symmetry

MDPI

Article

Big Data Clustering Using Chemical Reaction Optimization Technique: A Computational Symmetry Paradigm for Location-Aware Decision Support in Geospatial Query Processing

Ali Fahem Neamah [1], Hussein Khudhur Ibrahim [1], Saad Mohamed Darwish [2,*] and Oday Ali Hassen [3]

[1] College of Computer Science & Information Technology, Wasit University, Kut 52001, Iraq
[2] Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, 163 Horreya Avenue, El–Shatby, Alexandria 21526, Egypt
[3] Ministry of Education, Wasit Education Directorate, Kut 52001, Iraq
* Correspondence: saad.darwish@alexu.edu.eg; Tel.: +20-122-263-2369

Abstract: The emergence of geospatial big data has opened up new avenues for identifying urban environments. Although both geographic information systems (GIS) and expert systems (ES) have been useful in resolving geographical decision issues, they are not without their own shortcomings. The combination of GIS and ES has gained popularity due to the necessity of boosting the effectiveness of these tools in resolving very difficult spatial decision-making problems. The clustering method generates the functional effects necessary to apply spatial analysis techniques. In a symmetric clustering system, two or more nodes run applications and monitor each other simultaneously. This system is more efficient than an asymmetric system since it utilizes all available hardware and does not maintain a node in a hot standby state. However, it is still a major issue to figure out how to expand and speed up clustering algorithms without sacrificing efficiency. The work presented in this paper introduces an optimized hierarchical distributed k-medoid symmetric clustering algorithm for big data spatial query processing. To increase the k-medoid method's efficiency and create more precise clusters, a hybrid approach combining the k-medoid and Chemical Reaction Optimization (CRO) techniques is presented. CRO is used in this approach to broaden the scope of the optimal medoid and improve clustering by obtaining more accurate data. The suggested paradigm solves the current technique's issue of predicting the accurate clusters' number. The suggested approach includes two phases: in the first phase, the local clusters are built using Apache Spark's parallelism paradigm based on their portion of the whole dataset. In the second phase, the local clusters are merged to create condensed and reliable final clusters. The suggested approach condenses the data provided during aggregation and creates the ideal clusters' number automatically based on the dataset's structures. The suggested approach is robust and delivers high-quality results for spatial query analysis, as shown by experimental results. The proposed model reduces average query latency by 23%.

Keywords: evolutionary clustering; optimization; location-aware decision support; query processing; geospatial data management

Citation: Neamah, A.F.; Ibrahim, H.K.; Darwish, S.M.; Hassen, O.A. Big Data Clustering Using Chemical Reaction Optimization Technique: A Computational Symmetry Paradigm for Location-Aware Decision Support in Geospatial Query Processing. *Symmetry* **2022**, *14*, 2637. https://doi.org/10.3390/sym14122637

Academic Editor: João Ruivo Paulo

Received: 29 October 2022
Accepted: 2 December 2022
Published: 13 December 2022

1. Introduction

Big data is a byproduct of the fast growth of information technology, which has resulted in a flood of information from a wide variety of sources. Despite its ubiquitous presence, big data from cellular phones and GPS trackers has yet to fulfil its immense potential in environmental monitoring [1]. Location information is part of geospatial big data. Since the vast majority of information produced in the era of big data is spatial and gathered by location-aware sensors, position awareness is essential. Multiple efforts have

tried to use geographic big data for human activity tracking and environmental/urban studies. It is possible to examine the historical patterns of transportation services that operate inside certain regions to better detect urban functions and create job-housing functional dynamics than with conventional remote sensing images [2].

Because of its volume, velocity, diversity, veracity, and value, big data makes efficient data collection a challenge. Modern storage, maintenance, and protection requirements are required for vast amounts of data on a daily basis since traditional processing infrastructures can't manage such a huge volume of composite data in so many different formats. Cloud computing systems, which fall under the umbrella of High-Performance Computing (HPC), have advanced parallel processing capability, the capacity to scale to enormous sizes, and flexibility in handling enormous data sets [3–5]. Recently, shared-nothing architecture (SNA), a data storage paradigm used in high-performance computing, has become the de facto standard for storing data, and an HPC cluster may simply add nodes to increase both storage and processing capability. Its capacity for centralized processing also reduces network data transmission, and its capacity for local storage does away with the possibility of a single point of failure. One well-known use of SNA is the Hadoop Distributed File System (HDFS) [6,7].

In big data, the analysis of geospatial queries is complex. Enquiring such huge volumes of data is beyond the capabilities of conventional database management systems [8]. New management challenges have emerged for avoiding query processing bottlenecks brought on by an uneven distribution of work in the cloud [9]. Geospatial clustering combines geographic points into "clusters. The clustering of geospatial big data has to grow and speed up without losing effectiveness. You'll need an economical processing model with a low computing cost if you want to make use of this massive volume of data [10]. There are three ways to increase big data clustering speed and scalability. (1) Using sampling-based algorithms reduces iteration. Sampling-based algorithms cluster a portion of that data, not the complete set. (2) Randomize data to reduce its dimension. Dimensionality affects the complexity and speed of clustering techniques. (3) Use parallel and distributed methods to speed processing and enhance scalability. Traditional parallel and data-intensive applications use concurrent computing.

The field of computational symmetry studies the use of computers to model, evaluate, create, and manipulate symmetries in digital form. Symmetry plays a crucial role in the design and development of algorithms. Often, symmetry facilitates and simplifies the probabilistic analysis of randomized algorithms. Incorporating clever techniques of symmetry or tie-breaking into algorithms may sometimes improve their performance. The suggested model relies on symmetric clustering, in which each node simultaneously runs applications and monitors other nodes. The absence of a hot standby key makes this clustering more efficient than asymmetric clustering.

If a symmetric clustering technique is going to make use of Spark, it has to be modified so that it can make use of Spark's distributed computing environment. The algorithm's strengths—including its fault-resilient distributed dataset (RDD), in-memory iterative processing, minimal disc I/O load, direct acyclic graph execution technique, sophisticated local data caching system, quicker distributed file system, and Spark fault-tolerant mechanism—should serve it well [6,7]. Since most conventional clustering methods presume the data to be located in a single repository, dealing with the dispersed nature of big data presents a difficulty for big data clustering methods. Most of the time, the number of clusters present, the shape of the clusters, and the presence or absence of outliers are unknown in a clustering situation. This means that it is still important to do effective clustering regardless of whether one is aware of any preexisting clusters in the data.

1.1. Problem Statement and Aim of the Work

The amount and complexity of geographic data are growing at an exponential rate, beyond the capacity of traditional data processing methods and systems. Geospatial big data requires efficient data gathering and analysis frameworks. Large volumes of

geographic data increase query execution time since it's hard to scan the complete set in an acceptable amount of time. Clustering is beneficial. Contemporary researchers have suggested several clustering algorithms. These algorithms suffer from dimensionality or need several iterations to cluster. There have been few publications on concurrently distributed Apache Spark clustering research. Noisy input makes clustering an item harder, affecting clustering performance. Successful algorithms can handle outliers and noise. This research provides a dynamic clustering strategy for processing spatial queries on geospatial big data after examining recent clustering trends and advancements in response to the difficulties presented by big data. By repositioning the cluster heads to the most important nodes, this method creates robust clusters quickly and easily. The number of global clusters might be left undetermined; instead, it may change over time. The medoid of the cluster is optimized at this stage.

