Implementing Short-Term Traffic Flow Forecasting Based on Multipoint WPRA with MapReduce

Shuangshuang Li
State Key Laboratory of Management and Control for Complex Systems
Institute of Automation, Chinese Academy of Sciences
Beijing, China
shuangshuang.li@ia.ac.cn

Abstract—**The multipoint short-term traffic flow forecasting should deal with mass of historical traffic flow data from intersections in an area or urban. So the problem in runtime still remains to be the major obstacle for the practical and successful applications of the prediction algorithms. This problem becomes the key point to the evaluation of the data-driven methods especially for the nonparametric forecasting approach such as the weighted pattern recognition algorithm (WPRA). In order to solve this problem we use the MapReduce computing framework to implement the multipoint WPRA which is an improvement of the pattern recognition algorithm (PRA) based on the nonparametric regression method (NPR). By using MapReduce for the multipoint WPRA, the runtime successfully decreases, compared with using one computer.**

Index Terms—**Multipoint weighted pattern recognition algorithm, Short-term traffic flow forecasting, MapReduce**

I. Introduction

A variety of methods and techniques have been developed to the single point short-term traffic flow predicting, which has moved Intelligent Transportation Systems (ITS)[1][2][3][4] from passive to proactive control and management. So far, most of research attention is mainly put into the single point traffic flow forecasting while little in the multipoint traffic flow forecasting. For an urban traffic control system, it can obtain the predicted traffic flow of intersections in this urban by using a multipoint traffic flow forecasting method. According to those predicted traffic flow, the urban control system can give a global control to the urban traffic. In practical situations, the execution time of the multipoint traffic flow forecasting algorithm must meet the time allowed by the control system in a control period. If the execution time is so long and exceeds the allowed time in a control period of the urban control system, there will be a lag with the implementation of the control strategies.

Although most of prediction methods have alleviated the accuracy problem in traffic forecasting to some extent, the runtime problem is still not well solved. The run time problem becomes very obvious especially for the data-driven nonparametric forecasting approaches such as nonparametric regression (NPR). It is noted that the effectiveness of a data-driven nonparametric predicting method is directly dependent on the quality of the database. Clearly, it is desirable to possess a large database of cases that span the likely conditions that a system is expected to face [5]. However, while as large a database as possible is desirable for increasing the accuracy of the traffic flow forecasting method, the size of the database has significant implications on the timeliness of the method execution especially for the multipoint traffic flow predicting. B. L. Simith *et al.* [5] point that it is important to realize that the management of the database is a critically important issue for NPR. When NPR works, the majority of effort at runtime is expended in searching the database for neighbors. As the database grows, this search process grows accordingly. Taehyung Kim *et al.* [6] propose a pattern recognition algorithm (PRA) which is an improvement of NPR by regarding a sequence of "increasing", "equal" or "decreasing" of the traffic flows as a pattern and searching traffic flow patterns instead of traffic flow neighbors in database. Their experiments results show that PRA significantly (about 20% improvement) outperforms the *k*-nearest neighbor (*k*-NN) nonparametric regression model [7]. S. S. Li *et al.* [8] propose a weighted pattern recognition algorithm (WPRA) which is an improvement of PRA by weighting the traffic flow patterns according to the different clock time intervals. Compared with PRA, it is about 20% improvement is obtained by applying the results to the actual data and the simulation data.

The multipoint WPRA is defined that the predicted traffic flows of intersections in an urban or area are obtained by using WPRA. Based on that, the runtime of the multipoint WPRA is an urgent problem which needs to be solved. If the control system obtains the future traffic flows of intersections during the allowed time, the urban traffic control system can coordinate the intersections well by giving the control commands ahead of time. One simple idea to solve the runtime problem is to compute the predicted traffic flow in each intersection and then the urban traffic control system

This work was supported in part by NSFC 60921061, 61174172, 60904057, 31170670, 61101220, 70890084, 90920305 and CAS 2F11D03, 2F11N06, 2F11D01, 2F11N07.

collects the results together. There may be two problems for this idea. One is that the synchronization problem. The premise condition of sending coordination control instructions to intersections is after gathering all predicted traffic flow. The runtime of this process depends on the transmission time of the slowest result based on cask theory which shows that the cubage of a cask is dependent on the shortest wood plate. The other problem is that the traffic flow data packets may be lost during transmission process. In this paper, we use the MapReduce [9] for implementing the multipoint WPRA for short-term traffic flow forecasting.

