Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 756107, 9 pages
http://dx.doi.org/10.1155/2015/756107

Research Article

A Group Mining Method for Big Data on Distributed Vehicle Trajectories in WAN

Jie Yang,[1,2] Xiaoping Li,[1] Dandan Wang,[1] and Jia Wang[1]

[1]*School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*
[2]*Public Security Bureau of Jiangsu Province, Nanjing 210024, China*

Correspondence should be addressed to Xiaoping Li; xpli@seu.edu.cn

Received 11 August 2014; Revised 2 December 2014; Accepted 10 December 2014

Academic Editor: Xiaohong Jiang

A distributed parallel clustering method MCR-ACA is proposed by integrating the ant colony algorithm with the computing framework Map-Combine-Reduce for mining groups with the same or similar features from big data on vehicle trajectories stored in Wide Area Network. The heaviest computing burden of clustering is conducted in parallel at local nodes, of which the results are merged to small size intermediates. The intermediates are sent to the central node and clusters are generated adaptively. The great overhead of transferring big volume data is avoided by MCR-ACA, which improves the computing efficiency and guarantees the correctness of clustering. MCR-ACA is compared with an existing parallel clustering algorithm on practical big data collected by the traffic monitoring system of Jiangsu province in China. Experimental results demonstrate that the proposed method is effective for group mining by clustering.

1. Introduction

Recently, big data on vehicle trajectories collected by traffic monitoring systems are more and more important in practice, which are based on license plate identification and the RFID techniques. For example, there are more than 100 subsystems in a traffic monitoring system of Jiangsu province in China, which includes more than 2×10^4 data sensors (collecting devices) distributed in 13 cities, covering 3 million acres in size. Around 5×10^4 million data records have been collected so far with 70 million being increased every day. Usually, traveling features of human beings imply some patterns. For example, because of people's behavior habits or their fixed working locations, they always go out at the same time with the same trajectory. Therefore, the data collected by traffic monitoring systems imply features of vehicle trajectories, which illustrate characteristics of human behaviours. It is reported that 93% of human behaviors can be foreseen [1] and four spatiotemporal points are enough to uniquely identify 95% of the individuals [2]. Likewise, it is possible to identify a driver with high probability according to several trajectory records, and a driver group with the same or similar behaviour features can be found by mining conducted on big data of vehicle trajectories. It is quite important and applicable. For example, a band of criminal suspects would be found if they use cars for transportation. Big data on vehicle trajectories is distributively stored in WAN, which is huge in amount and hard to be physically centralized by extraction.

The biggest challenge in big data clustering is designing effective algorithms for clustering and distributed parallel computation [3]. For these issues, some distributed parallel computing frameworks based on Cloud Computing [4] and MapReduce [5] have been proposed in recent years, such as the batch computing framework [6, 7], the stream parallel computing framework [8], the customized parallel computing framework [9], and the mixed parallel computing framework [10]. Based on such computing frameworks, some distributed parallel clustering algorithms have been proposed. A new density-based clustering algorithm DBCURE-MR [11] was introduced, which is robust to find clusters with various densities and suitable for parallelizing the algorithm with MapReduce. A nonparametric accuracy estimation method and system [12] were proposed for speeding up big data analysis. Sampling with replacement was adopted to obtain

the sampling points according to the sampling distribution. The amount of data input to MapReduce can be decreased considerably. Taking into account the distributed nature of portioned data and model, three clustering algorithms [13], k-mean, canopy, and Fuzzy k-mean, were implemented in parallel on MapReduce. The efficiency of distributed clustering was improved significantly. For detecting in large community networks, a parallel structural clustering algorithm was introduced [14], which is based on the similarity of edge structures and MapReduce. The interfaces and implementations for user-defined aggregation in several states of the distributed computing systems were evaluated in [15], where communication overhead of data-intensive applications could be decreased largely by local clustering, which clusters the intermediate results generated by Map tasks and then transmits the clustered results to Reduce tasks.

The existing methods improve clustering efficiency either by parallel computing on physically centralized big data or by reducing data scale using sampling. However, the communication overhead of data centralization and the impact on sparse data for clustering accuracy have not been considered yet. In this paper, by integrating the ACA (ant colony algorithm) with the computing framework Map-Combine-Reduce (MCR), a MapReduce based distributed parallel clustering method MCR-ACA is proposed for group mining on big data of vehicle trajectories in WAN. Some parallel ACA methods based on MapReduce have been proposed [16, 17] (defined as MR-ACA). However, since these methods work on physically centralized big data in LAN, the communication overhead of data centralization is ignored. The MCR-ACA method contains three stages: Map operation, Combine operation, and Reduce operation. Both the computation tasks with the heaviest burden are conducted and their results are combined in parallel on data source nodes. The combined result is transmitted to the central node and new cluster centers are generated adaptively. The presented method avoids the communication overhead of big data migration, improves the clustering efficiency, and guarantees the accuracy of the global cluster among distributed nodes.

The rest of this paper is organized as follows. The problem of group mining for vehicle trajectories is described in Section 2. A distributed parallel clustering method MCR-ACA is proposed in Section 3. Section 4 shows the computational experiments, followed by the conclusion and future work in Section 5.

