			
	f	T		f	T	f	%	T	%
$k=2$	4.1942×10^{10} (0.0000)	5.22 (0.59)	5000	4.1973×10^{10} (0.0000)	3.03 (0.67)	4.1934×10^{10} (0.0000)	+0.02	1.15 (0.12)	+353.91
			10,000	4.1950×10^{10} (0.0000)	3.02 (0.59)	4.1930×10^{10} (0.0000)	+0.03	1.08 (0.05)	+383.33
			15,000	4.1945×10^{10} (0.0000)	3.20 (0.61)	4.1911×10^{10} (0.0000)	+0.07	0.99 (0.06)	+427.27
			20,000	4.1952×10^{10} (0.0000)	3.40 (0.15)	4.1947×10^{10} (0.0000)	−0.01	0.97 (0.06)	+438.14
$k=3$	2.5774×10^{10} (0.0000)	6.99 (0.22)	5000	2.5766×10^{10} (0.0000)	3.68 (0.91)	2.5749×10^{10} (0.0000)	+0.10	2.72 (0.11)	+156.99
			10,000	2.5777×10^{10} (0.0000)	3.71 (0.75)	2.5760×10^{10} (0.0000)	+0.05	2.57 (0.10)	+171.98
			15,000	2.5782×10^{10} (0.0000)	4.57 (0.68)	2.5774×10^{10} (0.0000)	0.00	2.47 (0.15)	+183.00
			20,000	2.5774×10^{10} (0.0000)	3.62 (0.70)	2.5734×10^{10} (0.0000)	+0.16	2.55 (0.20)	+174.12
$k=5$	1.5673×10^{10} (0.0000)	22.41 (0.21)	5000	1.5704×10^{10} (0.0000)	8.06 (0.50)	1.5702×10^{10} (0.0001)	−0.18	7.35 (0.39)	+204.90
			10,000	1.5695×10^{10} (0.0000)	8.31 (0.47)	1.5690×10^{10} (0.0001)	−0.11	7.81 (0.27)	+186.94
			15,000	1.5679×10^{10} (0.0000)	10.79 (0.39)	1.5676×10^{10} (0.0000)	−0.02	10.10 (0.41)	+121.88
			20,000	1.5678×10^{10} (0.0000)	10.08 (0.56)	1.5671×10^{10} (0.0000)	+0.01	9.75 (0.53)	+129.85
$k=10$	7.8899×10^{9} (0.0000)	160.91 (0.16)	5000	8.0881×10^{9} (0.0001)	26.31 (0.86)	7.8868×10^{9} (0.0000)	+0.04	30.19 (0.37)	+432.99
			10,000	8.0450×10^{9} (0.0001)	36.20 (0.19)	7.9019×10^{9} (0.0001)	−0.15	35.64 (0.35)	+351.49
			15,000	8.0020×10^{9} (0.0000)	117.78 (0.88)	8.0193×10^{9} (0.0002)	−1.61	123.81 (0.24)	+29.97
			20,000	7.9832×10^{9} (0.0001)	118.33 (0.30)	8.0872×10^{9} (0.0001)	−2.44	121.84 (0.28)	+32.07
$k=15$	5.2791×10^{9} (0.0000)	339.77 (0.84)	5000	5.4366×10^{9} (0.0001)	76.42 (0.58)	5.2758×10^{9} (0.0000)	+0.06	84.68 (0.11)	+301.24
			10,000	5.3832×10^{9} (0.0000)	237.93 (0.55)	5.3325×10^{9} (0.0002)	−1.00	272.66 (0.42)	+24.61
			15,000	5.3961×10^{9} (0.0000)	298.39 (0.31)	5.4164×10^{9} (0.0001)	−2.53	301.75 (0.46)	+12.60
			20,000	5.3904×10^{9} (0.0000)	313.48 (0.78)	5.3990×10^{9} (0.0001)	−2.22	324.70 (0.39)	+4.64