The open-source project Hadoop[10] is the most well-celebrated MapReduce[9] framework, which processes vast amounts of data in-parallel on a computer cluster. MapReduce is a new distributed programming paradigm and easy to use for users, while offering load balancing and fault tolerance mechanisms. MapReduce is based on two higher order functions: Map and Reduce. The two functions can be designed by users and MapReduce allows the parallel computing under a cluster, grid and cloud computing environment. We notice that the industry's introduction of the concepts of Cloud Computing and MapReduce gives scientists very viable alternatives for their computational needs. MapReduce frameworks are a distributed data analysis framework model which is introduced by Google [9] and well-suited for the execution of a huge amount of data. It can well meet the demand of the time of the algorithm execution in the multipoint WPRA for short-term traffic flow forecasting.

The purpose of this study is to use MapReduce to implement the multipoint WPRA for short-term traffic flow forecasting in order to meet the time demand of the traffic control system. The structure of the paper is organized as follows. After introducing the time demand for multipoint WPRA for short-term traffic forecasting, we introduce the related work about *k*-NN, PRA and WPRA in Section II. In Section III we describe the multipoint WPRA with MapReduce. In Section IV we show the experimental results of the multipoint WPRA with MapReduce, and compare the runtime of multipoint WPRA using MapReduce with that using one computer. In Section 5 we conclude this paper and give a discussion on future work.

II. Related Work

A. The k-NN Nonparametric Regression Algorithm

Single point short-term traffic flow forecasting plays an important role in the Intelligent Transportation System (ITS). The *k*-nearest neighbor (*k*-NN) nonparametric regression method [5] is a classic data-driven method which is well suited for application to single point short-term traffic flow forecasting.

The *k*-NN nonparametric regression method predicts the future traffic flow by using the current traffic flow and the historical traffic flow. A traffic flow vector, *V*(*t*), is defined as

$$V(t) = \left[x(t), x(t-1), x(t-m), x_h(t), x_h(t+1)\right]^T \qquad (1)$$

where $x(t)$ is the volumes at current time interval t, $x(t-1)$is the volumes at the previous time interval (t-1), and so on. $x_h(t)$ is the historical average traffic flow during the current time interval of historical average.

We can describe *k*-NN nonparametric regression method by using the following function:

$$x(t+1) = f(V(t)) \qquad (2)$$

where $x(t+1)$ is the predicted traffic flow and $f(\cdot)$ a nonlinear function.

The traffic flow vectors which are similar with the current traffic flow vector are viewed as the nearest neighbors. The output $x_i(t+1)$ of the *i*-th nearest neighbor is can be obtained by

$$x_i(t+1) = \frac{x(t)}{x_h(t)} x_h(t+1) \qquad (3)$$

According to the outputs of the nearest neighbors, the most common approach for predicting the traffic flow is the straight average approach [5] which applies equal weigh to each of these outputs. So the future traffic flow is calculated by the following equation

$$x(t+1) = \frac{1}{k}\sum_{i=1}^{k} x_i(t+1) \qquad (4)$$

where *k* is the total number of the nearest neighbors.

B. The Pattern Recognition Algorithm

Taehyung Kim *et al.* [6] propose a pattern recognition algorithm to predict the short-term traffic flow. This algorithm is an improvement of the *k*-NN method. It supposes that the predicted traffic flow in not only depends on the current traffic flow and the historical traffic flow but also on the sequences of the traffic flow patterns [8]. The traffic flow pattern is a vector that shows the variation of the traffic flow during a selected time interval.

In order to structure a traffic flow pattern, we consider a vector of traffic flow

$$X = \left[x_1, x_2, \ldots, x_n\right]^T \qquad (5)$$

where x_i is the number of cars passing in a given period *i*.

The definition of the current traffic flow can be expressed as $[x_{n-m}, x_{n-m+1}, \cdots, x_{n-1}, x_n]$, where m is a pattern size such that 1≤ m ≤ n-1. A difference vector which is corresponding to X is defined as

$$S = \left[s_1, s_2, \ldots, s_{n-1}\right]^T \qquad (6)$$

where $s_i = x_{i+1} - x_i$ (1≤ i ≤n-1).

