Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 926251, 8 pages
http://dx.doi.org/10.1155/2014/926251

Research Article

Traffic Flow Prediction Model for Large-Scale Road Network Based on Cloud Computing

Zhaosheng Yang,[1,2,3] Duo Mei,[2] Qingfang Yang,[1,2,3] Huxing Zhou,[1,2,3] and Xiaowen Li[2]

[1] *State Key Laboratory of Automobile Simulation and Control, College of Transportation, Jilin University, Changchun 130025, China*
[2] *College of Transportation, Jilin University, Changchun 130025, China*
[3] *Jilin Province Key Laboratory of Road Traffic, College of Transportation, Jilin University, Changchun 130025, China*

Correspondence should be addressed to Duo Mei; meiduo059@163.com

Received 10 June 2014; Accepted 21 July 2014; Published 12 August 2014

Academic Editor: Huimin Niu

To increase the efficiency and precision of large-scale road network traffic flow prediction, a genetic algorithm-support vector machine (GA-SVM) model based on cloud computing is proposed in this paper, which is based on the analysis of the characteristics and defects of genetic algorithm and support vector machine. In cloud computing environment, firstly, SVM parameters are optimized by the parallel genetic algorithm, and then this optimized parallel SVM model is used to predict traffic flow. On the basis of the traffic flow data of Haizhu District in Guangzhou City, the proposed model was verified and compared with the serial GA-SVM model and parallel GA-SVM model based on MPI (message passing interface). The results demonstrate that the parallel GA-SVM model based on cloud computing has higher prediction accuracy, shorter running time, and higher speedup.

1. Introduction

Large-scale road network has high complexity, strong nonlinear, and high dynamic. The mass of traffic flow data brings enormous difficulties to traffic flow prediction. The traditional serial traffic flow prediction methods cannot meet the needs of real-time and accuracy. In order to solve this problem, the experts and scholars at home and abroad have been dedicated to the study on parallel traffic flow prediction methods and have achieved some research results. For example, Li et al. proposed a parallel traffic flow prediction method of space-time two-dimensional integration based on SVM, but this method is more suitable for emergency cases, not very practical for the normal traffic condition [1]. Deng realized parallel neural network method based on dish network by Charm++ programming model and applied to traffic flow prediction. When the number of parallel nodes is 110, the running time of two thousand links was 520.88 s [2]. Wang et al. implemented a parallel generalized neural network method for traffic flow prediction based on MPI (message passing interface) programming model. The experimental results showed that the speed of the proposed method was more than two times as fast as the serial method [3]. Wang proposed a parallel traffic flow prediction method based on SVM, and the experimental results showed that the result of parallel SVM method is better than parallel BP neural network method, and when the number of parallel nodes is 100, the running time of two thousand links was 36.48 s [4]. Zhang presented a short-term traffic flow prediction method based on genetic neural network in cloud computing environment. Its running efficiency was more than fourteen times as fast as the serial genetic neural network method [5].

To some extent, the above research results can solve the problem of large-scale road network traffic flow prediction, but there are some limitations, such as huge resource consumption and long running time. A large number of research results show that SVM is widely used in traffic flow forecasting and has certain advantages [1, 4, 6]. However, SVM still has some shortcomings; for example, it needs large store space and longer training time when it deals with large amounts of traffic flow data, so people have developed parallel SVM to reduce the computing cost and improve the running efficiency.

2. Support Vector Machine

SVM is developed based on the statistical theory. The numerous research results show that it has advantages in solving the problem of nonlinear, high dimension, and local minimum points, which becomes a research hot spot [6]. Traffic flow prediction is a kind of nonlinear regression problem, so SVM is widely used in the field. The idea of solving this problem is as follows.

Known training set $T = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_l, y_l)\}$, $x_i \in R^n$, is the factors which impact traffic prediction, $y_i \in R$ is predictive value of traffic flow, $i = 1, \ldots,$ and l is the number of training samples. The traffic flow has the inevitable connection to the traffic flow in several periods before, so x_i is the traffic flow in several periods before. A nonlinear function $\phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_N(x)]^T$ is introduced to map the training data from the lower dimensional feature space to high-dimensional feature space. Then we build linear decision function to make the original nonlinear problem into a linear problem in the high dimensional feature space:

$$f(x) = \sum_{m=1}^{l} w_m \phi_m(x) + b. \tag{1}$$

An insensitive loss function

$$L(x, y, f(x)) = \begin{cases} 0, & |f(x) - y| \le \varepsilon \\ |f(x) - y| - \varepsilon, & \text{others} \end{cases} \tag{2}$$

is introduced, where ε is insensitive loss factor. And then $f(x)$ is brought in to minimize $C\sum_{i=1}^{l} L(x, y, f(x)) + (1/2)\|w\|^2$, where C is punishment factor and $w = [w_1, w_2, \ldots, w_N]^T$ is linear weight vector.