1.2. Contribution

The goal of this research is to improve the scalability and speed of geospatial big data queries by using a distributed, optimized dynamic clustering approach implemented in Apache Spark. Parallel processing, wherein each individual node generates its own set of local clusters from the whole dataset, and aggregate processing, in which these clusters are merged in order to form clusters that are both compact and stable, make up the two phases of the recommended distributed dynamic clustering approach. Cluster representatives reduce the overhead of these exchanges. Clustering in a distributed manner is an adaptive method for geographic data as the number of valid clusters is not fixed beforehand (as an input parameter). This solves one of the k-medoids' problems. The proposed model is novel in that it combines the CRO algorithm, which aids in obtaining the optimum solution and best medoids for large geospatial datasets with a high number of outliers. The best medoids are utilized as seeds for the two-phase dynamic clustering.

This article continues as follows: Section 2 reviews the related big data clustering approaches. Section 3 provides an outline of the recommended strategy. Section 4 tests the suggested approach and analyzes the results. Section 5 summarizes the findings of the research and recommends next directions.

2. Related Work

Many studies have been conducted to enhance clustering in big data techniques. Each researcher has a distinct method and idea. Each researcher has a distinct method and idea. Studies [10–14] show that no one clustering method is adequate for solving all the problems associated with huge data. Implementing the techniques can cluster vast volumes of data, but the techniques are complex. MapReduce or Spark will help implement these parallel algorithms. To analyze a lot of data with enough resources, we need to make clustering methods take less time and use less memory. The authors in [15] adapted MapReduce to medoids. During mapping, it places each object next to its closest medoid, and during reduction, it moves the real medoid to the center of the group. To parallelize medoid updating, it uses a local search instead of the traditional partitioning around medoids. This method is quick but inaccurate. The GREEDY algorithm [16] finds k-medoids from each data partition. Due to the partitioning requirement, GREEDY's sample size and number depend on the overall data size. Large data sets make this technique unworkable.

Using simultaneous analysis and a combination of resampling and the weighted k-medoids technique, the authors in [17] provide a single optimal sample. As the number of distances collected rises, its efficiency decreases. The work presented in [18] suggests a MapReduce-based k-medoids++ spatial clustering algorithm for large geographical data. Initialization and MapReduce decrease iterations. k-medoids++ has two advantages over previous spatial clustering methods. To decrease iterations, k-medoids++ spatial clustering implements an efficient initial medoids search method. Second, MapReduce parallelizes k-medoids++ spatial clustering. Their approach outperforms ordinary k-medoids and is scalable effectively on commodity hardware. Using Apache Spark, the same authors sped

up k-medoids clustering [19]. As a constant distance separates any two patterns, k-medoids iterations may avoid calculating pairwise distances between cluster components. (Dis) similarity measures determine validity. Real-world route map datasets revealed distributed implementations' utilization of structured data.

M. Bendechache et al. [20] recommended combining local and global data. Each node builds local clusters depending on its subset. During this phase, parallelism is completely employed. MapReduce makes this framework easy to implement. Distributed clustering produces a dynamic number of global clusters. In [21], the same author improved dynamic distributed clustering by adding a phase to efficiently reduce data. This technique was designed for geographic data sets, and it helps cut down on the transmission overhead. If local clustering is NP, then the technique scales effectively and the results are robust to changes in communication protocol.

To address the multiple-criteria decision analysis issue of optimal business site selection from a given collection of candidate sites, the authors in [22] provide a combination of the Analytic Hierarchy Process and the Technique for Order of Preference strategies. A step-by-step case study is provided to demonstrate the efficacy of the suggested method by finding a prime spot in New York City for a gasoline station. With the goal of estimating potential geographic references expressed by the users, the authors in [23] introduced Paval (location-aware virtual personal assistant), a semantic assisting engine for suggesting local points of interest (POIs) and services through the analysis of users' natural language queries. The system uses natural language processing (NLP) and semantic approaches to provide suggestions for POIs and services within the user's current geographic location that best meet their needs, as determined by a search of the Km4City Knowledge Base. By comparing the proposed system to the most popular virtual assistants, like Google Assistant, Apple Siri, and Microsoft Cortana, with a focus on the request of geolocated POIs and services, the authors show that it has promising capabilities in successfully estimating the users' information needs and multiple geographic references.

In [24], the authors introduced GEOSPARKVIZ, a fully functional system that facilitates the loading, processing, integration, and execution of GeoViz operations on massive amounts of geographical data. The GEOSPARKVIZ project is an extension of a cutting-edge distributed data management system that adds native functionality for geographic map display. The system incorporates the fundamental phases of map visualization, such as pixelizing spatial objects, pixel aggregation, and map tile rendering, into a collection of massively parallelized map construction operators. Because of this, the system can simultaneously improve both the spatial query operators and the map building operators. Additionally, GEOSPARKVIZ has a spatial partitioning operator that is GeoViz-aware, which distributes GeoViz workloads evenly among the cluster's nodes.

In [25], the authors offered a collaborative filtering-based methodology for predicting the quality of GI services in a given area. From the user's and the GIService's viewpoints, the model employs a mixed collaborative filtering (CF) technique based on the time zone characteristic. GIServices' degree of resemblance to a target was quantified using a time zone-adjusted Pearson correlation coefficient technique, which aided in the identification of services with a high degree of similarity. In [26], the authors evaluated the performance of numerous parallel and distributed Distance Join Queries (DJQ) algorithms, comparing two of the most recent and prominent ones, Spatial Hadoop and Location Spark, in different scenarios with huge real-world spatial datasets. When large spatial datasets are joined, SpatialHadoop proves to be the superior system, but when medium spatial datasets are combined, LocationSpark emerges as the clear winner in total execution time efficiency (Spark's in-memory processing). However, more memory needs to be set aside when big spatial datasets are used in DJQs (especially Location Spark).

GeoSpark will be used in the present endeavor since it is a vital solution that has been extensively applied to spatial data. Specifically, it operates on top of Apache Spark, the key framework utilized by the scientific community and companies for big data transformation, processing, and visualization. To this purpose, the authors in [27] concentrated on trajectory

data format so as to be adaptable to the GeoSpark environment, and a GeoSpark-based solution is devised for the efficient administration of actual spatio-temporal data. The next stage is to acquire a better knowledge of the data through the implementation of *k*-nearest neighbor (k-NN) searches. In [28], the authors developed and deployed a novel Voronoi-Diagram-based data splitting approach in SpatialHadoop. SpatialHadoop also has a revised version of the Nearest Neighbors Join Query and Closest Pairs Query MapReduce algorithms that make use of the new partitioning mechanism.

In [29], the authors presented a PID-based kNN query processing method (PIDKNN) for geographic large data using Spark, bringing proportional integral derivative (PID) control technology into kNN query processing. This approach employs a grid partition technique to split the whole data space into uniform grid cells, after which a grid-based index is built. The PID parameters are modified once the geographic data is grouped using a grid-based density peak clustering method. Predicting the radius expansion step size of kNN searches using the PID algorithm improves their throughput. This makes it possible to implement kNN query processing with a feedback-based, variable query radius growth step. The PIDKNN algorithm outperforms other approaches currently in use for processing parallel kNN queries in terms of both performance and scalability, as shown by a number of experiments. For more recent research in this field, readers can refer to [30–35].

