ISSN 1870-4069

HOK-Means: A Hybrid and Parallel Clustering Algorithm Oriented to Big Data

Joaquín Pérez Ortega, Nancy Salgado Antunez[1], Sandra Silvia Roblero Aguilar[1, 2], Yasmín Hernández[1], Nelva Nely Almanza Ortega[2], Vanesa Landero Nájera[3]

[1] Tecnológico Nacional de México, Centro Nacional de Investigación y Desarrollo Tecnológico, Mexico

[2] Tecnológico Nacional de México, Instituto Tecnológico de Tlalnepantla, Mexico

[3] Universidad Politécnica de Apodaca, Computer Systems, Mexico

jpo_cenidet@yahoo.com.mx, {m20ce047, yasmin.hp}@cenidet.tecnm.mx, {sandra.ra, nelva.ao}@tlalnepantla.tecnm.mx, vlandero@upapnl.edu.mx

Abstract. Using the K-Means algorithm to analyze large datasets demands much time and computational resources. An approach to reducing the time is to parallelize the algorithm. However, the processing time is still high to process large datasets like those presented in Big Data. In this sense, a hybrid clustering algorithm with parallel execution is proposed to solve large datasets. The proposed algorithm is inspired by a highly efficient sequential variant of the K-Means algorithm named O-K-Means. Experimental results with synthetic and real large datasets with conventional equipment showed that Hybrid OK-Means reduces the time to 7.54 times compared to the sequential variant. It is noteworthy that as the size of the datasets grows, the speedup results tend to improve, preserving the quality of the solution. Highlighted, the proposal presented in this article shows a significant improvement in speedup, surpassing the works reported by other researchers.

Keywords: Big data, clustering, k-means, parallel programming.

1 Introduction

Technological development has caused an exponential increase in data generation and storage in recent years. Therefore, there is an interest in extracting knowledge from these massive amounts of data since it would allow us to make better decisions [1, 2].

One of the ways to gain insight from large amounts of data is by identifying clustering patterns. To perform clustering of massive amounts of data (Big Data) with standard tools is generally limited by computing resources [3]. In this regard, our contribution is to provide a strategy to deal with the problem of clustering objects according to their attributes in a Big Data environment.

Clustering techniques have been used in various areas, such as data science, data engineering, and business [1]. Clustering consists of partitioning a set of n objects in k non-empty subsets called clusters in such a way that the objects in one cluster have attributes similar to each other and, at the same time, different from the objects in other clusters [2].

In this article, a parallel algorithm, which we call Hybrid O-K-Means (HOK-Means), is proposed, which is inspired by an improvement of the K-Means algorithm called O-K-Means[2]. This variant has shown to be highly efficient in solving large datasets of the Big Data type.

The structure of this paper is organized as follows. Section 2 presents related work. Section 3 describes the algorithms used in this research and shows the improvement proposal. Section 4 describes performance metrics and the design of the experiment. Section 5 reports the results obtained. Conclusions and ideas for future research are given in Section 6.

Algorithm 1: HOK-Means

Master Node
Initialization:
  $P := \{p_1, \ldots, p_t\}$; // The set of available processors is allocated
  $N := \{x_1, \ldots, x_n\}$; // Load Dataset
  $M := \{\mu_1, \ldots, \mu_k\}$; // Initialize random centroids
  $U := 0.72$; //Threshold value for determining O-K-Means convergence;
  Send start order, centroids (M), *and* n_o to slaves;
Slave Node
Classification:
  For $x_i \in N_p$ and $\mu_k \in M${
    Calculate the Euclidean distance from each x_i to each k centroid;
    Assign object x_i to the nearest centroid μ_k;
  Calculate the number of objects that changed groups;
  Send *MR* results matrix;
  Send the number of objects that changed the group v_r
Master Node
Receive end status and information from all slave nodes;
Calculate the percentage of change $\gamma_r = 100(v_r/n)$;
Calculate centroids:
  Calculate the centroid M;
Convergence:
  If ($\gamma \leq U$):
    Stop the algorithm;
  Otherwise:
    Go to step **7**;
End of algorithm

