Clustering Big Data Based on Distributed Fuzzy K-Medoids: An Application to Geospatial Informatics

MAGDA M. MADBOULY¹, SAAD M. DARWISH¹, NOHA A. BAGI², AND MOHAMED A. OSMAN³

¹Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, Alexandria 21526, Egypt
²IT Engineer and Senior Programmer, Alexandria Water Company, Alexandria 21581, Egypt
³Management Information System Department, Higher Institute of Management Information Technology, Kafr El-Sheikh 33511, Egypt

Corresponding author: Saad M. Darwish (saad.darwish@alexu.edu.eg)

ABSTRACT
The advent of big data related to spatial position knowledge, called geospatial big data, provides us with opportunities to recognize the urban environment. Existing database processing methods are inadequate to rapidly provide reliable results in a geospatial big data context due to the need for defining approximation “measures” and the increasing execution time for the queries. The clustering method yields the functional effects. How to scale and accelerate clustering algorithms while maintaining high clustering efficiency, on the other hand, remains a significant challenge. The paper’s primary contribution is the introduction of a modified hierarchical distributed k-medoid clustering method that is specific to spatial query analysis for big data. To improve the efficiency of the k-medoid algorithm and obtain more precise clusters, the suggested model utilizes the Fuzzy k-Medoids method to overcome outliers in the spatial data set and to deal with data uncertainty. The method is complex in nature since it is not predicated on the number of right clusters. The proposed model is divided into two phases: the first step creates local clusters based on a portion of the entire dataset; this stage makes extensive use of the parallelism paradigm provided by the Apache Spark framework; and the second phase aggregates the local clusters to produce compact and reliable final clusters. The proposed model greatly reduces the amount of knowledge shared during the aggregation process and automatically produces the appropriate number of clusters based on the dataset characteristics. The results show that the proposed model outperforms the traditional K-medoids in terms of accuracy of obtained centers in big data applications.

INDEX TERMS
Geospatial informatics, big data clustering, dynamic clustering, Apache Spark, fuzzy K-medoids.

I. INTRODUCTION

Over the last few decades, the rapid development of information technology has resulted in an explosion in data from a variety of devices, propelling us into the era of big data. Big data derived from devices such as smartphones and portable Global Positioning System (GPS) devices has now permeated our daily lives and shown immense potential in practical applications such as climate science, disaster management, public health, crop protection, smart cities, emergency management, and environmental monitoring [1]. Geospatial big data is a subset of big data that includes position data.

Location knowledge is critical in the age of big data, as the majority of the data collected today is spatial in nature and collected by pervasive location-aware sensors. Numerous attempts have been made to use geospatial data to track patterns in human activity and to conduct urban and environmental research using remote sensing imagery. By analyzing the temporal characteristics of pick-up and drop-off operations within geographic units, we cannot only identify urban functions but also determine job-housing functional dynamics more precisely than traditional remote sensing images, allowing us to further investigate intra- and inter-city spatial activity [2].

Collecting such valuable information and patterns efficiently is challenging due to big data’s 5V...
characteristics — volume, velocity, variety, veracity, and value. Traditional processing infrastructures are no longer capable of managing such large amounts of complex data in a variety of data formats, necessitating the development of modern standards for storing, maintaining, and preserving massive amounts of data on a daily basis. High-Performance Computing (HPC) architectures, especially cloud computing systems, are characterized by their superior parallel computing capability, broad-scale scalability, and versatility in the processing of large amounts of data through the use of virtualization technology, automated distribution, and distributed computing [3]–[5].

In high-performance computing (HPC), three common data storage paradigms exist: shared everything architecture (SEA), shared-disk architecture (SDA), and shared-nothing architecture (SNA). For more details, see [1]–[3]. SNA distributes data through cluster processors and processes a subset of it locally. SNA has been the de facto standard for data storage because (1) it is scalable, allowing an HPC cluster to efficiently increase its storage and processing power with the addition of additional device nodes; (2) each processor can be processed centrally, reducing data transfer through the network; and (3) single-point error is eliminated due to the computer’s being stored locally. The Hadoop Distributed File System (HDFS) is a widely used implementation of SNA, the Hadoop ecosystem’s central storage mechanism. HDFS divides data into blocks and stores them in a Hadoop cluster, which spans many computer nodes and allows for concurrent processing [6], [7].

In the world of geospatial big data, query analysis is a significant obstacle. Traditional database management systems, due to their inability to scale, are no longer useful for querying such massive amounts of data [8]. Recently, the adoption of large-scale and parallel computing infrastructures built on cost-effective commodity clusters and cloud computing environments has introduced new management problems for avoiding query processing bottlenecks induced by an unbalanced load of parallel tasks. This requires the use of a clustering algorithm [9]. Geospatial clustering is a technique for categorizing a set of spatial points into distinct classes called “clusters.” The challenge for geospatial big data clustering is to scale and accelerate clustering algorithms while maintaining the highest possible clustering efficiency.

To exploit this large amount of data, an effective processing model with a rational computing cost for this massive, complicated, diverse, and heterogeneous data is needed [10]. There are three methods for increasing the speed and scalability of big data clustering algorithms. (1) The first approach is to minimize iteration by the use of sampling-based algorithms. Sampling-based algorithms run clustering algorithms on a subset of data sets rather than the whole data set. (2) The second approach is to use randomized approaches to minimize the dimension of the data. Another factor to consider is the dimensionality of the data collection, which has an impact on the difficulty and speed of clustering algorithms. (3) The final method is to use parallel and distributed algorithms, which accelerate computing and improve scalability by using many computers. Applications for concurrent computing cover both traditional parallel and data-intensive applications. Parallel programs in the standard sense assume that data may be stored in the memory of distributed computers.

