2011 14th International IEEE Conference on
Intelligent Transportation Systems
Washington, DC, USA. October 5-7, 2011

# Dynamic Route Guidance using Maximum Flow Theory and its MapReduce Implementation

Peijun Ye, Cheng Chen and Fenghua Zhu

*Abstract*— **Road traffic load balancing can avoid network congestion and improve traffic efficiency. This paper proposes a method of dynamic route guidance based on Maximum Flow Theory to balance traffic load of road network. A modified Ford-Fulkerson algorithm is used for searching the optimal route. In addition, the algorithm is implemented by using MapReduce primitives, which introduces Cloud Computing Platform for large-scale traffic network guidance. Computational experimental environment is built by integrating Artificial Transportation Systems (ATS) and Hadoop. Results in ATS and performances on Hadoop show that the method proposed can improve the traffic situation effectively.**

## I. Introduction

With the urbanization development, transportation problems are increasingly prominent. Intelligent Transportation System (ITS), composed of the Advanced Traffic Management System (ATMS), Advanced Traveler Information System (ATIS), Advanced Vehicular Controlling System (AVCS) and Advanced Public Transportation System (APTS), is an efficient means to solve the traffic problem on the premise of ensuring travel safety. As a crucial part of ATIS, the dynamic route guidance can reduce travel time, avoid congestion, and raise road network efficiency. The dynamic route guidance system can readily release real-time traffic information so that drivers can choose the optimal route available in time.

Scholars have conducted a lot of research in this area so far and acquired valuable results. Reinforcement learning has been used in the dynamic route guidance by Zhang [1], which is a piece of representative work. In his paper, time costs were described by Q-factors which were iterated in order to obtain the optimal route-choice by observing probe vehicles running in the road network. Later, a natural algorithm, Particle Swarm Optimization (PSO) algorithm, was introduced into this filed by Chen [2] for the requirement of real-time guidance. Yuan [3] concentrated on the travel time reliability during the process of route guidance. However, on a macro level, traffic management must balance the traffic load to avoid traffic jams in particular roads. And the traffic volume on each road is an important index to measure the congestion level. Therefore, guiding the traffic flow to low-volume roads can contribute to an effective utilization of road network and balance traffic load. In this light, the maximum flow theory can be used for traffic guidance. In addition, it can be more effective to search the optimal route between any two intersections and publish the results via broadcasting or Variable Message Signs (VMS). This enumerates the optimal routes of all node pairs. In this way, nearly all the calculations converged at the traffic control center while networked terminal devices (VMS, radio or vehicle navigation equipment) only need to receive the optimal results. This method involves massive computation, which can resort to the Cloud Computing Platform. In fact, some researchers and engineers have been studying traffic control on the basis of such a platform and have proposed the concept of Intelligent Traffic Clouds (ITC) [4]. This paper aims to introduce some work about this. Section II briefly reviews the maximum flow theory that is used during the search of optimal path in graph theory. Then, we establish a route guidance model according to this theory and present a modified algorithm. Meanwhile, an example helps illustrate the computation process. For the massive computation, section III illustrates the implementation of the algorithm in the form of MapReduce primitives and shows the foundation for introducing a Cloud Computing Platform in an abstracted road network. Section IV offers some statistical results of computational experiments in Artificial Transportation System and gives the performance analysis. Finally, we draw a conclusion and suggest our future work.

Manuscript received May 15, 2011. This work was supported in part by NSFC 60921061, 70890084, 90920305, 90924302, 60904057, 60974095; CAS 2F09N05, 2F09N06, 2F10E08, 2F10E10; 2F11D01.

Peijun Ye is with State Key Laboratory for Intelligent Control and Management of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. (e-mail: dreamflight@163.com).

Cheng Chen is with State Key Laboratory for Intelligent Control and Management of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. (e-mail: chengchen.cas@gmail.com).

Fenghua Zhu is with State Key Laboratory for Intelligent Control and Management of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. (e-mail: fenghua.zhu@ia.ac.cn).

## II. Traffic Guidance Process based on Maximum Flow Theory

The maximum flow theory enjoys a wide scope of applications but few of them has to do with traffic route guidance. Assume that N=(V, A) is a connected directed network with no loop, where V and A are the sets of vertices and arcs, with the following requirements met: 1)there is a subset X of vertices that the in-degree of each is zero; 2)with no mutual vertices of X, there is a subset Y of vertices that the out-degree of each is zero, too; 3)each arc has a non-negative weights, known as the arc capacity. Further, we use N=(V, X, Y, A, C) to represent the network above, where V is vertex set, A is arc set and X, Y, C are called source set, convergence set and capacity function respectively. The maximum flow problem can be described as seeking the maximum feasible flow in a given network.