2. Group Mining for Vehicle Trajectories

Distributed frameworks in WAN are always adopted by traffic monitoring systems. Hierarchical ones are even utilized by some complex systems. For the traffic monitoring system of Jiangsu examined in this paper, a 3-layer framework is applied. There are 13 independent branch centers in 13 cities, respectively, which are responsible for the integration of independent data branches within each city. A head center is in charge of all the branch ones. Therefore, the main characteristics of big data of vehicle trajectories in WAN are multiple data sources and hard to be physically centralized. In this paper, the data branches are called source nodes, city branch centers are city nodes, and the head center is the central node. The topological network is shown in Figure 1.

Traffic monitoring systems which contain multiple independent subsystems are being developed in the cities. They form distributed data sources which increase rapidly. There are more than one hundred subsystems in the system of the Jiangsu province. The amount of data in the subsystems is quite huge and growing rapidly (data increment in one city of Jiangsu is over 12 million records every day). Multimedia data in various formats (such as photos and videos) increase several TBs each day, which is very difficult to be physically centralized in the central node. The trajectory data of the cars collected by a subsystem is listed in Table 1.

Group mining for vehicle trajectories (GMVT for short) is critical for data clustering on big data of distributed traffic monitoring systems in WAN. The main idea of mining groups on big data of vehicle trajectories is to implement the automatic partition of vehicle trajectories with the same or similar features by clustering on attributes, which consist of the metadata (e.g., time and location) of the vehicle trajectories. The information (the number of license plates) of the vehicle trajectories in the same clusters can be drawn from the partition result. A complete vehicle trajectory record includes those metadata: the number of the license plates, passing time, location, direction, speed, and car color. These attributes are set as metadata. Records collected by different sensors in various subsystems are normalized as a 6-tuple (the number of license plates, passing time, location, direction, speed, and car color). The first record in Table 1 is normalized as S032V0, 20130521073907, checkpoint at the crossroad of Suyuan road and west Qingshuiting road, 1, 41, and A. Every element is assigned to a weight. Features of vehicle trajectories with the same or similar elements are clustered on the data records.

Suppose there are $\mathbb{Z}$ distributed data sources (subsystems) $\{S_1, S_2, \ldots, S_{\mathbb{Z}}\}$. Each data source S_i with 6 attributes has T_i tables $\{L_{i,1}, L_{i,2}, \ldots, L_{i,T_i}\}$. A table $L_{i,j}$ has $Q_{i,j}$ records. In total, there are $\sum_{i=1}^{\mathbb{Z}} \sum_{j=1}^{T_i} Q_{i,j}$ records. The objective of group mining is to partition set A with $\sum_{i=1}^{\mathbb{Z}} \sum_{j=1}^{T_i} Q_{i,j}$ records into N subsets $A_1, A_2, \ldots, A_N$. In each subset, records have the same or similar attributes and $\bigcup_{i=1}^{N} A_i = A$, $A_i \cap A_j = \emptyset$, $\forall i \neq j$. In an arbitrary subset A_i, all records are grouped according to the number of license plates. The clustering accuracy rate is the ratio of the clustered records of one car to the total records which this car is involved in (e.g., if the total records of one car are Q_c, Q records of the car are clustered into one class; the corresponding accuracy rate is $(Q/Q_c)100\%$). If the accuracy rate is greater than a given threshold η, the car is merged into this class. All cars in this class have the same or similar trajectory features.

3. Group Mining Methods for Vehicle Trajectory in WAN

In WAN, GMVT is critical for clustering data of vehicle trajectories by attribute features such as time or location to construct various classes with the same or similar attributes,

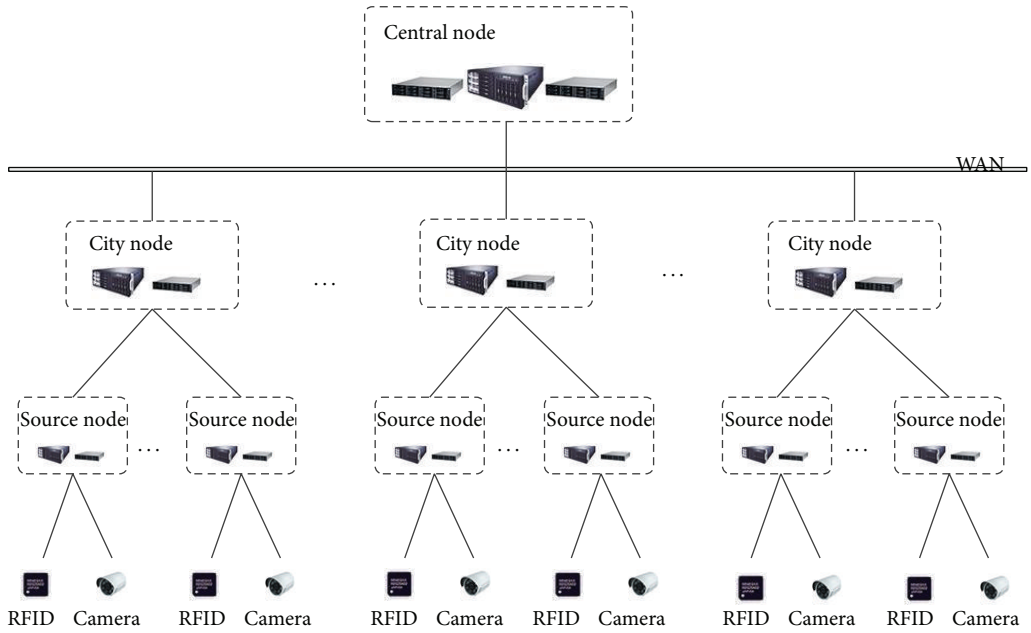

Figure 1: Topological network of traffic monitoring system.

