IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 19, 2016, accepted May 6, 2016, date of current version June 24, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2570021

A MapReduce-Based Nearest Neighbor Approach for Big-Data-Driven Traffic Flow Prediction

DAWEN XIA[1,2], HUAQING LI[3], BINFENG WANG[1], YANTAO LI[1], AND ZILI ZHANG[1,4]

[1]School of Computer and Information Science, Southwest University, Chongqing 400715, China
[2]School of Information Engineering, Guizhou Minzu University, Guiyang 550025, China
[3]School of Electronics and Information Engineering, Southwest University, Chongqing 400715, China
[4]School of Information Technology, Deakin University, Geelong, VIC 3220, Australia

Corresponding author: Z. Zhang (zhangzl@swu.edu.cn)

This work was supported in part by the Science and Technology Foundation of Guizhou Province under Grant LH20147386, in part by the Fundamental Research Funds for the Central Universities under Grant XDJK2016B016, Grant XDJK2015B030, and Grant SWU114006, in part by the Scientific Project of State Ethnic Affairs Commission of the People's Republic of China under Grant 14GZZ012, in part by the Natural Science Foundation of Chongqing CSTC under Grant cstc2014jcyjA40016 and Grant cstc2015jcyjA40044, and in part by the National Natural Science Foundation of China under Grant 61403314, Grant 61402380, and Grant 61528206.

ABSTRACT In big-data-driven traffic flow prediction systems, the robustness of prediction performance depends on accuracy and timeliness. This paper presents a new MapReduce-based nearest neighbor (NN) approach for *traffic flow prediction* using *correlation* analysis (TFPC) on a Hadoop platform. In particular, we develop a real-time prediction system including two key modules, i.e., offline distributed training (ODT) and online parallel prediction (OPP). Moreover, we build a parallel k-nearest neighbor optimization classifier, which incorporates correlation information among traffic flows into the classification process. Finally, we propose a novel prediction calculation method, combining the current data observed in OPP and the classification results obtained from large-scale historical data in ODT, to generate traffic flow prediction in real time. The empirical study on real-world traffic flow big data using the leave-one-out cross validation method shows that TFPC significantly outperforms four state-of-the-art prediction approaches, i.e., autoregressive integrated moving average, Naïve Bayes, multilayer perceptron neural networks, and NN regression, in terms of accuracy, which can be improved 90.07% in the best case, with an average mean absolute percent error of 5.53%. In addition, it displays excellent speedup, scaleup, and sizeup.

INDEX TERMS Big data analytics, traffic flow prediction, correlation analysis, parallel classifier, Hadoop MapReduce.

I. INTRODUCTION

Recently, the traffic data in transportation have been exploding rapidly with the characteristics of heterogeneity, autonomous sources, and complex and evolving associations (HACE) [1]. The big data generated by the Intelligent Transportation Systems (ITS) are worth further exploring to bring all their full potential for more proactive traffic management [2], [3]. The ability to accurately predict the evolution of traffic in an online and real-time manner [4] that plays a crucial role in traffic management and control applications is particularly important. Data-driven intelligent transportation has drawn significant attention in recent years [5], [6]: Reference [7] provided a special session on big data services and computational intelligence for industrial systems, including ITS applications [8]. Furthermore, a special issue highlighted the most recent research progress in big data applications [9], which could help establish fundamental knowledge, concepts and technologies related to transportation and traffic engineering, covering transportation planning, traffic operations, and safety. Another special issue focused on the latest technical progresses on the big-data-driven applications for ITS [10], such as application-oriented system models, data-driven computing techniques, and data processing methods.

In the last several decades, considerable research studies were reported on the applications of different empirical and theoretical techniques to traffic flow prediction [11]. These works can be roughly categorized as parametric methods [12], nonparametric methods [13], [14], and hybrid integration methods [15], [16]. Recently, there have been various traffic flow prediction systems, models and algorithms using statistics-based approaches [17] and computational

intelligence (CI)-based approaches [18], [19]. Lv *et al.* [2] proposed a novel deep-learning-based traffic flow prediction method, using autoencoders as building blocks to represent traffic flow features for forecasting. Li *et al.* [20] presented a method which picks the most relevant data from the "Big Data" to build concise yet accurate traffic flow prediction model. Tchrakian *et al.* [11] developed an algorithm for the implementation of short-term prediction of traffic with real-time updating based on spectral analysis. Min and Wynter [21] put forward an approach to providing predictions of speed and volume over 5-min interval for up to 1 h in advance. However, most of them employed the stand-alone learning models with the sequential algorithms on a single machine and were still somewhat unsatisfactory in processing the increasingly explosive traffic big data in real time, such as massive taxi trajectory data used in this work, containing large-scale GPS points which generate the road network of Beijing as shown in Fig. 1.

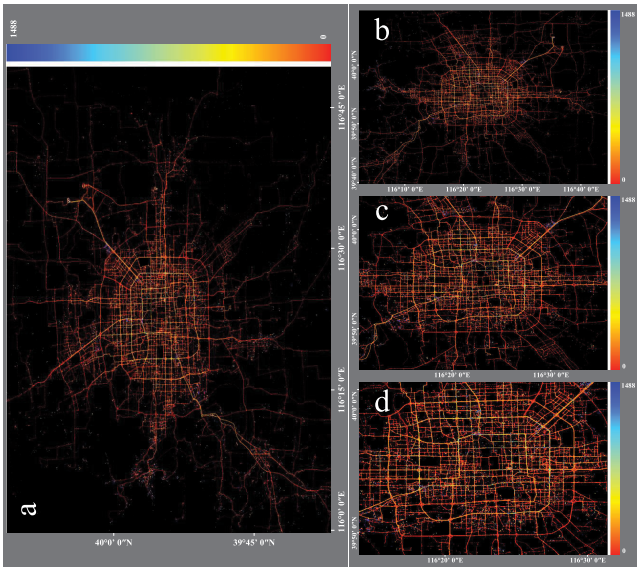


FIGURE 1. Distribution of the GPS points (1,232,048 records) generated by 12,000 taxicabs in Beijing at 00:14:35 AM-01:14:34 AM on 1 November 2012, where the color represents the density of the points. (a) Data overview in Beijing. (b) Within the 6th Ring Road, (c) the 5th Ring Road, and (d) the 4th Ring Road of Beijing.

Traffic flow conditions are extremely uncertain in current complex transportation network situation, due to the heterogeneous and dynamic nature of traffic with nonlinear interactions between drivers and environments [22]. Moreover, the traffic state of a specific location is highly influenced by its upstream and downstream traffic conditions [23], [24] besides the traffic conditions in the past and future periods, and thus the spatial and temporal correlations are the inherent features of traffic flow. Most of traditional classification approaches take the traffic flows as the individual and independent instances, which do not consider the correlation information among traffic flows. In particular, the most studied methods take into account the single road segment forecast and only seldom consider the relation among links of a road network. In other words, they take into account the temporal dimension, but ignore the spatial one [25]. On the other hand, the nearest neighbor-based approach has manifested superior classification performance with three remarkable advantages: (i) no needs of the complicated training procedure, (ii) no risk of overfitting of parameters, and (iii) naturally being capable of processing a large number of classes [26]. In this point of view, the nearest neighbor classifier is more suitable for traffic flow prediction. However, the performance of nearest neighbor classifiers can be seriously affected if the size of the training samples is extremely large. Especially for k-nearest neighbor (KNN), the computational and the storage issue is a critical problem [27]. That is, it requires the storage of the whole training set which would be an excessive amount of storage for large data sets and leads to a large computation time in the classification stage, and usually involves storing the training data in memory and finding relevant data to answer a particular query. However, these encourage us to rethink the real-time Traffic Flow Prediction (TFP) problem using distributed models with parallel algorithms based on correlation analysis and so rich amount of traffic data [5].

In this paper, we propose a new nearest neighbor approach using correlation analysis under a MapReduce framework on a Hadoop platform, to address the difficult problem of real-time prediction with very large training data. Following our previous work [28], the correlation information among traffic flows is utilized in a robust nonparametric classifier to improve the classification performance for real-time prediction applications. More specifically, this approach is implemented by a real-time prediction system (RPS) including offline distributed training (ODT) and online parallel prediction (OPP), based on a parallel k-nearest neighbor optimization (ParKNNO) classifier considering traffic flow correlations in the modeling. In ODT, flow correlation analysis and large-scale historical data are taken as the input of ParKNNO, which will generate the classification results. With the output of ODT and the current data observed, we can achieve real-time prediction results using a novel prediction calculation method in OPP. The experimental results demonstrate that the proposed approach for real-time traffic flow prediction has superior performance by speeding up the classification process and reducing the storage requirements. More importantly, with reasonable execution time, the classification performance can be significantly improved by flow correlation information, even under the extremely difficult circumstance of very large training samples.

This paper aims to develop a robust approach for the accurate and real-time *Traffic Flow Prediction* using *Correlation* analysis (TFPC). The major contributions of our work are summarized as follows:

- A new prediction system (RPS) on a Hadoop platform is built to improve the capacity of big traffic data processing for forecasting traffic flow in real time, which contains two key modules of offline distributed training (ODT) and online parallel prediction (OPP).