According to the study, earlier research focused on the fact that (1) no clustering technique can handle all big data concerns. (2) Parallel categorization helps cluster big data, but implementation is tough. (3) The MapReduce architecture is suitable for parallel algorithms but not iterative ones. (4) No current k-medoids technique is accurate or efficient for large amounts of geographic data. Existing methodologies for query processing are insufficient to quickly offer correct answers in the context of massive data sets. Few attempts have been made to build a unique dynamic distributed k-medoids clustering approach for geospatial big data with many outliers. K-medoids are a kind of clustering method that has been studied extensively in recent years, and its potential for parallelism has been the subject of several academic articles, but applying it to spatially large data presents significant challenges due to the presence of many outliers and disturbances. In the next part, we'll go over the proposed model in more depth; it uses a dynamic distributed clustering approach to deal with spatially large data query processing and includes the CRO algorithm to improve the selection of the initial medoids.

3. The Proposed Geospatial Big Data Query Model

3.1. Problem Formulation

K-medoids is a partitioned clustering algorithm for datasets composed of N_p patterns, that is, it partitions a dataset $D = \{x_1, x_2, \ldots, x_{N_P}\}$ into k non-overlapping groups (clusters), i.e., $C = \{S_1, \ldots, S_K\}$ such that $S_i \cap S_j = \varnothing$ if $i \neq j$ and $\cup_{i=1}^{k} S_i = D$. To locate such clusters, *k*-medoids seeks to minimize the following objective function: the Within-Cluster Sum-of-Distances (WCSoD) [36].

$$WCSoD = \sum_{i=1}^{k} \sum_{x \in S_i} d(x, m(i))^2, \quad (1)$$

$m(i)$ is the medoid for the ith cluster, and it is defined as the data point that minimizes the sum of pairwise distances inside the cluster, where $d(.,.)$ is a (dis)similarity measure. Large data sets present the greatest difficulty when attempting to employ clustering algorithms, since doing so requires both scalable storage and a distributed querying strategy. In order to find and eliminate correlations in massive datasets, clustering approaches are much desired. However, there are a number of obstacles to bringing these methods to bear on large-scale geographic datasets, such as the volume of data and the complexity of certain algorithms. Thus, it is necessary to look into parallel clustering techniques that are both efficient and accurate. Using distributed algorithms is essential for achieving better scalability and accuracy when dealing with massive volumes of data. Given that huge data

sets are often generated in dispersed places and might benefit from being processed on their local hosts, distributed clustering is an attractive option.

3.2. Proposed Method

For the purpose of studying huge geospatial datasets, this section outlines an optimized distributed k-medoids clustering model. This approach comprises two phases: sampling-based local clustering and aggregation-based global clustering. By expanding the exploration for medoids and picking the ideal medoids, the local clustering technique boosts the effectiveness of k-medoids using the Chemical Reaction Optimization (CRO) algorithm. Hierarchical clustering describes this approach. The model uses Hadoop MapReduce and runs on Apache Spark. It integrates Hadoop MapReduce's features, but unlike MapReduce, it stores intermediate and performance task outputs in memory (memory computing). Memory computing boosts data processing. This model has a faster execution time and more consistent clusters. Figure 1 shows the proposed model's primary components. The following subsections discuss each part.

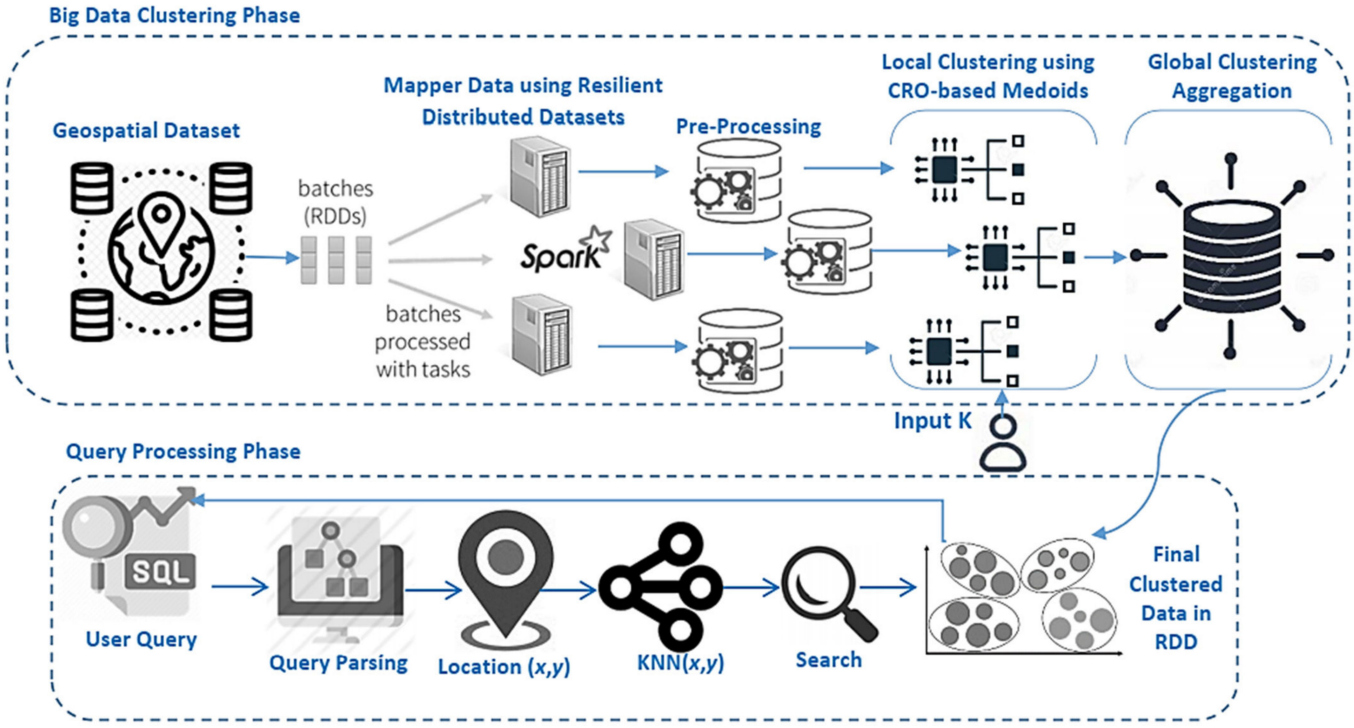

Figure 1. The Proposed Dynamic Distributed Clustering –based Geospatial Query Model.

Step 1: Dataset

The dataset comprises taxi cab movements in San Francisco, CA, USA. It comprises 500 taxis' 30-day GPS locations from the San Francisco Bay Area. Each file contains the taxi's identification (ID) and displays the taxi's latitude, longitude, occupancy, and UNIX epoch time.

Step 2: Distributed Computing with Apache Spark

Due to the inclusion of geographic data, the size and complexity of these massive databases make their utilization impractical without access to appropriate computational means. For example, Apache Spark paves the way for cutting-edge techniques to be developed that can convert enormous data volumes into useful knowledge. Hadoop Distributed File System (HDFS) is where datasets are first kept. A resilient distributed dataset (RDD) is the result of transforming this data [6,7,9,19]. RDD operations are parallelized on each partition. Data-storing worker nodes perform tasks [19]. The primary Apache Spark steps are shown in Figure 2.