Table 1: Data collected by a subsystem.

HPHM	JGSJ	JGDD	XSFX	XSSD	CSYS
S032V0	20130521073907	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	1	41	A
S032V0	20130522191001	Southwest corner of the crossroad of Jinxianghe road and east Beijing road	0	25	J
S032V0	20130524213427	Northwest corner of the crossroad of Xuanwu road and Huayuan road	2	85	B
S470A5	20130523071732	Northeast corner of the crossroad of south Fengtai road and Hexi street	6	31	J
S470A5	20130524101258	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	2	42	J

according to which drivers with the same or similar features are identified as a group. Because of the two characteristics mentioned in Section 2, traditional parallel clustering methods are no longer efficient, which motivates us to present the following method.

3.1. Clustering Framework for Distributed Big Data in WAN. Data clustering is very difficult for big data that is stored distributively in WAN. The reason lies in two aspects: (i) huge amount of data makes clustering computing more time-consuming, which leads to existing methods being infeasible. (ii) Communication overhead on data migration is generally more than the computing cost. It is better to migrate computation rather than migrate data. Therefore, a distributed parallel computing framework (MCR), which is based on MapReduce, is proposed in this paper. The framework of MCR is depicted in Figure 2.

Based on MCR, the traditional ACA is adapted to MCR-ACA for group mining for big data. The procedure of MCR is described as follows.

(i) Divide the data source S_i ($i = 1, 2, \ldots, \mathbb{Z}$) into H_i data chunks $\{B_{i,1}, B_{i,2}, \ldots, B_{i,H_i}\}$.

(ii) Map operations are carried out on each data chunk $B_{i,j}$ by a clustering strategy. All records in $B_{i,j}$ are clustered by the given strategy.

(iii) The clustered results are merged into intermediate ones by Combine. For example, $B_{i,j}$ is clustered and combined into a set of intermediate results with $m_{i,j}$ elements ($m_{i,j}$ is usually small).

(iv) The intermediate results are sent to the central node, where Reduce is conducted for global clustering.

(v) The method terminates if the global clustering converges to or reaches the maximal iterations $g_{\max}$. Otherwise, the comparison parameter is sent to each data chunk by Reduce. The next iteration starts from step (ii).

Computing operations with the heaviest burden are conducted in parallel at source nodes. Data in each source node is divided into data chunks. All chunks are clustered in parallel which leads to good efficiency. Communication overhead is significantly reduced by transmitting intermediate results combined at local source nodes rather than the source data. The global clustering is conducted on the intermediate results at the central node.

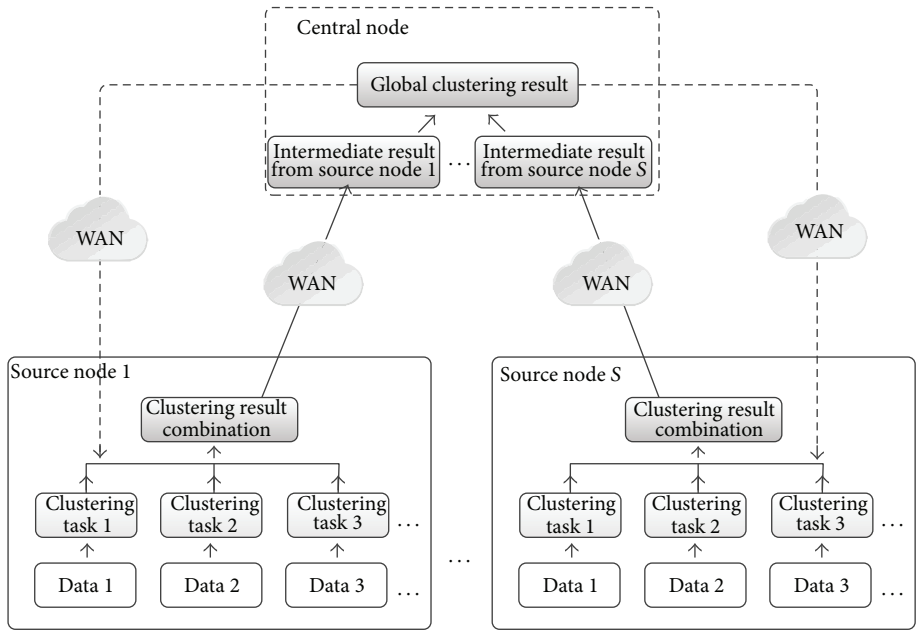

Figure 2: MCR distributed parallel computing framework.

3.2. MCR-ACA Method for Group Mining. ACA was inspired from the phenomenon that ant individuals gather at a location with food by pheromone interaction among them [18, 19]. By integrating ACA with the computing framework MCR, the group mining method MCR-ACA is proposed. The number of classes and the trajectory records in a class are determined adaptively, and clustering centers are generated in iterations without predefinition, which is desirable for the considered problem.