Table 3 Experimental results on YearPredictionMSD dataset

No. of clusters	k-means		Batch size	Mini Batch k-means		Proposed algorithm			
	f	T		f	T	f	%	T	%
$k=2$	1.0415×10^{9} (0.0000)	44.88 (0.66)	5000	1.0390×10^{9} (0.0000)	17.54 (0.81)	1.0420×10^{9} (0.0000)	−0.05	14.22 (0.10)	+215.61
			10,000	1.0389×109 (0.0000)	18.10 (0.16)	1.0418×10^{9} (0.0000)	−0.03	17.12 (0.16)	+162.15
			15,000	1.0381×10^{9} (0.0000)	17.36 (0.55)	1.0415Ex10^{9} (0.0000)	0.00	17.02 (0.10)	+163.69
			20,000	1.0389×10^{9} (0.0000)	19.47 (1.13)	1.0416×10^{9} (0.0000)	−0.00	16.29 (0.15)	+175.51
$k=3$	9.6949×10^{8} (0.0000)	136.82 (0.21)	5000	9.6523×10^{8} (0.0000)	30.15 (0.62)	9.7059×10^{8} (0.0000)	−0.11	27.12 (0.16)	+404.50
			10,000	9.6445×10^{8} (0.0000)	32.33 (0.40)	9.7097×10^{8} (0.0001)	−0.15	29.61 (0.13)	+362.07
			15,000	9.6491×10^{8} (0.0000)	31.15 (1.28)	9.6967×10^{8} (0.0000)	−0.02	29.55 (0.22)	+363.01
			20,000	9.6352×10^{8} (0.0000)	33.46 (0.75)	9.6969×10^{8} (0.0000)	−0.02	29.33 (0.26)	+366.48
$k=5$	9.0560×10^{8} (0.0005)	327.77 (0.86)	5000	9.0440×10^{8} (0.0000)	55.68 (0.90)	9.1073×10^{8} (0.0002)	−0.57	54.49 (0.31)	+501.52
			10,000	9.0531×10^{8} (0.0000)	62.89 (0.54)	9.0758×10^{8} (0.0001)	−0.22	62.88 (0.30)	+421.26
			15,000	9.0537×10^{8} (0.0000)	68.95 (1.06)	9.0599×10^{8} (0.0001)	−0.04	67.85 (0.48)	+383.08
			20,000	9.0247×10^{8} (0.0000)	80.05 (1.40)	9.0597×10^{8} (0.0001)	−0.04	78.18 (0.37)	+319.25
$k=10$	—	—	5000	8.2782×10^{8} (0.0000)	151.42 (0.66)	9.0079×10^{8} (0.0004)	—	153.60 (0.70)	—
			10,000	8.2783×10^{8} (0.0000)	162.97 (0.67)	8.5568×10^{8} (0.0002)	—	164.21 (0.53)	—
			15,000	8.2696×10^{8} (0.0000)	399.81 (1.09)	8.5352×10^{8} (0.0002)	—	456.02 (0.96)	—
			20,000	8.2832×10^{8} (0.0000)	495.33 (1.01)	8.3611×10^{8} (0.0002)	—	506.99 (0.86)	—
$k=15$	—	—	5000	7.9515×10^{8} (0.0001)	200.46 (0.57)	8.4650×10^{8} (0.0002)	—	218.28 (0.32)	—
			10,000	7.9407×10^{8} (0.0000)	603.09 (0.69)	8.3460×10^{8} (0.0002)	—	733.70 (0.54)	—
			15,000	7.9361×10^{8} (0.0000)	694.57 (1.19)	8.2542×10^{8} (0.0001)	—	707.42 (0.69)	—
			20,000	7.9400×10^{8} (0.0000)	704.57 (1.11)	8.2048×10^{8} (0.0001)	—	795.08 (0.97)	—