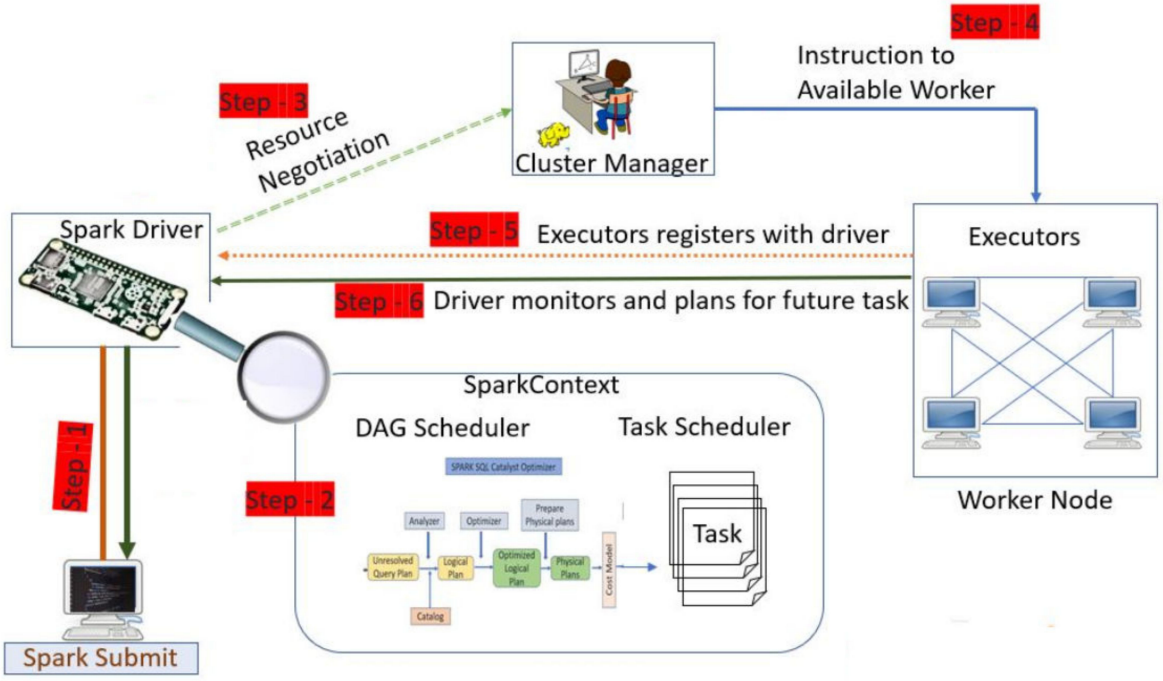

Figure 2. Apache Spark Architecture.

After partitioning and storing the dataset in nodes, preprocessing is conducted in parallel. Integrating raw data from several sources may be difficult. Practical data is contaminated by faults (errors) and missing values. Preprocessing assigns missing values, treats noise, normalizes, transforms, integrates, mitigates inconsistencies, and reduces and discretizes data [37]. Geospatial clustering datasets often include missing values. In certain techniques, missing data is eliminated to solve the problem [38]. Data scientists have two main options for dealing with missing data: imputation and data removal. The imputation technique generates acceptable replacements for missing information. This method is especially beneficial when there is a small amount of missing information. When there is a large amount of information that is missing, it is impossible to build a reliable model. The alternative is to delete the data. Deleted related data may help eliminate bias when dealing with missing data that occurred by chance. If there are not enough observations to provide a trustworthy analysis, then omitting data may not be the best decision. It may be necessary to keep an eye out for certain things at certain times. In our case, the imputation option based on the use of the attribute mean (the average value for that attribute) to fill the missing value is employed.

Clustering has two main steps for noise reduction [38]. First, use data cleaning to eliminate noise by restoring noisy instances rather than rejecting them. Then, the repaired instances are reintroduced. It's a tough task with low noise levels. Smoothing data is the process of reducing or eliminating noise in the data. For the purposes of smoothing, you may utilize the following methods: Binning is a method wherein the data is first sorted and then divided into groups with similar frequency distributions. The bin mean, bin median, or bin border may then be used in place of the noisy data. When there is extraneous information present, regression is utilized to smooth it out. Regression is useful for determining the most appropriate variable to use in an investigation. Noise filters are used to identify and remove noisy training data without affecting clustering [39]. Eliminating unnecessary data reduces noise and simplifies the investigation. Outlier detection and treatment were incorporated to discard data or establish a high and low boundary. Some techniques used to deal with outliers include: (1) Deleting observations; (2) Transforming values; (3) Imputation; (4) Separate treatment; and (5) Deleting observations. Sometimes it is best to completely remove those records from your dataset to stop them from skewing your analysis. In our case, the imputation technique is employed.

Step 3: Clustering, Initial Phase

Phase 1: The CRO Algorithm for Local Clustering

Local clusters rely heavily on the nodes' clustering techniques. Local clustering uses k-medoids. The method starts by finding starting medoids, which choose the (initial) medoids' IDs. The k-medoid method is simple but has various problems. (1) The procedure depends on the original random sample; various samples generate different results. (2) Finding the optimal value for k, the number of clusters, may be challenging. (3) The method relies on the input dataset's natural order. To overcome these problems, the proposed solution integrates the CRO algorithm to boost the k-medoid clustering algorithm by picking cluster medoids using the CRO meta-heuristic algorithm rather than randomly. The CRO algorithm can produce more optimal solutions by using objective functions, and the optimal solution or medoid for the cluster will be chosen using the objective function with the highest value. Herein, CRO is utilized as a blackbox with its default parameters. The algorithm's steps can be outlined as follows: (1) Begin with the initial population, which is composed of a collection of individuals, each of whom possesses potential energy (PE). Such CRO parameters, such as population size, number of iterations, and buffer, should be specified initially. (2) Create new reaction mixtures through chemical reactions. (3) Maintain a current state of potential energy. (4) Repeat steps before the termination state is reached. Figure 3 shows the flowchart of chemical reaction optimization.