The slack variables ξ_i and ξ_i^* are introduced; then the type is rewritten as follows:

$$\begin{aligned} \min \quad & \left\{ C\sum_{i=1}^{l} L(x, y, f(x)) + \frac{1}{2}\|w\|^2 \right\} \\ \text{s.t.} \quad & y_i - \left[\sum_{m=1}^{N} w_m \phi_m(x) + b\right] \le \varepsilon + \xi_i \\ & \left[\sum_{m=1}^{N} w_m \phi_m(x) + b\right] - y_i \le \varepsilon + \xi_i^* \\ & \xi_i \ge 0, \quad \xi_i^* \ge 0, \quad i = 1, 2, \ldots, l. \end{aligned} \tag{3}$$

The above question is a question of quadratic programming with inequality constraints. Then a kernel function $K(x_i, x) = \phi(x_i)^T\phi(x_j)$ is introduced. We select the radial basis kernel function. And then the method of Lagrange multiplier is used to get the following formula:

$$\begin{aligned} \min \quad & \left\{ \frac{1}{2}\sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j) \right. \\ & \left. - \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) y_i + \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) \varepsilon \right\} \\ \text{s.t.} \quad & \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0, \\ & 0 \le \alpha_i, \quad \alpha_i^* \le C, \quad i = 1, 2, \ldots, l, \end{aligned} \tag{4}$$

where α_i and α_i^* are the Lagrange multiplier. And then the above problem is solved to rewrite the regression function (1) as follows:

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(x_i, x) + b, \tag{5}$$

where $K(x_i, x) = e^{-\|x_i - x\|/\sigma^2}$, $\sigma > 0$.

Thus it can be seen that the SVM parameters C, insensitive loss factor ε, and the kernel function parameter σ have a greater influence on the calculation results, so we use parallel genetic algorithm based on cloud computing to optimize them.

3. Parallel Genetic Optimization SVM

Genetic algorithm (GA) has two shortcomings. First is easy to premature convergence, falling in a local optimum; the second is more time consuming in the selection, crossover and mutation steps, resulting in low efficiency. Considering the parallelism of genetic algorithm, the parallel GA is arisen at the historic moment. In this paper, Hadoop is used to implement the parallel genetic algorithm, which can avoid local convergence of genetic algorithm and improve the efficiency of genetic algorithm. The optimization of SVM based on the parallel genetic algorithm is a restricted area search problem. The three parameters are limited within a certain area based on the characteristics of traffic flow. The following research contents are several key problems of parallel genetic optimization SVM based on cloud computing.

3.1. Chromosome Coding. Chromosome coding is convenient to calculate and improve the computing speed of the optimal solution. It is the process of translating the form of the problem to solve into the string form encoded, which can be identified by genetic algorithm. The binary coding method is selected to encode the parameters of SVM. The desirable coding range of C, ε, and σ is [0.1, 150], [0.01, 0.5], and [0.01, 10] in traffic flow prediction. Because 150 is between $2^7 \sim 2^8$, the coding of the parameters needs 8-bit binary; encoding length is determined by the time.

Decoding is the process of translating the form of encoded string into the form of the solution. Decoded by the following formula:

$$X_j = \sum_{j=4k-3}^{4k} \left(x_j \cdot 2^{4k-j} \right), \tag{6}$$

where X_j is the parameter and x_j is the j-bit of binary coding for the parameters, $x_j = 0$ or 1.

3.2. Fitness Function. Define a fitness function to guide the evolution of next generation and obtain the optimal solutions to the problem. The precise fitness function can improve the quality of reconciliation and the speed of the algorithm. Because the purpose of the SVM parameter optimization problem is to find the optimal parameters, the average relative error is chosen for fitness evaluation.