We map these differences onto variables d_i, by encoded each value of them as 2, 0 or 1.

$$d_i = \begin{cases} 2 & s_i < 0 \\ 0 & s_i = 0 \\ 1 & s_i > 0 \end{cases} \qquad (7)$$

The process of the pattern recognition algorithm is shown as the following Table 1.

Table 1
The pattern recognition algorithm [6]

Given the current state $p_d = [d_{n-m}, d_{n-m+1}, \dots, d_{n-1}]^T$ correlated with pattern size m, the corresponding matched pattern $p'_d = [d_{j-m}, d_{j-m+1}, \dots, d_{j-1}]^T$ and the difference vector $s'_d = [s_{j-m}, s_{j-m+1}, \dots, s_{j-1}]^T$ associated with pattern p'_d.

1. Start with a minimal neighborhood size k and a minimal pattern size m.
2. Search the pattern $[d_1, d_2, \dots, d_{n-m-1}]^T$ to find the nearest matches p'_d for p_d. The traffic flow difference of the l-th neighborhood is s^l_j.
3. Calculate the predicted traffic flow x_{n+1} on the basis of the final differences for all of the nearest neighbors

$$x_{n+1} = x_n + \sum_{l=1}^{k} s^l_j \Big/ k \tag{8}$$

where x_n is the traffic flow during the current time interval.

4. Calculate the root mean squared error (RMSE) between the actual and predicted values for these choices of pattern size and neighborhood size for the entire estimation set.
5. Repeat step 2 to 4 for pattern size m+1, m+2,…, $m_{\max}$.
6. Repeat step 1 to 5 for neighborhood size k+1, k+2,…, $k_{\max}$.
7. Choose the optimal pattern recognition model which can obtain the minimal RMSE by optimizing the neighborhood and pattern sizes.

C. The Weighted Pattern Recognition Algorithm

S. S. Li *et al.* [8] propose a weighted pattern recognition algorithm for short-term traffic forecasting. This algorithm is an improvement of PRA. It assumes that the corresponding traffic flow errors of the same patterns which belong to different clock time interval have different contribution to the predicted traffic flow. The corresponding errors of same patterns, whose clock time is similar to the current clock time, weigh larger than others.

The flow of weighted pattern recognition algorithm is shown in Table 2.

Table 2
The weighted pattern recognition algorithm [8]

Give a sequence of traffic flow $x_1, x_2, \dots, x_n$.

1. Start with a pattern size L. From the pattern size L describing the current traffic flow pattern, i.e., $p_d = [d_{n-L}, d_{n-L+1}, \dots, d_{n-1}]^T$.
2. Search the historical traffic flow pattern $[d_1, d_2, \dots, d_{n-L-1}]^T$ to find the same patterns for p_d. If the same patterns can be found in the historical traffic flow pattern, go to step 3. Otherwise, go to step 1 and start with another pattern size.
3. Find the traffic flow difference s^l_j at next clock time period of the l-th same pattern in a time interval.
4. Calculate the value x_{n+1} on the basis of the differences for all of the same patterns which belong to different time interval:

$$x_{n+1} = x_n + \sum_{i=1}^{m} \omega_i c_i \Big/ \sum_{i=1}^{m} \omega_i \quad \text{where} \quad c_i = \sum_{l=1}^{k} s^l_j \Big/ k \tag{9}$$

where c_i is the average difference in the i-th time interval. m is the total number of the time intervals. k is the total number of the same patterns the i-th time interval.

As stated earlier, the accuracy of the WPRA is better than k-NN and PRA according to the previous literatures [6][7][8]. In this paper, our contribution is to reduce the running time of the multipoint WPRA.