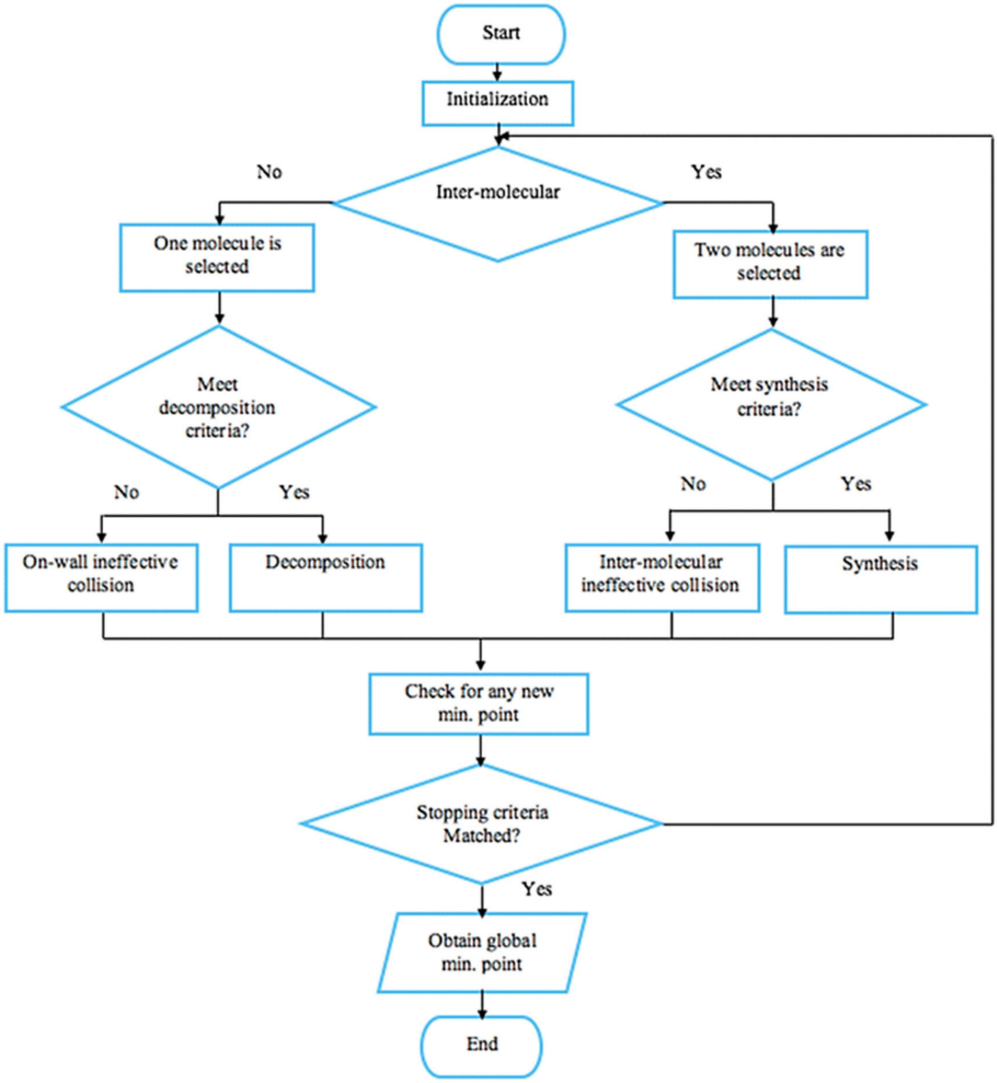

Figure 3. Flowchart of chemical reaction optimization.

The CRO algorithm differs from other evolutionary methods primarily in that the population size is subject to change at the end of each iteration, while in other methods the population size is fixed from the beginning to the end of the run. In the CRO community, the molecule is the basic building block since it is assumed to store the individual's fitness in the form of potential energy. Depending on the problem at hand, some aspects of a molecule—its kinetic energy, its structure, and so on—may be more important than others. Molecules may be altered by either intra- or intermolecular collisions (two or more molecules collide with each other). The end goal is to transform it into a stable compound with the lowest feasible potential energy. For additional details, see [40].

Combining the principles of natural chemical reactions with mathematical optimization, CRO is distinguished by its ability to converge quickly and its need for fewer adjustable parameters. See [41] for further information on CRO and its applications. CRO, unlike other optimization approaches, does not need a large number of initial parameters; just the required number of initial reactants is required for implementation. In this study, we employ binary coding for reactants and the uniform population approach to generate the starting population. Initialize the starting reactants uniformly throughout the viable search space. Thus, it is possible to acquire all vectors in a space by linearly combining components of the base set. The absence of one member from the base set causes a decrease in the associated dimension. Consequently, it is essential that the starting reactants comprise each member of the base set. Moreover, the starting reactants must be uniform and maintain the base set. The CRO demonstrates great performance in two crucial optimization metaheuristics characteristics: intensification and diversification. It also gets the benefits of the genetic algorithm (GA) since it employs the crossover operator and mutation that are often used in GA.

Phase 2: Context-aware Reduction

Data reduction strategies may provide a smaller dataset with the same consistency [42,43]. Samples, data compression, and data discretization are discussed in the literature. The bulk of these techniques are focused on database storage size rather than information (context-aware). The most basic method of information reduction is sampling, in which we randomly choose samples from the complete dataset. Random, deterministic, and density-biased sampling strategies are explained. Naive sampling techniques are unsuitable for real-world applications with noisy data because algorithm performance might vary substantially. Random sampling ignores knowledge in samples not picked for the smaller subset. The basic idea behind random projection is to use a matrix whose column lengths are all one to transfer high-dimensional data to a lower-dimensional subspace. Controls are applied to the size and distribution of random projection matrices to ensure that the distances between any two dataset samples remain unchanged. Therefore, the distance-based approach may benefit from random projection as an approximation strategy. To construct a reduction set, the suggested model follows the same reduction technique as [44]. The employed technique concerns the shape and density of the clusters that are formed. Its border points indicate the shape of a cluster. Meanwhile, its density may be represented by a mean density value or by a series of density values representing the density in distinct regions of the cluster. The reduction set that may be seen as the local model at site S_i in the system will be constructed using cluster boundary and density information. The server receives this local model and uses it to generate global models. Herein, only cluster members who account for 1% to 2% of data are sent. Medoids and border points are cluster members.

Step 4: Global Clustering "Aggregation"

In the recommended model's aggregation phase, global models are formed. Unlike the first phase, this distributed operation incurs communications expenses. This procedure has two parts that may be repeated forever to construct all global clusters. First, each leader gathers its neighbors' clusters. Leaders can combine small clusters with the overlay method. Before reaching the root node, we'll repeat cluster fusion. The root node holds global clusters. Costs may be incurred in stage two due to local cluster communications to top-level managers. For nodes to interact in local clustering, just the cluster boundaries

need to be shared. Global clustering involves sharing each node's reduced cluster (medoid + boundary points) with its neighbors. This identifies overlapping leaders. Each leader unites overlapping group members to create new ones. These steps must be repeated until the first node is reached. After aggregating data from smaller clusters, the big view is placed at the tree's highest node (the root node). A leader is selected based on their capacity, computational power, and ease of access. Leaders combine and regenerate data items based on cluster representations. This step improves global cluster consistency, since local clusters lack essential information [21,45,46].

Step 5: User Querying Processing

The user specifies a query in accordance with the requirements. The query has entered the users' query processing unit, which is responsible for query decoding. This is advantageous for mapping queried data and moving to the appropriate partition in order to retrieve the desired data from the final clustered data contained in the RDD. Lexical analysis is used to verify the syntax and derive significant tokens from the user question included in the input. Tokens are produced using a lexical analyzer from a stream of input string characters that have been broken down into small components to form meaningful expressions [47].

The use of spatial querying inside a geospatial big data environment is used to demonstrate the proposed distributed clustering model's efficacy. A spatial query is subtype of a database query that spatial databases provide. It enables the representation of basic geometric structures such as points. The queries vary significantly from non-spatial SQL queries in many respects [48]. The two most often used spatial queries, given the derived tokens, are the k-nearest neighbor (kNN) queries and window queries [48]. The kNN queries return the k-nearest objects to the querying user's position, while window queries return all objects within the requested window. A real-world example of a kNN question is "tell me the three closest restaurants to my current location"; a window query is "display all coffee shops inside the navigation frame based on my place."