3.3. Genetic Manipulation

3.3.1. Individual Choice. The purpose of individual choice is passing on excellent individual whose fitness value is higher than the next generation by copying. It can make excellent individual evolving continually. The roulette wheel selection method is chosen in this paper. The probability of the individual whose fitness value is $G(i)$ is selected as follows:

$$P_1(i) = \frac{G(i)}{\sum_{i=1}^{N} G(i)}. \tag{7}$$

3.3.2. Crossover and Mutation. Crossover and mutation are the keys to affect the behavior of GA. Crossover operation is to reserve excellent genes of these parent individuals as far as possible and form a new individual. The aim of mutation is to avoid the algorithm trapping in local optimal solution and keep the diversity of population. Because the variation in nature is in order to adapt to the environment, the adaptive adjustment functions of crossover rate and mutation rate are introduced. In this case, the crossover rate and mutation rate are adjusted constantly to maintain the population diversity, so as to avoid GA into premature convergence. Probability functions are as follows [7]:

$$\begin{gathered} P_2(i) = \begin{cases} \dfrac{K_1 N \left(g_{\max} - g' \right)}{\left(g_{\max} - \widehat{g} \right)}, & g' \geq \widehat{g}, \\ K_2, & g' < \widehat{g}, \end{cases} \\ P_3(i) = \begin{cases} \dfrac{K_1 N \left(g_{\max} - g_i \right)}{\left(g_{\max} - \widehat{g} \right)}, & g_i \geq \widehat{g}, \\ K_2, & g_i < \widehat{g}, \end{cases} \end{gathered} \tag{8}$$

where $g_{\max}$ is the maximum fitness value in current generation, $\widehat{g}$ is the average fitness value in current generation, g_i is the fitness value of the individual i in the current generation, g' is the bigger fitness value in two crossover individuals in the current generation, N is the length of the chromosome, $(g_{\max} - g')/(g_{\max} - \widehat{g})$ is the degree of the advantages and disadvantages whose fitness value is bigger of two crossover individuals in the current generation, $(g_{\max} - g_i)/(g_{\max} - \widehat{g})$ is the degree of the advantages and disadvantages of the individual i in the current generation, and K_1 and K_2 are the adjustment coefficient.

4. The Genetic SVM Based on Cloud Computing

MapReduce programming model is a cloud computing programming mode of Google, whose parallelism, fault tolerance and data distribution, load balancing, and so forth are implemented by the system, which is very suitable for processing and generating large data sets [8, 9]. Meanwhile, MapReduce has the advantages that MPI and any other programming models do not have, such as the function of balanced load and elastic computing and the ability to reduce bandwidth consumption and read latency, which can further improve the running efficiency of GA-SVM [10–12].

The parallel computing process is abstracted to the two functions in MapReduce: map( ) and reduce( ). They are as shown in Table 1 [13, 14]. The data processing of MapReduce is demonstrated by Figure 1 [15]. In the map( ), the original data is set into M fragments, which is composed of countless key/value pairs $\langle k1, v1\rangle$. And then, they are input to map( ) for processing. A group of intermediate key/value pairs $\langle k2, v2\rangle$ are output. After key/value pairs $\langle k2, v2\rangle$ are integrated and sorted by system, the key/value pairs which have the same $k2$ are output. And then they are decomposed into R fragments to input to reduce( ). Finally, the key/value pairs $\langle k3, v3\rangle$ needed are output.

In cloud computing environment, the genetic optimization SVM requires three steps: preparing of the training sample data, training of SVM, and traffic flow forecasting based on SVM trained. Specific steps are as follows.

4.1. Preparation of the Training Sample Data. In order to improve the running speed of the algorithm, first of all is the pretreatment of the collected traffic flow data to generate the data set. The normalized processing by the following formula:

$$\begin{aligned} f(x) &: x \longrightarrow y, \quad y \in [-1, 1], \\ y &= \frac{\left(y_{\max} - y_{\min} \right) \left(x - x_{\min} \right)}{\left(x_{\max} - x_{\min} \right)} + y_{\min} \\ &= \frac{2 \left(x - x_{\min} \right)}{\left(x_{\max} - x_{\min} \right)} + y_{\min}, \end{aligned} \tag{9}$$

where x is the collected data and y is the mapped data.

4.2. Training of SVM Based on Parallel Genetic Algorithm. SVM is trained by genetic algorithm based on MapReduce. Its process is a problem of quadratic programming. The specific steps are as follows [16, 17].

(1) Generate an initial parameter population. Generate an initial population of SVM parameters randomly. The populations are encoded and uploaded to Hadoop as a local file.

Table 1: Map and Reduce functions.

Function	Input	Output	Instructions
Map	$\langle k1, v1\rangle$	list $(k2, v2)$	(1) Parse data into key/value pairs, input to map( ). (2) Input $\langle k1, v1\rangle$, output intermediate result $\langle k2, v2\rangle$.
Reduce	$\langle k2, \text{list}(v2)\rangle$	$\langle k3, v3\rangle$	Input $\langle k2, \text{list}(v2)\rangle$, list $(v2)$ stands for the value, which belongs to the same $k2$.