978-1-4577-2196-0/11/$26.00 ©2011 IEEE

Ford and Fulkerson first proposed the concept of maximum flow and designed a pseudo-polynomial algorithm in 1957 [5]. Another concept, augmenting path, should be explained here. We call P as an augmenting path if any arc $\vec{a}$ in P meets $\Delta f(\vec{a})=c(\vec{a})-f(\vec{a})>0$ when $\vec{a}$ is a forward arc, and $\Delta f(\vec{a})=f(\vec{a})>0$ when $\vec{a}$ is a backward arc. In the above equations, $c(\vec{a})$, $f(\vec{a})$ and $\Delta f(\vec{a})$ are the capacity, current flow and augmenting flow of arc $a$. We define $\Delta f(P)=\min_{\vec{a}\in P}\Delta f(\vec{a})$ as augmenting flow in P. Network flow can be increased according to the following formula:

$$\hat{f}(\vec{a})=\begin{cases} f(\vec{a})+\Delta f(P), & if\ \vec{a}\ is\ a\ forward\ arc \\ f(\vec{a})-\Delta f(P), & if\ \vec{a}\ is\ a\ backward\ arc \\ f(\vec{a}), & if\ \vec{a}\ is\ not\ in\ P \end{cases} \quad (*)$$

Generally, the urban road network can be abstracted as a network in graph theory, with roads and intersections viewed as arcs and vertices. The maximal flow of each road can be defined as its capacity that can be calculated according to the formula in Traffic Engineering [6]. Obviously, according to the two definitions, the path with maximum augmenting flow means that the traffic volume can be maximally increased. Guiding the traffic flow to this path can contribute to an effective utilization of road network to the maximal extent. By searching the augmenting paths between each node pair based on maximum flow theory dynamically, we can achieve the purpose of traffic guidance by publishing that kind of paths on traffic networked terminal devices (VMS, radio or vehicle navigation equipment). However, augmenting path in graph theory may contain a backward arc which poses a vital problem: in the actual road network, the backward arc in an augmenting path is forbidden to drive reversely given the traffic rule. Thus, the augmenting path which contains backward arc should not be the guidance route. With this considered, *the rest of this paper defines augmenting path as those only contain forward arcs*. Let $f(\vec{a})$ represent the original flow of arc $\vec{a}$ in the input network. The whole process of guidance can be illustrated in Figure 1 while the main searching operation details are as follows:

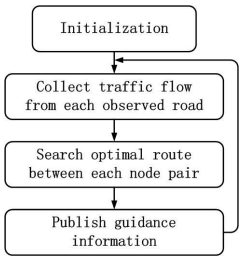



Fig. 1. Dynamic route guidance process

Input: current volume of each arc in network N(V, A, C);
Output: maximal augmenting path without backward arc between each node pair;

For each node pair (x, y) begin

Step 0: PA=∅. (PA is set of augmenting paths)

Step 1: for each arc $\vec{a}$, let $\Delta f(\vec{a})=c(\vec{a})-f(\vec{a})$, then we get a new graph called residual network.

Step 2: let Stack=x, mark x with VISITED while other vertices with NOT VISITED. (Stack is the record of a particular path)

Step 3: for the top vertex u in the Stack, if there is a adjacent vertex v which meets: 1° (u, v)∈A; 2° v∉Stack; 3° v has not been visited through u; then go to Step 4, else go to Step 5.

Step 4: push v and go to Step 6.

Step 5: mark the vertices which are visited through u with NOT VISITED and pop u, then go to Step 6.

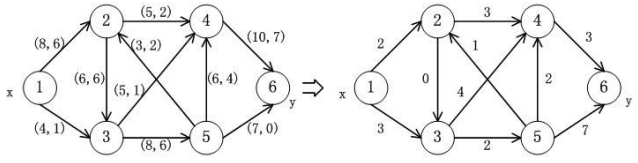
Step 6: if y is at the top of the Stack, mark y with NOT VISITED, copy the path P recorded in the Stack to PA and pop y.

Step 7: if the Stack is not empty, go to Step 2, else go to Step 8.