Table 4 Experimental results on US Census (1990) dataset

No. of clusters	k-means		Batch size	Mini Batch k-means		Proposed algorithm			
	f	T		f	T	f	%	T	%
$k=2$	5.1331×10^{7} (0.0000)	594.68 (0.75)	5000	5.2134×10^{7} (0.0000)	17.20 (1.30)	5.1330×10^{7} (0.0000)	+0.00	10.61 (0.29)	+5504.90
			10,000	5.2276×10^{7} (0.0000)	10.22 (1.33)	5.1331×10^{7} (0.0000)	0.00	9.86 (0.54)	+5931.24
			15,000	4.9196×10^{7} (0.0000)	9.48 (1.75)	5.1331×10^{7} (0.0000)	0.00	9.39 (0.60)	+6233.12
			20,000	5.2299×10^{7} (0.0000)	9.89 (1.48)	5.1331×10^{7} (0.0000)	0.00	9.39 (0.59)	+6233.12
$k=3$	4.6351×10^{7} (0.0008)	2874.99 (0.56)	5000	4.5407×10^{7} (0.0000)	20.13 (1.11)	5.1085×10^{7} (0.0002)	−9.27	19.65 (0.55)	+14530.99
			10,000	4.5432×10^{7} (0.0000)	24.66 (1.32)	4.9740×10^{7} (0.0001)	−6.81	19.61 (0.79)	+14560.84
			15,000	4.5645×10^{7} (0.0000)	21.46 (0.95)	4.8781×10^{7} (0.0001)	−4.98	18.75 (0.45)	+15233.28
			20,000	4.5590×10^{7} (0.0000)	20.45 (0.82)	5.0697×10^{7} (0.0002)	−8.57	18.84 (0.40)	+15160.03
$k=5$	2.6269×10^{7} (0.0001)	12,877.15 (0.78)	5000	3.7240×10^{7} (0.0001)	70.59 (0.55)	3.4704×10^{7} (0.0001)	−24.31	58.53 (0.22)	+21900.94
			10,000	3.7634×10^{7} (0.0001)	59.47 (0.96)	3.6823×10^{7} (0.0001)	−28.66	58.94 (0.52)	+21747.90
			15,000	3.7738×10^{7} (0.0001)	59.94 (1.05)	3.0490×10^{7} (0.0001)	−13.84	64.65 (0.29)	+19818.25
			20,000	3.7781×10^{7} (0.0001)	71.11 (1.22)	2.9575×10^{7} (0.0001)	−11.18	63.80 (0.80)	+20083.62
$k=10$	—	—	5000	2.5437×10^{7} (0.0001)	188.58 (1.74)	2.2295×10^{7} (0.0000)	—	171.73 (0.20)	—
			10,000	2.5385×10^{7} (0.0001)	180.62 (0.41)	2.4565×10^{7} (0.0001)	—	179.16 (0.19)	—
			15,000	2.5423×10^{7} (0.0001)	511.44 (0.70)	2.6572×10^{7} (0.0001)	—	502.85 (0.24)	—
			20,000	2.5455×10^{7} (0.0001)	512.28 (1.13)	2.2433×10^{7} (0.0000)	—	547.75 (0.51)	—
$k=15$	—	—	5000	2.1666×10^{7} (0.0000)	222.25 (0.60)	2.8379×10^{7} (0.0001)	—	245.00 (0.56)	—
			10,000	2.1639×10^{7} (0.0000)	812.30 (0.71)	2.1128×10^{7} (0.0000)	—	828.94 (0.40)	—
			15,000	2.1633×10^{7} (0.0000)	912.34 (0.63)	2.1257×10^{7} (0.0000)	—	936.91 (0.59)	—
			20,000	2.1631×10^{7} (0.0000)	932.16 (0.91)	2.3052×10^{7} (0.0001)	—	989.64 (0.67)	—

objective function was less than that of k-means and Mini Batch k-means, and with the batch size equal to 15,000, it coincided with the k-means algorithm. At the same time, the relative improvement in the average execution time compared to the second was 183%.

As mentioned above, because of the impossibility of clustering on the whole 'big' dataset with a large number of clusters using the k-means algorithm, the results are carried out in Table 3 only for $k=2$, $k=3$, and $k=5$.

Despite the considerable size of the YearPredictionMSD dataset requiring a large number of memory resources, the proposed approach showed a satisfactory result. For example, with $k=2$ and the size of the batch equal to 15,000, the mean value of the objective function coincided with the k-means.

Evaluation of the proposed approach at the largest of the considered in this paper datasets (US Census (1990) dataset) has proved the applicability of the proposed approach (Table 4). With $k=2$ and different batch sizes, the performance did not deteriorate, and when the batch is equal to 5000, it even surpassed the k-means algorithm. The average clustering time has significantly decreased.

Our approach showed the best results according to the mean value of the objective function in comparison with Mini Batch k-means when $k=2$, 5, 10 (batch size equal to 5000, 10,000, and 20,000) and $k=15$ (batch size equal to 10,000 and 15,000).

The influence of the number of iterations on the values of the objective function and computation time for six datasets was studied to evaluate the performance of the proposed approach.

Figs. 2–4 show the results of the objective function and calculation time with the number of clusters equal to five, depending on the number of iterations for all considered batch sizes.