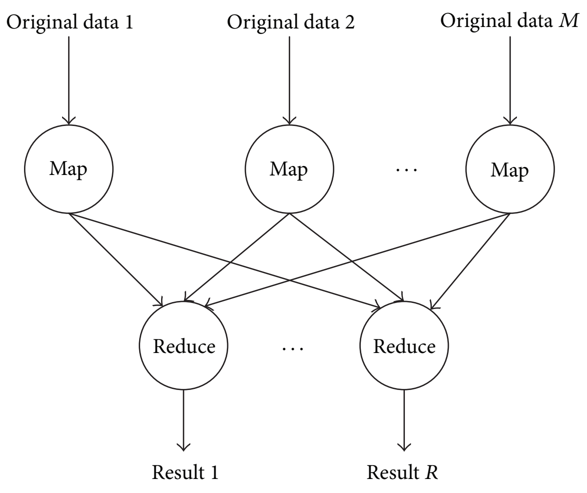


Figure 1: The data processing of MapReduce.

(2) *Initial population.* The initialization of population is completed by the master machine (Job Tracker). All individuals are divided into multiple child populations. Set up the parameters of genetic algorithm. And then, assign the parameters and the populations to the slave machine (Task Tracker).

(3) *Fitness evaluation.* Call map( ). Define the child population number as key. Define the chromosomes as value. The fitness evaluation is carried through for every population by Task Tracker to get the fitness value of each individual. The values of key/value pairs which have the same key are reduced and stored in the local HDFS file system.

(4) *Select operation.* Call reduce( ). Job Tracker reads the position of the intermediate file, and messages to reduce( ). The reduce( ) reads intermediate file from a Data Node after receiving instructions. And then the reduce( ) completes selection operation of the child population. Each child population selects two individuals.

(5) *Crossover and mutation operation.* Crossover operation is performed on the two individuals selected from the child population through the method of inserting genes. And then produce two new individuals. Perform mutation operation using the method of adaptive mutation to produce new individuals, which make up offspring populations. They are read in HDFS file system in terms of key/value pairs.

(6) *Termination conditions.* Job Tracker judges whether the evolutionary generations approach the optimum. If true, the algorithm is terminated. Job Tracker consolidates the results, outputs the optimal SVM parameters; if not, turn to step (7).

(7) Update the evolutionary generations; turn to step (3).

4.3. Parallel Genetic SVM Prediction Algorithm. In this paper, the process of the traffic flow prediction algorithm based on GA-SVM in cloud computing environment is as follows [18].

(1) The traffic flow sample data collected of large-scale road network is preprocessed. Part of it is used as the training sample, and then the other part is used as the forecasting sample. They are uploaded to the Hadoop.

(2) Job Tracker divides training sample and forecasting sample automatically. And then it reads SVM parameters and assigns them to Task Trackers together with the sample data. At this point, each Task Tracker has a small training sample.

(3) Task Trackers call map( ). Each small training sample is trained to output prediction results.

(4) The prediction results of each training sample are sorted by Job Tracker. And then, Job Trackers call reduce( ). At last, the prediction data tables of the entire road network are output. The performed process of MapReduce is over. The whole algorithm is terminated.

To sum up, the idea of "divide and rule" is adopted in parallel prediction model based on MapReduce. The sample data is divided into the child populations. GA algorithm is realized for child population, respectively, through map( ) and reduce( ). Training SVM only needs the shorter time. Parallel traffic flow prediction is realized using SVM trained to reduce the running time of the algorithm.

4.4. Evaluation Indices. In this paper, choose the relative error (RE), mean relative error (MRE), maximum relative error (MAXRE), and root-mean-square error (RMSE) as the evaluation index of prediction accuracy. Running time and the speedup (Sn) are chosen as the efficiency evaluation index. The related evaluation index expression is as follows:

$$\begin{gathered}
\mathrm{RE} = \frac{\left|\hat{y}(t) - y(t)\right|}{y(t)} \cdot 100\%, \\
\mathrm{MRE} = \frac{1}{n} \sum \frac{\left|\hat{y}(t) - y(t)\right|}{y(t)}, \\
\mathrm{MAXRE} = \max \frac{\left|\hat{y}(t) - y(t)\right|}{y(t)} \cdot 100\%, \qquad (10) \\
\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum \left[\frac{\hat{y}(t) - y(t)}{y(t)} \right]^2}, \\
S_n = \frac{T_s}{T_p},
\end{gathered}$$

where $y(t)$ is the actual value, $\hat{y}(t)$ is the average prediction value, n is the number of prediction, T_s is the running time of the serial algorithm, and T_p is the running time of the parallel algorithm.