3.2.1. Map Function of MCR-ACA. A vector of m elements (attributes) is developed for each record of vehicle trajectories; that is, a data record R_k to be clustered is denoted as $R_k = (r_k^1, r_k^2, \ldots, r_k^m)$. $a_{i,j}$ ants are assigned to data chunk $B_{i,j}$, each of which randomly serves for a record R_k in the initial step. A neighborhood is established with the center R_k and radius $\mathbb{R}$ (from experience), denoted as $N(R_k, \mathbb{R})$. The comprehensive similarity between the center R_k and all records within its neighborhood $N(R_k, \mathbb{R})$ is defined by

$$f(R_k) = \sum_{R_l \in N(R_k, \mathbb{R})} \left[1 - \frac{d_{kl}}{\lambda}\right], \tag{1}$$

where λ is the similarity factor, representing the range of dimensions (the difference between the maximum and minimum of dimension). Let d_{kl} be the space distance between two records R_k and R_l, which is calculated by the weighted distance according to

$$d_{kl} = \left\| P(R_k - R_l) \right\| = \sqrt{\sum_{h=1}^{m} P_h \left(r_k^h - r_l^h\right)^2}, \tag{2}$$

where P_h is the weight based on the experience and data collecting accuracy. Therefore, the probability of clustering record R_k into class $N(R_k, \mathbb{R})$ is computed by

$$p_{kl}(t) = \frac{\tau_{kl}^{\alpha}(t) f_t^{\beta}(R_k)}{\sum_{\in N(R_l, \mathbb{R})} \tau_{zl}^{\alpha}(t) f_t^{\beta}(R_z)}, \tag{3}$$

where α, β are control parameters and $\tau_{kl}(t)$ is the pheromone amount on the path from R_k to R_l at time t ($\tau_{kl}(0) = 1$).

The decision of putting down or moving R_k is made in terms of the clustering probability $p_{kl}(t)$: (i) If $p_{kl}(t)$ is greater than or equal to the given threshold p_0, the ant puts down R_k and clusters it into class $N(R_k, \mathbb{R})$. The traveled path length of the ant is saved, and the location where R_k was put down is set as the start of a new traversal path; then another record is randomly assigned to the ant. (ii) If $p_{kl}(t)$ is less than p_0, the ant carrying R_k keeps moving to the next point R_l with the largest $p_{kl}(t)$. R_k is dropped when the path length reaches the maximum or the ant has not found the proper clusters until the travel ends (it can be regarded as abandoned). The ant gets a new record. After all records in $B_{i,j}$ are travelled by ants, that is, $|B_{i,j}|$ records are clustered or abandoned, local clustering stops.

The Map function takes $(\langle \text{key}, R_k \rangle, \langle p, d, s \rangle)$ as the input key/value pair, where key is the key value of R_k, p is the clustering probability, and d is the path length where the ant carries R_k. p and d are initialized as 0. s is the coordinate of nodes along the path with length d, which is initialized as $\emptyset$. g is the index of the current iteration. $\tau_{kl}(g)$ is the pheromone value on path R_k to R_l after gth iteration with the initial value 1. p_k is the clustering probability for R_k, d_k is the path length while R_k is clustered or abandoned, and s_k is the node where R_k is clustered or abandoned. d_g is the minimum d_k after the

Input: Data chunk $B_{i,j}$
(1) $p \leftarrow 0$, $d \leftarrow 0$, $Num \leftarrow 0$, $s \leftarrow \emptyset$, $\tau_{kl}(0) \leftarrow 1$, p_0 is given.
(2) **while** $(Num \leqslant |B_{i,j}|)$ **do**
(3) Calculate the weight distance d_{kl} between R_k with all records in $N(R_k, \mathbb{R})$ by (2).
(4) Calculate the comprehensive similarity $f(R_k)$ between R_k with all records in $N(R_k, \mathbb{R})$ by (1).
(5) Read the pheromone value $\tau_{kl}(g-1)$, calculate the clustering probability $p_{kl}(t)$ by (3).
(6) **if** $p_{kl}(t) \geq p_0$ **then**
(7) Cluster R_k into $N(R_k, \mathbb{R})$, save p_k, d_k, s_k, $Num \leftarrow Num + 1$.
(8) Go to Step 17.
(9) **if** $d \geq d_{g-1}$ **then**
(10) Abandon R_k, save p_k, d_k, s_k, $Num \leftarrow Num + 1$.
(11) Go to Step 17.
(12) Select the node with largest $p_{kl}(t)$ into s.
(13) **if** all records are examined by the ant **then**
(14) Abandon R_k, save p_k, d_k, s_k.
(15) Go to Step 17.
(16) $d \leftarrow d + d_{kl}$, go to Step 2.
(17) Randomly assign the ant a new record which has not been clustered or abandoned.
(18) Output the clustering result $(\langle key, R_k \rangle, \langle p_k, d_k, s_k \rangle)$.
(19) **return**

Algorithm 1: Map function of MCR-ACA.

Input: Probability threshold p_0.
(1) **if** $p_k < p_0$ **then**
(2) $d_{i,j} \leftarrow \min\{d_k\}$.
(3) Output $d_{i,j}$ and go to Step 6.
(4) Combine records with the same s_k and generate $\mathbb{N}_{s_k}$.
(5) Calculate C_{s_k}, output $\left(s_k, \left\langle \mathbb{N}_{s_k}, C_{s_k} \right\rangle\right)$.
(6) Update pheromone $\tau_{kl}(g)$.
(7) **return**

Algorithm 2: Combine function of MCR-ACA.

gth iteration, which is the comparing parameter for the next iteration. Map function on $B_{i,j}$ is described in Algorithm 1.