Scholars have investigated the possibility of requiring the cluster representative to be an object that exists within each cluster (i.e., the cluster’s center-most data object). These objects are called medoids, which are defined as the objects with the shortest sum of distances to all other points in their respective clusters. The K-medoids method, as a partitioning clustering algorithm, is one of the most widely used approaches in data mining and knowledge discovery applications [11]. However, the algorithm’s two primary issues are the determined cluster numbers as an input and the effect of the initial value of cluster centers on the clusters’ quality [12]. One of the most fundamental assumptions behind medoid-based clustering models is that items must be assigned to a single (and unique) cluster. However, scholars have shown that this limitation may be arbitrary and that many items are better represented by medoids from various clusters. Similarly, objects allocated to a cluster may sometimes be represented even more accurately by numerous medoids. Both scenarios are possible, depending on the nature of the data [11]–[13]. Numerous approaches exist for incorporating fuzziness into a clustering model. Often, fuzziness is introduced by allowing non-medoid objects to be allocated to multiple clusters to varying degrees while still requiring that all medoids be assigned to a single and unique cluster.

A. PROBLEM STATEMENT AND MOTIVATION

Today, the exponential growth in the volume and complexity of produced geospatial data is so great and complicated that conventional data processing technology tools and systems are insufficient to handle it. As a result, geospatial big data needs appropriate analysis frameworks for effective data collection and analysis. Generally, when dealing with vast amounts of geospatial data, the execution time of queries increases, since it is impossible to scan the whole set of data in a reasonable amount of time. Clustering produces the most positive effects. Numerous clustering methods have been proposed by the contemporary research group. These methods either suffer from the curse of dimensionality or need several iterations to complete the clustering process. Thus, current clustering techniques are insufficient for the study of geospatial big data. Historically, very few studies involving parallel, distributed clustering with Apache Spark have been published. However, these approaches suffer from noisy or uncertain input, which makes clustering an entity difficult, thus impacting the clustering performance. Successful algorithms must be capable of coping with outliers and noisy results. As a result, the current research endeavors to establish a framework for clustering large geospatial datasets that is robust to outlier results and deals effectively with uncertain data.
B. AIM OF THE WORK
After an analysis of the trends and advancements in clustering strategies for addressing big data problems, this research would suggest a dynamic clustering algorithm to enable spatial query processing for geospatial big data. This algorithm generates high-quality clusters quickly and efficiently by adjusting the cluster heads to the most appropriate ones without setting the cluster head as static. One of the main features of this distributed clustering strategy is that the number of global clusters is dynamic; there is no need to predetermine the number of global clusters. Herein, a fuzziness approach is used to boost the selection of the cluster’s medoid for uncertain data.

C. CONTRIBUTION AND METHODOLOGY
The paper discusses how to increase the scalability and efficiency of geospatial big data clustering by using a distributed dynamic fuzzy clustering paradigm built on top of Apache Spark. The suggested distributed dynamic clustering method is divided into two phases: the entirely parallel process, in which each node of the system calculates its own local clusters based on its portion of the entire dataset; and the aggregate phase that combines the local clusters to produce compact and reliable final clusters. This process is devoid of communication; it makes good use of the parallelism model. Since the second step is distributed as well, it does include some coordination between the nodes. The overhead associated with these interactions, on the other hand, has been reduced by the use of a novel concept of cluster representatives. Additionally, the distributed clustering technique is dynamic for spatial datasets since it does not require the number of valid clusters to be defined in advance (as an input parameter). This effectively addresses one of the big drawbacks of K-Medoids.

The proposed model makes use of fuzzy K-medoids and is intended to deal with dynamic geographic big data that contains noise and uncertain data. Fuzzy K-medoids provide more accurate implications for data points that overlap. In contrast to fuzzy k-means, which uses artificial objects calculated as weighted means as cluster prototypes, fuzzy k-medoids use a subset of observed objects as cluster prototypes (medoids). The noise cluster is an extra cluster (in comparison to the k standard clusters) to which outlier objects with high membership degrees are allocated.

The remainder of this article is structured as follows: Section II conducts a review of the literature on clustering methods for big data. Section III discusses the suggested approach, which is focused on an overview of prior approaches. Section IV validates the proposed model through a series of experiments, followed by a review of the results. The final section, Section V, summarizes the work focused on the previous chapter and suggests possible directions for future work.

II. RELATED WORK
Numerous studies have been conducted to improve the efficiency of clustering in big data algorithms. Each researcher approached the problem from a unique perspective and with a unique concept. Based on current survey work [10] [14]–[17], it can be inferred that no clustering algorithm can resolve all of the problems associated with big data. Although parallel classification has the potential to be extremely useful for clustering large amounts of data, the difficulty of implementing these algorithms remains a significant barrier. However, the MapReduce or Spark frameworks will serve as an excellent foundation for implementing these parallel algorithms. Generally, in order to process a large amount of data while maintaining reasonable resource requirements, we must develop clustering algorithms by reducing their time and memory complexity.