KNN is a non-parametric algorithm, which implies that some assumptions must be fulfilled in order to execute it. Additionally, KNN does not create a standard explicitly; it simply marks new data entries based on learning from historical data. The latest data entry will be associated with the closest neighbor's plurality class. KNN is a memory-based technique due to its instance-based learning. The classifier automatically adapts when new training data is collected. This lets the algorithm react quickly to changes in the input when it is being used in real time [49]. In order to locate the K spatial objects that are closest to the query location, KNN queries are employed. To do so, the KNN query (Spark object, query point, K) API from Spark data frames is used [49].

4. Experimental Results

In this section, many experiments have been performed with real data sets to test how well the model works. The method of prototype verification was made using MATLAB in a modular way, and it has been tested on a DellTM InspironTM N5110 Laptop computer, Dell Computer Corporation. For distributed parallel processing, the model that was suggested was run on Hadoop 2.7.1 and Spark 2.2.0. For spatial query processing, studies use a 167 million-record, 30-GB benchmark taxi dataset [50]. Each record details a taxi ride at a certain date and time. The query uses two 2D pickup coordinates. All tests use a 30 million-record, 5 GB sample dataset. Clustering error, average latency, clustering accuracy, and convergence time are assessed [37].

4.1. Experiment 1: KNN Search Analysis

This series of tests evaluates the efficiency of various queries in terms of average latency against datasets of varying sizes. To evaluate the performance of the suggested model, various datasets ranging in size from 5 million to 30 million records (800 MB to 5 GB) are created from the benchmark taxi dataset. The pickup position is regarded as a two-dimensional spatial object. For assessment purposes, various test cases involving

KNN search queries are developed. Ten test queries are executed to determine the nearest K (K = 3 to 15) pickup locations to a specified query point. The sampling procedure is used to pick different query points from densely populated regions. Table 1 shows KNN query efficiency. The average delay grows with the number of kNN neighbors and dataset rows, as expected. A 5-million-row dataset's average latency grows 4% between K = 3 and K = 15. The average rises 3.7% as the dataset expands from 5 million to 30 million rows.

Table 1. Performance of KNN Query Regarding Number of *K* and database size.

No. of K	K = 3			K = 5			K = 15		
Number of Rows in million	5	15	30	5	15	30	5	15	30
Average Latency (Sec)	10	23	50	20	48	90	52	140	200

In general, utilizing distributed clustering (the Spark framework) is regarded as a critical aspect of the proposed model since it reduces the run time required to achieve the final clustered data. This contributes to the acceleration of spatial query analysis in a large data environment. Additionally, the procedure for aggregating spatial local clusters into global clusters requires just a small portion of the original datasets to be exchanged, which is extremely effective. In general, the clustering construction stage (offline) requires more time, while the query stage (online) requires significantly less.

4.2. Experiment 2: Comparative Study for KNN Search

The second set of experiments compares the suggested model's query performance in terms of average latency to a similar technique described in [51], which is developed on top of the Spark architecture and a NoSQL database, Cassandra. The same dataset as in the previous experiment is used. Ten test queries are executed to determine the nearest *K* pickup locations (K = 5 to 25) to a specified query point. Table 2 illustrates the efficiency of the KNN search query on both versions. The proposed model reduces average latency by 23%. The proposed model's dominance stems from the fact that it is founded on two levels of clustering: local and global clustering. The first step produces local models or trends, while the second phase attempts to combine the effects of the first phase to produce global models. In contrast to the comparative model, it is based on a single level of clustering. As a result, the total number of clusters produced by the proposed model is significantly less than the alternative approach. By and large, looking through a few clusters is faster.

Table 2. Comparative Study for KNN Query Performance (Average Latency in Seconds).

Database Size (Number of Rows in Million)	5	10	15	20	25	30
Proposed Model	16	33	45	76	48	90
Related Technique [51]	22	42	57	88	95	113

4.3. Experiment 3: Clustering Error

This series of tests contrasts standard and suggested clustering methodologies to show the superiority of the proposed model. Clustering errors were evaluated. This investigation considers several clustering techniques briefly discussed in [37]. Table 3 displays the cluster error of both conventional and novel clustering methods. The recommended hierarchical clustering model was able to reduce the error rate to an average of 0.36%, which is much lower than the other approaches. K-means clustering is not as effective as the global cluster, and it is difficult to determine the *k* value. Furthermore, the approach performs poorly for clusters of variable dimensions and densities, and different beginning partitions lead to different end clusters. The k-prototype ensures the approach converges to a local average, not a global minimum. Furthermore, the similarity computation procedures in the OCIL

method are not very efficient. Each variable affects distance calculation independently in similarity-based k-medoids clustering. Redundant values might dominate a relationship between data points. These drawbacks are what make the proposed hierarchically distributed clustering so effective.

Table 3. The Clustering Error Analysis.

Clustering Algorithm	Clustering Error
C1 (k-means)	0.41
C2 (k-prototype)	0.39
C3 (Object Clustering Iterative Learning)	0.25
C4 (Similarity-based k-medoids clustering)	0.23
Proposed Model	0.19

Clustering errors are cut by 16%, 25%, 51.6%, and 51.7%, respectively, when the suggested method is used instead of the C4, C2, C3, and C3 clustering algorithms. Clustering errors are reduced more effectively using multiphase k-medoids, although they are more complicated to implement than k-means. The usage of distributed computing helped overcome this shortcoming. In the proposed model, compact clusters exhibit the same fundamental properties as local clusters. The reliability of the overall model is significantly affected by the accuracy of the local cluster.

4.4. Experiment 4: Convergence Time

In this experiment, we compare the convergence times of conventional clustering models with those of the suggested improved two-level clustering technique (see Table 4). Convergence times are reduced in the similarity-based k-medoids, k-prototype, and OCIL methods. The research shows that more time is required for the suggested model to converge. When compared to similarity-based k-medoids, k-prototype, and OCIL clustering algorithms, the suggested model speeds up convergence by 51%, 65%, and 24%, respectively. The suggested model incorporates local and global clustering, which may account for these results. On the other hand, the local clustering infrastructure is based on the Spark framework, which speeds up processing even when dealing with massive volumes of data. Nevertheless, global clusters function iteratively, with the nodes of the leaders aggregating smaller clusters; this lengthens the time needed for the clusters to converge.

Table 4. The Average Convergence Time Analysis (s).

Clustering Algorithm	Average Time
C2 (K-prototype)	6
C3 (Object Clustering Iterative Learning)	12
C4 (Similarity-based k-medoids clustering)	8
Proposed Model	16

4.5. Experiment 5: Clustering Accuracy

Here, a clustering accuracy measure is used to assess the efficiency of the various clustering methods. Table 5 displays the clustering accuracy of the recommended model, as well as that of the parallel k-medoids clustering [52]. The primary concept of the paper [52] is to individually apply clustering via two cost-free phases: parallel seeding and parallel refining. While in the first stage a global search is conducted over a subset of the data, in the second stage a local search is conducted over the full dataset. Improved clustering accuracy of 3.5% is clear evidence of the study's value. Our technique combines the k-medoid with the chemical reaction optimization (CRO) algorithm, which may explain the result. CRO is used to increase the quality of data utilized for clustering and to increase the search space for the appropriate medoid.

Table 5. The Clustering Accuracy Analysis.