- A robust nearest neighbor classifier (ParKNNO) on a MapReduce framework is presented to enhance the accuracy, efficiency and scalability of traffic flow prediction, by discovering correlation information among traffic flows and incorporating it into the classification process.
- A novel prediction calculation method is put forward to generate the real-time traffic flow prediction, with the current data observed in OPP and the classification results obtained from large-scale historical data in ODT.
- The prediction performance of TFPC is investigated with real-world traffic flow data collected from 12,000 taxis of Beijing during 15 days. The empirical study indicates that the proposed approach is superior to other comparable prediction methods in terms of accuracy, speedup, scaleup, and sizeup.

The rest of this paper is organized as follows. Section II provides a background on the nearest neighbor classifier and introduces the MapReduce paradigm. The proposed approach is described in Section III in detail. Section IV presents the extensive experiments and results of performance evaluation. Finally, this paper is concluded in Section V.

II. BACKGROUND

This section provides some background information about the nearest neighbor (NN) classifier, and then describes the MapReduce paradigm used in this paper.

Algorithm 1 The Basic k-Nearest Neighbor Algorithm

Input: k: The number of nearest neighbors;
te: The testing sample;
Tr: The set of training samples.
Output: Class of each testing sample.

1: **for** each testing sample $te = (x', y')$ **do**
2: Compute $d(x', x)$, the distance between te and each sample, $(x, y) \in Tr$;
3: Select $Tr_{te} \subseteq Tr$, the set of k closest training samples to te;
4: $y' = \arg\max_{v} \sum_{(x_i, y_i) \in Tr_{te}} I(v = v_{y_i})$;
5: **end for**

A. NN CLASSIFIER

KNN is a widely applied nonparametric NN approach [13], and commonly makes classification decision on the basis of closest training samples in the feature space [29]. When $k = 1$, the NN-based classifier assigns a testing traffic flow into the class of its nearest training sample [26]. The summary of the basic KNN [30], [31] is given in Algorithm 1. The algorithm calculates the distance (or similarity) between each testing sample $te = (x', y')$ and all the training samples $(x, y) \in Tr$ to determine its nearest-neighbor list, Tr_{te}. Once the nearest-neighbor list is obtained, the testing sample is classified based on the majority class of its nearest neighbors by the Majority Voting approach (see Eq. 3).

Traditionally, the basic KNN classifier for traffic flow prediction employs the following search mechanisms:

(i) It defines the state space in the temporal dimension which only considers the temporal correlation of traffic flow to describe traffic conditions [13]. For instance, a state vector $x(t)$ of flow rate measurements collected every 5 minutes with an appropriate number of lags=4, can be defined as

$$x(t) = [V(t), V(t-1), V(t-2), V(t-3), V(t-4)], \quad (1)$$

where $V(t)$ denotes the flow rate at the current time interval; $V(t-1)$ and $V(t-2)$ represent the flow rate at the previous 5-min interval and at the previous 10-min interval, respectively, and so forth.

(ii) It measures the similarity via the basic Euclidean distance [30], [32], which is given by

$$d_i(x, y) = \sqrt{\sum_{j=1}^{n} \left(x_j - y_j^i\right)^2}, \quad (2)$$

where $d_i(x, y)$ represents the distance between current data and the i^{th} components in the historical database, x_j is the value of the j^{th} attributes in the current data vectors, y_j^i is the value of the j^{th} attributes of the i^{th} components in the historical data vectors, and n is the number of attributes.

(iii) It utilizes the Majority Voting approach in [30] where each neighbor has the same influence on the classification to determine the class label, and thus makes the classifier sensitive to the choice of k. The Majority Voting approach is given by

$$y' = \arg\max_{v} \sum_{(x_i, y_i) \in Tr_{te}} I(v = v_{y_i}), \quad (3)$$

where v is a class label, v_{y_i} is the class label for one of the nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

Especially, with the basic KNN classifier, the average of the nearest neighbors [32] which is commonly used to generate the prediction, can be defined as

$$\hat{X}_{T+1} = \frac{1}{k} \sum_{k=1}^{K} X_{T+1}^k, \quad (4)$$

where $K \in N^+$ represents the total number of the selected nearest neighbors from the training samples; $\hat{X}_{T+1}$ is the predicted traffic flow at the next time intervals $T+1$, and X_{T+1}^k is the traffic flow of the k^{th} nearest neighbors searched from the historical traffic flow database at the corresponding time intervals (i.e., $T+1$).

Moreover, the weighted average of the nearest neighborhoods [13] is also widely adopted to enhance the prediction accuracy. It is given by

$$\hat{X}_{T+1} = \sum_{k=1}^{K} \frac{d_k^{-1}}{\sum_{k=1}^{K} d_k^{-1}} X_{T+1}^k, \quad (5)$$

where d_k denotes the distance between the current data and the k^{th} nearest neighbors. During the performance evaluation (see Section IV), KNN employs this method for the forecast generation.

Based on the search mechanisms of the basic KNN classifier and prediction calculation method, we can observe that it ignores the correlation information in traffic flows, especially for the spatial correlations. Furthermore, according to Algorithm 1, the computational cost of finding the nearest neighbors of testing samples would be very high in the sequential execution mode on a single machine, particularly when the number of training samples is extremely large. To tackle the above problems, we aim to optimize the state space, the similarity measure, the choice of k and the prediction calculation of KNN using correlation analysis for improving prediction accuracy. On the other hand, to reduce computational costs and save storage requirements of real-time prediction applications with large training samples (e.g., big traffic flow data), we implement the KNN optimization classifier under a MapReduce parallel processing framework on a Hadoop distributed computing platform for enhancing prediction timeless.

B. MAPREDUCE PARADIGM

The Apache Hadoop project provides open-source software for reliable, scalable and distributed computing, and allows for the distributed processing of large-scale data sets across clusters of computers with simple programming models. Hadoop Distributed File System (HDFS) [33] and Hadoop MapReduce [34], [35] are the key components of Hadoop. Specifically, HDFS is a distributed file system designed for storing very large files by offering high-throughput access to application data [35], and MapReduce is a YARN-based programming paradigm for parallel processing of massive data sets [36]. The process of MapReduce jobs includes the Map phase and the Reduce phase. Each phase has *key-value* pairs as input and output, the types of which may be chosen by the programmer through specifying two functions: the Map function and the Reduce function [35]. For more details, see the Apache Hadoop website.[1]

In this work, with correlation analysis, we employ a general architecture of distributed modeling on a MapReduce framework for traffic flow forecasting (MF-TFF) [28].

III. PROPOSED APPROACH: TFPC

This section presents a MapReduce-based nearest neighbor approach to solve the real-time application problem of traffic flow prediction using correlation analysis (TFPC) with current and historical traffic big data, by discovering flow correlation information and incorporating it into the classification process.

We aim to develop a new framework (RPS) to handle real-time prediction applications. Also, a robust nonparametric classifier (ParKNNO) is put forward to improve the classification performance by effectively incorporating correlation of traffic flows, and a novel prediction calculation method is proposed to accurately generate the prediction in real time.

[1] http://hadoop.apache.org/

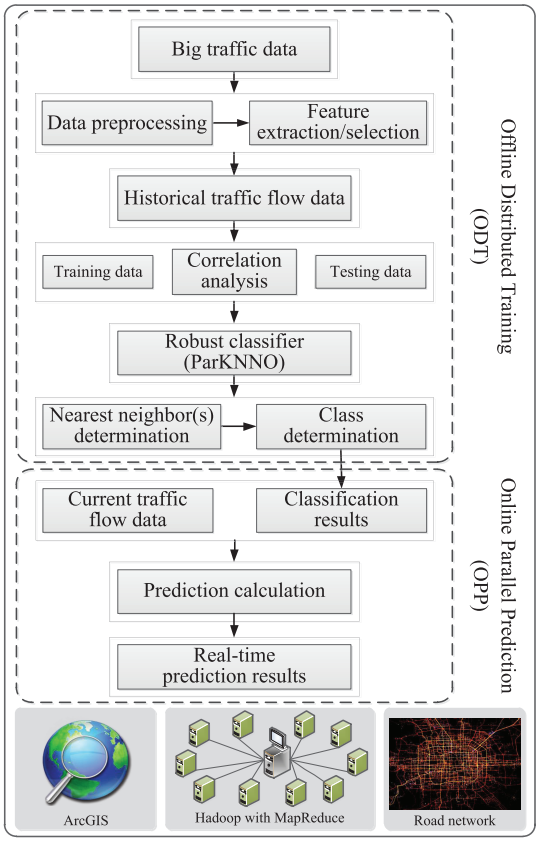


FIGURE 2. A real-time prediction system (RPS) framework.

A. SYSTEM FRAMEWORK

Figure 2 shows the real-time prediction system (RPS) framework on a Hadoop platform combining MapReduce, ArcGIS and road network. There are two important modules: offline distributed training (ODT) and online parallel prediction (OPP). The module of ODT takes correlation analysis (see Section III-B) and the training and testing data (or samples) as input to build a robust classifier, ParKNNO, for the accurate traffic flow classification (see Section III-C), and particularly focuses on fast finding the nearest neighbors of a testing sample through speeding up the classification process and reducing the storage requirements on a distributed Hadoop platform. The module of OPP aims to generate prediction results in real time with the current traffic flow data and the classification results of ParKNNO obtained from large-scale historical traffic flow data, using a novel prediction calculation method (see Section III-D), by reducing computational cost and saving memory consumption of big traffic flow data processing on a parallel MapReduce framework (see Section III-E).