3.2.2. Combine Function of MCR-ACA. The results obtained by the Map function $(\langle \text{key}, R_k \rangle, \langle p_k, d_k, s_k \rangle)$ are combined into intermediate results $(s_k, \langle \mathbb{N}_{s_k}, C_{s_k} \rangle)$ by the Combine function at local nodes. The minimum $d_{i,j}$ along the path of this iteration is found. Pheromone is updated by the Combine function, which is increased when ants pass by and decreased in time. The pheromone along the path from R_k to R_l after the gth Map function is updated by

$$\tau_{kl}(g) = (1-\rho) \cdot \tau_{kl}(g-1) + \Delta e, \tag{4}$$

where $\rho \in (0, 1]$ denotes the evaporating rate of pheromone. Δe is the pheromone left by passing of ant. Δe is set as 1 if an ant passes by; otherwise it is set to 0.

For the records with the clustering probability less than the given threshold p_0, the minimal d_k is set as the minimum path length $d_{i,j}$ in the data chunk $B_{i,j}$. For all the records with clustering probability not less than p_0, they are combined according to the clustered nodes s_k. Records with the same s_k are merged into the same class s_k. The number is denoted as $\mathbb{N}_{s_k}$. For the records $R_{s_k,k}$ in class s_k, the sum of the attribute vector is $C_{s_k} = \sum_{k=1}^{\mathbb{N}_{s_k}} R_{s_k,k} = (\sum_{k=1}^{\mathbb{N}_{s_k}} r_{s_k,k}^1, \sum_{k=1}^{\mathbb{N}_{s_k}} r_{s_k,k}^2, \ldots, \sum_{k=1}^{\mathbb{N}_{s_k}} r_{s_k,k}^m)$.

Multiple Combine functions can be conducted in parallel for one data source S_i, each of which works on one or several data chunks. The Combine function for $B_{i,j}$ is described in Algorithm 2.

There are only two possible outputs from the Combine function, either $(s_k, \langle \mathbb{N}_{s_k}, C_{s_k} \rangle)$ or the minimal path length $d_{i,j}$. Therefore, the data sent to the central node in WAN can be greatly reduced. Data chunk $B_{i,j}$ is combined into $m_{i,j}$ classes; the communication overhead is $m_{i,j}$ intermediate results $(s_k, \langle \mathbb{N}_{s_k}, C_{s_k} \rangle)$ and a $d_{i,j}$. By testing on practical data, the data chunk with the amount of 1.8 GB only needs to transmit 30 KB after combination, which is only $(1/6) \times 10^{-4}$ of the original data.

3.2.3. Reduce Function of MCR-ACA. At the gth iteration, two parts obtained from the Combine phase on data chunks, that is, intermediate results $(s_{k,g}, \langle \mathbb{N}_{s_{k,g}}, C_{s_{k,g}} \rangle)$ and $d_{i,j,g}$, are recombined by the Reduce function. New clustering centers are generated. The weighted distances among clustering centers in different data chunks are calculated by (2). If it is less than or equal to $\mathbb{R}$, the parts are merged into one class $N(s_{k,g}, \mathbb{R})$. The global cluster center $\overline{C}_{s_{k,g}}$ at the gth iteration is computed by $\sum_{N(s_{k,g},\mathbb{R})} C_{s_{k,g}} / \sum_{N(s_{k,g},\mathbb{R})} \mathbb{N}_{s_{k,g}}$. $\overline{C}_{s_{k,g}}$ converges to and outputs the global clustering result if $|\overline{C}_{s_{k,g}} - \overline{C}_{s_{k,g-1}}| \leq |\overline{C}_{s_{k,g-1}} - \overline{C}_{s_{k,g-2}}|$. Otherwise, the minimal $d_{i,j,g}$ is output as d_g and sent to each source node for the next comparison. The Reduce function is described in Algorithm 3.

(1) Calculate the weighted distance $d_{s_{k,g},s_{k',g}}$ between different
cluster centers $s_{k,g}$ and $s_{k',g}$ respectively located in different data chunks by (2).
(2) **if** $d_{s_{k,g},s_{k',g}} \leq \mathbb{R}$ **then**
(3) Combine $s_{k,g}$ and $s_{k',g}$ into the same class $N(s_{k,g}, \mathbb{R})$.
(4) Calculate the global cluster center $\overline{C}_{s_{k,g}} = (\sum_{N(s_{k,g},\mathbb{R})} C_{s_{k,g}} / \sum_{N(s_{k,g},\mathbb{R})} \mathbb{N}_{s_{k,g}})$ of the gth iteration.
(5) **if** $\overline{C}_{s_{k,g}}$ converges **then**
(6) output the global clustering result.
(7) Go to Step 9.
(8) Output the minimal $d_{i,j,g}$ as d_g.
(9) **return**

Algorithm 3: Reduce function of MCR-ACA.

(1) $p \leftarrow 0, d \leftarrow 0, d_0 \leftarrow \infty, Num \leftarrow 0, s \leftarrow \emptyset, \tau_{kl}(0) \leftarrow 1$, p_0 is given.
(2) **while** $(g \leqslant g_{\max})$ **do**
(3) Conduct Map function in parallel, output the clustering result $(\langle key, R_k \rangle, \langle p_k, d_k, s_k \rangle)$.
(4) Perform Combine functions in parallel, output intermediate results $\left(s_k, \left\langle \mathbb{N}_{s_k}, C_{s_k} \right\rangle\right)$ and $d_{i,j}$, for each data chunk $B_{i,j}$.
(5) Reduce functions are carried out in parallel to develop the global cluster $\overline{C}_{s_{k,g}}$.
(6) $g \leftarrow g + 1$.
(7) **if** ($\overline{C}_{s_{k,g}}$ does not converge) **then**
(8) Output d_g as the minimal $d_{i,j,g}$ to each source node.
(9) Output the global classes.
(10) **return**

Algorithm 4: MCR-ACA.