In [18], the authors generalized the partitioning around medoids (PAM) algorithm for MapReduce. Similar to the k-means extension, it assigns each object to the nearest medoid during the mapping process and updates the actual medoid to the most central object during the reduce phase using pair-wise computation. In contrast to the original PAM algorithm, which performs a global search, it performs a local search to parallelize the medoid update operation. In general, this algorithm exhibits excellent scalability and implementation speed but does not guarantee high accuracy. The GREEDY algorithm [19] was presented, which utilizes a greedy algorithm to find a candidate collection of k-medoids from each partition of the entire data set, where a partition can be thought of as a sample. GREEDY would not address the efficiency problem entirely, since the amount and scale of samples are depend on the total size of the data due to the partitioning criterion. However, this algorithm is impractical when dealing with large data sets.

The algorithm suggested in [20] generates a single best sample by parallel, iterative sampling and then operates a weighted k-medoids algorithm on the sample. Iterative sampling is costly, especially for large data sets, since it accesses the entire data collection repeatedly to measure the distances between sampled and un-sampled objects. Yet, its performance often degrades as k grows, as the number of measuring distances rises. In general, existing algorithms have chosen reliability over precision by running a local search rather than a global search or by utilizing sampled data rather than whole data.

The authors [21] introduced a new K-Medoids++ spatial clustering algorithm focused on MapReduce for large spatial data clustering. The initialization algorithm is paired with the MapReduce method to reduce the number of iterations. There are two primary facets to the improvement of the suggested K-Medoids++ spatial clustering algorithm. To begin with, the efficient initial medoids search algorithm is implemented into K-Medoids++ spatial clustering in order to find appropriate medoids and thus reduce iterations. Second, the MapReduce architecture supports parallelization of
the K-Medoids++ spatial clustering algorithm. In terms of speed, their algorithm was better than standard K-Medoids. It also works well when dealing with a lot of spatial data on common computer hardware.

The same authors discussed techniques for possible speedups in k-medoids clustering using the Apache Spark framework [22]. They discussed the benefits of pre-caching the pairwise distance matrix, which is at the heart of the k-medoids clustering algorithm, not only to speed up the algorithm’s execution but also to speed up the validation of clusters. The central idea is based on the observation that because the distance between any two patterns remains constant regardless of whether they belong to the same cluster or not, it is possible to avoid evaluating the pairwise distances between cluster elements at each k-medoids iteration. However, the validity of this observation is contingent upon the (dis) similarity measure used. The results from real-world pathway map datasets demonstrated the robustness of such distributed implementations, as well as their efficacy with structured data.

M. Bendechache et al. [23] proposed a novel distributed clustering approach that makes use of both local results and aggregation to generate global models. The fully parallel phase in which each system node calculates its local clusters based on its subset of the entire dataset. There is no communication during this phase; the parallelism paradigm is fully utilized. Although the second phase is distributed as well, it does require some communication between the nodes. The overhead associated with this communication, on the other hand, has been minimized through the use of a novel concept of cluster representatives. Each cluster is denoted by a contour and a density value. This framework is extremely simple to implement via the MapReduce mechanism. The fact that the number of global clusters is dynamic is one of the key outputs of this distributed clustering technique.

The same author’s [24] enhanced the aforementioned dynamic distributed clustering techniques by incorporating an efficient data reduction phase that significantly reduces the size of the data exchanged; as a result, this technique addresses the issue of communication overhead and was developed for spatial datasets. The results indicate that the approach accelerates linearly and scales well when the complexity of the local clustering is NP, as its results are unaffected by the type of communication.

Although clustering approaches have been studied for nearly two decades, there is still room to make them more efficient and practical in real-world big data applications. According to the aforementioned review, it can be found that past studies were primarily devoted to: (1) no clustering algorithm can be used to solve all the big data issues. (2) parallel classification is potentially very useful for big data clustering, but the complexity of the implementation of these algorithms remains a great challenge. (3) The MapReduce framework can provide a very good basis for the implementation of such parallel algorithms, but is not useful when dealing with iterative algorithms. (4) None of the existing algorithms satisfy both high accuracy and high efficiency in the k-medoids algorithm for geospatial big data environment. (5) Existing approaches for query processing are insufficient to quickly provide accurate results in a big data environment due to the need to identify approximation “measures” and execution time for the queries is increasing. Effective results are obtained through the clustering process. However, how to scale up and speed up clustering algorithms with minimum sacrifice to the clustering quality remain a big problem.

To the best of our knowledge, little attention has been paid to devising a new dynamic distributed fuzzy K-medoids clustering approach applicable in spatial query processing for big geospatial data that has a high number of outliers and uncertain data. Although there are many works that deal with the idea of parallelism within K-medoids-based clustering approaches, implementing this idea in geospatial big data faces many challenges due to the high number of outliers and noise in this type of data. The next section discusses in detail the suggested model that integrates the fuzzy K-medoids algorithm to boost the selection of the cluster’s medoid for uncertain data with a dynamic distributed clustering technique to handle geospatial big data clustering processing.

III. PROPOSED GEOSPATIAL BIG DATA FUZZY CLUSTERING MODEL

A. PROBLEM FORMULATION

K-medoids clustering is among the most popular methods for cluster analysis, despite its use requiring several assumptions about the nature of the latent clusters. Our ability to unearth valuable knowledge from large sets of data is often impaired by the quality of the data. Data may be imprecise (e.g., due to measurement errors) or originate from unreliable sources (such as crowd-sourcing). In Fuzzy K-medoids, its underlying formulation allows medoids to represent multiple clusters. However, their implementation into massive geospatial databases, though, faces various problems, including high-dimensional data and the complexity of certain algorithms. As a result, effective and precise parallel clustering algorithms must be investigated. The use of distributed algorithms is critical for improving scalability and accuracy while processing large amounts of data. Distributed clustering is a promising solution for large data sets, as they are typically produced in distributed locations and processing them on their local hosts greatly reduces response times and communications.