In particular, to build robust classification models, flow correlation analysis is presented to correlate information in

the traffic flows. Moreover, the robust classification engine classifies traffic flows into application-based classes by taking all information of relevant features and flow correlations into account.

The novelty of the proposed framework is to consider correlation information in the traffic flows and incorporate it into the classification process. In this work, we employ ParKNNO to model traffic flow correlations using the MF-TFF framework.

B. CORRELATION ANALYSIS

Naturally, the spatial and temporal correlations are the inherent features of traffic flow in a complex urban transportation networks. The traffic flow of the target road segment at the future time interval is closely related to that of the same road segment at the previous and current time intervals. Moreover, the traffic flow of a particular road segment is highly affected by its upstream traffic conditions and its downstream traffic conditions. In other words, traffic on a road is influenced by traffic on nearby roads, and the traffic flow on a road segment is correlated with previous traffic flows on the same road segment. Temporal correlations refer to the correlations of the current and past (or historical) traffic flows at the current time interval and the future traffic flow at the next time interval on the same road segment, while spatial correlations refer to the correlations of the traffic flow of the target road segment and that of its upstream and downstream road segments at the same time interval.

To predict the traffic flow at the next time interval, it is extremely important to consider the correlations of traffic flow at the past-future time intervals on the same road segment and its upstream-downstream road segments at the same time interval, with the current traffic flow condition. In addition, by capturing recurring traffic patterns, the trend adjustment feature of traffic flow is merged into our approach to improve prediction accuracy.

In this work, we take the aforementioned correlation among traffic flows into account in the traffic flow prediction, and conduct correlation analysis by ParKNNO. The results reported in Section IV indicate that the correlation information can significantly improve the classification performance, particularly for very large training samples.

C. ROBUST CLASSIFIER

We propose a robust nonparametric classifier (ParKNNO) to model the correlation analysis for improving the accuracy of traffic flow prediction. ParKNNO incorporates the correlation information among traffic flows into the classification process by optimizing the definition of state space, the method of similarity measure, and the choice of k.

1) STATE SPACE

To more sufficiently and accurately describe traffic conditions, we consider the spatio-temporal correlation of traffic flow (i.e., the state vector in both spatial and temporal dimensions) in the definition of state space. That is, we employ the state vector of describing spatial and temporal correlation information among traffic flows. For example, when $T = 2$, the hybrid state vector, $x(t)$, is defined as:

$$x(t) = \Big[V_{t-1}^{r_{j-1}}, V_t^{r_{j-1}}, V_{t-1}^{r_j}, V_t^{r_j}, V_{t-1}^{r_{j+1}}, V_t^{r_{j+1}}, \tilde{V}_{t-1}^{r_{j-1}}, \tilde{V}_t^{r_{j-1}}, \tilde{V}_{t-1}^{r_j}, \tilde{V}_t^{r_j}, \tilde{V}_{t-1}^{r_{j+1}}, \tilde{V}_t^{r_{j+1}}\Big], \tag{6}$$

where T is the continuous time intervals as a combined group, $t = 1, 2, \ldots, T$; $\tilde{V}_t^{r_j}$, $\tilde{V}_t^{r_{j-1}}$, and $\tilde{V}_t^{r_{j+1}}$ are the traffic flow obtained from the historical traffic data on the target road segment r_j, its upstream road segment r_{j-1}, and its downstream road segment r_{j+1} at the current time interval t, respectively; $\tilde{V}_{t-1}^{r_j}$, $\tilde{V}_{t-1}^{r_{j-1}}$, and $\tilde{V}_{t-1}^{r_{j+1}}$ are the traffic flow of the corresponding road segment at the previous time interval $t-1$.

2) SIMILARITY MEASURE

Based on the hybrid state vector with spatio-temporal correlations, the definition of distance function for similarity measure not only considers the spatio-temporal correlation and weight of traffic flows, but also exploits the signs and magnitudes of changes in traffic flows. It can be given by

$$d_i(x, y) = \alpha \sqrt{\sum_{t=1}^{T} \beta^{T-t+1} \left(x_t - y_t^i\right)^2} + (1-\alpha) \sqrt{\sum_{t=2}^{T} \sum_{\delta=1}^{t-1} \left[(x_t - x_\delta) - \left(y_t^i - y_\delta^i\right)\right]^2}, \tag{7}$$

where $0 \le \alpha \le 1$, $0 < \beta < 1$, and $T \in N^+$ is the total number of continuous time intervals included; $d_i(x, y)$ is the distance between the current traffic flow data and the i^{th} components in the historical traffic flow data, and x_t is the value of the t^{th} time intervals in the current traffic flow data; i is the sequence number of the historical traffic flow data, and y_t^i is the value of the t^{th} time intervals of the corresponding traffic flow record (i.e., the i^{th} components) in the historical data.

Hence, the distance for both spatial and temporal dimensions can be computed by

$$d_i^{sum}(x, y) = d_i^{up}(x, y) + d_i^{tar}(x, y) + d_i^{down}(x, y), \tag{8}$$

where $d_i^{sum}(x, y)$ is the total distance; $d_i^{up}(x, y)$ and $d_i^{down}(x, y)$ are the distance of calculating the corresponding upstream road segment and downstream road segment of the target road segment by Eq. 7, respectively, and $d_i^{tar}(x, y)$ is the distance of computing the target road segment.

3) K SELECTION

As discussed previously, each neighbor has the same influence on the classification by the Majority Voting approach, thus making the classifier sensitive to the choice of k. To reduce the impact of k, the Distance Weighted Voting

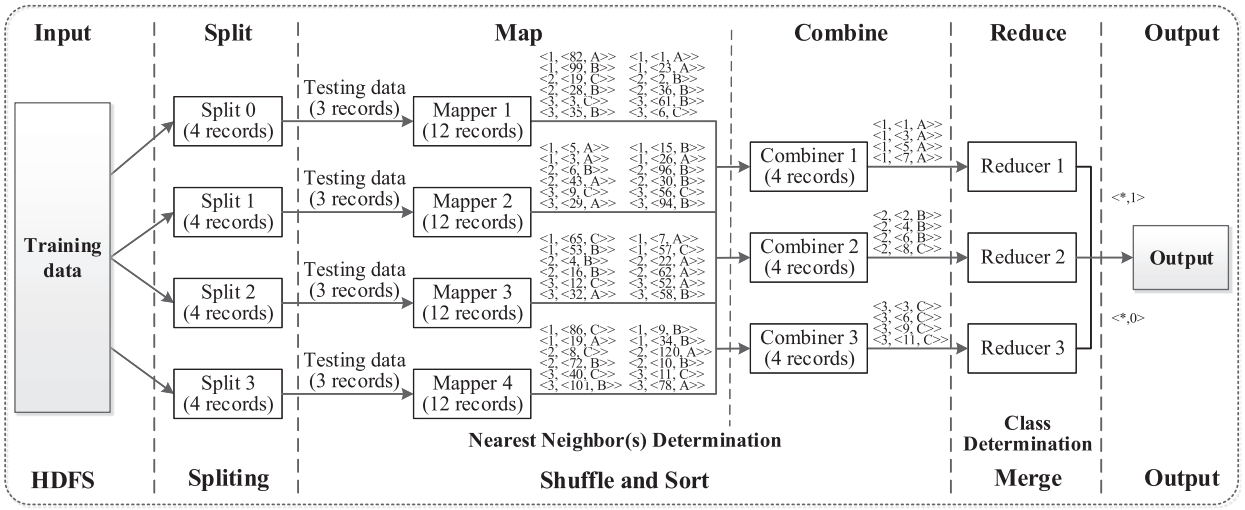


FIGURE 3. The computation flowchart of ParKNNO on MapReduce and its data flow with $k = 4$.

scheme is used in ParKNNO, and the class label can be determined by

$$y' = \arg\max_{v} \sum_{(x_i, y_i) \in Tr_{te}} \omega_i \times I(v = v_{y_i}), \tag{9}$$

where $\omega_i = \frac{1}{d(x', x_i)^2}$ is the weight of each nearest neighbor x_i according to its distance, v is a class label, v_{y_i} is the class label for one of the nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

D. PREDICTION CALCULATION

Given the selected nearest-neighbors and the predicted point as defined above with correlation analysis, the method of prediction calculation which takes the combined weights between the weighted average of the nearest neighbors and the trend adjustments (between T and t) of each nearest neighbor, is utilized to generate the traffic flow prediction. It is defined as

$$\hat{X}_{T+1} = \gamma \sum_{k=1}^{K} \frac{d_k^{-1}}{\sum_{k=1}^{K} d_k^{-1}} X_{T+1}^k + (1-\gamma)\left[X_T + \frac{1}{KT}\sum_{k=1}^{K}\sum_{t=1}^{T}\left(X_{T+1}^k - X_t^k\right)\right], \tag{10}$$

where $0 \le \gamma \le 1$, d_k is the distance between the current data and the k^{th} nearest neighbors, $K \in N^+$ is the total number of the selected nearest neighbors from the training samples; X_T is the current traffic flow at the time interval T, $\hat{X}_{T+1}$ is the predicted traffic flow at the next time interval $T+1$, and X_{T+1}^k is the traffic flow of the k^{th} nearest neighbors searched from the historical traffic flow database at the corresponding time intervals (i.e., $T+1$).

