KYBERNETIKA — VOLUME *61* (2025), NUMBER 4, PAGES 492–508

IBK-MEANS: AN ITERATIVE BATCH *K*-MEANS ALGORITHM FOR BIG DATA CLUSTERING

Rasim Alguliyev, Ramiz Aliguliyev, Adil M. Bagirov, and Mustafa Aliyev

Information technologies such as social media, mobile computing, and the realization of the industrial Internet of Things (IoT) produce huge amounts of data every day. The development of powerful tools for knowledge-discovery is imperative to deal with such a volume of data. Clustering methods are among the most important knowledge-discovery techniques. The growth in computational power and algorithmic developments allow us to efficiently and accurately solve clustering problems in large datasets. However, these developments are insufficient to deal with clustering problems in big datasets. This is because these datasets cannot be processed as a whole due to hardware and computational restrictions. In this paper an iterative batch k-means (*ibk*-means) algorithm is proposed that yields good clustering results with low computation costs on big datasets. It is designed to cluster datasets using batch data. The efficiency and accuracy of the proposed algorithm are investigated depending on the size of batches, the number of attributes and clusters. The algorithm is compared with the classic k-means and mini batch k-means algorithms using computational results on several real-world datasets, all of which are available from the UCI Machine Learning Repository. The smallest dataset has 500000 data points and 2 attributes and the largest one contains 43930257 data points and 16 attributes. Results demonstrated that the *ibk*-means algorithm outperforms both the k-means and mini batch k-means algorithms in the sense of both efficiency and accuracy and it is applicable for the clustering of big datasets. The proposed algorithm provides real time clustering and may have direct applications in expert and intelligent systems. Furthermore, results from this paper will have a clear impact in the sense of designing more accurate and efficient clustering algorithms for big datasets taking into account available computer resources.

Keywords: big data, cluster analysis, k-means algorithm, batch clustering, mini batch k-means

Classification: 68T09, 90B99

1. INTRODUCTION

With the explosive growth in the number of devices connected to the Internet of Things (IoT) and rapid development of diversified data sources, such as scientific experiments, and online social networks, the volume of data increases at an exponential rate [22, 34]. Significant problems arise during the collection, analysis, and visualization of such data. The current techniques and technologies are not always able to handle storage and

DOI: 10.14736/kyb-2025-4-0492

efficient processing of such an amount of data. The development of efficient and accurate tools for knowledge discovery is imperative to deal with such a volume of data, which is the subject of big data analytics.

Big data has become a strategic asset for organizations, industries, businesses, and for the security of a nation. It is one of the most promising frontiers for innovative research and development in computer science, industry, and business [21,49]. The number of IoT devices has increased several times since 2016 [2] and the the global big data analytics market is growing significantly [49]. The popularity of IoT has made big data analytics challenging [42].

There is no exact definition of big data; however, it is accepted that this type of data has the following characteristics [22]:

- **Volume**: really big (although the size depends on the resources available for processing them).
- **Variety**: poorly structured and heterogeneous.
- **Velocity**: processing should be very fast (and the results are often needed quickly if it is about online services).

It is assumed that big data cannot be stored in the RAM of a computer. For example, datasets containing $10^{\geq 12}$ objects cannot be stored in most current computers. Furthermore, a dataset containing 10^{12} objects with 10 attributes, stored in short integer (4 bytes) format, would require 40 TB of storage. Storage and analysis of big data will continue to be a problem [33, 49].

Big data analytics is a series of approaches, tools, and methods for processing structured and unstructured data of huge volumes and significant diversity to obtain human-readable results that are effective under conditions of continuous growth, distribution over numerous nodes of the computer network. The goal of big data analytics is to extract knowledge from the huge volume of data by using data mining techniques to make predictions, identify recent trends, find hidden information, and make decisions. Thus, under big data, we do not understand any specific amount of data, or even the data itself, but the methods for processing them. For discovering valuable knowledge from a huge volume of data, we need either to improve existing techniques and technologies or to develop new methods and technologies to analyze big data [2, 22].

In this paper, a new algorithm is introduced for clustering big datasets using batch data, which is effective in the limited memory space and computational resources. The proposed algorithm is based on the k-means algorithm and can be considered as a new version of k-means for big datasets. It offers a new approach for clustering large datasets. In this algorithm, the batch size can be adjusted according to hardware limitations. The algorithm is called an iterative batch k-means algorithm (*ibk*-means). Numerical experiments on very large datasets demonstrate that the *ibk*-means algorithm is more efficient than the classic k-means and mini-batch k-means algorithms. Note that this work is an extended version of the paper [4].

The main contributions of this paper are as follows:

- The new *ibk*-means algorithm for big data clustering is developed.

- The performance of the proposed algorithm is compared with that of the classic k-means [40] and mini-batch k-means (mbk-means) [47] algorithms using seven real-world datasets from [24]. The smallest dataset has 500,000 data points with 2 attributes, and the largest one has 43,930,257 data points and 16 attributes.

The rest of this paper is structured as follows. Section 2 presents an overview of the related work. The new ibk-means algorithm is discussed in Section 3. Numerical results and the comparison of algorithms are reported in Section 4. Section 5 contains concluding remarks.

2. RELATED WORK

One of the tools to process big data with unlabeled elements (aka unsupervised learning) is data clustering. The main task of cluster analysis is to partition a set of data points into tightly organized groups based on a suitable similarity measure. There are different types of clustering [1, 11, 29, 45, 51] and in this paper, we consider the partition clustering problem in big datasets. In general, clustering algorithms for big datasets can be classified into three groups [26]:

1. **Sampling methods.** These methods select a small subset of the dataset and then the clustering is executed on this subset. Since the whole dataset is not processed, these methods can speed up the execution time; however, the clustering precision may be considerably affected.

2. **Data transformation methods.** These methods alter the structure of the data so that it can be clustered more efficiently.