An increase in the batch size results in a decrease in the value of the objective function and an increase in the computation time at each iteration. The method achieves good performance after six iterations according to the value of the objective function.

An analysis of the proposed k-means based parallel batch clustering for different numbers of computer nodes on three datasets with $k=15$ and batch size equal to 20,000 was considered. The effect of changing in the number of nodes is shown in Fig. 5.

With an increase in the number of nodes, a decrease in the execution time of the proposed algorithm is observed. After four nodes, the reduction in running time becomes less noticeable. Significant improvement is observed for eight nodes. Thus, there is an increase of about two times in the speed of the parallel

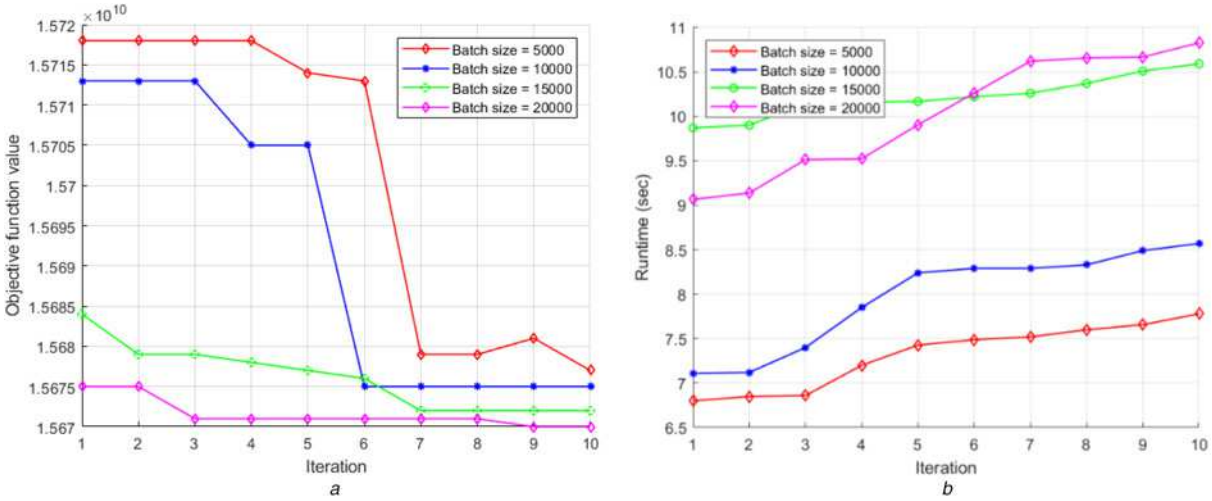

Fig. 2 *Performance of the proposed approach with different number of iterations on Phone Accelerometer dataset*

a Objective function value
b Runtime (seconds)

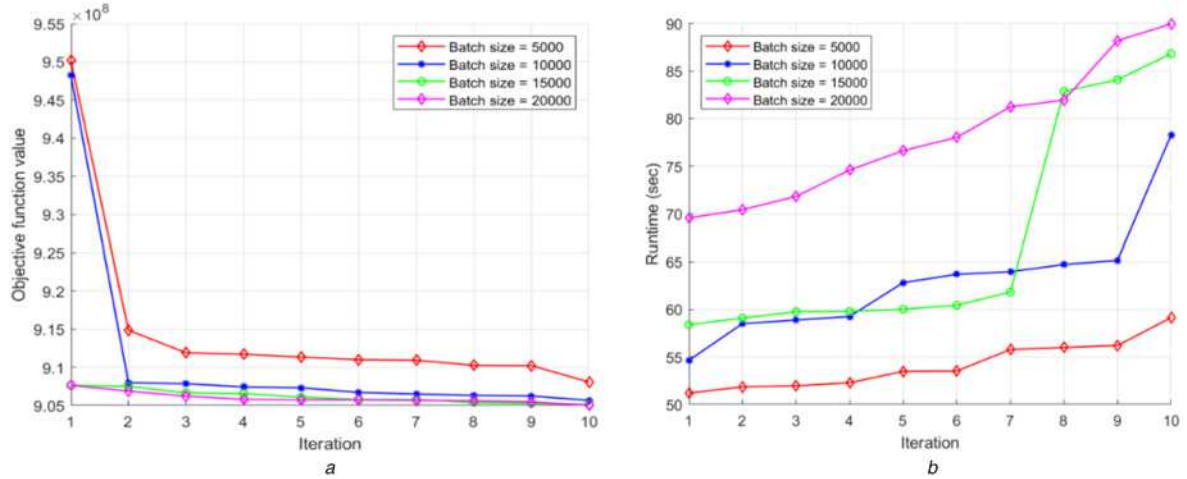

Fig. 3 *Performance of the proposed approach with different number of iterations on YearPredictionMSD dataset*

a Objective function value
b Runtime (seconds)