The fuzzy K-medoids (FKM) method optimizes the underlying mathematical optimization model expressed as follows [12], [13]:

\[
\min Z_{FKM} = \sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} (e_{ij})^h \\
\text{s.t. } \sum_{j=1}^{n} e_{ij} = 1, \quad \forall i \in \{1, 2, \ldots, n\} \\
e_{ij} \leq e_{jj}, \quad \forall i, j \in \{1, 2, \ldots, n\} \\
\sum_{j=1}^{n} e_{jj} = k \quad (1) (2) (3) (4)
\]
The fuzzy membership degree of object $O_i$ to the cluster whose medoid is $O_j$ is denoted by $e_{ij}$, and it is defined as:

$$e_{ij} = \frac{1}{\sum_{t|e_t=1} \left( \frac{d_{ij}}{d_{it}} \right)^h}$$ (7)

where $e_{ij}$ is the degree of membership of $O_i$ to the cluster whose medoid is $O_j$ ($e_{ij} = 0$ if $O_j$ is not a medoid), and $h$ is the fuzziness factor. The fuzziness factor is a hyperparameter that indicates the desirable level of overlapping between the clusters to be found, or, in other words, how much the degrees of membership will be spread among the clusters. As $h \rightarrow 1^+$, the objects tend to be assigned to a single cluster, creating crisp partitions. As $h \rightarrow \infty$, the objects tend to be equally spread among clusters. That is, for each non-medoid $O_i$ and each medoid $O_j$, the degree of membership $e_{ij}$ tends to $1/k$.

Given a known set of medoids, the degrees of membership of each object $O_i$ to a chosen medoid $O_j$ can be found by computing the following expression:

$$e_{ij} = \frac{1}{\sum_{t|e_t=1} \left( \frac{d_{ij}}{d_{it}} \right)^h}$$ (7)

**B. MATERIALS AND METHODS**

This section details the proposed optimized hierarchical distributed Fuzzy K-medoids clustering paradigm for increasing the reliability and scalability of the analysis of massive geospatial datasets. This model consists of two phases: local clustering generation and global clustering generation through aggregation. Inside step 1, the local clustering algorithm makes use of the degree of fuzzification to improve the efficiency of the k-medoid by extending the search for medoids, resulting in the best medoids according to the nature of the data in terms of uncertainty. This technique is classified as hierarchical clustering. The model is a distributed computing architecture that runs on top of Apache Spark and is built on the Hadoop MapReduce algorithm. It incorporates the benefits of Hadoop MapReduce, but unlike MapReduce, it allows for the storage of intermediate and performance outcomes of jobs in memory, a concept known as memory computing. Memory computing significantly increases the performance of data processing. As a result, Apache Spark is best adapted to iterative applications like geospatial data mining. This model performs better in terms of execution time, cluster consistency, cluster count, and record count. The proposed model’s major components and their connections are shown in Figure 1, and the following subsections describe each part in depth [6], [7].

**STEP 1: INPUT DATASET**

The employed dataset includes the movement records of taxi cabs in San Francisco, California, United States of America (https://crawdad.org/epfl/mobility/20090224/). It includes the GPS coordinates of approximately 500 taxis gathered in the San Francisco Bay Area during a 30-day period. Every file is named after the taxi’s ID, and the data is displayed as follows: 37.75134 -122.39488 0 1213084687, with the coordinates 37.75134 -122.39488 0 1213084687 denoting the latitude, longitude, whether the taxi is occupied or not (0 for empty), and the timestamp in the UNIX epoch period. The data is combined into a single CSV file, with each row containing the taxi's ID.

**STEP 2: DISTRIBUTED DATASET IN APACHE SPARK FRAMEWORK**

Given the spatial dataset, these enormous databases are much too wide and complicated (with data associated with noise and uncertainty) for humans to efficiently collect valuable knowledge without the assistance of computing resources.
Apache Spark, for example, enables novel and exciting methods for processing and transforming big data, which is characterized as complicated, unstructured, or massive volumes of data, into usable information. Initially, datasets are housed in a database structure such as the Hadoop Distributed File System (HDFS). These datasets are subjected to operations (transformations) that produce resilient distributed datasets (RDDs), or dataset fragments [6]. Thus, RDD is simply a format for describing datasets that are spread across many devices that can run in parallel. RDDs are robust in the sense that they can still be re-computed [7], [9], [21], [22]. On each partition, RDD operations are performed in parallel. Tasks are carried out on the worker nodes that store the data. For additional information, see [18], [21], [22].

Following the division of the dataset into several trunks and storing it in a series of computer nodes, the preprocessing step is performed in parallel. It might not be easy to integrate raw data obtained from one or more sources. Real-world data is often tainted by errors, missed values, and user bias. Preprocessing allows for the assignment of missing values, noise treatment, normalization, transformation, integration, mitigation of inconsistencies, and greatly increased data efficiency by reduction and discretization [23]. Missing values are a frequent occurrence during the acquisition process of clustering strategies in geospatial datasets. In certain approaches, it would be discarded in instances that might have missing values in order to address the issue created by missing values [24].