III. MapReduce for Data-driven Methods

A. The runtime demand of an urban control system

The time for the traffic flow forecasting algorithm execution is very important for a traffic control system. Suppose that the control cycle is T. In general situation, T consists of four parts: the sampling period T_{sp}, the computing time T_{cp}, the telecommunication time T_{tm} and the time of the actions with site actuators T_{ad}.

$$T = T_{sp} + T_{cp} + T_{tm} + T_{ad} \tag{10}$$

When considering how the data-driven methods work, one will see that the majority of effort at runtime is expended in searching the database for neighbors. As the database grows, this search process grows accordingly. For real-time traffic flow forecasting, it must guarantee that T_{cp} be less than T. If the traffic control system is capable of achieving it, there will be little lag between the collection of traffic data and implementation of traffic control strategies.

B. MapReduce

In order to meet the time demand for the multipoint WPRA, we use MapReduce programming model. MapReduce is a distributed computing model and is suitable for processing and generating large data sets. It helps programmers run their own programs on the distributed system even though they do not know the parallel and distributed programming in details. The computing process of MapReduce is as follows. Firstly, it divides a huge data set into many small data sets. Secondly, every split data set is handled by one node or a cluster for generating intermediate results. Thirdly, these intermediate results are handled by several nodes for combining final results, as shown in Figure 1.

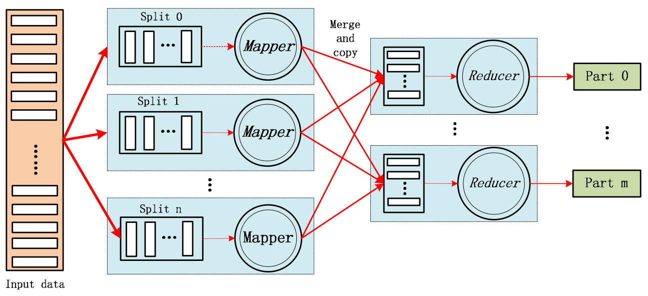

Fig.1 MapReduce data flow with multiple map and reduce tasks

The basic components of an application on MapReduce include one or more Mappers and Reducers class, as well as a program to create a JobConf. The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types, as shown in Table 3.

Table 3
MapReduce programming paradigm

	Input	Output
Map	(key1,value1)	(key2,value2)
Reduce	(key2,list<value2>)	(key3,value3)

C. MapReduce for multipoint WPRA method

In order to meet the time demand for the multipoint WPRA, we use MapReduce programming model to implement the multipoint WPRA. When considering programming this algorithm, we change the Integer type of the traffic flow patterns into String type of traffic flow patterns. So we can use the string searching algorithms to find the matched traffic flow string in the historical traffic flow string. It can reduce the execution time of the multipoint WPRA.

In the following part, we are about to go into the details of multipoint WPRA based on MapReduce. There are six steps for multipoint WPRA. There are one Mapper, one Joiner Reducer and four Reducers with the implementation of multipoint WPRA using MapReduce, as shown in the following steps.

- **Step 1** Divide the historical traffic data of every intersection into several partitions according to the time intervals. For example, we can divide daily traffic flow into four time intervals: A (5:30, 9:30), B (9:30, 15:30), C (15:30, 18:30), D (18:30, 5:30), as shown in [8]. A is the morning peak. B and D are the smooth. C is the evening peak. We program this process corresponding to Mapper 1, as shown in Table 4. The format of input traffic data is shown is Table 5.

Table 4
Mapper 1

Input: the historical traffic flow data or the current traffic flow data
Output: (IntersectionID + Time interval flag, Traffic flow)

```
1: for every traffic flow F
2:     If(Time belongs to time interval M)
3:     Set the time interval flag M for F;
4: end for;
```

Table 5
The format of input traffic flow data

IntersectionID	Time	Traffic flow

- **Step 2** Compute the traffic flow pattern vector for every intersection according to the Equation 6 and 7. Unlike [6][8], we set the variable d_i in Equation 6 is equal to 2, 0 and 1 to indicate the "decreasing", "equal" and "increasing" of the traffic flows. Based on this, it is more convenient to convert numerical processing to character processing. The input of this step is dependent on Mapper1, as shown in Table 6, where T_c is the sampling period, $x(k{\cdot}T_c)$ is traffic flow value at the k^{th} sampling period and the traffic flow of its next period is $x(k{\cdot}T_c+1)$ (k=1,2,3…). This step is programmed corresponding to Reducer 1, as shown in Table 7.