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 1, pp. 9–14

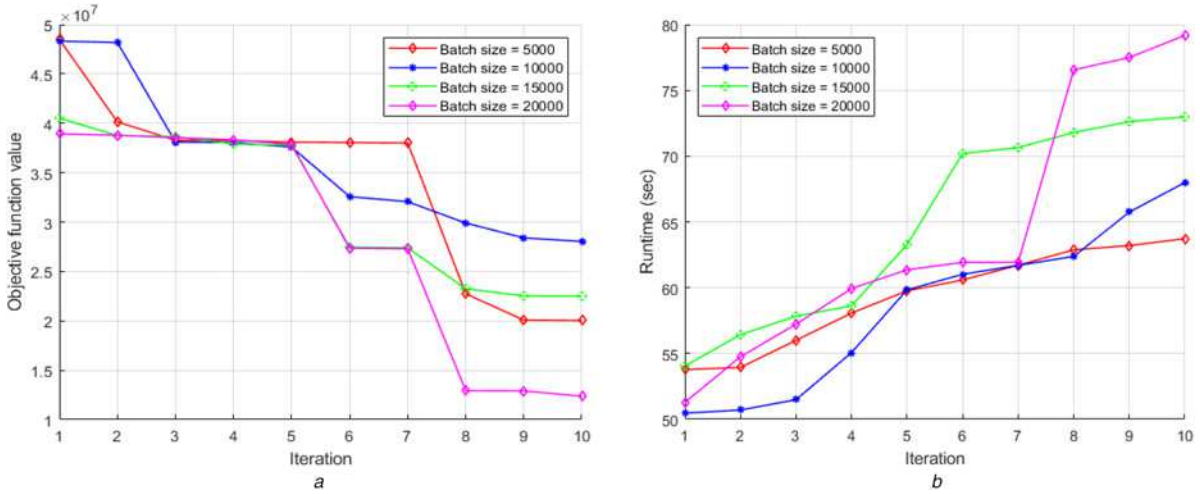

Fig. 4 *Performance of the proposed approach with different number of iterations on US Census (1990) dataset*

a Objective function value
b Runtime (seconds)

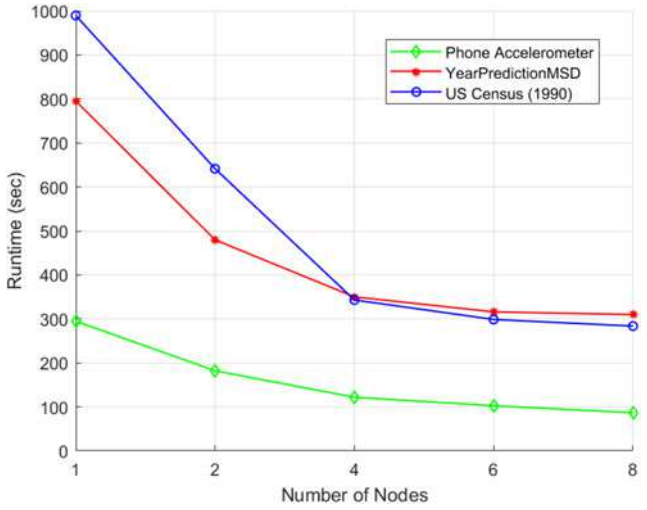

Fig. 5 *Runtime performance of the proposed approach for different number of nodes on the three datasets*

implementation of the proposed approach compared to its sequential version.

The proposed algorithm is useful for big data clustering. The running of the k-means algorithm takes a long time at each iteration, and the running time of the proposed algorithm based on parallel clustering takes only a small percentage of the total time of the first one.

The partitioning data into batches helps reduce the response time of the clustering algorithm and the computational costs, despite the increase in the number of clusters.