5. Example and the Result Analysis

5.1. Design of Experiment. Parallel traffic flow prediction program for the large-scale road network is developed by Java language, Hadoop, GA algorithm, and SVM. The parallel computing experiment platform is set up by 20 PCs. The experiments are carried out to test the proposed algorithm based on the real-time data of Haizhu District in Guangzhou City. Software environment is Redhat Enterprise Linux 5.0 virtual machine, Hadoop0.17.1, JRE1.5, and JAVA. Hardware environment is 20 PCs. Among them, one PC is as the master nodes and also as a slave node, the rest of 19 PCs are only as slave nodes. Hadoop0.17.1 and JRE1.5 are installed on the master node and configured to the slave nodes through the SCP command.

Haizhu District of Guangzhou City contains 3,174 nodes and 8,914 links, whose network diagram is shown in Figure 2. Haizhu District is from Keyun Road in the east to Binjiang Road in the west and from Yuejiang Road in the north to Nanzhou Road in the south. Among them, one thousand links are chosen for traffic flow prediction. The data is from the SCATS traffic information collection system. A group of data is generated every five minutes. Acquisition times are 7 am to 7 pm. With $4 * 144 = 576$ groups of data from Monday to Thursday as the training samples, the traffic flow data on Friday is predicted. The 144 groups of data on Friday are as the actual value to compare with the prediction value.

The number of parallel nodes is 1, 2, 4, 8, 16, and 20. The one thousand links are predicted by the serial GA-SVM algorithm, the parallel GA-SVM algorithm based on MPI, and the parallel GA-SVM algorithm based on MapReduce. The basic idea of parallel GA-SVM algorithm based on MPI is "divide and conquer." Firstly, SVM is trained by parallel GA algorithm, and then the SVM trained is used to traffic flow prediction [19–21]. The performance of three algorithms is compared through a numerical example.

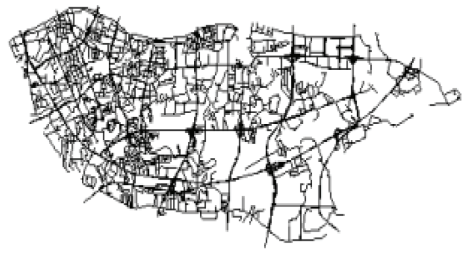


Figure 2: Road network of Guangzhou Haizhu District.

5.2. Selection of Experimental Parameters. When the parameters of SVM are optimized, the number of parallel nodes is 4, the GA population size $m = 120$, and the maximum evolution algebra 400.

5.3. Result Analysis. The parameters of SVM are optimized by three kinds of algorithms. They are the serial GA, the parallel GA based on MPI, and the parallel GA based on MapReduce. The results of parameter optimization are shown in Table 2.

The performance of the serial algorithm and the parallel algorithm (the number of parallel nodes is 16) is contrasted. It is analyzed from two aspects: prediction accuracy and operation efficiency.

5.3.1. Prediction Accuracy. The prediction results and RE curves of Link 103-104 by three algorithms are shown in Figures 3, 4, and 5. It can be seen that the imitative effect of the prediction values and the actual values by two parallel algorithms is better than serial algorithm. When the traffic flow is fluctuating greatly, the absolute relative error of parallel algorithm based on MapReduce is relatively stable. Table 3 is the evaluation index of the prediction precision by three kinds of algorithm. It can be seen that MRE, MAXRE, and RMSE of parallel algorithms are smaller than the serial algorithm, so their prediction precision is higher. Because the parallel genetic algorithm can avoid the shortcomings of traditional genetic algorithm, the parameters of SVM are optimized better, and then the prediction precision is improved.

5.3.2. Operation Efficiency. Figure 6 is the running time contrast of two kinds of parallel algorithm. From the diagram, it can be seen that when the number parallel nodes is less than 4, the advantage of the parallel algorithm based on MapReduce is not obvious, because too little number of parallel nodes that the map phase will spend more time. With the increase in the number of parallel nodes, the advantage of MapReduce is manifested gradually. The running time is reduced greatly. But when the number of parallel nodes is increased to 16, the running time of two kinds of parallel algorithm is reduced slightly; the reason is that, with the increase of the number of parallel nodes, communication costs among the different nodes are increased gradually to increase communication time. Therefore, the proper number of parallel nodes in the process of forecasting can achieve high performance, save resources, and improve efficiency.