There are two primary strategies for dealing with noise in clustering approaches [24]. The first is to eliminate noise with the use of data polishing techniques, which identify noisy instances but restore them rather than discarding them by replacing damaged values with more suitable ones. Following that, the corrected instances are reintroduced to the dataset. Whoever it is, it is a difficult job that is typically restricted to low levels of noise. The second method, which is used for the proposed model, is to employ noise filters, which recognize and delete noisy instances in the training data without modifying the clustering technique [25]. Eliminating extraneous data decreases the noise to a degree that would not complicate the study. Outlier identification and treatment were included in this section to either delete records or set a higher and lower limit.

**STEP 3: FIRST STAGE CLUSTERING**

**C. LOCAL CLUSTERING USING ENTROPY-BASED FUZZY K-MEDOIDS ALGORITHM**

Local clusters are strongly reliant on the clustering strategies utilized by the nodes that they belong to. The fuzzy K-medoids algorithm is used to do the local clustering. Each node generates \( k \) local clusters by executing fuzzy K-medoids on its local dataset. The algorithm begins by locating initial medoids, a process that is responsible for choosing the (initial) collection of medoids IDs. While the Fuzzy k-medoid algorithm is straightforward, it does have several pitfalls, including the following: (1) the algorithm is dependent on the initial random sample; using a different initial selection produces different results; and (2) the optimum value of \( k \), which is the number of clusters, is difficult to define. Finally, (3) the algorithm is order-sensitive in the input dataset. Commonly, to decrease the effect of these issues, the rehearsal method is applied and the best result is selected as the output. A rehearsal strategy uses repeated practice of information to learn it [26]. In this paper, a modified fuzzy \( k \)-medoids algorithm is utilized, which covers the problems of clustering based on instance entropy.

In order to improve the efficiency of the fuzzy clustering algorithm, the sum of objects’ entropy as a complementary factor is considered in the objective function. Thus, Eq. (1) plus the sum of objects’ entropy formed a manipulated objective function.

\[
\min Z_{FKM} = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} (e_{ij})^h + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij} \log(e_{ij})
\]

Herein, the amount of fuzziness relies on the \( h \) coefficient. The empirical results show that the amount of \( h \) depends on the type of the data objects. A data object with a small value anticipates a small \( h \), and for a large data object value, a large \( h \) value is expected. Moreover, in [26], it was demonstrated that the \( h \) value should be within a certain interval. If it is too large, the number of unearthed clusters is converging to 1, and for too small a parameter value, the number of uncovered clusters is more than the actual one. The Euclidian distance is applied for the dissimilarity criterion as follows:

\[
d_{ij} (x_i, z_j) = \sqrt{\sum_{l=1}^{m} (z_{il} - x_{il})^2}
\]

The algorithm returns the full distance matrix by evaluating pairwise distances according to a given (dis)similarity metric. Each iteration proceeds as follows: the current medoids IDs are stored based on the initial medoids retrieved using the entropy-based fuzzy \( k \)-medoids algorithm in order to search for convergence later before an ad-hoc function stop Criteria.

The distance matrix can be used to calculate the distances between points and medoids by slicing only the columns corresponding to the medoids’ IDs and then determining which medoid is closest to each point (i.e. row). The new medoid is determined by multiplying the intra-cluster distance matrix by rows and choosing the element with the smallest sum of pairwise distances. Algorithm 1 illustrates the local clustering process [21], [27]. Both local clustering algorithms are performed in parallel without coordination between nodes in this implementation.

**D. CONTEXT-AWARE REDUCTION**

Data reduction methods may be used to create a compressed version of the dataset that is significantly smaller in volume but retains the initial data’s consistency [27]–[33]. Numerous data reduction techniques are described in the literature;
M. M. Madbouly et al.: Clustering Big Data Based on Distributed Fuzzy K-Medoids: Application to Geospatial Informatics

Algorithm 1 Local Clustering Procedure

**Initialization**

Input: \( X_i \): Dataset Fragment, \( k_i \), Initial medoids\(_i\)
Output: \( C_i \): Cluster’s dataset of \( X_i \)

1. Apply entropy-based fuzzy K-medoid algorithm
2. Select the first medoids (initial seeds) based on step 1
3. Evaluate pairwise distances between patterns \( x_j \in X_i, j = 1 \) to \( n_i \) and already-selected medoids, mapping each pattern with its closest medoids.
4. A pattern from the dataset \( X_i \) is selected as new medoids with probability \( \frac{D^*(x)}{\sum_{x \in X} D^*(x)} \)
5. Loop 3–4 until \( k \) initial medoids are selected
6. Cluster all instance in data to the nearest distance using \( D^*_i(x_j) = d(x_i, x_j) \)
7. Calculate new medoids of all cluster. New medoids of each cluster can be calculated using Eq. 8
8. Either medoids stop updating (if the medoids of new cluster and old cluster are equal then it is said to be convergence), or a maximum number of iterations is reached.

Return \( C_i \);

Algorithm 2 Boundary Detecting Procedure

**Input:** \( C_i \): Cluster’s dataset of \( X_i \)
**Output:** Boundary points \( B_c \) of the cluster \( C_i \)

1. **Step 1:** \( B_c \leftarrow C_i \)
2. **Step 2:**
   a. For all points \( p \) in \( B_c \), do
      i. For all points \( q \neq p \) in \( C_i \), do
         1. if \( \text{dist}(q, p) < \varepsilon \) then
            2. Discard \( p \) from \( B_c \)
      end if
   end for
   break
end for

Algorithm 3 Global Clustering Procedure

**Input:** \( D \): Tree degree, \( C_i \): Local cluster’s representatives generated by \( X_i \)
**Output:** \( C_{G_{k,0}} \): Global Cluster’s representative (global results, level = 0)