Table 6
The format of the element for Reducer 1

IntersectionID	Time interval flag	Traffic flow $x(k{\cdot}T_c)$	Traffic flow $x(k{\cdot}T_c+1)$

Table 7
Reducer 1

Input: The improvement of the output of Mapper 1
Output: (IntersectionID + Time interval flag, Traffic flow pattern vector)

```
1: for $i$= 1 to $n$ in each intersection
2:     $d(i)$=sign($x(k{\cdot}T_c+1)-x(k{\cdot}T_c)$);        // According to Equation 7.
3: end for;
```

- **Step 3** Calculate the traffic flow error vector for every intersection based on Equation 6. This step is programmed corresponding to Reducer 2, as shown in Table 8. The input of this step is the same to that of step 2.

Table 8
Reducer 2

Input: The improvement of the output of Mapper 1
Output: (IntersectionID + Time interval flag, Traffic flow error vector)

```
1: for $i$= 1 to $n$ in each intersection
2:     $s(i)=x(k{\cdot}T_c+1)-x(k{\cdot}T_c)$;        // According to Equation 6.
3: end for;
```

- **Step 4** Join the traffic flow pattern vector and traffic flow error vector together. As stated earlier, the WPRA must find the same current traffic flow patterns in historical traffic flow patterns. Based on this, the predicted traffic flow can be calculated by the errors corresponding to those traffic flow patterns. It will be more convenient to find the corresponding traffic flow errors by the join. This step is programmed corresponding to Reducer 3, as shown in Table 9.

Table 9
Reducer 3

Input: the output of Reducer 1 and Reducer 2
Output: (IntersectionID + Time interval flag, Traffic flow pattern vector + Traffic flow error vector)

```
1: for each intersection
2:     Join the traffic flow pattern vector and traffic flow error vector;
       //According to the same (IntersectionID + Time interval flag)
3: end for;
```

- **Step 5** Find the traffic flow patterns which are the same as the current traffic flow pattern in historical traffic flow patterns and save the errors corresponding to that traffic flow patterns. We program this step in Reducer 4, as shown in Table 10.

Table 10
Reducer 4

Input: the output of Reducer 3
Output: (IntersectionID + Time interval flag, The corresponding traffic flow errors)

```
1: for each intersection
2:     Add the traffic flow pattern in a List<String>;
3:     Find the same traffic flow pattern strings to the current traffic flow
       pattern string based on the character matching algorithm;
4:     Save the corresponding traffic flow errors in a list<Integer>;
5: end for;
```

- **Step 6** Calculate the predicted traffic flow of each intersection by the corresponding traffic flow errors in step 5. What's more, the traffic flow error must multiply the corresponding weights according to the time flag based on Equation 8 in the calculation of the final results. This step is programmed corresponding to Reducer 5, as shown in Table 11.

Table 11
Reducer 5

Input: the output of Reducer 5
Output: (IntersectionID, The predicted traffic flow)

```
1: for each intersection
2:     Multiply the corresponding weight for every matched traffic flow error
       according to the time flag;
3:     Sum up weighted traffic flow error and the traffic flow at current
       sampling period;
4: end for;
```

IV. EXPERIMENTS

A. *The traffic flow data set*

In order to evaluate the execution time of the multipoint WPRA with MapReduce, it is necessary to prepare the traffic flow data. The traffic data were collected at a site monitored by the Center for Intelligent Control and Systems Engineering (CICSE) of Institute of Automation Chinese Academy of Sciences (CASIA). The site chosen within CICSE is located on Suzhou City, Jiangsu Province, China. This road is called Liukun Line. There are more than twenty intersections in this line. We select the twenty intersections in Liukun Line. Every intersection has four data collection points and the data size of one point is nearly 50KB. On the analogy of this, for a year the traffic data for thirty intersections is about 2.2GB.