Techniques	Parallel K-medoids clustering [52]	Proposed Model (Parallel CRO-Based K-medoids Clustering)
Clustering Accuracy	94.6	97.4

4.6. Limitation

- Many decision-making processes in our world include optimization issues that are NP-hard. The large-scale, dynamism, and vagueness of these problems limit the use of independent optimization techniques.
- Metaheuristics approaches (such as Chemical Reaction Optimization) typically assume that problem inputs, underlying objective functions, and optimization constraints are either deterministic or follow basic probabilistic rules. As a result of these high assumptions, several deterministic models are oversimplified copies of real-world systems.
- In the absence of uncertainty in the optimization formulation, the optimal solutions for these systems may be unstable and sensitive to modest changes in input parameters.
- Numerous metaheuristics use some type of stochastic optimization, such that the solution discovered is dependent on the provided random variables.
- Compared with conventional approaches, more computing resources are needed.

5. Conclusions

As most current data is spatial and collected by sensors, geospatial big data is essential in the big data age. While big data has many potential benefits, it may be difficult to strike a good balance between the data's volume, diversity, velocity, validity, and usefulness. Solving geographic issues with massive volumes of data requires high-performance computation. Geospatial big data query analysis is difficult. Improving query success via data management is tricky. The rising complexity of data queries requires a clustering technique.

In the context of geospatial big data, this paper examined the specifics of the suggested dynamic distributed clustering paradigm for analyzing spatial queries. There are two phases to the suggested paradigm. By employing Spark's task parallelism model, local clustering builds subset-focused clusters. The "aggregation" phase of global clustering is responsible for mechanically producing the required quantity of compact final clusters. These dense clusters are similar to their neighboring counterparts in most respects. The k-medoid technique is suggested to be used with the Chemical Reaction Optimization (CRO) algorithm to improve efficiency and provide more precise clusters. This technique employs CRO to improve clustering and expand the search for the optimal medoid by amassing more reliable results. Experiments demonstrated that the recommended model balanced clustering error, execution time, clustering accuracy, and convergence speed. The suggested model could be enhanced using soft computing, the ideal number of processing nodes could be discovered, and the model could be scaled up to handle big data analytics for streaming geographic data.

Author Contributions: Conceptualization, S.M.D. and O.A.H.; methodology, S.M.D., A.F.N. and O.A.H.; software, A.F.N. and H.K.I.; validation, S.M.D. and O.A.H.; formal analysis, S.M.D., A.F.N. and H.K.I.; investigation, S.M.D. and O.A.H.; resources, A.F.N. and H.K.I.; data curation, S.M.D., A.F.N. and H.K.I.; writing—original draft preparation, S.M.D. and A.F.N.; writing—review and editing, S.M.D. and H.K.I.; visualization, A.F.N. and H.K.I.; supervision, S.M.D. and O.A.H.; project administration, A.F.N. and O.A.H.; funding acquisition, A.F.N. and H.K.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets for this research are available in http://crawdad.org/epfl/mobility/20090224/cab (accessed on 1 January 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Deng, X.; Liu, P.; Liu, X.; Wang, R.; Zhang, Y.; He, J.; Yao, Y. Geospatial big data: New paradigm of remote sensing applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 3841–3851. [CrossRef]
2. Li, S.; Dragicevic, S.; Castro, F.A.; Sester, M.; Winter, S.; Coltekin, A.; Pettit, C.; Jiang, B.; Haworth, J.; Stein, A.; et al. Geospatial big data handling theory and methods: A review and research challenges. *ISPRS J. Photogramm. Remote Sens.* **2016**, *115*, 119–133. [CrossRef]
3. Li, Z. Geospatial Big Data Handling with High Performance Computing: Current Approaches and Future Directions. In *High Performance Computing for Geospatial Applications*; Springer: Cham, Switzerland, 2020; pp. 53–76.
4. Wang, H.; Zhu, S. Multisource Aggregation Search and Scheduling for Remote Sensing Data Cluster. *IEEE Geosci. Remote Sens. Lett.* **2019**, *7*, 352–356. [CrossRef]
5. Limkar, S.V.; Jha, R.K. A novel method for parallel indexing of real time geospatial big data generated by IoT devices. *Future Gener. Comput. Syst.* **2019**, *97*, 433–452. [CrossRef]
6. Eldawy, A.; Mokbel, M.F. Spatialhadoop: A mapreduce framework for spatial data. In Proceedings of the IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 1352–1363.
7. Lenka, R.K.; Barik, R.K.; Gupta, N.; Ali, S.M.; Rath, A.; Dubey, H. Comparative analysis of SpatialHadoop and GeoSpark for geospatial big data analytics. In Proceedings of the 2nd International Conference on Contemporary Computing and Informatics, Greater Noida, India, 14–17 December 2016; pp. 484–488.
8. Lee, K.; Ganti, R.K.; Srivatsa, M.; Liu, L. Efficient spatial query processing for big data. In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas, TX, USA, 4 November 2014; pp. 469–472.
9. Aljawarneh, I.M.; Bellavista, P.; Corradi, A.; Montanari, R.; Foschini, L.; Zanotti, A. Efficient spark-based framework for big geospatial data query processing and analysis. In Proceedings of the IEEE Symposium on Computers and Communications, Heraklion, Greece, 3–6 July 2017; pp. 851–856.
10. Shirkhorshidi, A.S.; Aghabozorgi, S.; Wah, T.Y.; Herawan, T. Big data clustering: A review. In Proceedings of the International Conference on Computational Science and Its Applications, Guimarães, Portugal, 30 June 2014; pp. 707–720.
11. 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 Trans. Emerg. Top. Comput.* **2014**, *2*, 267–279. [CrossRef]
12. Ayed, A.B.; Halima, M.B.; Alimi, A.M. Survey on clustering methods: Towards fuzzy clustering for big data. In Proceedings of the 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), Tunis, Tunisia, 11–14 August 2014; pp. 331–336.
13. Arora, S.; Chana, I. A survey of clustering techniques for big data analysis. In Proceedings of the 5th International Conference-Confluence: The Next Generation Information Technology Summit, Noida, India, 25–26 September 2014; pp. 59–65.
14. Shi, Z.; Pun-Cheng, L.S. Spatiotemporal data clustering: A survey of methods. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 112. [CrossRef]
15. Xinxiang, H.; Henan, X. A new data mining algorithm based on Mapreduce and Hadoop. *Int. J. Signal Proc. Image Process. Pattern Recognit.* **2014**, *7*, 131–142.
16. Mirzasoleiman, B.; Karbasi, A.; Sarkar, R.; Krause, A. Distributed sub-modular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*; ACM Digital Library: New York, NY, USA, 2013; pp. 2049–2057.
17. Ene, A.; Im, S.; Moseley, B. Fast clustering using MapReduce. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21 August 2011; pp. 681–689.
18. Yue, X.; Man, W.; Yue, J.; Liu, G. Parallel k-medoids++ spatial clustering algorithm based on mapreduce. *arXiv* **2016**, arXiv:1608.06861.
19. Martino, A.; Rizzi, A.; Mascioli, F.M. Distance matrix pre-caching and distributed computation of internal validation indices in k-medoids clustering. In Proceedings of the International Joint Conference on Neural Networks, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
20. Bendechache, M.; Kechadi, M.T.; Le-Khac, N.A. Efficient large scale clustering based on data partitioning. In Proceedings of the IEEE International Conference on Data Science and Advanced Analytics, Montreal, QC, Canada, 17–19 October 2016; pp. 612–621.
21. Bendechache, M.; Le-Khac, N.A.; Kechadi, M.T. Performance evaluation of a distributed clustering approach for spatial datasets. In *Australasian Conference on Data Mining*; Springer: Singapore, 2017; pp. 38–56.
22. Shaikh, S.; Memon, M.; Kim, K. A multi-criteria decision-making approach for ideal business location identification. *Appl. Sci.* **2021**, *11*, 4983. [CrossRef]