E. IMPLEMENTATION ON MAPREDUCE

This subsection describes the implementation of ParKNNO classifier following a MapReduce procedure to speed up the classification process and reduce the storage requirements for real-time traffic flow prediction.

To improve the efficiency and scalability, we implement ParKNNO under a MapReduce framework on a Hadoop platform. ParKNNO is split into multiple MapReduce jobs, which are carried out in the Map, Combine, and Reduce phases by the corresponding functions, respectively.

- Map phase: the *Mapper function* produces the *key-value* pair of the testing and training samples with the spatio-temporal state vector (see Eq. 6). Next, it calculates the distance between each testing sample and the training samples by Eqs. 7 and 8, and finally outputs the intermediate data to the *Combiner function*.
- Combine phase: the *Combiner function* sorts the intermediate results by the distance. Subsequently, it selects the local k nearest neighbors and then outputs them to the *Reducer function*.
- Reduce phase: the *Reducer function* determines the class label via Eq. 9. After that, it judges the classification results, and lastly outputs the global k nearest neighbors.

Figure 3 illustrates the computation flowchart of ParKNNO on MapReduce and its data flow of MapReduce with an example of 4-nearest neighbors. After the Map, Combine and Reduce tasks, we can utilize current traffic flow data to predict traffic flow for the next time interval in real time using Eq. 10, based on the selected k nearest neighbors outputted from MapReduce jobs.

As described above, we design the Mapper, Combiner, and Reducer functions to reduce computational costs and save memory consumption for classification. The number of Mappers relies on input splits, whereas the number of Combiners and Reducers is equal to the group of keys. As shown in Fig. 3, this example has 3 records of testing data,

Algorithm 2 *Mapper Function*()

Input: $\langle key, value\rangle$
key: the record id of the testing and training samples
$value$: the value of the testing and training samples
Output: $\langle key_1, value_1\rangle$
key_1: the record id of the testing samples, id
$value_1$: vector (the calculated distance, d, and the selected class, c)

```
for i = 0 to n do
    //the training samples, i, and the extracted class labels, C_Tr
    C_Tr = GetClassLabel (i);
    for all p ∈ testing samples do
        //each testing sample, p
        d = Our DistanceFunction (p, i);
        // Eqs. 7 and 8
        Context.write(id, vector (d, C_Tr));
    end for
end for
return ⟨id, ⟨d, c⟩⟩;
```

Algorithm 3 *Combiner Function* ()

Input: $\langle key_1, value_1\rangle$
key_1: the same key of Map output for each Combiner
$value_1$: the corresponding value with the same key of Map output
Output: $\langle key_2, value_2\rangle$
key_2: id
$value_2$: $\langle d, c\rangle$

```
//Walk through each testing sample and add them to an ArrayList to sort key-value pairs by the distance (i.e., d)
for all key and value do
    ArrayList.add (vector(⟨d, c⟩));
    Sort (ArrayList);
    //Select the k key-value pairs with the smallest distance and output them to Reducer
    Context.write(id, ArrayList.get(k));
end for
return ⟨key_2, value_2⟩ pairs;
```

and 16 records of training data which are partitioned into 4 splits (i.e., each split contains 4 records). In addition, each Mapper has an entire testing set with 3 data records.

1) MAPPER FUNCTION

In the Map phase, the *Mapper function* (see Algorithm 2) receives each line in the training and testing sets as a different *key-value* pair, then walks through each testing sample and computes the similarity (or distance) between each pair of testing data points and training data points, and finally outputs the intermediate results, $\langle id, \langle d, c\rangle\rangle$, which form the input to the *Combiner function*. That is, each produced pair is composed of the corresponding record id emitted as the *key*, and the calculated distance and the selected class, $\langle d, c\rangle$, emitted as the *value*. For example, $\langle 1, \langle 82, A\rangle\rangle$ can be interpreted as: record id is 1, distance is 82, and the data point belongs in class A. Evidently, there is a total of $N_{Te} \times N_{Tr}$ pairs, where N_{Te} and N_{Tr} are the number of all the testing sample records and the training sample records, respectively.

Furthermore, it is an example (see Fig. 3) where a single Mapper receives 4 training records and 3 testing records, and then calculates the distance between each data point and produces 12 *key-value* pairs with 3 different *keys* (1, 2 and 3). All the Mappers output 48 *key-value* pairs in total to *Combiners*.

2) COMBINER FUNCTION

To reduce the computational complexity of MapReduce jobs and save the limited bandwidth available on a Hadoop cluster, the *Combiner function* is employed to cut down the amount of data shuffled between the Map tasks and the Reduce tasks.

In the Combine phase, the *Combiner function* (see Algorithm 3) is specified to be conducted on the Map output and its output forms the input to the *Reducer function*. In this work, the *Combiner function* is employed to perform the determination task of the local k nearest neighbors. Each *Combiner function* receives the intermediate data (i.e., a set of $\langle id, \langle d, c\rangle\rangle$ pairs) with the same *key* (i.e., record id), and then sorts the *key-value* pairs by the distance in ascending order. According to the sorted pairs, only the k *key-value* pairs with the smallest distance will be output to Reducer. That is, the selected pairs are confirmed as the nearest neighbors. Obviously, the intermediate data would be reduced through the following rate equation:

$$R = \left(1 - \frac{k}{N_{Tr}}\right) \times 100\%, \tag{11}$$

where k is the number of the selected nearest neighbors, N_{Tr} is the number of all the training sample records.

In this example mentioned above, the *key-value* pairs with the same *key* are sent to the same Combiner, and thus 3 Combiners are invoked owing to 3 different *keys*. For instance, the sorted pairs of Combiner 1 are $\langle 1, \langle 1, A\rangle\rangle$, $\langle 1, \langle 3, A\rangle\rangle$, $\langle 1, \langle 5, A\rangle\rangle$, $\langle 1, \langle 7, A\rangle\rangle$, $\langle 1, \langle 9, B\rangle\rangle$, $\langle 1, \langle 15, B\rangle\rangle$, $\langle 1, \langle 19, A\rangle\rangle$, $\langle 1, \langle 23, A\rangle\rangle$, $\langle 1, \langle 26, A\rangle\rangle$, $\langle 1, \langle 34, B\rangle\rangle$, $\langle 1, \langle 53, B\rangle\rangle$, $\langle 1, \langle 57, C\rangle\rangle$, $\langle 1, \langle 65, C\rangle\rangle$, $\langle 1, \langle 82, A\rangle\rangle$, $\langle 1, \langle 86, C\rangle\rangle$ and $\langle 1, \langle 99, B\rangle\rangle$, and the nearest neighbor list is the first 4 pairs which will be sent to Reducers. Most importantly, 75% of the intermediate data can be reduced by the *Combiner function*.

3) REDUCER FUNCTION

In the Reduce phase, the Reduce task is designed to determine the global nearest neighbors. The *Reducer function* (see Algorithm 4) emits the Combiner output (i.e., the k *key-value* pairs with the smallest distance) as an input. Next, the Distance Weighted Voting scheme (see Section III-C.3) is employed to determine the class label. Then, Reducer judges

Algorithm 4 *Reducer Function* ()

Input: $\langle key_2, value_2 \rangle$
key_2: the key of Combiner output
$value_2$: the value of Combiner output
Output: $\langle key_3, value_3 \rangle$
key_3: *id*
$value_3$: 0 or 1
Add the *key-value* pairs to the ArrayList for the sorting operation
for all *key* and *value* **do**
  ArrayList (vector($\langle d, c \rangle$));
  Sort (ArrayList);
  New ArrayList result;
  //Add k samples to result
  result.add (id, ArrayList.get(k));
  //Determine the class label of testing sample using Eq. 9, and return 1 or 0 to indicator function by judgement
  Context.write(id, ParKNNO(result));
  Sent the classification results to the main program and compute the fitness value
end for
return $\langle key_3, value_3 \rangle$ pairs;

whether the classification result is accurate and accordingly returns the value 1 or 0 to indicator function. Finally, the classification result will be sent to the main program and the fitness value will be computed.

From the aforementioned example, Reducer 1 receives $\langle 1, \langle 1, A \rangle \rangle$, $\langle 1, \langle 3, A \rangle \rangle$, $\langle 1, \langle 5, A \rangle \rangle$ and $\langle 1, \langle 7, A \rangle \rangle$ pairs from the Combine phase as input. In this case, the class of its neighbors is determined as A via the Distance Weighted Voting scheme. Once the class is identified, Reducer judges if the class is true and then returns the value 1 to the main program, and 0 otherwise (i.e., $\langle *, 1 \rangle$ will be written; otherwise, $\langle *, 0 \rangle$).

F. COMPLEXITY ANALYSIS

As the fact that MapReduce is a programming model for big data processing and MapReduce programs especially are inherently parallel, ParKNNO with MapReduce implementation puts a huge number of computational tasks into different nodes of a cluster. In terms of this parallel processing paradigm, the time complexity of ParKNNO is $O(n \times s) \big/ (p \times m)$, where n is the total number of the training samples, s is the number of sample attributes, and p is the number of nodes and each node has m core(s). Moreover, the space complexity is $O(n) \big/ p$.

In comparison with the corresponding sequential version, ParKNNO can improve the classification efficiency by the following rate equation:

$$O = \left(1 - \frac{1}{p \times m}\right) \times 100\%. \tag{12}$$

Based on the above-mentioned analysis, we can find that ParKNNO significantly speeds up the classification process and reduces the storage requirements, with the desired computational complexity.