Repeat

\( \text{level} = \text{treeHeight}; \)
\( \text{Node}_i \) joins a group \( C_{G_{k,level}} \) of \( D \) elements;
\( \text{Node}_j = \text{ElectLeaderNode} \left( G_{K, \text{Level}} \right) \)
For each \( \text{Node}_i \in [\text{space}] G_{K, \text{Level}} \) do
   If \( i \neq j \) then
      Send \( (C_i, \text{Node}_j) \);
   Else
      Receive \( (C \equiv (\{C_i\}, \text{Node}_i)) \);
      \( G_{K, \text{Level}} = \text{Merge} \left( C_i, C \right) \);
   end if
\end{algorithm}

**STEP 3: GLOBAL CLUSTERING “AGGREGATION”**

Global models (patterns) are created during the suggested model’s aggregation phase. This process is still distributed, but unlike the first, it incurs communications costs. This process consists of two primary steps that can be replicated indefinitely until all global clusters have been created. To begin with, each leader collects its neighbors’ local clusters. Second, the leaders can use the overlay strategy to integrate the local clusters. The method of cluster fusion will be repeated before we reach the root node. Global clusters would be contained on the root node. As discussed previously, communicating the local clusters to the leaders during the second step can produce a significant overhead. As a consequence, the aim is to reduce data transmission and computing time while also obtaining reliable global outcomes. In the local clustering phase, only the cluster boundaries are exchanged between system nodes, rather than the whole cluster.

The global clustering model entails sharing the cluster’s reduced version (medoid + cluster boundary points) located in each node with its neighboring nodes. This enables one to
recognize leaders that are overlapping. Each leader makes an effort to unify the group’s overlapping members. As a result, each leader produces new leaders (new clusters). These steps are replicated before the root node is reached. Aggregation of sub-clusters is performed using a tree layout, with the global results placed at the top level of the tree (the root node). The leaders are chosen based on a variety of characteristics, including their capacity, computing strength, and accessibility. The leaders are responsible for combining and regenerating data objects depending on the representations of the local cluster. The aim of this step is to enhance the consistency of global clusters, as local clusters often lack critical details. Algorithm 3 contains the pseudo-code for the algorithm. For additional details, see [21], [30]–[35].

IV. RESULTS AND DISCUSSION

We have conducted extensive experiments on real data sets to verify the performance of the developed model. The prototype verification technique was built in a modular fashion using the MATLAB language and has been implemented and tested on a DellTM InspironTM N5110 Laptop machine by Dell Computer Corporation, Texas, which had the following features: Processor: Intel Xeon E5-2620v3 2.4 GHz (12 CPUs), RAM: 32 GB, SSD Hard Disk: 3x1T.B (Raid 5), System type: 64-bit operating system, and Microsoft Server 2008 R2 Enterprise 64 bit. The suggested model has been run over Hadoop 2.7.1 and Spark 2.2.0 for distributed parallel processing. For spatial query processing, the experiments are conducted using a benchmark taxi dataset [35] that is an open database containing 167 million records in 30 GB. Each record describes a taxi trip made by a particular driver at a particular date and time. Each record has 16 attributes. Only two 2D coordinates, which represent the pickup location, are used in the query. All experiments are done on a sample dataset containing about 30 million records in 5 GB of storage. Here, the results are analyzed and evaluated in terms of clustering error, execution time, clustering accuracy, and convergence time [24].

- Cluster error \( \phi \): is defined to be the sum of the absolute error (distance) as:

\[
\phi = \sum_{i=1}^{k} \phi_{C_i}(\theta_i), \quad \text{where } \phi_{C_i}(\theta_i) = \sum_{O \in C_i} \text{dist}(O, \theta_i)
\]

(10)

\( \theta_i \) denotes the medoid of \( C_i \), and \( K \) is the number of clusters.

- Cluster Accuracy: clustering accuracy is calculated as follows:

\[
\text{Clustering Accuracy} = \frac{\sum_{i=1}^{k} N_{C_i}}{|O|}
\]

(11)

the cluster \( C \) is partitioned into a set of clusters \( c_1, c_2, \ldots, c_k \) on a dataset \( O \) with \( n \) number of objects, \( N_{C_i} \) is the number of objects that occurred in cluster \( C_i \), and \( |O| = n \) represents the number of objects in the dataset.

FIGURE 2. The Clustering Error versus Different Clustering Algorithms.

A. EXPERIMENT 1: (CLUSTERING PERFORMANCE–CLUSTERING ERROR)

This set of experiments compares the efficiency of traditional clustering approaches with proposed ones to demonstrate the superiority of the proposed clustering model as a clustering engine. Herein, evaluation was performed in terms of clustering error. \( K \)-means, \( K \)-prototype, Object Clustering Iterative Learning (OCIL), and similarity-based \( k \)-medoids clustering are the algorithms considered in this analysis. See [23], [24] for information on how each of them works. The cluster error of both existing and proposed clustering strategies is shown in Figure 2. The results indicate that, as opposed to the other methods, the suggested hierarchical clustering model decreased the error rate to 0.36 percent on average. It is difficult to estimate the \( k \) value in the \( k \)-means algorithm, and it does not provide stronger clustering results than the global cluster. Additionally, differing initial partitions result in varying final clusters, and the algorithm does not operate well for clusters of varying sizes and densities. The method then converges to a local mean, rather than a global minimum, using the \( k \)-prototype algorithm.