3. **Single-pass methods.** These methods divide the data into batches and perform clustering on each batch, then combine the clustering results. These methods are also classified into two classes: (i) Incremental methods; (ii) Divide and conquer methods.

Parallelization of clustering algorithms is considered as one of the directions to develop efficient algorithms for big datasets. Such methods can be classified into two groups [48]:

1. **Single-machine clustering methods:** For example, CLARANS, BIRCH, CURE.

2. **Multiple-machine clustering methods:** For example, ParMETIS, GPU-based methods, MapReduce based on k-means, MapReduce based on GPU.

The k-means [40], for its good time performance, is one of the widely used clustering algorithms. However, it was designed for solving single-view data clustering problems and therefore, with the increasing size of the datasets being analyzed, this algorithm loses its effectiveness as it requires the entire dataset to be stored in the main memory [15]. For this reason, several methods were proposed to improve the performance of the k-means algorithm. For example, [19] proposed a new robust large-scale multi-view k-means algorithm which can be easily parallelized and performed on multi-core processors. The computational complexity of this method is similar to that of the classical k-means algorithm. Algorithms based on nonsmooth optimization techniques were proposed in

[12,13,37,38] to solve the minimum sum-of-squares clustering problems in large datasets where the new clustering algorithms is designed by modifying the limited memory bundle method of nonsmooth optimization and combining it with an incremental approach.

There are several papers on the comparative assessment of the clustering algorithms for big datasets [48, 53]. The paper [25] provides an analysis of the existing clustering algorithms for big datasets and presents a framework for classification of the clustering algorithms to guide the selection of algorithms for big data. A detailed review of density-based clustering methods, their advantages and disadvantages, and also their performance comparison for different big datasets are presented in [16].

In [23], MapReduce was modified by proposing a novel processing model to eliminate the iteration dependence and obtain high performance. Experiments demonstrated that this algorithm is efficient, robust, and scalable. Another similar approach with a good initial seeding was proposed in [46]. First, they describe the existing clustering algorithms: the serial k-means++ and the competitive k-means. Then, they propose a new algorithm called DisK-Means based on the above two algorithms. This algorithm divides the complete dataset into parts, where each part must have more than the minimum sample size to be representative of the dataset.

A parallel version of the k-means++ algorithm is much faster than existing parallel versions of the k-means algorithms. The k-means++ algorithm obtains a nearly optimal solution after a logarithmic number of passes [14]. A mini-batch k-means based algorithm with low computation cost on large datasets was presented in [47]. This algorithm reduces the computational cost and shows the best results compared to the stochastic gradient descent algorithm. The parallel versions of the k-medoids and k-means algorithms for big data clustering is developed in [8] and the parallel batch k-means for big data clustering is introduced in [9]. In [43], it was shown that the triangle inequality-based bounding can be used to accelerate the mini-batch k-means algorithm.

A version of the k-means algorithm that leads to higher speed and considerably lower clustering distortion was presented in [55]. In this paper, a clustering objective function that is feasible for the whole ℓ_2-space was developed. The method is implemented by top-down bisecting. The superior performance over most of the algorithms based on the k-means was observed across different scenarios.

The paper [30] proposes the $I-k$-means$-+$ algorithm. This is an iterative approach to improve the quality of clustering by the k-means via removing one cluster (minus), dividing another one (plus), and applying re-clustering again, at each iteration. The algorithm can be applied to datasets with a large number of data points or a large number of clusters. Although this process is time-consuming, the use of some heuristics effectively speeds up $I-k$-means^{-+}. The robust fusion method [20] could be addressed to some fundamental statistical problems in the context of big data.

Recently, a high-performance analysis of IoT big data has been a promising research direction in big data analytics. Tsai et al. [52] presented an efficient framework for a metaheuristic algorithm on a cloud computing environment for clustering big datasets. Zhang et al. [54] proposed two high-order possibilistic c-means algorithms, called CP-HOPCM and TT-HOPCM, for clustering IoT big data with a large number of attributes which can be used in IoT systems with limited memory space and computing power.

Ilango et al. [35] proposed the artificial bee colony optimization algorithm for cluster-

ing in big data to reduce the execution time. For improving the scalability and precision of FCM for big data, authors of the work [26] proposed scalable FCM on Hadoop, called BigFCM. Havens and Bezdek [32] presented an efficient formulation of the iVAT (an improved visual assessment of cluster tendency) algorithm which significantly reduces the computational complexity of the iVAT algorithm from $O(n^2)$ to $O(n \log n)$, where n is the number of objects to be clustered. Kumar et al. [39] presented a new clusiVAT algorithm, based on sampling the data. Havens, Bezdek and Palaniswami [33] proposed an extension to the scalable VAT (sVAT) algorithm [31], called sVAT-S, whose computational complexity is significantly smaller than $O(n^2)$.

The paper [27] proposed an algorithm that combines parallel clustering with single-pass, stream-clustering algorithms. This approach has two key advantages. First, while the dataset may not fit in the memory, one can still keep its portion in the memory and cluster them before reading the rest of the data. The second advantage is that multiple chunks can be clustered in parallel using multiple CPU cores utilizing their maximum computational capability to cluster the dataset as quickly as possible. The limited network bandwidth, limited storage, and low computational capacities of a single machine do not allow merging big datasets generated across multiple distributed sources. To address these problems, [50] proposed a three-stage distributed clustering scheme using boundary information.

Clustering algorithms for big datasets were applied in various areas where such data are available. An improved DBSCAN clustering method was proposed in [28] for predicting drilling overflow accidents. For intrusion detection systems over big data environments, [44] proposed a clustering method combining mini-batch k-means with the principal component analysis. For anomaly and DoS attack detection over big data environments in [3,5,6,36], a clustering, an optimization, and deep learning methods were proposed, respectively.