3.2.4. MCR-ACA Method Description. MCR-ACA is constructed by integrating MCR with the Map function conducted on various data chunks in different source nodes, the Combine one on the local clustering results, and the Reduce one on global cluster centers. Assume the maximum iteration is $g_{\max}$. The MCR-ACA method is described in Algorithm 4.

4. Experimental Results on Practical Big Data

In the experiment, the MCR-ACA method is compared with the existing MR-ACA method on the traffic monitoring system of Jiangsu province in China. The two cities Nantong and Changzhou are selected; two subsystems are chosen from each of them, respectively. Nanjing is the central city. Subsystems are linked by fiber with 1000 Mbps within each city. The distance between Nantong and the central city is 270 kilometers. There are 140 kilometers from Changzhou and the center Nanjing. The cities are connected by the Internet with network width of 200 Mbps. We adopt Hadoop, Mahout, and IK as software tools. Two PCs are used in the two cities, respectively, while four PCs work in the central node. All of them are configured with Intel 5620CPU, 2.4 GHZ, 6-core, 4 G memory, and 300 G disk. In the MCR-ACA experiment, Map and Combine operations are conducted parallel in four PCs in two cities, and the Reduce function is conducted in the central node. In the MR-ACA experiment, all data is transmitted to the central node and processed by the four PCs in the central node, where Map, Combine, and Reduce functions are performed.

The records on vehicle trajectories are represented by a set of 6 elements (HPHM, JGSJ, JGDD, XSFX, XSSD, and CSYS) with the weight P_h being 0.05, 0.3, 0.3, 0.15, 0.15, and 0.05. Since both MCR-ACA and MR-ACA are based on MapReduce, the experiments focus on the scale growth of vehicle trajectories. According to the experiments, the neighborhood radius $\mathbb{R}$ and the threshold probability p_0 of the two methods are found to be key parameters of Map functions. The parameters are mainly determined by the combination of requirements like accuracy and efficiency. Meanwhile, the two parameters affect each other and thus are usually given in pairs. As a result, the comparison experiments of the parameters are conducted in advance based on the MCR-ACA method since the Map function in both methods are roughly the same. In the experiments, there are 56 Map functions and 3 ants for each Map function, with 4 Combine functions and 4 Reduce functions, whose maximal iteration is 5. The experiment data consists of 4×10^7 vehicle trajectory records. The experiment results are shown in Tables 2 and 3.

It can be seen from Table 2 that as $\mathbb{R}$ increases the total time of clustering computation increases and the accuracy of clustering is robust. Table 3 illustrates that the total time of clustering computation tends to decrease and then increase while the accuracy of clustering fluctuates as p_0 increases. For further comparison of the effect of different parameter pairs

Table 2: The clustering time (H)/the accuracy (%) of MCR-ACA with different neighborhood radius $\mathbb{R}$.

$\mathbb{R}$	p_0	Total time	Accuracy	Total time/accuracy
0.004	0.441	52.60	86.45	1.64
0.005	0.441	53.58	87.56	1.63
0.006	0.441	53.88	88.72	1.65
0.007	0.441	54.56	88.63	1.62
0.008	0.441	54.79	87.25	1.59

Table 3: The clustering time (H)/the accuracy (%) of MCR-ACA with different threshold probability p_0.

$\mathbb{R}$	p_0	Total time	Accuracy	Total time/accuracy
0.006	0.431	54.53	87.49	1.60
0.006	0.436	54.01	87.92	1.63
0.006	0.441	53.88	88.72	1.65
0.006	0.446	54.21	88.73	1.64
0.006	0.451	55.23	88.52	1.60

of $\mathbb{R}$ and p_0 on clustering time and accuracy, the comparison is based on the accuracy in unit time (i.e., accuracy/total time). $\mathbb{R} = 0.006$ and $p_0 = 0.441$ are the ideal parameter pair according to Tables 2 and 3.

Therefore, the experimental parameters are set as below: 56 Map functions, 4 Combine functions, 4 Reduce functions, the maximal iteration being 5, 3 ants for each Map function, the neighbourhood radius $\mathbb{R}$ taking 0.006, $p_0 = 0.441$, and the similarity factor set by the range of dimensions. In the experiments, the records on vehicle trajectories are divided into 56 chunks on average with one Map function working on one data chunk. The communication overhead of data extraction is the time cost of extracting the records from Nantong and Changzhou to the central node. The results are given in Table 4. The metrics for Map, Combine, Reduce, and total time are hours, and the unit for accuracy is percentage.

Table 4 implies that the accuracy of the two methods is similar and rising as the data amount becomes larger. The data amount is key to the clustering accuracy. Reducing data scale by sampling also reduces the accuracy. The increasing rate of clustering accuracy for MCR-ACA is greater than MR-ACA, as depicted in Figure 3.