Taking into account the results of the experiments, we can conclude that the speed of the proposed algorithm is much higher, compared to the k-means algorithm, which allows us to apply it for analysis of large-dimensional data. Unlike the k-means, the proposed approach showed good results on real datasets according to the speed and quality of clustering.

5 Conclusion

Analysis of large-scale datasets requires the presence of large computing powers, which is not always feasible. And so it became necessary to develop new clustering algorithms capable of such data processing based on batches using their parallel clustering. This approach was proposed in the paper.

Three datasets were used in this study for comparison to explore the performance of the proposed approach, including Phone Accelerometer, YearPredictionMSD, and US Census (1990) datasets. The proposed approach was compared with k-means and Mini Batch k-means algorithm. Based on the results of the experiments, the computational efficiency of the proposed approach based on the batch in comparison with the k-means was proved. Analysing the effect of the size of the batch on the mean value of the function and the mean running time of the algorithm, we can conclude that efficiency can be achieved even with small batch size. Despite the increase in the number of clusters, the speed of the proposed algorithm is significantly higher compared to the k-means. The experimental results showed that the proposed algorithm is a promising tool for clustering large sets of data in comparison with the k-means algorithm.

6 References

[1] Aggarwal, C.C., Reddy, C.K.: 'Data clustering: algorithms and applications' (CRC Press, New York, 2014)

[2] Shirkhorshidi, A.S., Aghabozorgi, S., Wah, T.Y., *et al.*: 'Big data clustering: a review'. Proc. Int. Conf. Computational Science and its Applications, Portugal, June–July 2014, pp. 707–720

[3] Karmitsa, N., Bagirov, A.M., Taheri, S.: 'New diagonal bundle method for clustering problems in large data sets', *Eur. J. Oper. Res.*, 2017, **263**, (2), pp. 367–379

[4] Karmitsa, N., Bagirov, A.M., Taheri, S.: 'Clustering in large data sets with the limited memory bundle method', *Pattern Recognit.*, 2018, **83**, pp. 245–249

[5] Alguliyev, R., Aliguliyev, R., Sukhostat, L.: 'Anomaly detection in big data based on clustering', *Stat. Optim. Inf. Comput.*, 2017, **5**, (4), pp. 325–340

[6] Alguliyev, R., Aliguliyev, R., Imamverdiyev, Y., *et al.*: 'An anomaly detection based on optimization', *Int. J. Intell. Syst. Appl. Eng.*, 2017, **9**, (12), pp. 87–96

[7] Alguliyev, R., Aliguliyev, R., Imamverdiyev, Y., *et al.*: 'Weighted clustering for anomaly detection in big data', *Stat. Optim. Inf. Comput.*, 2018, **6**, (2), pp. 178–188

[8] Alguliyev, R., Aliguliyev, R., Imamverdiyev, Y., *et al.*: 'An improved ensemble approach for DoS attacks detection', *Radio Electronics Comp. Sci. Control*, 2018, **45**, (2), pp. 73–82

[9] Boutsidis, C., Drineas, P., Mahoney, M.W.: 'Unsupervised feature selection for the k-means clustering problem'. Proc. 22nd Int. Conf. on Neural Information Processing Systems (NIPS), Vancouver, Canada, December 2009, pp. 153–161

[10] Jain, A.K.: 'Data clustering: 50 years beyond k-means', *Pattern Recognit. Lett.*, 2010, **31**, (8), pp. 651–666

[11] Sculley, D.: 'Web-scale k-means clustering'. Proc. Int. Conf. World Wide Web, North Carolina, USA, April 2010, pp. 1177–1178

[12] Alguliyev, R., Aliguliyev, R., Karimov, R., *et al.*: 'Batch clustering algorithm for big data sets'. Proc. Int. Conf. Application of Information and Communication Technologies, Baku, Azerbaijan, October 2016, pp. 79–82

[13] Jie, S., Zhongyi, M., Yichuan, Z., *et al.*: 'Rim: a reusable iterative model for big data', *Knowl.-Based Syst.*, 2018, **153**, pp. 105–116