Analysis of algorithms introduced in the above-mentioned papers shows that most existing algorithms for clustering of big datasets are extensions or significant modifications of conventional clustering algorithms such as the k-means algorithm, the fuzzy c-means algorithm, and hierarchical clustering algorithms. In all cases, the main aim is to reduce the execution time using parallelization techniques or using partitioning of the whole dataset into smaller groups and then defining representatives of each group. The performance of these algorithms is reported using small to large datasets, and only a few of them use datasets containing millions or tens of millions of data points. Most existing clustering algorithms for big datasets are based on the so-called distributed clustering schemes. In this paper, we propose a rather different approach to modify the k-means algorithm for clustering big datasets using a single machine. The proposed algorithm uses batch data and is based on the iterative clustering scheme.

3. K-MEANS AND THE PROPOSED IBK-MEANS ALGORITHMS

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of finite number of points given in an m-dimensional space. Assume that this set should be clustered into k clusters. Denote the collection of these clusters as $C = \{C_1, C_2, \ldots, C_k\}$. The k-means algorithm seeks a partition having an overall minimum squared error between the centers of the clusters and the

data points in the clusters. The center o_q of cluster C_q is calculated as follows:

$$o_q = \frac{\sum\limits_{x_i \in C_q} x_i}{|C_q|}, \quad q = 1, 2, \ldots, k, \tag{1}$$

where $|C_q|$ is the number of data points in the cluster C_q. For the cluster C_q, the squared error is defined as:

$$f_q = \sum_{x_i \in C_q} \|x_i - o_q\|^2, \quad q = 1, 2, \ldots, k. \tag{2}$$

Here $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^m$. The goal of the k-means algorithm is to minimize the sum of squared errors for all clusters, defined as:

$$f = \sum_{q=1}^{k} f_q = \sum_{q=1}^{k} \sum_{x_i \in C_q} \|x_i - o_q\|^2. \tag{3}$$

As stated in [10], minimizing the objective function (3) is an NP-hard problem. In this study, we have used the classic k-means algorithm [41] which uses an iterative refinement technique [40]. The pseudocode for the k-means algorithm is given below:

THE K-MEANS CLUSTERING ALGORITHM

Input:

- $X = \{x_1, x_2, \ldots, x_n\}$ // Set of data points.
- k // Number of desired clusters.

Output:

- A set of k clusters: $C = \{C_1, C_2, \ldots, C_k\}$.

Steps:

1. Sample k data points uniformly at random from X as initial centroids.
2. **Repeat**
   (a) Assign each point x_i to the cluster which has the closest centroid.
   (b) Calculate the new centroid for each cluster.
3. **Until** no data point changes its cluster.

The main drawback of the k-means algorithm is that the quality of the final clusters heavily depends on the initial centroids. The k-means algorithm produces different clusters for different initial centroids. The time complexity of this algorithm is $O(n \times k \times m \times t)$, where:

- n is the number of m-dimensional vectors to be clustered,

- k is the number of clusters,
- t is the number of iterations needed until convergence.

Lloyd's k-means algorithm is efficient for real-world clustering tasks, in other words, for clustering datasets that can be stored in the memory of a single machine. However, this algorithm is slow for large datasets, and its accuracy deteriorates as the number of data points increases.

As mentioned above, clustering of big datasets can be a problem due to memory space and computational restrictions. To address both these problems, we propose a new version of the k-means algorithm by combining it with batching of the dataset. The algorithm takes some batch of data from the given dataset and processes it. Then, it takes the next batch along with centroids from the previously processed batch and processes it, and so on until all the elements of the dataset are processed.

This algorithm is called the **iterative batch k-means (*ibk*-means)** algorithm and proceeds as follows:

1. Divide the given dataset into non-overlapping subsets (batches) with equal size p ($p < n$). The maximum value of p is determined by the capabilities of the personnel computer (PC), ensuring that each batch can be processed within RAM and in a reasonable time.
2. Take the first p data elements from the given dataset of n elements.
3. Apply the k-means algorithm to find k centroids ($k < p$) in this batch. If all batches are processed, proceed to Step 5.
4. Use the k centroids as the starting cluster centers for the next batch of p new data elements from the remaining dataset and return to Step 3.
5. Assign elements of the whole dataset to the k centroids calculated in the last iteration to obtain the k-partition of the dataset.

It can be seen from the description that the main tool in solving the clustering problem in the proposed algorithm is the k-means algorithm. The use of batches improves the accuracy of k-means since the number of data points in these batches is significantly smaller than in the whole dataset. Moreover, in all batches except the first one, the starting cluster centers are derived from the previous iteration of the k-means algorithm. This is another advantage of using batches in clustering large datasets.

The time complexity of the *ibk*-means algorithm is similar to that of the k-means algorithm and can be described as $O((n \times k \times m \times t)n_b)$, where:

- n_b is the number of batches,
- n is the number of m-dimensional vectors in each batch,
- k is the number of clusters,
- t is the maximum number of iterations over all batches.

4. COMPUTATIONAL RESULTS

In this section, we compare the *ibk*-means algorithm against both the classic k-means [40] and *mbk*-means [47] algorithms using numerical results. In the comparison, we use indicators such as the CPU time (T) required to compute centroids and assigning data points to centroids; and the value of the objective function (f).

The parameters of the PC on which the experiments were conducted are as follows:

- CPU: Core i7 2.2 GHz.
- RAM: 8 GB.
- Video Card: 1 GB.
- HDD: 1 TB.
- Operating System: Microsoft Windows 7 Ultimate 64-bit.

The whole datasets are also clustered using the k-means algorithm for efficiency and quality comparison with the proposed algorithm using the above-mentioned indicators. Due to the memory restriction of the above hardware, it is sometimes impossible to process a whole dataset for comparison purposes.

4.1. Datasets

In the numerical experiments, we use real-world datasets, all of which are available from the UCI Machine Learning Repository [24]. The brief description of the datasets is given in Table 1.