Table 1. Datasets used in experiments.

id	1	2	3	4	5	6	7	8	9	10	11	DSAS	EPCS
n	2	2	2	2	2	1	1	1	0.5	1.2	1	1,140,000	2,049,280
d	2	4	6	5	7	4	7	10	11	5	11	45	7

2 Related Work

According to [3, 4], the clustering problem type K-Means when k>=2 or d>=2 is NP-Hard, so obtaining an optimal solution for a dataset of considerable size is intractable. An approach to reducing the processing time of sequential algorithms is parallelizing them. The standard K-Means algorithm has been parallelized using different platforms and architectures.

For example, in Map Reduce [3, 4], Spark [5], Peer to Peer [6], GPU [7, 8], Multi-core [9], and FGPA [10], among others. Improvements have been made to the K-Means algorithm that increases its efficiency. However, there are few parallelized versions of improvements to the K-Means algorithm. Three of the parallelized enhancements are described below.

In [9], a parallel variant of K-Means++ is proposed, using multi-core on a single computer. This work presents the division of the dataset in such a way that each processor works a subset of objects in parallel. The best speedup achieved was 7.7, with a computer of 12 cores and a parallel efficiency of 0.64.

In [11], the K-Means++ algorithm is implemented on three different architectures for shared memory: multicore CPU, high-performance GPU, and the massively multithreaded Cray XMT platform. The objective of this research was to show a performance relationship of each platform with the number of objects, attributes, and groups.

In [5], an improvement in the initialization phase of the K-Means algorithm named Canopy is proposed. This improvement consists of selecting the set of centroids using the weighted density method to reduce the impact of outliers on the clustering results. The best result is a speedup of 4.0 and a parallel efficiency of 2.0.

The size of the largest dataset is 1.72 GB. The algorithm is parallelized using the Spark platform with eight cores. Although there are already algorithms that parallelize K-Means improvements, the efficiency they report is limited, and it is foreseeable that the solution of large datasets is long time-consuming.

In this sense, the proposal presented in this article shows a significant improvement in speedup, surpassing the works mentioned. Consequently, enabling the solution of larger datasets in less time.

3 New HOK-Means Algorithm

This section briefly describes the background that gave rise to the algorithm and then describes the proposed algorithm in detail. K-Means is an iterative method that consists

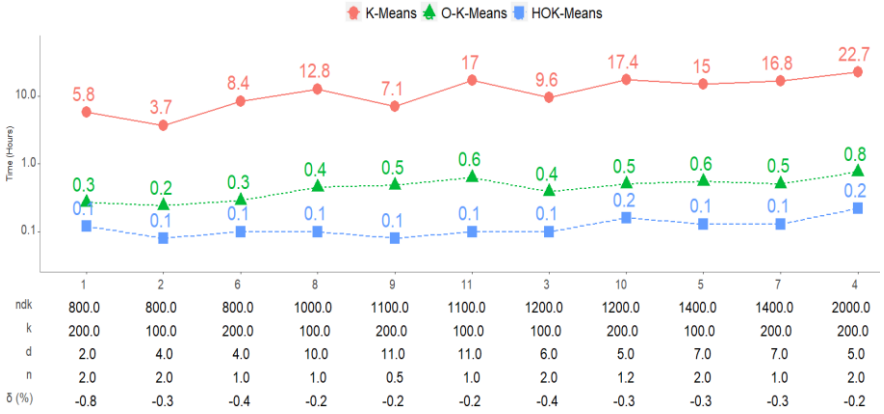

Fig. 1. Solution time with synthetic datasets.

of partitioning a set of N objects into $k \geq 2$ non-empty groups so that the objects of each group have similar attributes and, at the same time, are different from the objects of any other group [2].