IV. PERFORMANCE EVALUATION

This section reports the experiments and results of an empirical study, which is conducted to validate the performance of the proposed approach in terms of accuracy, efficiency and scalability. The evaluation setup and evaluation metrics are introduced, then different prediction methods are tested on the same data sets, and finally the results are analyzed in detail.

To evaluate the performance of the proposed approach (TFPC) in accuracy, we compare it with the state-of-the-art prediction methods: autoregressive integrated moving average (ARIMA), Naïve Bayes (NB), multilayer perceptron neural networks (MLP-NN), and conventional k-nearest neighbor (KNN). Four metrics are utilized to validate the accuracy: mean absolute percent error (MAPE), root mean square error (RMSE), mean absolute error (MAE), and maximum error (ME). Furthermore, for the efficiency and scalability, we adopt three well-accepted metrics: speedup, scaleup, and sizeup.

A. EVALUATION SETUP

Technically, based on a Hadoop platform including 1 Master machine and 10 Slave machines with Intel Xeon (R) E7-4820 2.00GHz CPU (4-cores) and 8.00GB RAM, all the experiments are performed on Ubuntu 12.04 OS with Hadoop 1.0.4 and JDK 1.6.0.

Moreover, we utilize real-world trajectory data set[2] which is composed of large-scale GPS trajectories generated by 12,000 taxis between 1 November and 15 November 2012. The total number of GPS points reaches 484 million and the total distance of the data set surpasses 25 million kilometers, and particularly the total size is approximately 25GB. Since the available taxi trajectory data for 15 days are finite in this case study, the test is conducted using the leave-one-out cross validation (LOO-CV) method [37] to provide a consistent validation test. During testing, we select the data of 14 days as training set and choose the remaining 1 day's data as testing set, and then the average performance across the 15 test days is employed to compare the prediction accuracy of different approaches.

Additionally, we perform the map-matching task to correct the deviation between GPS points of taxis and road network of Beijing after data preprocessing (see Fig. 4). Furthermore, we select the *Yuetan N. St.* $\rightarrow$ *Yuetan S. St.* of Sanlihe E. Rd. in the city of Beijing as the target road segment, *Fuchengmen Outer St.* $\rightarrow$ *Yuetan N. St.* as its upstream road segment, and *Yuetan S. St.* $\rightarrow$ *Fuxingmen Outer St.* as its downstream road segment, respectively, where we then extract the relevant features of the taxi trajectory records in the morning peak hours (from 07:00 AM to 09:00 AM) and evening peak hours (from 17:00 PM to 19:00 PM) from the aforementioned

[2]http://www.datatang.com/

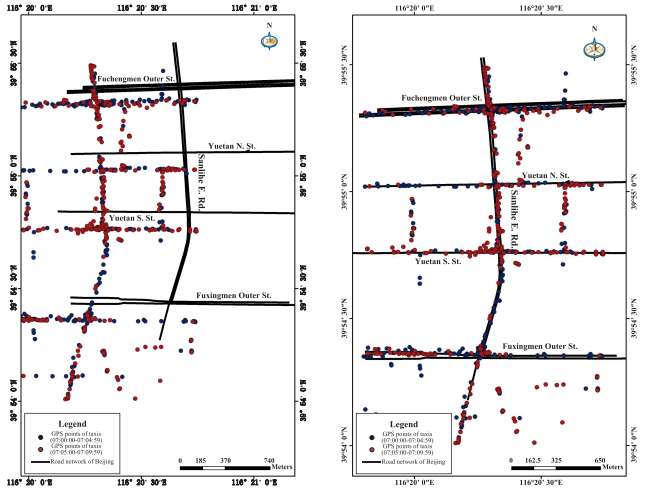


FIGURE 4. An example of map-matching between GPS points of taxis and road network of Beijing (e.g., GPS points of taxis in Sanlihe E. Rd. at 07:00 AM-07:10 AM on 1 November 2012.). (a) Before map-matching and (b) after map-matching.

data set, based on the ArcGIS platform and the road network of Beijing[3] composed of 106,579 road nodes and 141,380 road segments. Finally, we take 5 minutes as the time interval for prediction, and choose 5 continuous time intervals as a combined group (i.e., $T = 5$).

B. EVALUATION METRICS

In this performance evaluation, we validate the effectiveness of TFPC on accuracy, speedup, scaleup and sizeup using the following metrics.

1) MOEs

Four different accuracy metrics are commonly applied to the measures of effectiveness (MOEs) [38]: (i) MAPE, (ii) RMSE, (iii) MAE, and (vi) ME. The lower the MOEs represents, the more accurate predictions. The MAPE, RMSE, MAE, and ME metrics are defined as

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|X_t - \hat{X}_t|}{X_t} \times 100\%, \tag{13}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(X_t - \hat{X}_t)^2}, \tag{14}$$

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|X_t - \hat{X}_t|, \tag{15}$$

$$\text{ME} = \max_{t=1,\ldots,n}|X_t - \hat{X}_t|, \tag{16}$$

where X_t is the actual value of traffic flow observed at the time interval t, $\hat{X}_t$ is the prediction value for the same time interval; and n is the total number of the traffic flow observations processed during the time intervals provided.

[3]http://www.datatang.com/data/43855

As in many other research studies [31], [38], we compare mainly the prediction accuracy of each method with MAPE [13], [39]. In particular, to find the best combination of parameters k, α, β, γ to achieve prediction results with the highest accuracy, each combination of four parameters is evaluated for the lowest MAPE value within the reasonable range where $k = 1, 2, \ldots, 50$, $\alpha = 0, 0.1, \ldots, 1$, $\beta = 0, 0.1, \ldots, 1$ and $\gamma = 0, 0.1, \ldots, 1$, and the optimal combinations are produced for KNN(k) and TFPC(k, α, β, γ).

2) 3Ss

For further verifying the efficiency and scalability, we evaluate the 3Ss characteristics (i.e., Speedup, Scaleup, and Sizeup [40], [41]) of the parallel ParKNNO classifier through employing the following three metrics.

(i) The speedup metric measures how much the parallel algorithm is faster than the corresponding sequential algorithm, which is defined as

$$Speedup = \frac{T_1}{T_p}, \tag{17}$$

where T_1 represents the sequential execution time of the algorithm on 1 node for traffic flow prediction using the given data set, and T_p denotes the parallel execution time of the algorithm for solving the same issue using the same data set on a cluster with p nodes.

(ii) The scaleup metric verifies how well the parallel algorithm processes larger data sets when more nodes are available, which is given by

$$Scaleup = \frac{T_1}{\tilde{T}_p}, \tag{18}$$

where T_1 is the sequential execution time of the algorithm for dealing with the given data set on 1 node, and $\tilde{T}_p$ is the parallel execution time of the algorithm for handling p-times larger data sets on p-times larger nodes.

(iii) The sizeup metric validates how much longer the parallel algorithm takes on a given node, when the size of the data set is larger than the original data set. It can be defined as

$$Sizeup = \frac{\tilde{T}_x}{T_x}, \tag{19}$$

where T_x denotes the execution time of the algorithm for processing the given data set on the given node, and $\tilde{T}_x$ represents the execution time of the algorithm for coping with p-times larger data sets on the same node. Here, sizeup analysis is to keep the number of nodes constant and grow the size of the data set by the factor p, to evaluate the variation of execution time.

C. ACCURACY EVALUATION

We investigate the accuracy of the proposed TFPC approach compared to four well-known prediction methods consisting of conventional KNN, MLP-NN, NB and ARIMA using MOEs metrics, and then report the results in Table 1.