#	Dataset	# Attributes	# Instances	Attribute Types	Year	Source
1.	D500×2	2	500000	Integer, Real	2015	Online Retail
2.	D500×3	3	–	–	–	–
3.	D500×5	5	–	–	–	–
4.	D2000×3	3	2000000	Real	2012	Individual Household Electric Consumption
5.	D2000×5	5	–	–	–	–
6.	D2000×9	9	–	–	–	–
7.	D44000×16	16	43930257	Real	2015	Heterogeneity Activity Recognition

Tab. 1. The brief description of datasets.

The datasets D500×2, D500×3, and D500×5 are extracted from the Online Retail dataset, which contains 541,909 data points and 8 attributes. The Online Retail dataset contains records of transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based online retail company specializing in unique all-occasion gifts, with many wholesale customers.

The datasets D2000×3, D2000×5, and D2000×9 are extracted from the Individual Household Electric Power Consumption dataset, which contains 2,075,259 data points and 9 attributes. This dataset consists of measurements gathered in a house located in Sceaux, France, between December 2006 and November 2010 (47 months).

Finally, the dataset D44000×16 is from the Heterogeneity Activity Recognition dataset, containing 43,930,257 data points and 16 attributes. This dataset consists of two datasets designed to investigate the impact of sensor heterogeneities on human activity recognition algorithms, including classification, automatic data segmentation, sensor fusion, and feature extraction.

4.2. Results

Results of numerical experiments are presented in Tables 2 – 8. We used batches with 10000, 50000 and 250000 data points for the largest dataset D44000×16. This dataset cannot be stored in the RAM of the computer and therefore, we did not get results using the k-means algorithm and report only results for the mbk and ibk algorithms. To provide a proper evaluation of results, in the tables we include columns showing improvements by the ibk algorithm in comparison with other two algorithms for both the objective function value and CPU time. Positive numbers show improvement by the ibk algorithm and negative numbers show deterioration of results of other algorithms.

# of clus.	k-means		Batch size(p)	mbk		ibk		Relat. improv. of ibk over k-means (%)		Relat. improv. of ibk over mbk (%)	
	f	T		f	T	f	T	f	T	f	T
2	1452.7	16.5	10000	1459.6	17.2	1459.6	17.7	-0.45	-7.27	+0.03	-2.91
			50000	1457.8	18.7	1453.3	16.8	-0.04	-1.82	+0.31	+10.16
			250000	1455.2	19.2	1450.5	15.4	+0.15	+6.67	+0.32	+19.79
5	494.6	18.2	10000	499.3	18.7	501.2	19.6	-1.33	-7.69	-0.38	-4.81
			50000	502.8	19.2	497.5	18.7	-0.59	-2.75	+1.05	+2.60
			250000	504.6	20.1	492.7	17.5	+0.38	+3.85	+2.36	+12.94
10	247.6	20.4	10000	252.5	20.6	253.9	22.5	-2.54	-10.29	-0.55	-9.22
			50000	254.1	21.3	249.4	21.5	-0.73	-5.39	+1.85	-0.94
			250000	253.5	22.1	245.7	19.9	+0.77	+2.45	+3.08	+9.95

Tab. 2. Clustering results on D500×2 dataset.

4.3. Discussion of experimental results

The analysis of the experimental results shows that:

- As the batch size increases, the performance of the ibk-means algorithm improves across all datasets with respect to both indicators: CPU time (T) and clustering quality (the value of the objective function f). For example, for the D500×2 dataset, if the size of the batch increases from 10000 to 250000, then the CPU time decreases from 17.7 seconds to 15.4 seconds, and the value of f from 1459.2 to 1450.5. In other words, with the increase of the batch size, the performance of the ibk-means algorithm improves.

- It is clear from Table 2 that when the batch size is 250000, the ibk-means algorithm outperforms the k-means algorithm with respect to both indicators. From

# of clus.	k-means		Batch size(p)	*mbk*		*ibk*		Relat. improv. of *ibk* over k-means (%)		Relat. improv. of *ibk* over *mbk* (%)	
	f	T		f	T	f	T	f	T	f	T
2	2102.2	19.9	10000	2104.6	20.5	2107.6	21.1	-0.26	-6.03	-0.14	-2.93
			50000	2102.4	21.7	2105.6	20.5	-0.16	-3.02	-0.15	+5.53
			250000	2105.7	22.1	2101.8	19.6	+0.02	+1.51	+0.19	+11.31
5	709.0	23.4	10000	710.0	23.9	714.8	25.6	-0.82	-9.40	-0.68	-7.11
			50000	709.3	24.7	711.6	24.4	-0.37	-4.27	-0.32	+1.21
			250000	709.2	25.5	708.5	23.1	+0.07	+1.28	+0.10	+9.41
10	352.3	26.5	10000	359.2	27.4	357.6	28.8	-1.50	-8.68	+0.45	-5.11
			50000	357.3	28.2	354.4	27.7	-0.60	-4.53	+0.81	+1.77
			250000	356.5	29.7	351.4	26.4	+0.26	+0.38	+1.43	+11.11

Tab. 3. Clustering results on D500×3 dataset.

# of clus.	k-means		Batch size(p)	*mbk*		*ibk*		Relat. improv. of *ibk* over k-means (%)		Relat. improv. of *ibk* over *mbk* (%)	
	f	T		f	T	f	T	f	T	f	T
2	3022.1	27.9	10000	3027.3	28.6	3028.5	29.3	-0.21	-5.02	-0.04	-2.45
			50000	3028.7	28.9	3025.6	28.8	-0.12	-3.23	+0.10	+0.35
			250000	3030.6	29.5	3021.8	27.5	+0.01	+1.43	+0.29	+6.78
5	1211.0	34.7	10000	1215.8	35.2	1217.3	36.3	-0.52	-4.61	-0.12	-3.12
			50000	1214.1	36.4	1214.4	35.6	-0.28	-2.59	-0.02	+2.20
			250000	1213.5	37.4	1210.3	34.3	+0.06	+1.15	+0.26	+8.29
10	700.6	39.1	10000	702.8	38.2	705.3	40.6	-0.67	-3.84	-0.36	-6.28
			50000	707.4	39.1	703.4	39.4	-0.40	-0.77	+0.57	-0.77
			250000	709.1	41.3	699.9	38.7	+0.10	+1.02	+1.30	+6.30

Tab. 4. Clustering results on D500×5 dataset.