There are different improvements to the K-Means algorithm [12]. In [2], a heuristic called O-K-Means is proposed, which accelerates the convergence process, stopping the algorithm when the total number of objects that change the cluster in an iteration is less than a threshold. This value expresses a relationship between the computational effort and the quality of the solution.

Therefore, although the gain in solution quality is minimal, the algorithm invests the same computational effort to perform each iteration. The proposed stopping threshold decreases the number of iterations while preserving most of the quality of the solution of the K-Means algorithm. In [2], the experimental results presented achieved an average execution time reduction of 93.88%, with only a 0.4% loss in clustering quality compared to standard K-Means.

HOK-Means is a parallel version of the O-K-Means sequential algorithm, and for both algorithms, the same outputs will be obtained with the same inputs. HOK-Means is an algorithm that parallelizes the classification and convergence phases of the sequential O-K-Means algorithm.

To describe the HOK-Means algorithm, we will rely on Algorithm 1, which shows the sequence of execution of instructions in the master node and the slave nodes.

Initialization: The initialization of the master node is shown in lines 2 to 7 in Algorithm 1. In this phase, the variables are initialized. A remarkable parameter is the value of the threshold; in this case, a value of 0.72 U is assigned. In Line 7, the master node sends the start order, data of the centroids, and the subset of objects that each slave node will work on.

Classification: Lines 8 to 15 show the instructions carried out in parallel by the slave nodes. Lines 10 and 11 have the instructions for calculating the distance of each object to each of the centroids. Line 12 shows the instruction to assign an object to the group whose centroid is closest to the object. The next line determines the number of objects that changed groups. This information is relevant to determining when the algorithm should stop.

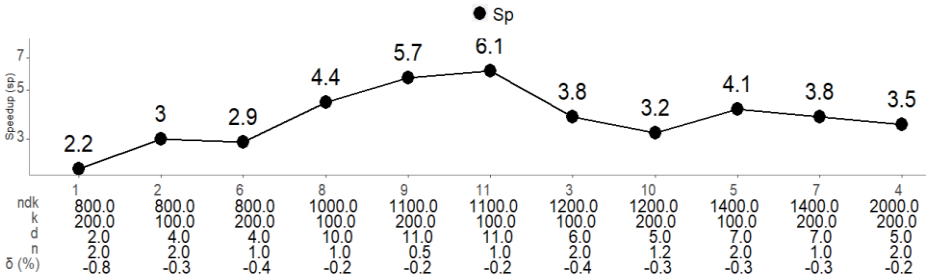

Fig. 2. Speedup obtained when solving synthetic datasets.

In line 14, the slave nodes transmit to the master node a matrix, whose number of rows corresponds to the number of centroids. In the first column, there is the value of the centroid identifier; in the second, the sum of the distance of all the objects that belong to this group; and the third corresponds to the number of objects in the group. Line 15 transmits the number of objects that changed the group.

Calculation de γ: Line 17 of the master node concentrates the data that all the slave nodes have transmitted to it and calculates the percentage of change of objects γ. Note that the value of γ is used in the stopping criterion of the algorithm.

Calculation of the Centroid: In line 20, the calculation of the new centroids is performed.

Convergence: In line 22, it is determined if the object change percentage is less than or equal to the predetermined threshold. If it is affirmative, the algorithm stops. Otherwise, the algorithm continues at step 7.

The HOK-Means programming was coded in Python 3.8 language on a Windows 10 operating system. Some libraries used were: Pandas, Numpy, and parallel Python (PP). The equipment used was a Dell laptop processor Intel® Core™ i7-6700HQ CPU 2.60GHz, 8 processors, 16GB of memory, and 1TB fixed hard drive.

4 Experimental Evaluation

In this section, the metrics to evaluate the parallelization of the algorithm are first described, then the experiments carried out are shown and, finally, the solved data sets are shown.