23. Massai, L.; Nesi, P.; Pantaleo, G. PAVAL: A location-aware virtual personal assistant for retrieving geolocated points of interest and location-based services. *Eng. Appl. Artif. Intell.* **2018**, *77*, 70–85. [CrossRef]
24. Yu, J.; Sarwat, M. GeoSparkViz: A cluster computing system for visualizing massive-scale geospatial data. *VLDB J.* **2021**, *30*, 237–258. [CrossRef]
25. Peng, Q.; You, L.; Dong, N. A location-aware GIServices quality prediction model via collaborative filtering. *Int. J. Digit. Earth* **2017**, *11*, 897–912. [CrossRef]
26. García-García, F.; Corral, A.; Iribarne, L.; Vassilakopoulos, M.; Manolopoulos, Y. Efficient distance join query processing in distributed spatial data management systems. *Inf. Sci.* **2019**, *512*, 985–1008. [CrossRef]
27. Dritsas, E.; Kanavos, A.; Trigka, M.; Vonitsanos, G.; Sioutas, S.; Tsakalidis, A. Trajectory clustering and k-NN for robust privacy preserving k-NN query processing in GeoSpark. *Algorithms* **2020**, *13*, 182. [CrossRef]
28. García-García, F.; Corral, A.; Iribarne, L.; Vassilakopoulos, M. Improving distance-join query processing with Voronoi-diagram based partitioning in SpatialHadoop. *Future Gener. Comput. Syst.* **2019**, *111*, 723–740. [CrossRef]
29. Qiao, B.; Ma, L.; Chen, L.; Hu, B. A PID-Based kNN Query Processing Algorithm for Spatial Data. *Sensors* **2022**, *22*, 7651. [CrossRef]
30. Schmidtke, H. Location-aware systems or location-based services: A survey with applications to CoViD-19 contact tracking. *J. Reliab. Intell. Environ.* **2020**, *6*, 191–214. [CrossRef]
31. Ghosh, S.; Das, J.; Ghosh, S. Locator: A cloud-fog-enabled framework for facilitating efficient location based services. In Proceedings of the International Conference on Communication Systems & Networks, Bengaluru, India, 7–11 January 2020; pp. 87–92.
32. Manna, P.; Bonfante, A.; Colandrea, M.; Di Vaio, C.; Langella, G. A geospatial decision support system to assist olive growing at the landscape scale. *Comput. Electron. Agric.* **2019**, *168*, 105143. [CrossRef]
33. Sadeghi-Niaraki, A.; Jelokhani-Niaraki, M.; Choi, S.M. A volunteered geographic information-based environmental decision support system for waste management and decision making. *Sustainability* **2020**, *12*, 6012. [CrossRef]
34. Keenan, P.; Jankowski, P. Spatial decision support systems: Three decades on. *Decis. Support Syst.* **2018**, *116*, 64–76. [CrossRef]
35. Shin, H.; Lee, K.; Kwon, H. A comparative experimental study of distributed storage engines for big spatial data processing using GeoSpark. *J. Supercomput.* **2021**, *78*, 2556–2579. [CrossRef] [PubMed]
36. Sajana, T.; Rani, C.S.; Narayana, K.V. A survey on clustering techniques for big data mining. *Indian J. Sci. Technol.* **2016**, *9*, 1–12. [CrossRef]
37. Narayana, G.S.; Vasumathi, D. An attributes similarity-based K-medoids clustering technique in data mining. *Arab. J. Sci. Eng.* **2018**, *43*, 3979–3992. [CrossRef]
38. Alasadi, S.A.; Bhaya, W.S. Review of data preprocessing techniques in data mining. *J. Eng. Appl. Sci.* **2017**, *12*, 4102–4107.
39. Uma, K.; Hanumanthappa, M. Data Collection Methods and Data Pre-processing Techniques for Healthcare Data Using Data Mining. *Int. J. Sci. Eng. Res.* **2017**, *8*, 1131–1136.
40. Hudaib, A.; Khanafseh, M.; Surakhi, O. An improved version of K-medoid algorithm using CRO. *Mod. Appl. Sci.* **2018**, *12*, 116. [CrossRef]
41. Majumder, S.; Sayed, A.; Jerin, J.; Inzamam-Ul-Hossain, M. Prediction of diabetics using chemical reaction optimization. In Proceedings of the International Conference on Computing Communication and Networking Technologies, Kharagpur, India,, 6–8 July 2021; pp. 1–5.
42. Martino, A.; Rizzi, A.; Mascioli, F.M. Efficient Approaches for Solving the Large-Scale k-medoids Problem. In Proceedings of the 9th International Joint Conference on Computational Intelligence, Funchal-Madeira, Portugal, 1–3 November 2017; pp. 338–347.
43. Whelan, M.; Le Khac, N.A.; Kechadi, M.T. Data reduction in very large spatio-temporal datasets. In Proceedings of the 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, Larissa, Greece, 28–30 June 2010; pp. 104–109.
44. Laloux, J.F.; Le-Khac, N.A.; Kechadi, M.T. Efficient distributed approach for density-based clustering. In Proceedings of the IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Paris, France, 27–29 June 2011; pp. 145–150.
45. Wang, B.; Yin, J.; Hua, Q.; Wu, Z.; Cao, J. Parallelizing k-means-based clustering on spark. In Proceedings of the International Conference on Advanced Cloud and Big Data, Chengdu, China, 13–16 August 2016; pp. 31–36.
46. Bendechache, M.; Kechadi, M.T. Distributed clustering algorithm for spatial data mining. In Proceedings of the 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services, Fuzhou, China, 8–10 July 2015; pp. 60–65.
47. Naacke, H.; Curé, O.; Amann, B. SPARQL query processing with Apache Spark. *arXiv* **2016**, arXiv:1604.08903.
48. Aly, A.M.; Aref, W.G.; Ouzzani, M. Spatial queries with k-nearest-neighbor and relational predicates. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 3 November 2015; pp. 1–10.
49. Papadias, D.; Zhang, J.; Mamoulis, N.; Tao, Y. Query processing in spatial network databases. In Proceedings of the VLDB Conference, Berlin, Germany, 9–12 September 2003; pp. 802–813.
50. Piorkowski, M.; Sarafijanovic-Djukic, N.; Grossglauser, M. CRAWDAD Dataset Epfl/Mobility (v2009-02-24), Trace Set: Cab. 2019. Available online: http://crawdad.org/epfl/mobility/20090224/cab (accessed on 1 January 2022).

51. Shah, P.; Chaudhary, S. Big data analytics framework for spatial data. In Proceedings of the International Conference on Big Data Analytics, Langkawi, Malaysia, 22 November 2018; pp. 250–265.
52. Song, H.; Lee, J.; Han, W. PAMAE: Parallel k-medoids clustering with high accuracy and efficiency. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 4 August 2017; pp. 1087–1096.