Tables 2 – 7, it is easy to see that this observation is true for all datasets and the number of clusters. Furthermore, for the batch sizes 10000 and 50000 the difference between the clustering function values obtained by these two algorithms is insignificant across all datasets. However, the k-means algorithm cannot be applied to very large data sets containing millions of data points and the *ibk* algorithm can be considered as an extension of k-means for clustering such datasets.

- The size of the batch cannot be increased arbitrarily due to the following reasons:

  1. The batch size should be such that it can be efficiently processed on the computer, i e., the RAM and computational power of the computer should allow it.
  2. The batch size strongly depends on the capabilities of the selected clustering algorithm. It is known that the efficiency and accuracy of clustering algorithms vary depending on the size of the dataset.

# of clus.	k-means		Batch size(p)	mbk		ibk		Relat. improv. of ibk over k-means (%)		Relat. improv. of ibk over mbk (%)	
	f	T		f	T	f	T	f	T	f	T
2	8698.4	77.7	50000	8709.1	76.3	8705.4	80.2	-0.08	-3.22	+0.04	-5.11
			250000	8717.0	77.6	8701.7	78.8	-0.04	-1.42	+0.18	-1.55
			500000	8715.5	80.6	8697.7	77.1	+0.01	+0.77	+0.20	+4.34
5	3512.4	94.1	50000	3529.3	94.9	3517.5	97.3	-0.15	-3.40	+0.33	-2.53
			250000	3528.7	95.3	3515.7	95.6	-0.09	-1.59	+0.37	-0.31
			500000	3524.5	96.1	3511.8	93.2	+0.02	+0.96	+0.36	+3.02
10	1869.3	111.2	50000	1906.0	111.2	1873.2	115.9	-0.21	-4.23	+1.72	-4.23
			250000	1895.9	114.4	1871.7	113.4	-0.13	-1.98	+1.28	+0.87
			500000	1891.4	115.8	1868.6	110.5	+0.04	+0.63	+1.21	+4.58

Tab. 5. Clustering results on D2000×3 dataset.

# of clus.	k-means		Batch size(p)	mbk		ibk		Relat. improv. of ibk over k-means (%)		Relat. improv. of ibk over mbk (%)	
	f	T		f	T	f	T	f	T	f	T
2	12653.6	109.1	50000	12653.8	110.2	12657.3	112.6	-0.03	-3.21	-0.03	-2.18
			250000	12653.6	113.4	12654.7	110.5	-0.01	-1.28	-0.01	+2.56
			500000	12653.4	114.7	12652.9	108.5	+0.01	+0.55	0.00	+5.41
5	5902.3	137.2	50000	5909.2	137.7	5905.4	140.9	-0.05	-2.70	+0.06	-2.32
			250000	5911.3	139.2	5903.6	138.2	-0.02	-0.73	+0.13	+0.72
			500000	5909.6	140.1	5901.8	137.0	+0.01	+0.15	+0.13	+2.21
10	3115.9	153.4	50000	3122.7	155.7	3118.6	156.5	-0.09	-2.02	+0.13	-0.51
			250000	3121.7	157.9	3116.5	154.1	-0.02	-0.46	+0.17	+2.41
			500000	3123.4	158.6	3115.0	152.9	+0.03	+0.33	+0.27	+3.59

Tab. 6. Clustering results for D2000×5 dataset.

- Results presented in the last column of Tables 2 – 7 (the relative improvement) lead to some interesting observations. The increase in the batch size reduces the CPU time required by the *ibk*-means algorithm. Preliminary analysis (see Tables 2 – 7) shows that this is due to the fact that, with the decrease in the number of batches (if the batch size increases), the number of requests to the hard drive also reduces. In turn, this leads to time-saving.

- For both the *mbk*- and *ibk*-means algorithms the value of the objective function f, which represents the quality of clustering solutions, does not strongly depend on the batch size. This dependence becomes even weaker as the size of the dataset increases.

- The comparison of the *mbk*-means and *ibk*-means algorithms demonstrates that there is no significant difference in accuracy when the batch sizes are not large. However, the *ibk*-means algorithm performs better than the *mbk*-means algorithm as the batch size increases. In the largest D44000×16 dataset, the *ibk*-means algo-

# of clus.	k-means		Batch size(p)	mbk		ibk		Relat. improv. of ibk over k-means (%)		Relat. improv. of ibk over mbk (%)	
	f	T		f	T	f	T	f	T	f	T
2	29313.7	154.6	50000	29585.3	153.1	29318.5	157.5	-0.02	-1.88	+0.90	-2.87
			250000	29582.4	154.2	29314.6	155.4	0.00	-0.52	+0.91	-0.78
			500000	29579.5	155.7	29312.6	153.9	0.00	+0.45	+0.90	+1.16
5	11812.8	193.5	50000	11852.3	194.1	11817.3	196.7	-0.04	-1.65	+0.30	-1.34
			250000	11854.2	195.4	11813.6	194.6	-0.01	-0.57	+0.34	+0.41
			500000	11854.5	197.7	11811.6	193.0	+0.01	+0.26	+0.36	+2.38
10	6234.8	215.5	50000	6264.7	216.2	6239.2	218.5	-0.07	-1.39	+0.41	-1.06
			250000	6261.9	219.3	6235.6	216.8	-0.01	-0.60	+0.42	+1.14
			500000	6260.2	221.9	6233.6	214.8	+0.02	+0.32	+0.42	+3.20

Tab. 7. Clustering results on D2000×9 dataset.

rithm finds considerably more accurate solutions than the mbk-means algorithm.