Additionally, the OCIL algorithm’s similarity computation operations are inefficient. The primary disadvantage of using similarity-based \( k \)-medoids clustering is that the variables contribute individually to the distance calculation. The relation between data points can be dominated by redundant values [24]. As a consequence of these disadvantages, the proposed hierarchical distributed clustering achieves superior performance. The proposed model reduces clustering error by 16 %, 25 %, 51.6 %, and 51.7 %, respectively, as opposed to similarity-based \( k \)-medoids clustering, \( K \)-prototype clustering, OCIL clustering, and \( K \)-means clustering. Multiphase \( K \)-Medoids are superior in terms of clustering error but have a higher difficulty than \( K \)-Means. This disadvantage was overcome by the use of distributed computing. The compact clusters in the proposed model share the primary characteristics
of the local clusters. The precision of the local cluster has a significant impact on the final model’s consistency.

B. EXPERIMENT 2: (CLUSTERING PERFORMANCE—CONVERGENCE TIME)

The convergence time of traditional clustering models and the proposed two-level clustering approach is assessed in this experiment to compare them as a clustering engine (see Figure 3). Less convergence time is needed for the k-prototype, OCIL, and similarity-based k-medoids techniques. As a result of the analysis, it is clear that the proposed model needs additional convergence time. The proposed model accelerates convergence by 51%, 65%, and 24%, respectively, as opposed to similarity-based k-medoids clustering, K-prototype clustering, and OCIL. One reason for these findings is that the proposed model employs two levels of clustering: local clustering and global clustering. While local clustering is built on the Spark architecture, which reduces the run time required to efficiently manage large amounts of data. However, global clusters operate in an iterative fashion, with leaders’ nodes aggregating local clusters; this requires additional convergence time.

C. EXPERIMENT 3: (CLUSTERING PERFORMANCE—CLUSTERING ACCURACY)

In this scenario, the clustering algorithms’ performance is evaluated using a clustering accuracy metric. The precision of clustering for both similarity-based K-medoids [24], [23] and the suggested clustering model is seen in Figure 4. The proposed model significantly increased clustering precision by 3.5%, indicating the proposed work’s efficacy. One reason for this finding is that the aggregation step in the proposed model is structured in such a way that the final clusters are compact and reliable while the overall method is time and memory-free.

V. CONCLUSION AND FUTURE WORK

Geospatial big data is critical in the age of big data, as the majority of data today is basically spatial, collected via ubiquitous location-aware sensors. However, balancing the “Vs” of big data (volume, variety, velocity, veracity, and value) is challenging. High-performance computing is a critical component in solving geospatial problems involving large amounts of data. In the world of geospatial big data, query analysis is a significant obstacle. Managing data to improve query success is an especially difficult challenge. This is because the difficulty of computational data computations involved in querying data is increasing, necessitating the use of a clustering algorithm.

This article discussed the details of the proposed dynamic distributed entropy-based fuzzy clustering paradigm for geospatial big data. The proposed model is divided into two stages. The local clustering process is responsible for creating local clusters centered on a subset of the entire dataset; this stage makes extensive use of the Spark framework’s task parallelism paradigm. The global clustering “aggregation” process is in charge of automatically generating the appropriate number of final compact clusters. These compact clusters share the majority of the characteristics of the nearby clusters. To increase the performance of the k-medoid algorithm and achieve more accurate clusters, a hybrid algorithm combining the k-medoid, fuzzy membership sets, and entropy is suggested. By collecting more reliable results, entropy-based fuzzy k-medoids is used in this method to extend the search for the ideal medoid and boost clustering.

This research demonstrates the critical nature of the K-medoids algorithm, as the precision of the local cluster has a significant effect on the consistency of the final models. Fuzzy K-Medoids for clustering is superior in all respects, including execution time, insensitivity to outliers, and noise reduction, but has the disadvantage of being more complex than K-Means. This disadvantage was overcome by the use of distributed computing. The experimental findings demonstrated that the proposed model achieved an appropriate trade-off between clustering error, clustering precision, and convergence time. Future work includes the following: enhancing the proposed model with the use of other soft computing methods; determining the optimum number of...
processing nodes in advance; and finally, scaling up the proposed model to address the big data analytics issue for geospatial streaming data.

REFERENCES

[1] X. Deng, P. Liu, X. Liu, R. Wang, Y. Zhang, J. He, and Y. Yao, “Geospatial big data: New paradigm of remote sensing applications,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 12, no. 10, pp. 3841–3851, Oct. 2019.

[2] S. Li, S. Dragicevic, F. Castro, M. Sester, S. Winter, A. Colleclin, C. Petit, B. Jiang, J. Haworth, A. Stein, and T. Cheng, “Geospatial big data handling theory and methods: A review and research challenges,” ISPRS J. Photogramm. Remote Sens., vol. 115, pp. 33–119, 2016.

[3] Z. Li, “Geospatial big data handling with high performance computing: Current approaches and future directions,” in High Performance Computing for Geospatial Applications. Cham, Switzerland: Springer, 2020, pp. 53–76.

[4] H. Wang and S. Zhu, “Multisource aggregation search and scheduling for remote sensing data cluster,” IEEE Geosci. Remote Sens. Lett., vol. 17, no. 2, pp. 352–356, Feb. 2020.

[5] S. V. Limkar and R. K. Jha, “A novel method for parallel indexing of real time geospatial big data generated by IoT devices,” Future Gener. Comput. Syst., vol. 97, pp. 433–452, Aug. 2019.

[6] A. Eldawy and M. F. Mokbel, “SpatialHadoop: A MapReduce framework for spatial data,” in Proc. IEEE 31st Int. Conf. Data Eng., Apr. 2015, pp. 1352–1363.