The speedup (*Sp*) and parallel efficiency (*Ep*) are two metrics used to evaluate the quality of a parallel algorithm. The rate of acceleration or speedup, indicates the relationship between the sequential execution time (*Ts*) and the parallel execution time (*Tp*) Eq. (1).

The ideal speedup is a linear value $Sp = p$, where p is the number of processors [13]. The parallel efficiency indicates the relationship between the speedup and the number of processors used (p) Eq. (2). The ideal parallel efficiency value is 1 [13]:

$$\text{speedup: } Sp = \frac{Ts}{Tp}, \tag{1}$$

$$\text{parallel efficiency: } Ep = \frac{Sp}{p}. \tag{2}$$

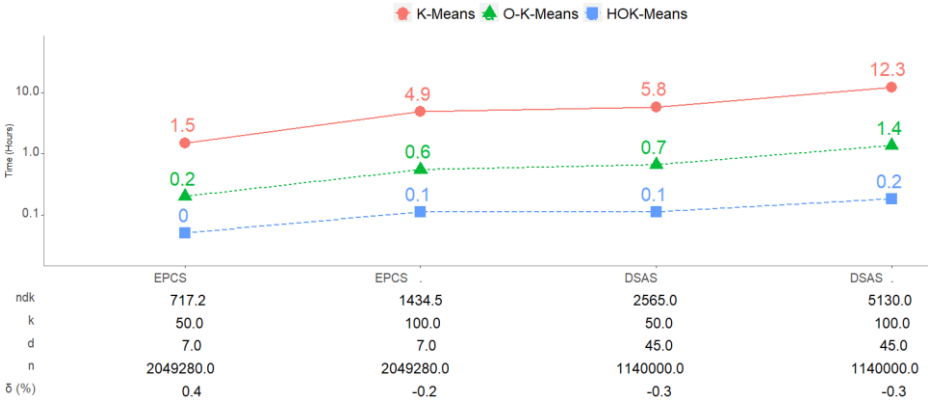

Fig. 3. Solution time with real datasets.

The measure of the quality of the clustering is the objective function value, the sum of the squared error (*SSE*), according to the K-Means algorithm[14]. Eq. (3) shows *SSE*. The clustering quality is better when the value of SSE is lower.

The percentage of quality loss (δ (%)) is the ratio of two quality measures $\delta(\%)=100(1-z/z_o)$, where z is the SSE obtained by solving with standard K-Means and z_o is the SSE obtained by O-K-Means, which is equal to that obtained with HOK-Means.

The *ndk* is an indicator related to the size of the dataset, it is the product of the number of objects in the dataset *n*, the number of attributes *d*, and the number of groups to form *k*:

$$SSE = \sum_{j=1}^{k} \sum_{x \in \mu_j} \left|\left|x - \mu_j\right|\right|^2. \tag{3}$$

4.1 Design of Experiments

To evaluate the performance HOK-Means, 1350 experiments were realized in total. Two real datasets were resolved, with k=50 and 100. 11 synthetic datasets with k=100 and 200 were resolved. All datasets were resolved with the HOK-Means, standard K-Means, and O-K-Means algorithms.

Each configuration of dataset and k was solved 30 times with different initial centroids. Note that the configuration is the same used in [2], but the initial centroids are not the same, so there is a small variation in the quality percentage δ(%).

4.2 Datasets Used in Experiments

In Table 1 describes the set of datasets synthetics and real used. The first row contains the identifier of the dataset; the second shows the number of objects (n) and the third row shows the number of attributes (d).