- The comparison of CPU time required by the mbk-means and ibk-means algorithms clearly shows that the latter algorithm uses less CPU time than the former one as the batch size, dataset size, and the number of clusters increase. This is due to the special procedure for selecting starting cluster centers in the ibk-means algorithm, which allows improvement in the quality of solutions obtained by the k-means algorithm and significantly reduces the number of iterations.

- As can be seen from Table 8, the ibk-means algorithm produces clusters in a reasonable time. For example, if the number of clusters is 10, the time for clustering the dataset with size $43,930,257 \times 16 = 702,884,112$ is approximately 3 hours 17 minutes (11796.9 sec.), which is reasonable for such a big dataset. In other words, we have a significant gain in using computational power with restricted resources.

5. CONCLUSIONS

Clustering of big datasets is challenging due to the restrictions of computational power. There are some research papers in this field to handle this challenge. In this paper, we proposed a batch clustering algorithm, called the ibk-means algorithm, which is based on the classic k-means algorithm, uses batches and thus efficiently manages computational power. A number of computational experiments have been carried out using real-world datasets. The largest dataset used in numerical experiments contains 43,930,257 instances and 16 attributes. The main advantage of the ibk-means algorithm is that it makes possible to process huge volume of datasets in batches, thus enabling of handling any size of the big dataset. Results demonstrate that the proposed ibk-means algorithm outperforms the classic k-means algorithm.

The proposed algorithm is fast in very large datasets. Another important strength of the algorithm is that it allows one to effectively use the available computer resources and provides real-time clustering in big datasets. However, this algorithm also has some limitations. Since the proposed ibk-means algorithm uses the k-means algorithm at

# of clus.	Batch size(p)	*mbk*		*ibk*		Relat. improv. of *ibk* over *mbk* (%)	
		f	T	f	T	f	T
2	10000	1687897.9	6591.1	1657385.6	6648.4	+1.81	-0.87
	50000	1683308.8	6603.2	1657381.4	6563.7	+1.54	+0.60
	250000	1688630.1	6614.8	1657375.3	6478.7	+1.85	+2.06
5	10000	683180.1	9267.0	668457.8	9283.5	+2.15	-0.18
	50000	691314.2	9224.6	668452.4	9194.4	+3.31	+0.33
	250000	692381.5	9208.2	668447.5	9108.6	+3.46	+1.08
10	10000	359200.8	12083.0	347597.7	11985.3	+3.23	+0.81
	50000	356364.4	12094.6	347596.4	11891.5	+2.46	+1.68
	250000	356416.7	12102.1	347591.8	11796.9	+2.48	+2.52

Tab. 8. Clustering results on D44000×16 dataset.

each iteration its accuracy deteriorates as the number of batches increases. Moreover, the increase of the number of batches may increase the computational time.

Results of numerical experiments demonstrated that the use of batch data allows one to significantly improve efficiency of the k-means algorithm for clustering very large datasets. The iterative nature of the proposed algorithm leads to the design of the efficient procedure for finding starting cluster centers for the next iteration. Numerical results also demonstrated that the accuracy of the proposed algorithm depends on the batch size. Larger size of batch data leads to more accurate clustering results. It is not unexpected as the larger batches are the better representatives of the whole dataset. However, the batch size should be restricted above by some number. Beyond this number the k-means algorithm may become inefficient and highly inaccurate. This number depends on the size of a dataset. In this case the use of more accurate clustering algorithms may improve overall accuracy, however they may also require prohibitively large computational time.

Results presented in this paper open new opportunities for future research for clustering of big datasets. One of the most important aspects of future research may be the determination of the optimal size of the batch. In other words, taking into account the capabilities of a computer, it is required to select the size of the batch so that to find the high-quality solution to clustering problem in a big dataset in a reasonable time. Another important aspect of the research may be to explore the possibility of combining the proposed iterative batch clustering with other clustering algorithms. The use of more accurate clustering algorithms with the optimal batch size may lead to more accurate clustering results in big datasets. Finally, the fourth direction is the development of parallel batch clustering algorithms. These all will be subjects for future research.

ACKNOWLEDGEMENTS

The research by Dr. Ramiz Aliguliyev was supported by the Azerbaijan Science Foundation – Grant No. AEF-MCG-2023-1(43)-13/04/1-M-04.

Competing interest. The authors have no competing interests to declare.

(Received February 28, 2025)

REFERENCES

[1] C. C. Aggarwal and C. K. Reddy: Data Clustering: Algorithms and Applications (First edition). CRC Press, Taylor and Francis Group, Boca Raton, London, New York 2014. DOI:10.1201/b17320

[2] E. Ahmed, I. Yaqoob, I. A. T.Hashem, et al.: The role of big data analytics in Internet of Things. Computer Networks *129* (2017), 2, 459–471. DOI:10.1016/j.comnet.2017.06.013

[3] R. Alguliyev, R. Aliguliyev, and L. Sukhostat: Anomaly detection in big data based on clustering. Statistics, Optim. Inform. Computing *5* (2017), 4, 325–340. DOI:10.19139/soic.v5i4.365

[4] R. Alguliyev, R. Aliguliyev, A. Bagirov, and R. Karimov: Batch clustering algorithm for big data sets. In: Proc. 2016 IEEE 10th International Conference on Application of Information and Communication Technologies, IEEE Press 2016, pp. 79–82. DOI:10.1109/ICAICT.2016.7991657

[5] R. Alguliyev, R. Aliguliyev, Y. Imamverdiyev, and L. Sukhostat: An anomaly detection based on optimization. Int. J. Intell. Systems Appl. *9* (2017), 12, 87–96. DOI:10.5815/ijisa.2017.12.08