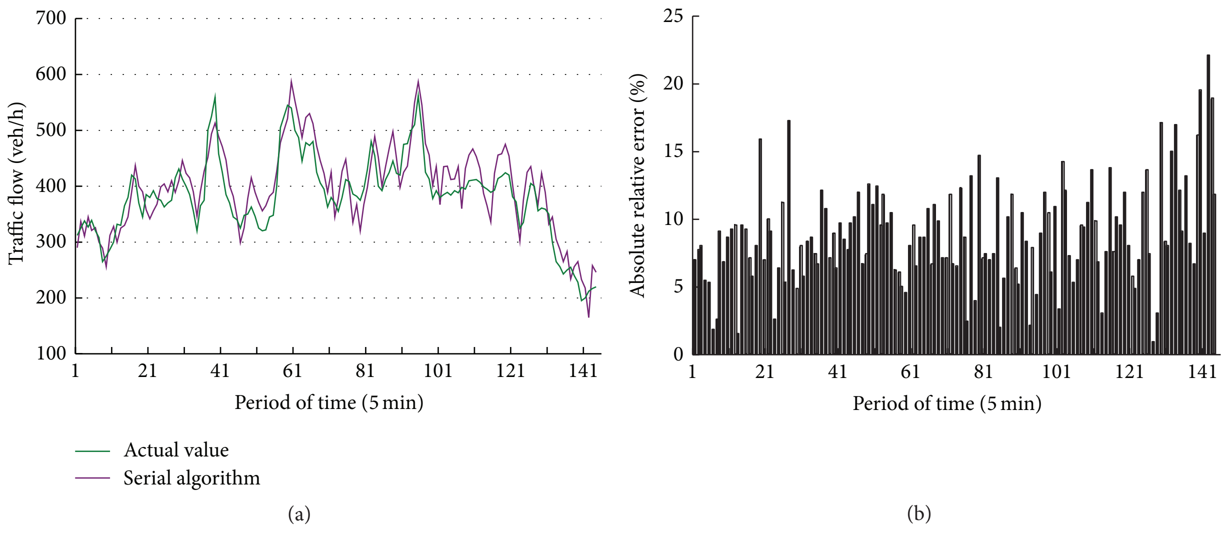


Figure 3: (a) Prediction results based on the serial algorithm, (b) RE based on the serial algorithm.

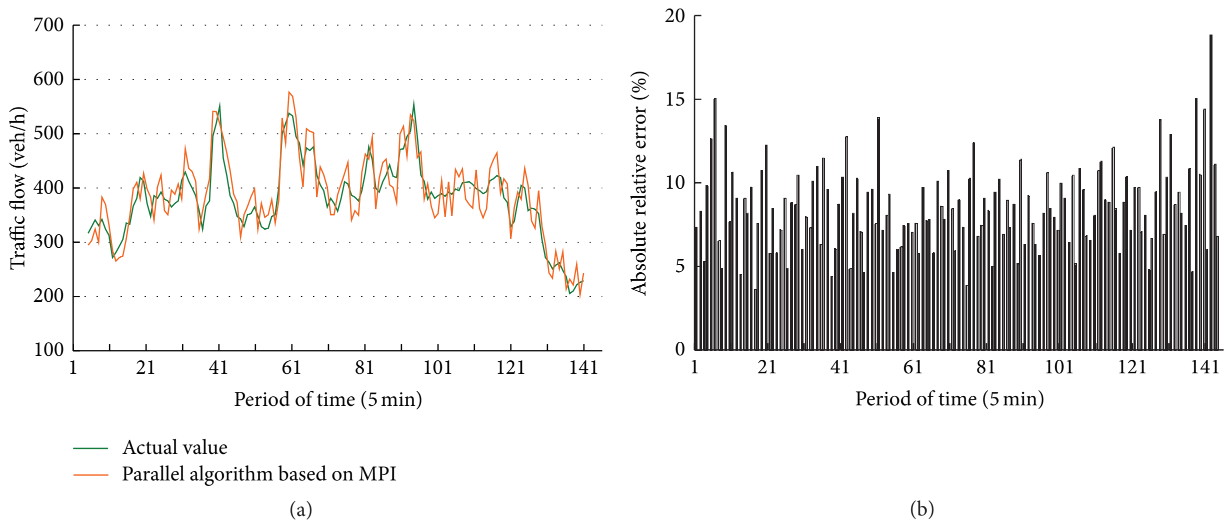


Figure 4: (a) Prediction results based on MPI, (b) RE based on MPI.

Table 2: SVM parameter values of three kinds of model.