Step 8: if PA=∅, then current flow is the maximum flow and searching process ends, else for each P∈PA, calculate augmenting flow $\Delta f(P)=\min_{\vec{a}\in P}\Delta f(\vec{a})$ and use P with maximal Δf(P) as the output and searching process ends.

End.

In this algorithm, steps 2 to 7 are the repeated process of finding augmenting paths. Step 3 only deals with forward arc which ensures a path containing no backward ones. Step 8 calculates augmenting flow of each path in PA and sets the path with the maximal Δf(P) as the optimal solution. PA=∅ means there is no augmenting path that any route is the optimal solution which we will not discuss here. Figure 2 gives a simple example, in which number pair besides each arc in the original network represents its (capacity, current flow). The residual network are calculated according to Step 1. Based on the process of computation, it is easy to get the result that $P_8$ is the optimal solution.




$P_1$: 1->2->3->5->4->6,
$\Delta f(P_1)=\min\{\Delta f(1,2), \Delta f(2,3), \Delta f(3,5), \Delta f(5,4), \Delta f(4,6)\}=\min\{2, 0, 2, 2, 3\}=0$
$P_2$: 1->2->3->5->6,
$\Delta f(P_2)=\min\{\Delta f(1,2), \Delta f(2,3), \Delta f(3,5), \Delta f(5,6)\}=\min\{2, 0, 2, 7\}=0$
$P_3$: 1->2->3->4->6,
$\Delta f(P_3)=\min\{\Delta f(1,2), \Delta f(2,3), \Delta f(3,4), \Delta f(4,6)\}=\min\{2, 0, 4, 3\}=0$
$P_4$: 1->2->4->6,
$\Delta f(P_4)=\min\{\Delta f(1,2), \Delta f(2,4), \Delta f(4,6)\}=\min\{2, 3, 3\}=2$
$P_5$: 1->3->5->2->4->6,
$\Delta f(P_5)=\min\{\Delta f(1,3), \Delta f(3,5), \Delta f(5,2), \Delta f(2,4), \Delta f(4,6)\}=\min\{3, 2, 1, 3, 3\}=1$
$P_6$: 1->3->5->4->6,
$\Delta f(P_6)=\min\{\Delta f(1,3), \Delta f(3,5), \Delta f(5,4), \Delta f(4,6)\}=\min\{3, 2, 2, 3\}=2$
$P_7$: 1->3->5->6,
$\Delta f(P_7)=\min\{\Delta f(1,3), \Delta f(3,5), \Delta f(5,6)\}=\min\{3, 2, 7\}=2$
$P_8$: 1->3->4->6,
$\Delta f(P_8)=\min\{\Delta f(1,3), \Delta f(3,4), \Delta f(4,6)\}=\min\{3, 4, 3\}=3$
Optimal solution: $P_8$ and $\Delta f(P_8)=3$

Computation of PA

Fig. 2. An example of computation

Here, we need to emphasize that the optimal solution is not unique, but every possible solution has the same augmenting flow. In this sense, all the possible solutions are equal. In fact,

PA has enumerated all the augmenting paths of the original graph, which ensures the optimality of our algorithm.

## III. Implementation in MapReduce Primitives for Cloud Computing

Although the algorithm above has shed light on searching augmenting path in theory, some practical problems still need to be considered. As mentioned in the introduction, nearly all the calculations converged at the traffic control center. So we are bound to encounter massive computation obstacle for a large scale network. Such calculation will trigger the sharp decrease of the update frequency of the optimal results, making the dynamic guidance meaningless for its unacceptable time interval. In this case, calculation speed is still the bottleneck.

It is believed that this problem can be solved through the Cloud Computing Platform which has recently aroused public attention. Google's use of cloud computing, and the subsequent open source Hadoop cloud computing infrastructure are most representative in this field. Since the Google's platform is not available to us, we adopt Hadoop to do our work. Both of these platforms provide data-processing capability by hosting so-called MapReduce jobs, which work by sequencing through data stored on disk. The technique increases scale by having a large number of independent (but loosely synchronized) computers running their own instantiations of the MapReduce job components on their data partition. Actually, having factored a problem in terms of MapReduce primitives, they are useful for computing in a streaming environment or on a single computer equipped with a large disk although its original design is for a cloud of computers [7]. So it can be concluded that the core issue of using a Cloud Computing Platform in calculation is to transform the algorithm into MapReduce primitives. The remainder of this section will expatiate on the implementation of the algorithm.