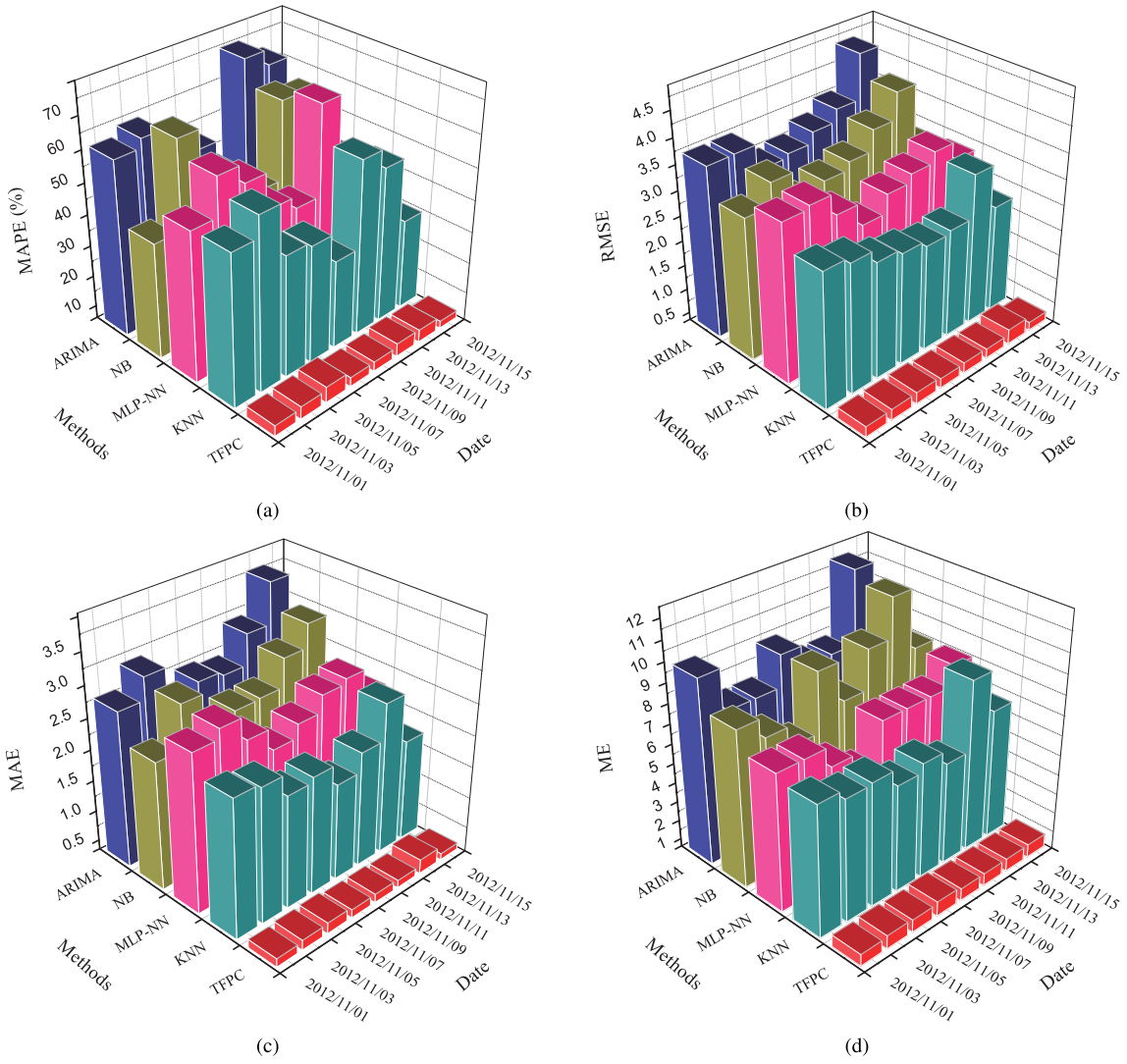


FIGURE 5. MOEs of TFPC, KNN, MLP-NN, NB, and ARIMA. (a) MAPE, (b) RMSE, (c) MAE, and (d) ME.

TABLE 1. Average MOEs comparison of five methods for all days.

MOEs	Methods				
	ARIMA	NB	MLP-NN	KNN	**TFPC**
MAPE(%)	65.3692	59.7228	55.6668	53.7348	**5.5290**
RMSE	3.4402	3.2282	2.9774	2.7116	**0.3163**
MAE	2.7539	2.5749	2.4225	2.1841	**0.2507**
ME	8.1200	7.8000	6.7333	6.8802	**0.7634**

Table 1 shows the average MOEs values of ARIMA, NB, MLP-NN, KNN and TFPC for all days. It could be found that the MAPE, RMSE, MAE and ME values of TFPC are much lower than other methods on an average (e.g., the average MAPE, RMSE, MAE and ME values of TFPC are respectively 89.7105%, 88.3367%, 88.5226% and 88.9042% lower than KNN, and even are over 90.0676%, 89.3781%, 89.6520% and 88.6622% lower than MLP-NN, NB and ARIMA, respectively). In particular, the average MAPE of TFPC is 5.5290%, which indicates TFPC has a highly accurate prediction capability because of MAPE $< 10\%$ [42].

For a more intuitive illustration, the MAPE, RMSE, MAE and ME values of all methods for 8 days selected from the experimental results are depicted in Figs. 5(a), 5(b), 5(c) and 5(d), respectively, based on the same trend of the experimental results for 15 days. From these results, we can clearly observe that TFPC achieves more accurate predictions than other comparable methods.

Furthermore, Tables 2 and 3 report the prediction results of TFPC compared to conventional KNN in the morning peak period and the evening peak period on 15 November 2012, respectively. It could be found that TFPC displays better prediction performance than the conventional KNN by significantly decreasing the MOEs values. In addition, to illustrate well the prediction performance of different methods, the prediction results of the morning peak hours and the evening peak hours generated by TFPC and four state-of-the-art

TABLE 2. Traffic flow prediction results in the morning peak period (KNN(48), TFPC(6, 0.8, 0.4, 0.9)).

Peak periods [a]	X_{T-4} [b]			X_{T-3}			X_{T-2}			X_{T-1}			X_T			X_{T+1} [c]		
	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Actual value	Prediction value KNN	Prediction value TFPC
11150725	2	6	10	7	8	1	3	3	8	5	10	8	8	9	7	7	6.21	7.16
11150730	7	8	1	3	3	8	5	10	8	8	9	7	5	7	10	7	7.03	7.16
11150735	3	3	8	5	10	8	8	9	7	5	7	10	4	7	6	4	6.88	4.18
11150740	5	10	8	8	9	7	5	7	10	4	7	6	5	4	11	6	6.28	5.72
11150745	8	9	7	5	7	10	4	7	6	5	4	11	8	6	9	7	5.86	6.86
11150750	5	7	10	4	7	6	5	4	11	8	6	9	9	7	10	7	5.57	6.92
11150755	4	7	6	5	4	11	8	6	9	9	7	10	12	7	5	9	5.28	8.62
11150800	5	4	11	8	6	9	9	7	10	12	7	5	6	9	11	5	6.30	5.24
11150805	8	6	9	9	7	10	12	7	5	6	9	11	6	5	7	7	6.36	6.83
11150810	9	7	10	12	7	5	6	9	11	6	5	7	6	7	3	2	6.44	2.47
11150815	12	7	5	6	9	11	6	5	7	6	7	3	5	2	4	5	5.48	4.72
11150820	6	9	11	6	5	7	6	7	3	5	2	4	9	5	5	3	5.22	3.01
11150825	6	5	7	6	7	3	5	2	4	9	5	5	10	3	7	4	5.04	3.96
11150830	6	7	3	5	2	4	9	5	5	10	3	7	10	4	7	5	4.79	4.77
11150835	5	2	4	9	5	5	10	3	7	10	4	7	2	5	6	7	4.59	6.83
11150840	9	5	5	10	3	7	10	4	7	2	5	6	8	7	4	6	5.35	6.04
11150845	10	3	7	10	4	7	2	5	6	8	7	4	9	6	8	12	5.36	11.41
11150850	10	4	7	2	5	6	8	7	4	9	6	8	12	12	13	6	6.91	6.53
11150855	2	5	6	8	7	4	9	6	8	12	12	13	6	6	10	9	6.63	8.78
MAPE(%)																	35.75	4.24
RMSE																	2.42	0.28
MAE																	1.77	0.22
ME																	6.64	0.59

[a] "11150725" represents the 07:25 on 15 November 2012, and so forth.
[b] "Target", "Upstream", and "Downstream" denote the traffic flow of the target road segment, its upstream road segment and its downstream road segment, respectively.
[c] "Actual value" and "Prediction value" are the actual traffic flow observed and the predicted traffic flow of corresponding methods on the target road segment at the next time interval, respectively.

TABLE 3. Traffic flow prediction results in the evening peak period (KNN(48), TFPC(6, 0.8, 0.4, 0.9)).

Peak periods [a]	X_{T-4} [b]			X_{T-3}			X_{T-2}			X_{T-1}			X_T			X_{T+1} [c]		
	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Up stream	Target	Down stream	Actual value	Prediction value KNN	Prediction value TFPC
11151725	9	11	10	9	4	11	7	5	10	9	8	12	10	8	12	8	5.87	7.91
11151730	9	4	11	7	5	10	9	8	12	10	8	12	5	8	10	8	6.68	7.99
11151735	7	5	10	9	8	12	10	8	12	5	8	10	4	8	8	5	6.93	5.10
11151740	9	8	12	10	8	12	5	8	10	4	8	8	7	5	10	7	6.43	6.71
11151745	10	8	12	5	8	10	4	8	8	7	5	10	6	7	12	5	6.32	5.15
11151750	5	8	10	4	8	8	7	5	10	6	7	12	8	5	12	6	5.70	5.81
11151755	4	8	8	7	5	10	6	7	12	8	5	12	6	6	14	8	5.86	7.66
11151800	7	5	10	6	7	12	8	5	12	6	6	14	8	8	10	8	5.95	8.03
11151805	6	7	12	8	5	12	6	6	14	8	8	10	11	8	7	6	5.88	6.19
11151810	8	5	12	6	6	14	8	8	10	11	8	7	9	6	4	7	6.65	6.87
11151815	6	6	14	8	8	10	11	8	7	9	6	4	4	7	9	4	6.62	4.02
11151820	8	8	10	11	8	7	9	6	4	4	7	9	3	4	7	6	5.78	5.82
11151825	11	8	7	9	6	4	4	7	9	3	4	7	7	6	17	7	6.01	6.90
11151830	9	6	4	4	7	9	3	4	7	7	6	17	13	7	12	9	5.61	8.84
11151835	4	7	9	3	4	7	7	6	17	13	7	12	11	9	10	8	5.35	7.93
11151840	3	4	7	7	6	17	13	7	12	11	9	10	5	8	9	5	6.45	5.03
11151845	7	6	17	13	7	12	11	9	10	5	8	9	2	5	11	11	6.78	10.24
11151850	13	7	12	11	9	10	5	8	9	2	5	11	14	11	7	10	6.97	10.15
11151855	11	9	10	5	8	9	2	5	11	14	11	7	6	10	12	6	6.96	6.39
MAPE(%)																	23.60	2.44
RMSE																	2.02	0.25
MAE																	1.67	0.18
ME																	4.22	0.76

[a] "11151725" denotes the 17:25 on 15 November 2012, and so on.
[b] "Target", "Upstream", and "Downstream" represent the traffic flow of the target road segment, its upstream road segment and its downstream road segment, respectively.
[c] "Actual value" and "Prediction value" are the actual traffic flow observed and the predicted traffic flow of corresponding methods on the target road segment at the next time interval, respectively.

methods are shown in Figs. 6(a) and 6(b), respectively. From these results, we can find that the prediction values of TFPC are the closest to the actual values.