Model	C	ε	σ
Serial GA	105.23	0.016	0.89
GA Based on MPI	102.45	0.021	1.22
GA Based on MapReduce	100.01	0.015	0.72

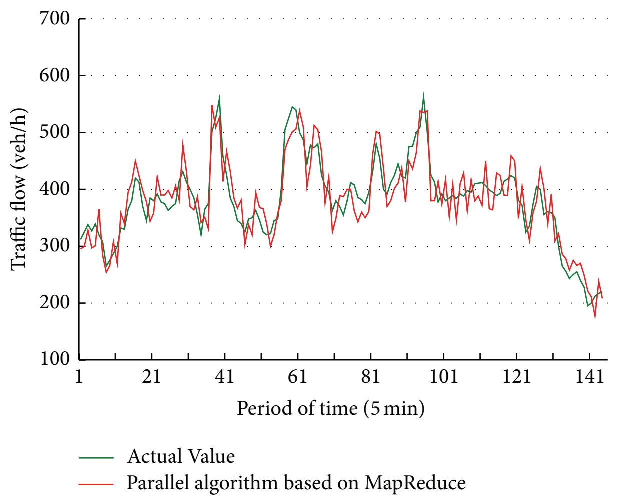
Table 3: Evaluation index of three kinds of algorithms.

Algorithm	MRE	MAXRE	RMSE
Serial GA-SVM	0.0914	0.2217	0.0956
GA-SVM Based on MPI	0.0881	0.1887	0.0887
GA-SVM Based on MapReduce	0.0779	0.1651	0.0807

The speedup is an important index to measure the efficiency of the parallel algorithm. The higher the speedup is, the higher the efficiency will be. Figure 7 is the speedup contrast of two parallel algorithms. It can be seen that with the increase in the number of parallel nodes, the speedup of two algorithms is higher and higher, and the speedup of parallel algorithm based on MapReduce is much higher than the parallel algorithm based on MPI. When the number of parallel nodes is 20, the speedup of parallel algorithm based on MapReduce is $S_n = T_s/T_p = 770.15\,\mathrm{s}/45.42\,\mathrm{s} = 16.96$. Its efficiency is 16.96 times as high as the serial algorithm.

6. Conclusions

In this paper, we presented traffic flow prediction model for large-scale road network based on cloud computing, which



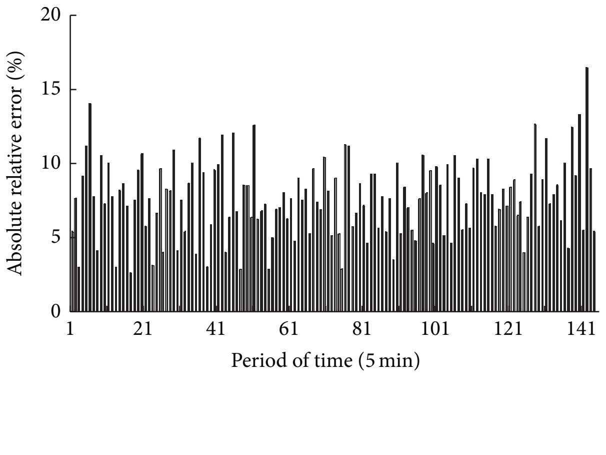


Figure 5: (a) Prediction results based on MapReduce, (b) RE based on MapReduce.

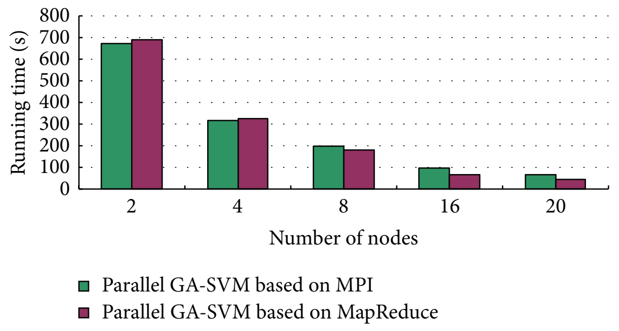


Figure 6: Running time comparison.

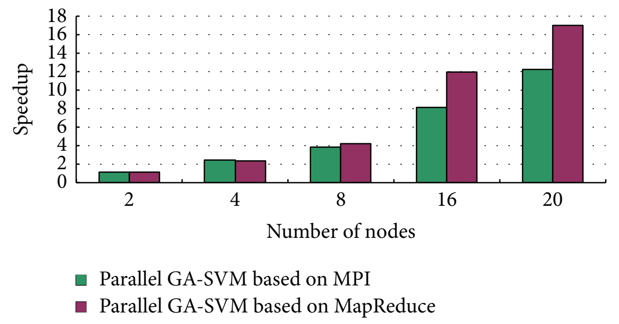


Figure 7: Speedup contrast.

has been implemented successfully by introducing genetic algorithm and support vector machine. In the process of traffic flow forecasting, we got the optimal parameters of support vector machine, the highest prediction accuracy, and the shortest running time. Finally, we verified the superiority of the proposed algorithm and the model through a numerical example based on Hadoop.