The whole process of MapReduce job is shown in Figure 3. Both map and reduce receive a sequence of records consisted of keys and values, and usually produce records in response. Input records presented to the mapper by its caller have no guaranteed order or relationship to one another. The mapper's job is to create some records (perhaps none) in response to each input record. During the partitions phase, records are sorted and spilt to the disk by key for the reducer. So those records with a given key are presented as a package to a reducer which then examines the packages sequentially through an iterator and gives an output file [8].

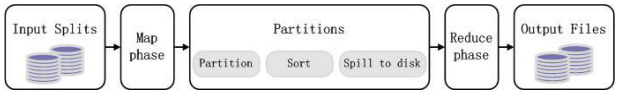



Fig. 3. MapReduce process

Due to the limitation of computing resources, the Hadoop platform in our work only contains 4 nodes (computers), the CPU and memory of which are all Intel(R) Core(TM)2 Quad Q8400 (2.66GHz) and 2.0 GB. Thus we cannot study

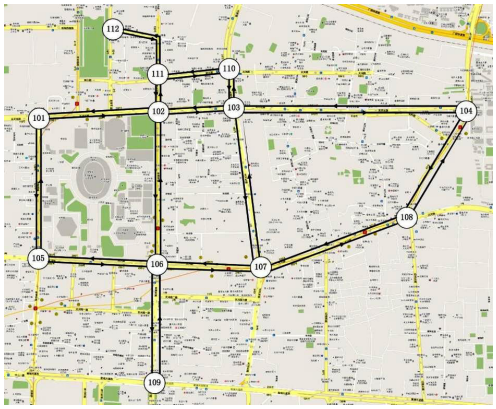


Fig. 4. Input map of Guangzhou

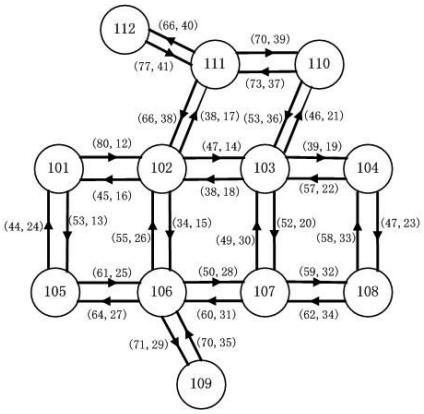



Fig. 5. Input graph

the guidance approach in a large-scale road network given the size of memory. However, the whole implementation process can be applied to a larger road network as long as we have enough nodes without any modification of MapReduce operation. Figure 4 shows the map of studied area of Tianhe Sports Center in the city of Guangzhou and Figure 5 is the input graph abstracted from the map. The two numbers on each arc represent the capacity and the current traffic volume of the road respectively, which are derived in a particular sampling moment from computational experiments in ATS (see the next section). Figure 6 summarizes the map and reduce operation intuitively. First, we need to input records that each holds an arc of the graph and its information (the rows on the left side in the Map phase). Then, for each arc record, the map operation emits two records, one keyed under each of the vertices that form the arc (the rows on the right side in the Map phase). The value of every record contains path (in the form of node pair) and augmenting flow ($\Delta f$). Each augmenting flow equals capacity minus volume. After this process, a series of bins are created, each of which holds records for every arc adjacent to its associated vertex. When it comes to the reduce phase, those records with the same key created in the map phase are put together by partition to search the path and calculate augmenting flow as is shown. Up to now, we have worked out all the 2-distance