[7] R. K. Lenka, R. K. Barik, N. Gupta, S. M. Ali, A. Rath, and H. Dubey, “Comparative analysis of SpatialHadoop and GeoSpark for geospatial big data analytics,” in Proc. 2nd Int. Conf. Contemp. Comput. Informat. (IC3I), Dec. 2016, pp. 484–488.

[8] K. Lee, R. Ganti, M. Srivatsa, and L. Liu, “Efficient spatial query processing for big data,” in Proc. ACM Int. Conf. Adv. Geographic Inf. Syst., 2014, pp. 469–472.

[9] I. M. Aljawarneh, B. Bellavista, A. Corradi, R. Montanari, L. Foschini, and A. Zanotti, “Efficient spark-based framework for big geospatial data query processing and analysis,” in Proc. IEEE Symp. Comput. Commun. (ISCC), Jul. 2017, pp. 851–856.

[10] A. Shirkhorshidi, S. Aghabozorgi, T. Wang, and T. Herawan, “Geospatial big data analytics,” in Proc. 31st Int. Conf. Data Eng. (ICDE), 2015, pp. 707–720.

[11] E. Schubert, A. Koons, T. Emrich, A. Zülfke, K. A. Schmid, and A. Zimek, “A framework for clustering uncertain data,” Proc. VLDB Endowment, vol. 8, no. 12, pp. 1796–1799, Aug. 2015.

[12] D. N. Pinheiro, D. Aloise, and S. J. Blanchard, “Convex fuzzy k-medoids clustering,” Fuzzy Sets Syst., vol. 389, pp. 66–92, Jun. 2020.

[13] B. Tavakkol and Y. Son, “Fuzzy kernel K-medoids clustering algorithm for uncertain data objects,” Pattern Anal. Appl., vol. 26, no. 1, pp. 1–6, 2021.

[14] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya, S. Foufou, and A. Bouras, “A survey of clustering algorithms for big data: Taxonomy and empirical analysis,” IEEE Trans. Emerg. Topics Comput., vol. 2, no. 3, pp. 267–279, Jun. 2014.

[15] A. Ben Ayed, M. Ben Halima, and A. M. Alimi, “Survey on clustering methods: Towards fuzzy clustering for big data,” in Proc. 6th Int. Conf. Soft Comput. Pattern Recognit. (SoCPaR), Aug. 2014, pp. 331–336.

[16] S. Arora and I. Chana, “A survey of clustering techniques for big data analysis,” in Proc. 5th Int. Conf. Confluence Next Gener. Inf. Technol. Summit (Confluence), Sep. 2014, pp. 59–65.

[17] Z. Shi and L. Pun-Cheng, “Spatiotemporal data clustering: A survey of methods,” ISPRS Int. J. Geo-Inf., vol. 8, no. 3, pp. 112, 2019–16, 2019.

[18] X. Yang and L. Lian, “A new data mining algorithm based on MapReduce and Hadoop,” Int. J. Signal Process., Image Process. Pattern Recognit., vol. 7, no. 2, pp. 131–142, Apr. 2014.

[19] B. Mirzaoleiman, A. Karbasi, R. Sarkar, and A. Krause, “Distributed sub-modal maximization: Identifying representative elements in massive data,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 2049–2057.

[20] A. Ene, S. Im, and B. Moseley, “Fast clustering using MapReduce,” in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2011, pp. 681–689.

[21] X. Yue, W. Man, J. Yue, and G. Liu, “Parallel K-medoids++ spatial clustering algorithm based on MapReduce,” 2016, arXiv:1608.06861.
SAAD M. DARWISH received the B.Sc. degree in statistics and computer science from the Faculty of Science, Alexandria University, Egypt, in 1995, the M.Sc. degree in information technology from the Department of Information Technology, Institute of Graduate Studies and Research (IGSR), University of Alexandria, in 2002, and the Ph.D. degree from Alexandria University for a thesis in image mining and image description technologies. Since June 2017, he has been a Professor with the Department of Information Technology, IGSR. He has supervised around 80 M.Sc. and Ph.D. students. He is the author or coauthor of more than 50 papers publications in prestigious journals and top international conferences. He has received several citations. His professional and research interests include image processing, optimization techniques, security technologies, database management, machine learning, biometrics, digital forensics, and bioinformatics. He has served as a reviewer for several international journals and conferences.

NOHA A. BAGI received the bachelor’s degree in computer engineering from the Alexandria Higher Institute of Engineering and Technology, Alexandria, in 2013. She has completed postgraduate courses in data analysis, Oracle administration, Oracle developer, Python, SQL server developer, Microsoft private cloud under supervision, graphic design, and Android programming. She is currently undergoing on-the-job training at Egyptian Telecom, Banha Company for Electronic Industries, and Hayek Company for programming, app development, and web development. Since 2018, she has been working as an IT Engineer and a Senior Programmer at Alexandria Water Company. She is also a full-time technical job in the main office. She is a Microsoft Certified Professional. Her professional and research interests include optimization techniques, security technologies, and database management.

MOHAMED A. OSMAN received the B.Sc. degree in management information systems from the Higher Institute of Management and Information Technology, Kafr El-Sheikh, Egypt, in 2009, and the M.Sc. degree in computer and information systems from the Department of Information Systems, Faculty of Management Sciences, Sadat Academy for Management Sciences, in 2019. His professional and research interests include optimization techniques, security technologies, and database management.