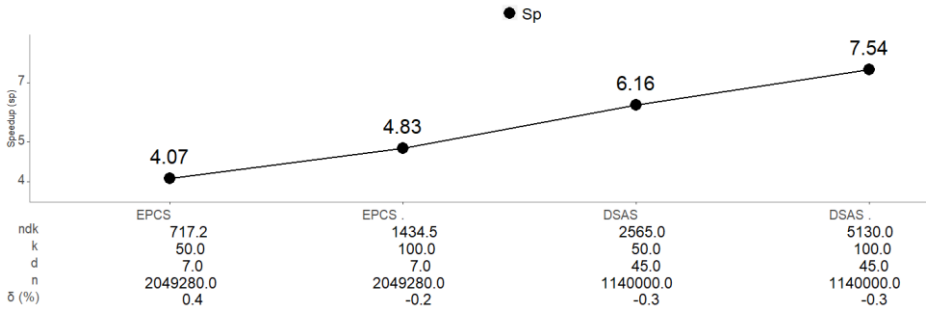

Fig. 4. Speedup obtained when solving real datasets with HOK-Means vs O-K-Means.

In the synthetics datasets (columns 2-12) the value of *n* is represented in millions. In the two last columns the real datasets used are described, which were obtained from the UCI machine learning repository [15].

5 Results

This section shows the results of the solution of the set of datasets using the K-Means, O-K-Means, and HOK-Means algorithms. Fig. 1 and 2 show the results obtained in the solution of synthetic datasets and Fig. 3 and 4 for real datasets.

The *x*-axis represents dataset *id*, which is ordered by *ndk* index from lowest to highest. In the graph in Fig. 1, the processing times are observed, when solving synthetic datasets, obtained with K-Means, O-K-Means, and HOK-Means.

In each case, the processing time with HOK-Means is the least. In the lower part of Fig. 1, it is shown: in the first row, the indicator *ndk* is given in millions; in the second, the percentage of quality loss of HOK-Means with respect to K-Means δ (%); in the third, the number of groups *k*; and in the last one the identifier of the dataset (Table 1).

It is remarkable that the quality of the solution (*SSE*) for the HOK-Means and O-K-Means algorithms is the same, and that the δ (%) was not greater than 0.8% in the experiment with id = 1 with the solution of synthetic datasets. For readers interested in seeing in detail the solution quality of the datasets, refer to [2].

Fig. 2 shows the speedup performance corresponding to each of the solved datasets. In the best case, with dataset 11, the solution time was reduced from 17 hours to less than six minutes, 99.4% of the processing time compared to K-Means, with a quality loss of only 0.21 %, achieving a speedup of 6.1 with respect to O-K-Means. Fig. 3 shows the processing times obtained when solving real datasets, using standard K-Means, O-K-Means and HOK-Means.

The processing time with HOK-Means (orange line) is the shortest. In the lower part of Fig. 3, it is shown: in the first row, the indicator *ndk* is given in millions; in the second, the percentage of quality loss of HOK-Means with respect to K-Means δ (%); in the third, the number of groups k; and in the last one the identifier of the dataset (Table 1). In Fig. 4, the speedup performance is observed, with each real dataset.

The best result was with the DSAS dataset, and k = 100. Processing time with the HOK-Means algorithm was reduced by 98.5% compared to K-Means, with a quality loss of only 0.29%.

A speedup of 7.54 was achieved with respect to O-K-Means. That is, using the same equipment, with HOK-Means the solution time was reduced from 1.36 hours to only 11 minutes, obtaining the same quality of the solution.

6 Conclusions and Future Work

This paper shows that it is feasible to parallelize a highly efficient general-purpose improvement of the K-Means algorithm. To validate the proposal, which we call HOK-Means, a set of experiments composed of real and synthetic datasets was designed. To compare the results of the algorithms, all the datasets were solved using HOK-Means, the standard K-Means, and O-K-Means algorithms. The HOK-Means algorithm is robust even when using larger data sets; computational resources limit it.

Based on the results, it was observed that HOK-Means, in the best of cases, reduce up to 7.54 times the solution time of a real dataset, with the following parameters n =1,140,000; d=45 y k=100. The proposal presented in this article shows a significant improvement in speedup, surpassing the works reported by other researchers.

It is underlined that HOK-Means tends to obtain better processing time as the number of attributes increases. It is noteworthy that due to the data structures implemented and the pagination in the Pandas library of the Python language, it was possible to process large datasets.