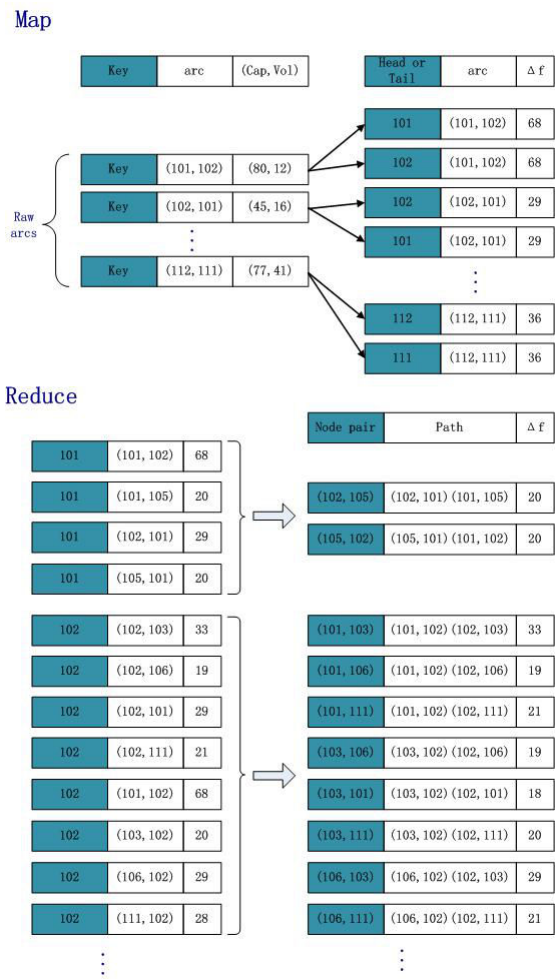



Fig. 6. The first round map and reduce operation

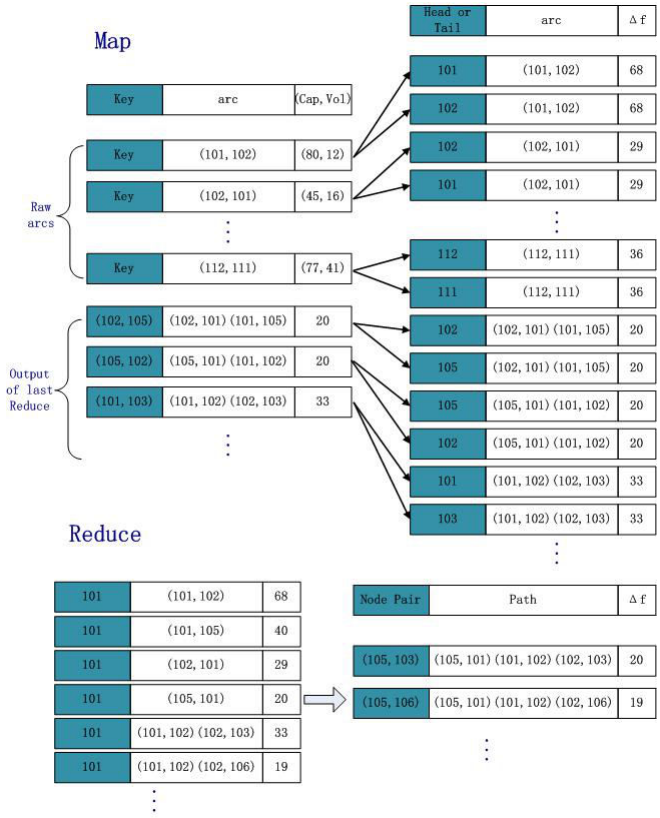



Fig. 7. The second round map and reduce operation

paths between node pairs from the output of the reduce operation. Then we'll move on to the second round map and reduce operation (see Figure 7). We need the combination of raw arcs and output of the last reduce to be the input file. Similarly, the map process generates a series of records. Again, records with the same key are prepared for a particular reducer. So we have calculated all the 3-distance (not only, perhaps longer) paths and the augmenting flows after this round. Similarly, we can begin the third, forth round and so on. It should be noted that during each round, loops must be excluded for their interference on obtaining the correct results. The iteration process is repeated until the outputs of two reduce operations are identical. This means that the paths and their augmenting flows between each node pair are contained in all of the outputs generated by each round reduce. All we need to do is sorting these results by node pair and augmenting flow. As mentioned above, MapReduce framework sorts the records by key before they reach the reducers but does not sort the values for any particular key. The sequence by which values appear is not even stable from one run to the next, since they come from different map tasks, which may finish at different times. Fortunately, designers of Haddop must have considered this problem and provided us with a smart means called Secondary Sort [9]. Here we change the keys to be composite: a combination of node pair and its augmenting flow. The framework will sort the order for keys by node pair and then by augmenting flow (descending) (see Figure 8). By setting a partitioner to partition by the node pair part of the key, we can ensure that records for the same node pair go to the same reducer. In addition, all the records for the same node pair will be in one reduce group on condition that we group values in the reducer by the node pair part of the key. Since they are sorted by augmenting flow in a descending order, the first is the maximum augmenting flow. So now we can publish the optimal routes between each node pair by extracting the first record from each reduce group.

Clearly, this implementation demonstrate that the whole process of solving this kind of problem is dependent on the Cloud Computing. In theory, it can be applied to road network of any size with sufficient hardware support. Results on Hadoop and its performance analysis will be shown in the next section.