[14] Aaron, C., Cholaquidis, A., Fraiman, R., *et al.*: 'Multivariate and functional robust fusion methods for structured Big Data', *J. Multivar. Anal.*, 2019, **170**, pp. 149–161
[15] Meng, Y., Liang, J., Cao, F., *et al.*: 'A new distance with derivative information for functional k-means clustering algorithm', *Inf. Sci.*, 2018, **463-464**, pp. 166–185
[16] Ismkhan, H.: 'I-k-means−+: an iterative clustering algorithm based on an enhanced version of the k-means', *Pattern Recognit.*, 2018, **79**, pp. 402–413
[17] Zhao, W.L., Deng, C.H., Ngo, C.W.: 'k-means: a revisit', *Neurocomputing*, 2018, **291**, pp. 195–206
[18] Tang, R., Fong, S.: 'Clustering big IoT data by metaheuristic optimized mini-batch and parallel partition-based DGC in Hadoop', *Future Gener. Comput. Syst.*, 2018, **86**, pp. 1395–1412
[19] Tang, R., Fong, S., Yang, X.-S., *et al.*: 'Integrating nature-inspired optimization algorithms to k-means clustering'. Proc. Int. Conf. Digital Information Management, Macau, China, August 2012, pp. 116–123
[20] Kantabutra, S., Couch, A.L.: 'Parallel k-means clustering algorithm on NOWs', *NECTEC Technical J.*, 2000, **1**, (6), pp. 243–247
[21] Kraus, J., Kestler, H.: 'A highly efficient multi-core algorithm for clustering extremely large data sets', *BMC Bioinformatics*, 2010, **11**, (169), pp. 1–16
[22] Zhang, G., Zhang, C., Zhang, H.: 'Improved k-means algorithm based on density Canopy', *Knowl.-Based Syst.*, 2018, **145**, pp. 289–297
[23] Hussain, S.F., Haris, M.: 'A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data', *Expert Syst. Appl.*, 2019, **118**, pp. 20–34
[24] Bahmani, B., Moseley, B., Vattani, A., *et al.*: 'Scalable k-means++'. *Proc. VLDB Endowment*, 2012, **5**, (7), pp. 622–633
[25] Ghesmoune, M., Lebbah, M., Azzag, H.: 'Micro-batching growing neural gas for clustering data streams using spark streaming', *Procedia Comput. Sci.*, 2015, **53**, pp. 158–166
[26] Cuomo, S, De Angelis, V., Farina, G., *et al.*: 'A GPU-accelerated parallel k-means algorithm', *Comput. Electr. Eng.*, 2019, **75**, pp. 262–274
[27] Othman, F., Abdullah, R., Rashid, N.A., *et al.*: 'Parallel k-means clustering algorithm on DNA dataset'. 'Proc. 5th Int. Conf. Parallel and Distributed Computing: Applications and Technologies, Singapore, December 8–10, 2004, LNCS, **3320**, pp. 248–251
[28] Parker, J.K., Hall, L.O.: 'Accelerating fuzzy-c means using an estimated subsample size', *IEEE Trans. Fuzzy Syst.*, 2014, **22**, (5), pp. 1229–1244
[29] Thompson, S.K.: 'Sample size for estimating multinomial proportions', *Am. Stat.*, 1987, **41**, (1), pp. 42–46
[30] 'UCI Machine Learning Repository', available at http://archive.ics.uci.edu/ml, accessed 7 September 2019
[31] Meek, C., Thiesson, B., Heckerman, D.: 'The learning-curve sampling method applied to model-based clustering', *Journal of Machine Learning Research*, 2002, **2**, pp. 397–418
[32] Chen, Z., Gehrke, J., Korn, F.: 'Query optimization in compressed database systems'. Proc. Int. Conf. Management of Data, California, USA, May 2001, pp. 271–282
[33] Giannella, C., Sayrafi, B.: 'An information theoretic histogram for single dimensional selectivity estimation'. Proc. ACM Symp. on Applied computing, Santa Fe, New Mexico, March 2005, pp. 676–677
[34] Bertin-Mahieux, T., Ellis, D.P.W., Whitman, B., *et al.*: 'The million song dataset'. Proc. Int. Society for Music Information Retrieval Conf., Florida, USA, October 2011, pp. 591–596
[35] Stisen, A., Blunck, H., Bhattacharya, S., *et al.*: 'Smart devices are different: assessing and mitigating mobile sensing heterogeneities for activity recognition'. Proc. Int. Conf. Embedded Networked Sensor Systems, Seoul, South Korea, November 2015, pp. 127–140