[6] R. Alguliyev, R. Aliguliyev, Y. Imamverdiyev, and L. Sukhostat: Weighted clustering for anomaly detection in big data. Statist. Optim. Inform. Comput. *6* (2018), 2, 178–188. DOI:10.19139/soic.v6i2.404

[7] R. M. Alguliyev, R. M. Aliguliyev, and R. G. Alakbarov: Constrained k-means algorithm for resource allocation in mobile cloudlets. Kybernetika *59* (2023), 88–109. DOI:10.14736/kyb-2023-1-0088

[8] R. Alguliyev, R. Aliguliyev, and L. Sukhostat: Improved parallel big data clustering based on k-medoids and k-means algorithm. Probl. Inform. Technol. *15* (2024), 18–25. DOI:10.25045/jpit.v15.i1.03

[9] R. Alguliyev, R. Aliguliyev, and L. Sukhostat: Parallel batch k-means for big data clustering. Comput. Industr. Engrg. *152* (2021), 107023, 1–11. DOI:10.1016/j.cie.2020.107023

[10] D. Aloise, A. Deshpande, P. Hansen, and P. Popat: NP-hardness of Euclidean sum-of-squares clustering. Machine Learn. *75* (2009), 2, 245–248. DOI:10.1007/s10994-009-5103-0

[11] S. E. Amrahov, Y. Ar, B. Tugrul, B. E. Akay, and N. Kartli: A new approach to Mergesort algorithm: Divide smart and conquer. Future Gener. Comput. Syst. *157* (2024), 330–343. DOI:10.1016/j.future.2024.03.049

[12] A. M. Bagirov, S. Taheri, and B. Ordin: An adaptive k-medians clustering algorithm. Probl. Inform. Technol. *13* (2022), 3–15. DOI:10.25045/jpit.v13.i2.01

[13] A. M. Bagirov, J. Ugon, and D. Webb: Fast modified global k-means algorithm for incremental cluster construction. Pattern Recogn. *44* (2011), 866–876. DOI:10.1016/j.patcog.2010.10.018

[14] B. Bahmani, B. Moseley, A. Vattani, et al.: Scalable k-means++. Proc. VLDB Endowment *5* (2012), 7, 622–633. DOI:10.14778/2180912.2180915

[15] J. Béjar: k-means vs mini batch k-means: A comparison. Technical Report, Universitat Politécnica de Catalunya, 2013. `http://upcommons.upc.edu/bitstream/handle/2117/23414/R13-8.pdf`.

[16] A. Bose, A. Munir, and N. Shabani: A comparative quantitative analysis of contemporary big data clustering algorithms for market segmentation in hospitality industry. 2017. `https://arxiv.org/abs/1709.06202`

[17] L. Bottou and Y. Bengio: Convergence properties of the k-means algorithm. In: Proc. 7th International Conference on Neural Information Processing Systems, MIT Press, Cambridge 1995, pp. 585–592.

[18] P. S. Bradley, U. Fayyad, and C. Reina: Scaling clustering algorithms to large databases. In: Proc. Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, New York 1998, pp. 9–15. DOI:10.5555/3000292.3000295

[19] X. Cai, F. Nie, and H. Huang: Multi-view k-means clustering on big data. In: Proc. Twenty-Third International Joint Conference on Artificial Intelligence, ACM Press, New York 2013, pp. 2598–2604. DOI:10.5555/2540128.2540503

[20] A. Catherine, C. Alejandro, F. Ricardo, and G. Badih: Multivariate and functional robust fusion methods for structured big data. J. Multivar. Anal. *170* (2019), 149–161. DOI:10.1016/j.jmva.2018.06.012

[21] P. Cetin and Ö. Ö. Tanriöver: Priority rule for resource constrained project planning problem with predetermined work package durations. J. Fac. Engrg. Architect. Gazi University *35* (2020), 3, 149–161. DOI:10.17341/gazimmfd.545873

[22] C. L. P. Chen and C.-Y. Zhang: Data-intensive applications, challenges, techniques and technologies: a survey on Big Data. Inform. Sci. *275* (2014), 314–347. DOI:10.1016/j.ins.2014.01.015

[23] X. Cui, P. Zhu, X. Yang, et al.: Optimized big data k-means clustering using MapReduce. J. Supercomput. *70* (2014), 3, 1249–1259. DOI:10.1007/s11227-014-1225-7

[24] D. Dua and E. Karra Taniskidou: UCI Machine Learning Repository. Irvine, Univ. California, School of Information and Computer Science, 2017. http://archive.ics.uci.edu/ml

[25] A. Fahad, N. Alshatri, Z. Tari, et al.: A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Trans. Emerging Topics Computing *2* (2014), 3, 267–279. DOI:10.1109/TETC.2014.2330519

[26] N. Ghadiri, M. Ghaffari, and M. A. Nikbakht: BigFCM: Fast, precise and scalable FCM on Hadoop. Future Gener. Computer Syst. *77* (2018), 29–39. DOI:10.1016/j.future.2017.06.010

[27] A. Hadian and S. Shahrivari: High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs. J. Supercomput. *69* (2014), 2, 845–863. DOI:10.1007/s10586-017-1687-5

[28] L. Haibo and W. Zhi: Application of an intelligent early-warning method based on DBSCAN clustering for drilling overflow accident. Cluster Comput. (2019). DOI:10.1007/s10586-017-1687-5

[29] J. Han, M. Kamber, and J. Pei: Data Mining: Concepts and Techniques (Third edition). Morgan Kaufmann, 2011. DOI:10.1016/C2009-0-61819-5

[30] I. Hassan: $I-k$-means-+: an iterative clustering algorithm based on an enhanced version of the k-means. Pattern Recogn. *79* (2018), 402–413. DOI:10.1016/j.patcog.2018.02.015

[31] R. J. Hathaway, J.C . Bezdek, and J. M. Huband: Scalable visual assessment of cluster tendency for large data sets. Patt. Recog. *39* (2006), 7, 1315–1324. DOI:10.1016/j.patcog.2006.02.011