## IV. Guidance Results and Performance Analysis

In this section, we will give the computational results of the particular input volume in Figure 5 on Hadoop to verify the correctness for the implementation of MapReduce. And performance indexes will then be listed to analyze its calculation speed. At last, computational experiments results

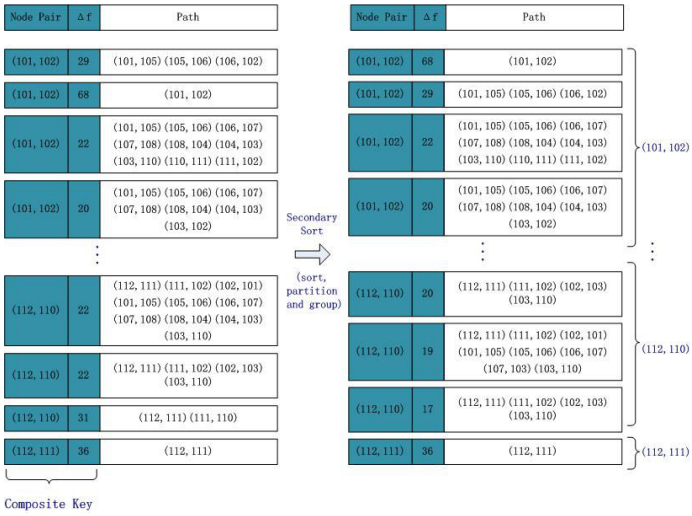



Fig. 8. Secondary Sort

TABLE I

PART OF THE GUIDANCE CALCULATION RESULTS

| Node Pair | Optimal Path | Augmenting Volume |
|---|---|---|
| (101,102) | (101,102) | 68 |
| (101,103) | (101,102)-(102,103) | 33 |
| (101,104) | (101,105)-(105,106)-(106,102)-(102,103) -(103,107)-(107,108)-(108,104) | 25 |
| (101,105) | (101,105) | 40 |
| (101,106) | (101,105)-(105,106) | 36 |
| (101,107) | (101,102)-(102,103)(103,107) | 32 |
| (101,108) | (101,102)-(102,103)-(103,107)-(107,108) | 27 |
| (101,109) | (101,105)-(105,106)-(106,109) | 36 |
| (101,110) | (101,102)-(102,103)-(103,110) | 25 |
| (101,111) | (101,105)-(105,106)-(106,102)-(102,103) -(103,110)-(110,111) | 25 |
| (101,112) | (101,105)-(105,106)-(106,102)-(102,103) -(103,110)-(110,111)-(111,112) | 25 |
| (102,101) | (102,101) | 29 |
| (102,103) | (102,103) | 33 |
| (102,104) | (102,103)-(103,107)-(107,108)-(108,104) | 25 |
| ... | ... | ... |

in ATS will be shown to investigate the effectiveness in the level of road network.

Running the program written in Java according to the implementation on Hadoop, we obtain 6 round iterations and Table I shows part of its final results which contain 132 node pairs and paths. Obviously, each path denoted by node pair with its augmenting volume is shown clearly and its optimality can be easily verified. In terms of computing performance, we have averaged run time of map and reduce phases and number of task distribution on 4 nodes in ten experiments (Table II and Table III). For the Map phase during iteration, the maximum and minimum time of executing one task is 3 and 9 seconds (which depends on computer performance), while the total time is less than 13 seconds. With the number of task distributed on each node considered, the time needed for executing one task might be reduced to 9 seconds at most, if given enough resources to ensure absolute task-node pairing. By contrast, the situation in Reduce phase is not so optimistic. As the iterated time increases, the amount of data sharply increases. Round 5 and round 6 have consumed 1024 and 735 seconds to finish the Reduce operation, which are evidently the bottleneck in computing. This problem can be alleviated to some extent by increasing the number of task in Reduce, which need to be studied further. Generally speaking, performance indexes have indicated that the total lapse of time in searching the optimal routes between each node pair seems a little long due to the small memory and the scarce cloud nodes. Increasing the number of computers may accelerate the speed.