For further issues, we should introduce other algorithms into traffic flow prediction model for large-scale road network and verify it by the larger road network to be closer to the actual situation.

Conflict of Interests

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

Acknowledgments

This work is partly supported by Chinese National High Technology Research and Development Program (Grant nos. 2012AA112307, and 2014BAG03B03), Postdoctoral Science Foundation of China (Grant no. 2013T60331), and National Science Foundation of China (Grant nos. 51308248, and 61104168). The authors thank the anonymous reviewers for their valuable input and suggestions.

References

[1] Q. R. Li, L. Chen, Z. Zhang, and X. J. Zhi, “Parallel spatio-temporal data fusion on traffic flow prediction of road section,” *China Journal of Hebei University of Technology*, vol. 37, no. 3, pp. 112–116, 2008.

[2] Q. Q. Deng, *Study on parallel algorithms of flow prediction and path optimization for traffic flow guidance system [M.S. thesis]*, China Dalian University of Technology, 2008.

[3] F. Wang, G. Z. Tan, H. M. Shi, and Y. X. Xu, “Traffic flow prediction based on parallel generalized neural network,” *China Computer Engineering and Applications*, vol. 46, no. 17, pp. 229–231, 2010.

[4] F. Wang, *Research on Methods of Traffic Flow Forecasting Based on SVM [Master thesis]*, China Dalian University of Technology, 2010.

[5] L. Zhang, *Design and implementation of short-term traffic flow prediction algorithm based on cloud platform [M.S. thesis]*, Dalian University of Technology, Dalian, China, 2013.

[6] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," *Parallel Computing. Theory and Applications*, vol. 29, no. 4, pp. 535–551, 2003.

[7] D. Lim, Y. S. Ong, Y. Jin, B. Sendhoff, and B. S. Lee, "Efficient hierarchical parallel genetic algorithms using grid computing," *Future Generation Computer Systems*, vol. 23, no. 4, pp. 658–670, 2007.

[8] A. Radenski and L. Ehwerhemuepha, "Speeding-up codon analysis on the cloud with local MapReduce aggregation," *Information Sciences*, vol. 263, pp. 175–185, 2014.

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

[10] D. Tapiador, W. O'Mullane, A. G. A. Brown, X. Luri, E. Huedo, and P. Osuna, "A framework for building hypercubes using MapReduce," *Computer Physics Communications*, vol. 185, no. 5, pp. 1429–1438, 2014.

[11] H. Mohamed and S. Marchand-Maillet, "MRO-MPI: MapReduce overlapping using MPI and an optimized data exchange policy," *Parallel Computing*, vol. 39, no. 12, pp. 851–866, 2013.

[12] V. Cherkassky and Y. Q. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, vol. 17, no. 1, pp. 113–126, 2004.

[13] S. J. Plimpton and K. D. Devine, "MapReduce in MPI for large-scale graph algorithms," *Parallel Computing*, vol. 37, no. 9, pp. 610–632, 2011.

[14] Q. F. Yang, D. Mei, Z. B. Han, and B. Zhang, "Ant colony optimization for the shortest path of urban road network based on cloud computing," *China Journal of Jilin University (Engineering and Technology Edition)*, vol. 43, no. 5, pp. 1210–1214, 2013.

[15] V. Vijayalakahmi, A. Akila, and S. Nagadivya, "The survey on MapReduce," *International Journal of Engineering Science and Technology*, vol. 4, pp. 3335–3342, 2012.

[16] R. Ben-Shalom, A. Aviv, B. Razon, and A. Korngreen, "Optimizing ion channel models using a parallel genetic algorithm on graphical processors," *Journal of Neuroscience Methods*, vol. 206, no. 2, pp. 183–194, 2012.

[17] N. Maheshwari, R. Nanduri, and V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 119–127, 2012.

[18] N. K. Alham, M. Z. Li, Y. Liu, and M. Qi, "A MapReduce -based distributed SVM ensemble for scalable image classification and annotation," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1920–1934, 2013.

[19] M. Acacio, O. Cánovas, J. M. García, and P. E. López-de-Teruel, "MPI–Delphi: an MPI implementation for visual programming environments and heterogeneous computing," *Future Generation Computer Systems*, vol. 18, no. 3, pp. 317–333, 2002.

[20] O. Devos, G. Downey, and L. Duponchel, "Simultaneous data pre-processing and SVMclassification model selection based on a algorithm applied to spectroscopic data of olive oils," *Food Chemistry*, vol. 148, pp. 124–130, 2014.

[21] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," *Neurocomputing*, vol. 64, no. 1–4, pp. 107–117, 2005.