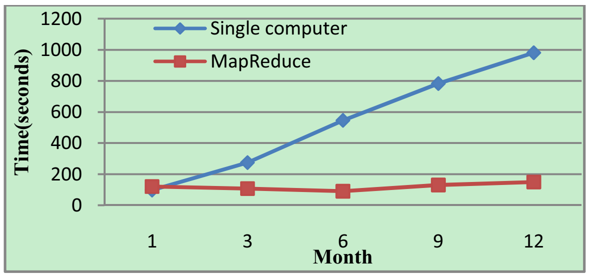

Fig. 2 The runtime of multipoint WPRA with MapReduce and one computer

B. *Result of the multipoint WPRA with MapReduce*

We compare the runtime of the multipoint WPRA which uses MapReduce with which uses one computer, as shown in Figure 2. The computer cluster for MapReduce consists of five computers which are the standard layout of DELL Optiplex 760. The runtime includes the time of accessing and storing data in database. It is concluded that: when we use traffic data in a month for multipoint WPRA, the runtime of the MapReduce is 121 seconds which is longer than when using one computer. But when the traffic data sets grow, the runtime of the single computer grows accordingly which is longer than that of the MapReduce. If a traffic control system's control cycle is 10 minutes, considering thirty intersections in Liukun Line, the multipoint WPRA with one computer couldn't meet the demand of the time of the control system when the traffic data sets is about three months. The reason is that the sampling period is 5 minutes, so the computing time must be less than 5 minutes according to the Equation 10.

Although the result shows that the multipoint WPRA using MapReduce can meet the time demand of the control system, the runtime for MapReduce increases from 6 months to 12months. That is to say, MapRedcuce can't meet the time demand when the data sets are big enough or more intersections are considered. This is because the number of the computer cluster is five. One approach would be to increase the size of the computer size. While examining this issue is beyond the scope of this paper, it is important to realize that runtime of multipoint WPRA is a critically important issue, particularly in real-time applications of it.

V. CONCLUSIONS AND FUTURE STUDIES

In this paper, we take use of the MapReduce to implement the multipoint WPRA for the short-term traffic flow forecasting by considering the real-time demand of the urban control system. And good result is obtained.

There is one thing we should do in the future research. We just use the multipoint WPRA to calculate the predicted traffic flow for several intersections at the same. What's more, we only focus on the runtime of the multipoint WPRA. For one intersection, the traffic flow information from its adjacent intersection is not be used for prediction. This information may be very useful to improve the performance of multipoint WPRA.

ACKNOWLEDGMENT

We would like to thank Prof. Fei-Yue Wang for his guidance and encouragement.

REFERENCES

[1] F.-Y. Wang, "Parallel Control and Management for Intelligent Transportation System: Concepts, Architectures, and Applications," *IEEE Transactions on Intelligent Transportation Systems*, September, 2010, vol.11, no.3, pp.630-638.
[2] F. H. Zhu, Z.J. Li, "A Case Study for Traffic Signal Control System Evaluation Based on Artificial Transportation Systems," *IEEE International Conference on Service Operations, Logistics, and Informatics*, July, Beijing, 2011, pp.347-352.
[3] K. Wang, Z. Shen, "Artificial Societies and GPU-Based Cloud Computing for Intelligent Transportation Management," *IEEE Intelligent Systems*, 2011, vol.26, no.4, pp.22-28.
[4] Z. Shen, K. Wang, F. H. Zhu, "Agent-based Traffic Simulation and Traffic Signal Timing Optimization with GPU," *14th International IEEE Conference on Intelligent Transportation Systems (ITSC 11)* Washington D. C., 2011, pp.145-150.
[5] B. L. Smith, B. M. Williams and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, August 2002,vol.10, no.4, pp. 303-321.
[6] T. Kim, H. Kim and D. J. Lovell, "Traffic Flow Forecasting: Overcoming Memoryless Property in Nearest Neighbor Non-Parametric Regression," *Proceeding of the 8th International IEEE Conference on Intelligent Transportation System*, Vienna, Austria, September 13-16, 2005, pp. 965-969.
[7] G. A. Davis, N. L. Nihan, "Nonparametric Regression and Short-Term Freeway Traffic Forecasting," *The American Society of Civil Engineers, Journal of Transportation Engineering*, 1991,vol.117,no.2, pp.178-188.
[8] S. S. Li, Z. Shen, F.-Y. Wang,"A Weighted Pattern Recognition Algorithm for Short-Term Traffic Flow Forecasting," *9th IEEE International Conference on Networking, Sensing and Control*, Accepted.
[9] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun.ACM*, 2008.vol. 51, no. 1, pp. 107-113.
[10] Apache, "Hadoop", http://hadoop.apache.org/common/,2006