To investigate the effectiveness of guidance approach based on maximum flow, we chose TransWorld, a computational platform in ATS, to carry out a series of computational experiments. It is based on ACP theory proposed by Fei-Yue Wang in 2004 for urban traffic study from the perspective of complex system [10]-[17]. Communications between TransWorld and Hadoop have been established regularly to exchange traffic volume data and optimal routes. In every

TABLE II

RUN TIME OF MAP AND REDUCE PHASES

| Iterate number | Time of Map (s) | | | Time of Reduce (s) | | |
|---|---|---|---|---|---|---|
| | min | max | total time | min | max | total time |
| 1 | 3 | 3 | 3 | 12 | 12 | 12 |
| 2 | 3 | 9 | 9 | 18 | 18 | 18 |
| 3 | 3 | 3 | 7 | 18 | 18 | 18 |
| 4 | 3 | 3 | 8 | 57 | 57 | 57 |
| 5 | 3 | 3 | 6 | 1024 | 1024 | 1024 |
| 6 | 3 | 9 | 13 | 735 | 735 | 735 |
| secondary sort | 3 | 3 | 31 | 39 | 39 | 39 |

sampling cycle, TransWorld sends volumes detected from each link and receives the optimal results from Hadoop. All the experiments adopted 70000 artificial population consistently and simulated a single day's traffic (07:00-22:00) in the road network as in Figure 5. To make it more reliable, we have set the guidance proportion as 0, 20, 50 and 80 percent. Experiments at each proportion have been carried out for ten times. Figure 9 and Figure 10 show the results of statistical average speed and vehicle number. It is clear that the traffic condition of the whole road network gradually improves with guidance proportion increase. This trend is more obvious as for the non-peak hours, while the effect is not so remarkable

TABLE III

TASK DISTRIBUTION ON 4 NODES

| Iterate number | Task distribution on 4 nodes in Map phase | | | | Task distribution on 4 nodes in Reduce phase | | | |
|---|---|---|---|---|---|---|---|---|
| node ID | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 3 | 4 | 2 | 4 | 0 | 1 | 0 | 0 |
| 3 | 3 | 2 | 4 | 4 | 0 | 0 | 1 | 0 |
| 4 | 2 | 4 | 4 | 3 | 0 | 1 | 0 | 0 |
| 5 | 3 | 2 | 4 | 4 | 0 | 0 | 1 | 0 |
| 6 | 4 | 4 | 3 | 2 | 0 | 1 | 0 | 0 |
| secondary sort | 18 | 19 | 18 | 18 | 0 | 1 | 0 | 0 |

in peak hours. The reason for this phenomenon is that people's going to work and going back home leads the surge of travel demand during morning and evening peak hour. In this case, the maximum capacity of the road network probably will be challenged or even surpassed, so that any optimal algorithm cannot improve the indexes obviously. So it is only through the expansion of road network can we solve this problem. On the other hand, in some time intervals, two indexes do not strictly improve with the increase of guidance proportion. This is because TransWorld adopts random travel principle and cannot guarantee the same vehicle number and status at a particular sample time in different experiments. However, the guidance algorithm for three proportions has roughly brought about 2.08%, 5.81%, 10.12% increase in the average speed and 6.17%, 12.70%, 15.98% decrease in the number of vehicles respectively. This proves that the dynamic guidance approach mentioned in section 2 is effective.