Furthermore, the computing time for Map function of MR-ACA is longer than MCR-ACA, and the difference becomes bigger as data amount increases. The reason lies in that all records are mixed up in hard disk after extraction, which makes the Map function more complicated in the central node than that in the data source nodes. The Map function of MR-ACA works on the data chunks that are divided from the data mixed stored on the central node, and these data chunks are more complex in elements as the data amount grows. Therefore, the Map function of MR-ACA costs more computing time. The comparison is indicated in Figure 4.

For Reduce function, the time consumed in MR-ACA is about twice as much as that in MCR-ACA, but less than the sum of computing time in both the Reduce function and the Combine function in MCR-ACA. Actually, the processing time in the Combine function in MCR-ACA is included in the Reduce function in MR-ACA.

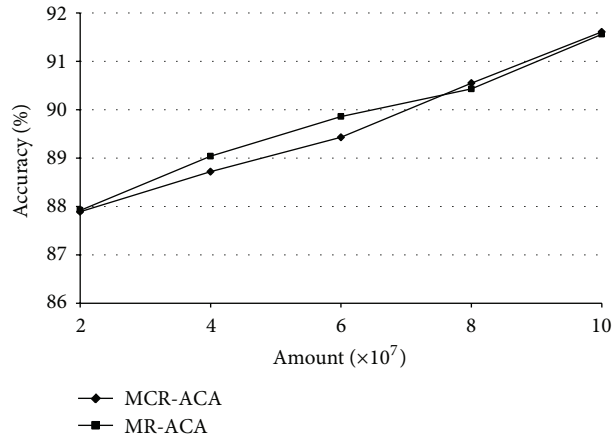

Figure 3: Comparison on group mining accuracy.

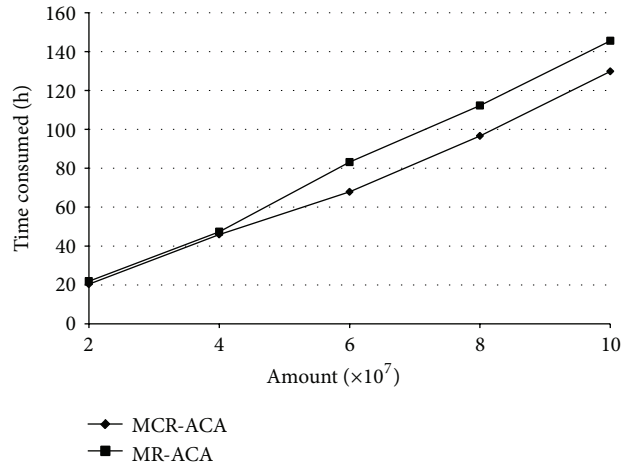

Figure 4: Comparison on Map function time consumed.

As shown in Table 4, the total time in MR-ACA is 50% longer than that in MCR-ACA due to the data extracting time on average, which indicates that the data extraction is the most essential influential factor for big data cluster.

Table 4 also demonstrates that as the data amount increases, the computation time of both of the two methods increases rapidly while the accuracy improvement is quite limited. The reason is that as the number of Map functions, Combine functions, and Reduce functions keeps the same, the amount of data in the data blocks among Map functions, Combine functions, and Reduce functions grows proportionally, which finally causes the computation time to grow rapidly. The clustering accuracy is a relative value, mainly determined by data amount (the ratio of the clustered records of one car to the total records which this car is involved in). As the data amount grows, the clustering accuracy increases gradually. However, there is no relationship between the clustering accuracy and the computation time, which leads

Table 4: Comparison experiments of MCR-ACA and MR-ACA.

Amount ($\times 10^7$)	MCR-ACA					MR-ACA					
	Map	Combine	Reduce	Total time	Accuracy	Extract	Map	Combine	Reduce	Total time	Accuracy
2	20.36	1.51	1.36	23.23	87.89	11.74	21.92	0	2.71	36.37	87.92
4	45.92	5.44	2.52	53.88	88.72	25.31	47.35	0	6.13	78.79	89.04
6	67.89	7.80	5.10	80.78	89.43	39.76	83.17	0	11.48	134.41	89.86
8	96.57	11.83	7.40	115.81	90.55	55.68	112.22	0	16.28	184.18	90.43
10	129.77	15.21	10.42	155.41	91.61	68.41	145.52	0	23.57	237.51	91.56

Table 5: The number of classes.

Amount ($\times 10^7$)	The number of classes
2	6
4	9
6	11
8	18
10	23

Table 6: Trajectories of the cars in a group.

HPHM	JGSJ	JGDD	XSFX	XSSD	CSYS
S032W0	20131227200308	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	3	62	C
S470A5	20131227200636	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	3	43	J
S560V8	20131227200638	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	3	47	A
S032V0	20131227202633	Northwest corner of the crossroad of Zhujiang road and north Taiping road	2	41	C
S560V8	20131227202957	Southeast corner of the crossroad of Hanzhongmen street and middle Jiangdong road	6	29	A
SFM979	20131227203021	Checkpoint at the crossroad of Suyuan road and west Qingshuiting road	3	39	A
S470A5	20131227203103	Southeast corner of the crossroad of Yangzijiang road and Qingliangmen street	4	40	A