HOK-Means is recommended for the solution of large datasets, and in particular with a large number of attributes. To continue this research, we suggest parallelizing other K-Means enhancements.

References

1. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing, vol. 2, no. 3, pp. 267–279 (2014). DOI: 10.1109/TETC.2014.2330519.
2. Pérez–Ortega, J., Almanza-Ortega, N. N., Romero, D.: Balancing effort and benefit of K-means clustering algorithms in big data realms. PLOS ONE, vol. 13, no. 9, pp. 1–19 (2018). DOI: 10.1371/journal.pone.0201874.
3. Ansari, Z., Afzal, A., Sardar, T. H.: Data categorization using hadoop MapReduce-based parallel K-means clustering. Journal of The Institution of Engineers, vol. 100, no. 2, pp. 95–103 (2019)
4. Sardar, T. H., Ansari, Z.: An analysis of distributed document clustering using MapReduce based K-means algorithm. Journal of The Institution of Engineers, vol. 101, no. 6, pp. 641–650 (2020). DOI: 10.1007/s40031-020-00485-2.
5. Wang, Z., Xu, A., Zhang, Z., Wang, C., Liu, A., Hu, X.: The parallelization and optimization of K-means algorithm based on spark. In: 15th International Conference on Computer Science and Education, pp. 457–462 (2020). DOI: 10.11 09/ICCSE49874.2020.9201770.
6. Azimi, R., Sajedi, H., Ghayekhloo, M.: A distributed data clustering algorithm in P2P networks. Applied Soft Computin, vol. 51, pp. 147–167 (2017). DOI: 10.1016/j.asoc.2016.11.045.
7. Al-Ayyoub, M., Yaseen, Q., Shehab, M. A., Jararweh, Y., Albalas, F., Benkhelifa, E.: Exploiting GPUs to accelerate clustering algorithms. In: IEEE/ACS 13th International

Conference of Computer Systems and Applications, pp. 1–6 (2016). DOI: 10.1109/AICCSA.2016.7945796.
8. Lutz, C., Breß, S., Rabl, T., Zeuch, S., Markl, V.: Efficient K-means on GPUs. In: Proceedings of the 14th International Workshop on Data Management on New Hardware, pp. 1–3 (2018). DOI: 10.1145/3211922.3211925.
9. Hadian, A., Shahrivari, S.: High performance parallel K-means clustering for disk-resident datasets on multi-core CPUs. The Journal of Supercomputing, vol. 69, no. 2, pp. 845–863 (2014). DOI: 10.1007/s11227-014-1185-y.
10. Dafir, Z., Lamari, Y., Slaoui, S. C.: A survey on parallel clustering algorithms for big data. Artificial Intelligence Review, vol. 54, no. 4, pp. 2411–2443 (2021). DOI: 10.1007/s10462-020-09918-2.
11. Mackey, P., Lewis, R. R.: Parallel K-means++ for multiple shared-memory architectures. In: 45th International Conference on Parallel Processing, pp. 93–102 (2016). DOI: 10.1109/ICPP.2016.18.
12. Pérez-Ortega, J., Almanza-Ortega, N. N., Vega-Villalobos, A., Pazos-Rangel, R., Zavala-Diaz, J. C., Martínez-Rebollar, A.: The K-means algorithm evolution. Introduction to Data Science and Machine Learning, Chaper 5 (2019)
13. Zavala-Díaz, J. C., Cruz-Chávez, M. A., López-Calderón, J., Hernández-Aguilar, J. A., Luna-Ortíz, M. E.: A multi-branch-and-bound binary parallel algorithm to solve the knapsack problem 0–1 in a multicore cluster. Applied Sciences, vol. 9, no. 24, p. 5368 (2019). DOI: 10.3390/app9245368.
14. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pp. 281–297 (1967)
15. UCI Machine Learning Repository, archive.ics.uci.edu/ml (2022)