Based on the aforementioned results, it could be concluded that correlation analysis used in TFPC is feasible, and especially discovering correlation information among

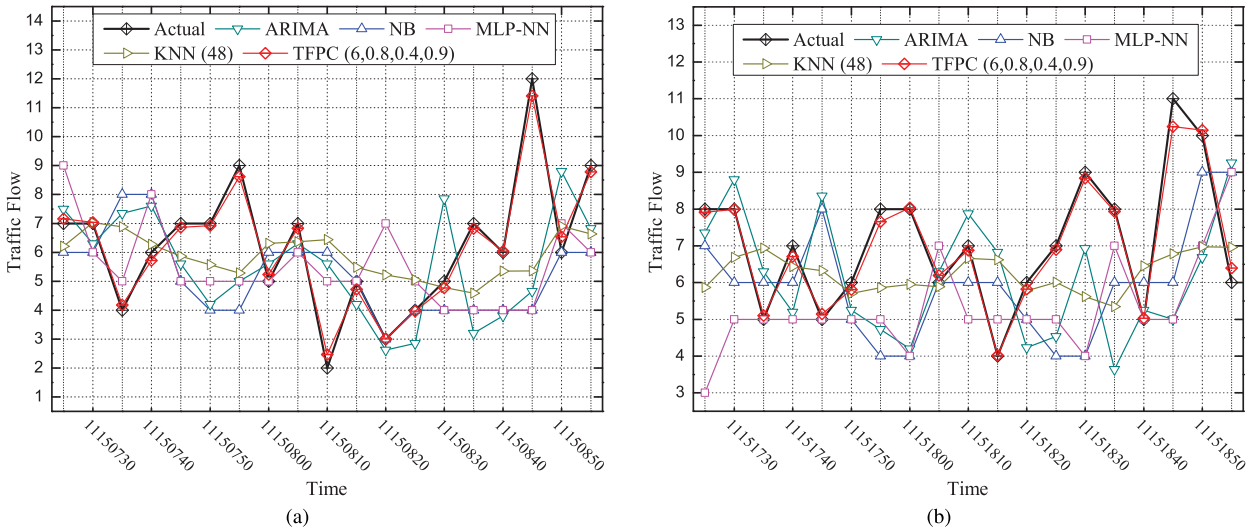


FIGURE 6. Prediction results of TFPC, KNN, MLP-NN, NB, and ARIMA. (a) Morning peak period and (b) evening peak period on 15 November 2012.

traffic flows and incorporating it into prediction are effective. Moreover, the results also demonstrate that the proposed ParKNNO classifier performs better than other methods by exploiting traffic flow correlations into the classification process.

D. SPEEDUP EVALUATION

To validate the speedup of ParKNNO, we perform respectively the experiments on a cluster of nodes ranging from 1 to 10 (i.e., the number of cores varying from 4 to 40), by holding six groups of data sets (128MB, 256MB, 512MB, 1GB, 2GB, and 4GB) constant, and then plot the results in Fig. 7(a).

From Fig. 7(a), we can observe that the speedup of ParKNNO increases relative linearly with the growth of the number of nodes, and particularly larger data sets obtains a better speedup. The speedup value reaches 8.863, which is 88.63% (8.863/10=88.63%) of the ideal speedup, when the size of the data set is 4GB on 10 nodes. In general, it is very difficult to achieve linear speedup due to the communication cost and the skew of the slaves [40], and especially the time to handle small-scale data sets is not dominant in comparison with the time consumed by communication and task arrangement. When the size of data sets is increased, the time of intensive computation becomes dominant. The results indicate that ParKNNO on MapReduce has a very good speedup performance over big data and performs better with larger data sets, which is nearly the same for data sets with very different sizes.

E. SCALEUP EVALUATION

For the scaleup evaluation of ParKNNO, we increase the number of nodes (varying from 1 node to 10 nodes) in direct proportion to the size of data sets (ranging from 128MB to 1.25GB, from 256MB to 2.5GB, and from 512MB to 5GB). The results are depicted in Fig. 7(b).

Figure 7(b) shows that all the scaleup values of ParKNNO are higher than 0.53, with the number of nodes and the size of data sets proportionally growing. Clearly, the ParKNNO classifier scales very well and has the adaptability of large-scale data sets under a MapReduce framework on a Hadoop platform.

F. SIZEUP EVALUATION

In order to measure the sizeup of ParKNNO, we carry out the experiments on a cluster with the fixed number of nodes (2, 4, 6, 8, and 10 respectively) by increasing the size of data sets varying from 1GB to 5GB, and then plot the results in Fig. 7(c).

From Fig. 7(c), we can find that the sizeup value of 2 nodes is approximately 2 while it is only about 1.5 for 8 nodes, when the size of data sets increasing from 1GB to 2GB. The reason is that the communication time of 2 nodes is shorter than that of 8 nodes. In particular, this communication time would not increase much in ParKNNO when the size of data sets is doubled. The results demonstrate that ParKNNO has a very good sizeup performance.

Based on the above-mentioned results, it could be found that ParKNNO under a MapReduce framework on a Hadoop platform offers nearly linear speedup, has excellent scaleup and sizeup, and especially achieves more efficiency than the corresponding sequential versions on a single machine with the same classification results.

Overall, the empirical study on real-world traffic flow big data proves the effectiveness of the proposed approach, and the experimental results indicate that TFPC outperforms four state-of-the-art prediction methods in accuracy and can significantly improve the scalability and efficiency of prediction. In other words, TFPC achieves superior performance and has a good robustness in addressing real-time TFP problem.

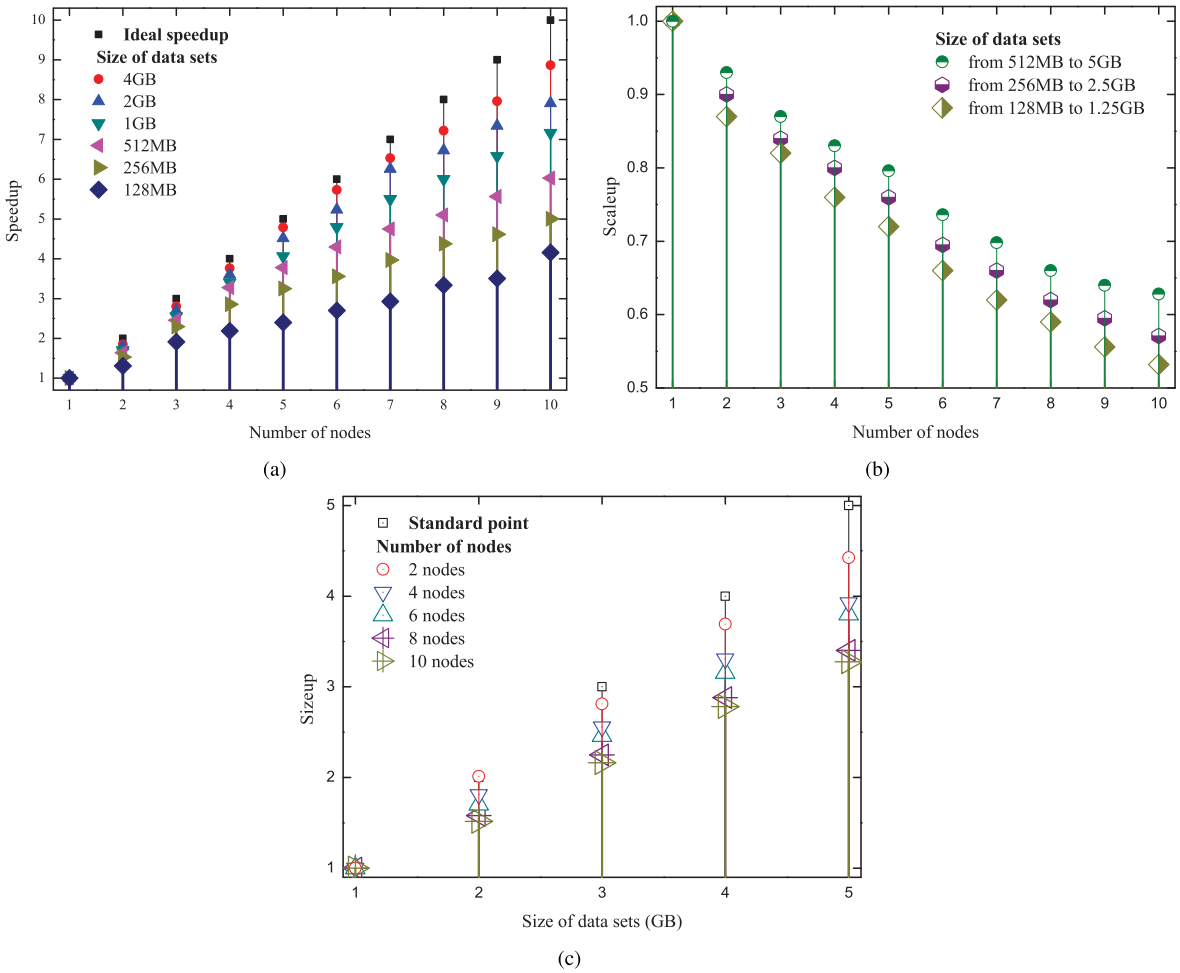


FIGURE 7. 3Ss of ParKNNO. (a) Speedup, (b) Scaleup, and (c) Sizeup.