[32] T. C. Havens and J. C. Bezdek: An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm. IEEE Trans. Knowl. Data Engrg. *24* (2012), 5, 813–822. DOI:10.1109/TKDE.2011.33

[33] T. C. Havens, J. C. Bezdek, and M. Palaniswami: Scalable single linkage hierarchical clustering for big data. In: Proc. 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, IEEE Press 2013. pp. 396–401. DOI:10.1109/ISSNIP.2013.6529823

[34] M. Hilbert and P. López: The world's technological capacity to store, communicate, and compute information. Science *332* (2011), 6025, 60–65. DOI:10.1126/science.1200970

[35] S. S. Ilango, S. Vimal, M. Kaliappan, and P. Subbulakshmi: Optimization using Artificial Bee Colony based clustering approach for big data. Cluster Comput. (2019). DOI:10.1007/s10586-017-1571-3

[36] Y. Imamverdiyev and F. Abdullayeva: Deep learning method for DoS attack detection based on restricted Boltzmann machine. Big Data *6* (2018), 2, 159–169. DOI:10.1089/big.2018.29026.zob

[37] N. Karmitsa, A. M. Bagirov, and S. Taheri: New diagonal bundle method for clustering problems in large data sets. European J. Oper. Res. *263* (2017), 2, 367–379. DOI:10.1016/j.ejor.2017.06.010

[38] N. Karmitsa, A. M. Bagirov, and S. Taheri: Clustering in large data sets with the limited memory bundle method. Pattern Recogn. *83* (2018), 245–249. DOI:10.1016/j.patcog.2018.05.028

[39] D. Kumar, J. C. Bezdek, M. Palaniswami, et al.: A hybrid approach to clustering in big data. IEEE Trans. Cybernet. *46* (2016), 10, 2372–2385. DOI:10.1109/TCYB.2015.2477416

[40] S. Lloyd: Least squares quantization in PCM. IEEE Trans. Inform. Theory *28* (1982), 2, 129–137. DOI:10.1109/TIT.1982.1056489

[41] J. B. MacQueen: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, University of California Press, 1967, pp. 281–297. `https://projecteuclid.org/euclid.bsmsp/1200512992`

[42] M. Marjani, F. Nasaruddin, A. Gani, et al.: Big IoT data analytics: architecture, opportunities, and open research challenges. IEEE Access *5* (2017), 5247–5261. DOI:10.1109/ACCESS.2017.2689040

[43] J. Newling and F. Fleuret: Nested mini-batch k-means. In: Proc. 30th International Conference on Neural Information Processing Systems, Curran Associates Inc. 2016, pp. 1360–1368. DOI:10.48550/arXiv.1602.02934

[44] K. Peng, V. C. M. Leung, and Q. Huang: Clustering approach based on mini batch k-means for intrusion detection system over big data. IEEE Access *6* (2018), 11897–11906. DOI:10.1109/ACCESS.2018.2810267

[45] M. Sabo: Consensus clustering with differential evolution. Kybernetika *50* (2014), 661–678. DOI:10.14736/kyb-2014-5-0661

[46] A. Saini, J. Minocha, J. Ubriani, and D. Sharma: New approach for clustering of big data: disk-means. In: Proc. International Conference on Computing, Communication and Automation, IEEE Press, 2016, pp. 122–126. DOI:10.1109/CCAA.2016.7813702

[47] D. Sculley: Web-scale k-means clustering. In: Proc. 19th International Conference on World Wide Web, ACM Press, New York 2010, pp. 1177–1178. DOI:10.1145/1772690.1772862

[48] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan: Big data clustering: a review. In: Proc. International Conference on Computational Science and its Applications, LNCS 8583, Part V, Springer 2014, pp. 707–720. DOI:10.1007/978-3-319-09156-3-49

[49] Z. Sun and P. P. Wang: A mathematical foundation of big data. New Math. Natur. Comput. *13* (2017), 2, 83–99. DOI:10.1142/S1793005717400014

[50] Q. Tong, X. Li, and B. Yuan: Efficient distributed clustering using boundary information. Neurocomput. *275* (2018), 2355–2366. DOI:10.1016/j.neucom.2017.11.014

[51] V. Torra, Y. Endo, and S. Miyamoto: On the comparison of some fuzzy clustering methods for privacy preserving data mining: towards the development of specific information loss measures. Kybernetika *45* (2009), 548–560.

[52] C.-W. Tsai, S.-J. Liu, and Y.-C. Wang: A parallel metaheuristic data clustering framework for cloud. J. Parallel Distribut. Comput. *116* (2018), 39–49. DOI:10.1016/j.jpdc.2017.10.020

[53] R. Xu and D. Wunsch: Survey of clustering algorithms. IEEE Trans. Neural Networks *16* (2005, 3, 645–678. DOI:10.1109/TNN.2005.845141

[54] Q. Zhang, L. T. Yang, Z. Chen, and P. Li: High-order possibilistic c-means algorithms based on tensor decompositions for big data in IoT. Inform. Fusion *39* (2018), 72–80. DOI:10.1016/j.inffus.2017.04.002

[55] W.-L. Zhao, C.-H. Deng and C.-W. Ngo: k-means: a revisit. Neurocomput. *291* (2018), 195–206. DOI:10.1016/j.neucom.2018.02.072

Rasim Alguliyev, Institute of Information Technology, Baku. Azerbaijan.
e-mail: r.alguliyev@gmail.com

Ramiz Aliguliyev, Institute of Information Technology, Baku. Azerbaijan.
e-mail: r.aliguliyev@gmail.com

Adil M. Bagirov, Centre for Smart Analytics, Institute of Innovation, Science and Sustainability, Federation University Australia, Ballarat. Australia.
e-mail: a.bagirov@federation.edu.au

Mustafa Aliyev, Azerbaijan University of Architecture and Construction, Baku. Azerbaijan.
e-mail: m.aliyev@gmail.com