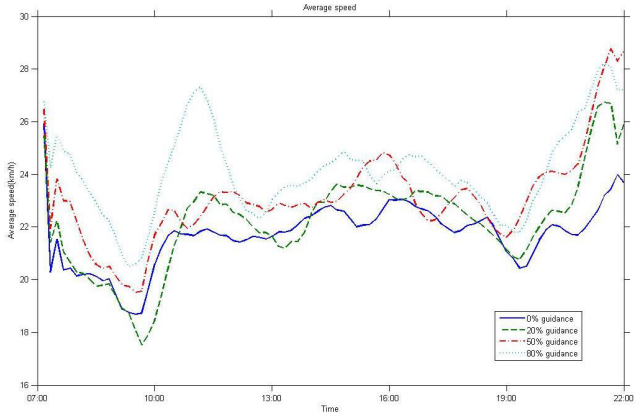



Fig. 9. Average speed

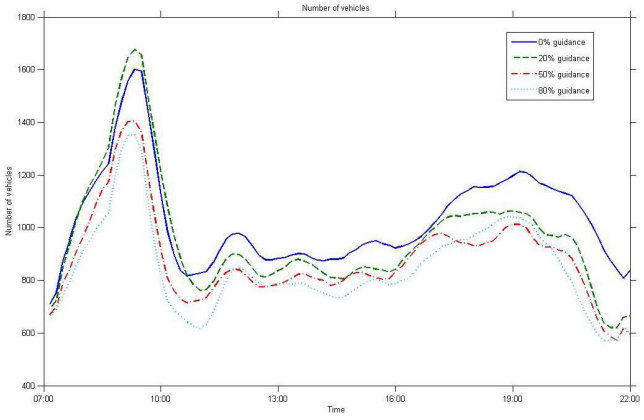



Fig. 10. Number of vehicles

## V. Conclusions and Future Works

By focusing on the dynamic route guidance, this paper introduces a new method based on the maximum flow theory and shows the main steps in searching the optimal path. Implementation in MapReduce primitives is outlined to introduce the Cloud Computing Platform so as to solve massive computation problems. Computational experiments in ATS have verified the guidance approach. Results and performance on Hadoop demonstrate the feasibility of cloud computing. We still have a lot to do in future, among which we will study how to improve the calculation performance and test the relationship between cloud and road network scale.

## VI. Acknowledgments

We would like to express our heartfelt gratitude for professor Fei-Yue Wang for his guidance and encouragement as well as those students without the help of whom this research cannot be fulfilled.

## References

[1] Zi Zhang and Jian-min Xu, "A Dynamic Route Guidance Arithmetic based on Reinforcement Learning", *in Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, China, 18-21 August 2005, pp. 3607-3611.

[2] Chen Qun, "Dynamic Route Guidance Method Based on Particle Swarm Optimization Algorithm", *in 2009 Second International Conference on Intelligent Computation Technology and Automation*, Changsha, China, 10-11 Oct 2009, pp. 267-270.

[3] Manrong Yuan, Zhaosheng Yang and Shifeng Niu, "Study on Dynamic Route Guidance Method of Vehicle Based on Travel Time Reliability", *in Advanced Computer Control (ICACC), 2010 2nd International Conference*, Shenyang, China, 27-29 March 2010, pp. 292-295.

[4] ZhenJiang Li, Cheng Chen and Kai Wang, Cloud Computing for Agent-Based Urban Transportation Systems, *IEEE Intelligent Systems*, vol. 26, no. 1, 2011, pp. 73-79.

[5] L.R.Ford and D.R.Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics*, no. 8, 1956, pp. 399-404.

[6] Research Institute of Highway Ministry of Transport, Road Capacity Manual, China, 2003.

[7] Jonathan Cohen, Graph Twiddling in a MapReduce World, *Computing in Science & Engineering*, vol. 11, no. 4, 2009, pp. 29-41.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplifed Data Processing on Large Clusters", *in Proceedings of OSDI'04: 6th Symposium on Operating System Design and Implementation*, San Francisco, CA, Dec. 2004.

[9] Tom White, *Hadoop: The Definitive Guide*, O'Reilly Media, Inc. June 2009.

[10] Fei-Yue Wang, Computational theory and methods for complex systems, *China Basic Sci.*, vol.6, no.41, 2004, pp. 3-10.

[11] Fei-Yue Wang, Artificial societies, computational experiments, and parallel systems: An investigation on computational theory of complex social-economic systems, *Complex System and Complex Science.*, vol.1, no.4, 2004, pp. 25-35.

[12] Fei-Yue Wang, "Integrated intelligent control and management for urban traffic systems," *in Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, 2003, pp.1313-1317.

[13] Fei-Yue Wang and Shu-ming Tang, Concepts and frameworks of Artificial Transportation Systems, *Complex Systems and Complexity Science*, vol.1, no.2, 2004, pp. 52-59.

[14] Shuming Tang, "A preliminary study for basic approaches in Artificial Transportation Systems," Ph.D. dissertation, Institute of Automation, Chinese Academy of Sciences, Beijing, 2005.

[15] Jinyuan Li, Shuming Tang and Xiqin Wang, "A software architecture for Artificial Transportation Systems - principles and framework", *in Proceedings of the 10th IEEE International Conference on Intelligent Transportation Systems*, Seattle, WA, Sept. 30-Oct. 3, 2007, pp.229-234.

[16] Jinyuan Li, Shuming Tang and Fei-Yue Wang, "An Investigation on ATS from the Perspective of Complex Systems", *in Proceedings of the 11th IEEE International Conference on Intelligent Transportation Systems*, Beijing, China, Oct. 12-15, 2008, pp.20-24.

[17] Fei-Yue Wang, Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications, *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, 2010, pp.630-638.