V. CONCLUSION

In this paper, we focused on the real-time prediction with big traffic flow data and thus proposed a new MapReduce-based nearest neighbor approach for traffic flow prediction using correlation analysis (TFPC) on a Hadoop platform. To save memory consumption and reduce the computational costs of big calculations, TFPC was carried out in a real-time prediction system (RPS) composed of the ODT module and the OPP module. In particular, to enhance the robustness of real-time applications with very large training samples, a parallel k-nearest neighbor optimization classifier (ParKNNO) was built to model traffic flow correlations in ODT, and a novel prediction calculation method was put forward to generate traffic flow prediction in OPP. Furthermore, we evaluated the performance of TFPC on accuracy, speedup, scaleup and sizeup using the LOO-CV method by an empirical study. The results demonstrated that our approach was superior to other comparable methods in terms of accuracy which can be enhanced 90.07% in the best case, and significantly improved the efficiency and scalability of traffic flow prediction.

ACKNOWLEDGMENT

The authors would like to thank Datatang (Beijing) Technology Co., Ltd. for providing the experimental data.

REFERENCES

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[2] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[3] Q. Shi and M. Abdel-Aty, "Big Data applications in real-time traffic operation and safety monitoring and improvement on urban expressways," *Transp. Res. C, Emerg. Technol.*, vol. 58, pp. 380–394, Sep. 2015.

[4] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

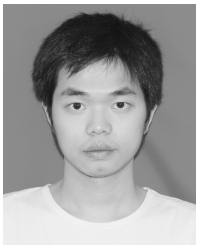
[5] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[6] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[7] Z. Zhou, W. Gaaloul, P. C. K. Hung, L. Shu, and W. Tan, "IEEE access special session editorial: Big data services and computational intelligence for industrial systems," *IEEE Access*, vol. 3, pp. 3085–3088, 2015.

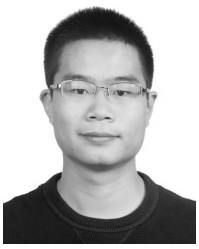
[8] Z. Zhao, W. Ding, J. Wang, and Y. Han, "A hybrid processing system for large-scale traffic sensor data," *IEEE Access*, vol. 3, pp. 2341–2351, 2015.
[9] E. I. Vlahogianni, B. B. Park, and J. W. C. van Lint, "Big data in transportation and traffic engineering," *Transp. Res. C, Emerg. Technol.*, vol. 58, p. 161, Sep. 2015.
[10] Y. Xia, L. Zhang, and Y. Liu, "Special issue on big data driven intelligent transportation systems," *Neurocomputing*, vol. 181, pp. 1–3, Mar. 2016.
[11] T. T. Tchrakian, B. Basu, and M. O'Mahony, "Real-time traffic flow forecasting using spectral analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 519–526, Jun. 2012.
[12] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: Overview of objectives and methods," *Transp. Rev.*, vol. 24, no. 5, pp. 533–557, Sep. 2004.
[13] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transp. Res. C, Emerg. Technol.*, vol. 10, no. 4, pp. 303–321, Aug. 2002.
[14] C. Chen, Y. Wang, L. Li, J. Hu, and Z. Zhang, "The retrieval of intra-day trend and its influence on traffic prediction," *Transp. Res. C, Emerg. Technol.*, vol. 22, pp. 103–118, Jun. 2012.
[15] M.-C. Tan, S. C. Wong, J.-M. Xu, Z.-R. Guan, and P. Zhang, "An aggregation approach to short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 60–69, Mar. 2009.
[16] Y. Zhang, Y. Zhang, and A. Haghani, "A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 65–78, Jun. 2014.
[17] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 3, pp. 387–399, Jun. 2011.
[18] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 644–654, Jun. 2012.
[19] F. Moretti, S. Pizzuti, S. Panzieri, and M. Annunziato, "Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling," *Neurocomputing*, vol. 167, pp. 3–7, Nov. 2015.
[20] L. Li, X. Su, Y. Wang, Y. Lin, Z. Li, and Y. Li, "Robust causal dependence mining in big data network and its application to traffic flow predictions," *Transp. Res. C, Emerg. Technol.*, vol. 58, pp. 292–307, Sep. 2015.
[21] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 4, pp. 606–616, Aug. 2011.
[22] Y. Zhang, "Special issue on short-term traffic flow forecasting," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 1–2, Jun. 2014.
[23] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *J. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 53–72, 2009.
[24] M. Schönhof and D. Helbing, "Empirical features of congested traffic states and their implications for traffic modeling," *Transp. Sci.*, vol. 41, no. 2, pp. 135–166, May 2007.
[25] P. Dell'Acqua, F. Bellotti, R. Berta, and A. De Gloria, "Time-aware multivariate nearest neighbor regression methods for traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3393–3402, Dec. 2015.
[26] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, Jan. 2013.
[27] H. A. Fayed and A. F. Atiya, "A novel template reduction approach for the k-nearest neighbor method," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 890–896, May 2009.
[28] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting," *Neurocomputing*, vol. 179, pp. 246–263, Feb. 2016.
[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York, NY, USA: Wiley, 2012.
[30] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. New York, NY, USA: Pearson Education Inc., 2006.
[31] Z. Zheng and D. Su, "Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 143–157, Jun. 2014.
[32] H. Wang, "Nearest neighbors by neighborhood counting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 942–953, Jun. 2006.
[33] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. 26th Symp. Mass Storage Syst. Technol. (MSST)*, Incline Village, NV, USA, 2010, pp. 1–10.
[34] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[35] T. White, *Hadoop: The Definitive Guide*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.
[36] P. Zikopoulos, C. Eaton, D. deRoos, T. Deutsch, and G. Lapis, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. New York, NY, USA: McGraw-Hill, 2011.
[37] H. Chen and H. A. Rakha, "Real-time travel time prediction using particle filtering with a non-explicit state-transition model," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 112–126, Jun. 2014.
[38] Q. Ye, W. Y. Szeto, and S. C. Wong, "Short-term traffic speed forecasting based on data recorded at irregular intervals," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1727–1737, Dec. 2012.
[39] C. D. Lewis, *Industrial and Business Forecasting Methods: A Practical Guide to Exponential Smoothing and Curve Fitting*. London, U.K.: Butterworth, 1982.
[40] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," *Data Min. Knowl. Discov.*, vol. 3, no. 3, pp. 263–290, Sep. 1999.
[41] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating MapReduce for multi-core and multiprocessor systems," in *Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Scottsdale, AZ, USA, Feb. 2007, pp. 13–24.
[42] C.-S. Li and M.-C. Chen, "A data mining based approach for travel time prediction in freeway with non-recurrent congestion," *Neurocomputing*, vol. 133, pp. 74–83, Jun. 2014.

DAWEN XIA is currently pursuing the Ph.D. degree with the School of Computer and Information Science and School of Software, Southwest University, Chongqing, China. He is an Associate Professor with the School of Information Engineering, Guizhou Minzu University, Guiyang, China. His research interests include big data analytics, multiagent systems, parallel and distributed computing, and spatio-temporal data mining.



HUAQING LI received the B.S. degree from the College of Mathematics and Physics, Chongqing University of Posts and Telecommunications, in 2009, and the Ph.D. degree from the College of Computer Science and Technology, Chongqing University, Chongqing, China, in 2013. He is an Associate Professor with the School of Electronic and Information Engineering, Southwest University, Chongqing, China. His research interests include consensus of multiagent systems, distributed optimization, and big data analytics.

BINFENG WANG is currently pursuing the M.Eng. degree with the School of Computer and Information Science and School of Software, Southwest University, Chongqing, China. His research interests include cloud computing, and big data analytics.



YANTAO LI received the Ph.D. degree from the College of Computer Science, Chongqing University, in 2012. He is an Associate Professor with the School of Computer and Information Science and School of Software, Southwest University, Chongqing, China. His research interests include wireless communication and networking, sensor networks and ubiquitous computing, and information security.

ZILI ZHANG received the B.Sc. degree from Sichuan University, the M.Eng. degree from Harbin Institute of Technology, and the Ph.D. degree from Deakin University, all in computing. He is a Professor with Southwest University, Chongqing, China, and a Senior Lecturer with Deakin University, Australia. He has authored or co-authored more than 130 refereed papers in international journals or conference proceedings and six monographs or textbooks. His research interests include multiagent systems, bio-inspired AI, and big data analytics.