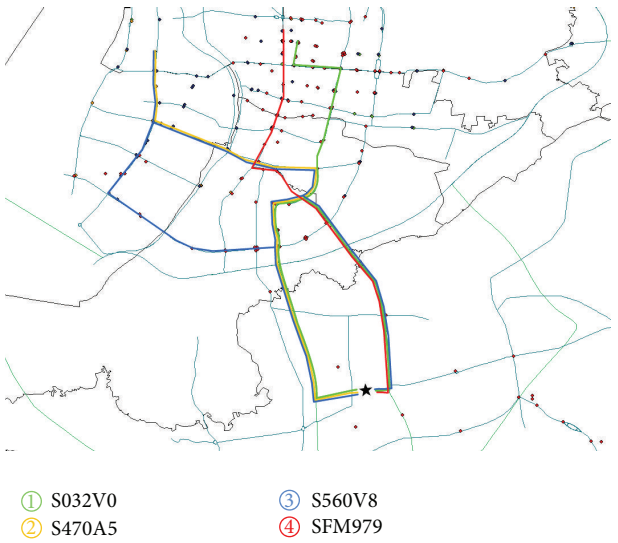

Figure 5: Trajectories of the cars in a group.

to the inconsistency of the computation time increasing and the accuracy increasing. According to the experiment, the number of obtained classes is listed in Table 5.

The results show that 4 cars in a research group illustrate the obvious similar trajectories, which is listed in Table 6.

Table 6 indicates that the 4 cars were caught by the same camera in half an hour. The car number "S032V0" was misidentified as "S032W0" by the camera capture. Through clustering, the trajectories of these 4 cars have plenty of traces with the same or similar features which construct a cluster. The time feature is on every Friday evening; the location feature is overlapped along the way to the university, as shown in Figure 5.

5. Conclusion and Future Work

Critical issues for group mining on big data of vehicle trajectories are centralization and source distribution. In this paper, a distributed parallel clustering method MCR-ACA is proposed for group mining on distributed vehicle trajectories. Parallel clustering is realized while communication overhead of big data is avoided. The method is tested on traffic monitoring systems of three cities (including the center city Nanjing) of Jiangsu province in China. Experimental results demonstrate that the proposed method achieves better performance on group mining.

Group mining can be used in many scenarios. According to the experiment results in this paper, two aspects are

promising for further work: (i) the forecast of group behavior based on specific features; for example, if the time feature of a group is in midnight and the location feature is somewhere with high crime incidence, the group can be regarded as a possible crime group with high possibility; (ii) outlier analysis for vehicle trajectory. Some vehicle trajectory outliers are formed in the clustering process (e.g., the abandoned vehicle trajectories defined in the paper); the reason that these vehicle trajectories are abandoned as outliers is useful for behavior forecast.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant 61272377) and the Key Technology R&D Program of Jiangsu Province (BE2014733).

References

[1] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[2] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: the privacy bounds of human mobility," *Scientific Reports*, vol. 3, article 1376, 2013.

[3] S. Wang, H. Wang, X. Qin et al., "Arc hitecting big data: challenges, studies and forecasts," *Chinese Journal of Computers*, vol. 34, no. 10, pp. 1741–1752, 2011.

[4] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[6] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] C. Jayalath, J. Stephen, and P. Eugster, "From the cloud to the atmosphere: running mapreduce across data centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 74–87, 2014.

[8] B. Chandramouli, J. Goldstein, and S. Duan, "Temporal analytics on big data for web advertising," in *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE '12)*, pp. 90–101, April 2012.

[9] A. Mukherjee, J. Datta, R. Jorapur et al., "Shared disk big data analytics with apache hadoop," in *Proceedings of the 19th International Conference on High Performance Computing (HiPC '12)*, pp. 1–6, IEEE, 2012.

[10] S. Fiore, A. D. Anca, C. Palazzo et al., "Ophidia: toward big data analytics for escience," *Procedia Computer Science*, vol. 1, pp. 2376–2385, 2013.

[11] Y. Kim, K. Shim, M.-S. Kim, and J. Sup Lee, "DBCURE-MR: an efficient density-based clustering algorithm for large data using MapReduce," *Information Systems*, vol. 42, pp. 15–35, 2014.

[12] N. Laptev, K. Zeng, and C. Zaniolo, "Very fast estimation for result and accuracy of big data analytics: the EARL system," in *Proceedings of the 29th International Conference on Data Engineering (CDE '13)*, pp. 1296–1299, Brisbane, Australia, April 2013.

[13] D. Garg, K. Trivedi, and B. Panchal, "A comparative study of clustering algorithms using mapreduce in hadoop," *International Journal of Engineering*, vol. 2, no. 10, 2013.

[14] W. Zhao, V. Martha, and X. Xu, "PSCAN: a parallel Structural clustering algorithm for big networks in MapReduce," in *Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA '13)*, pp. 862–869, Barcelona, Spain, March 2013.

[15] Y. Yu, P. K. Gunda, and M. Isard, "Distributed aggregation for data-parallel computing: interfaces and implementations," in *Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles (SOSP '09)*, pp. 247–260, October 2009.

[16] X. Cheng and N. Xiao, "Parallel implementa tion of dynamic positive and negative feedback aco with iterat ive mapreduce model," *Journal of Information and Computational Science*, vol. 10, no. 8, pp. 2359–2370, 2013.

[17] Y. Yang, X. Ni, H. Wang et al., "Parallel implementation of antbased clustering algorithm based on hadoop," in *Proceedings of the 3rd International Conference on Swarm Intelligence (ICSI '12)*, pp. 190–197, 2012.

[18] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, no. 6791, pp. 39–42, 2000.

[19] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851–871, 2000.