*Advanced Engineering Forum* *Online: 2012-07-26*
*ISSN: 2234-991X, Vol. 5, pp 133-137*
*doi:10.4028/www.scientific.net/AEF.5.133*
*© 2012 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.*

# Two-Stage Data Mining Based Vehicle Navigation Algorithm in Urban Traffic Network

Xiantong Li[1,a], Shi An[1,b]

[1]No.73, Huanghe Street, School of Transportation Science and Engineering, Harbin Institute of Technology, Harbin, China

[a]lxt@hit.edu.cn, [b]anshi@hit.edu.cn

**Keywords:** Urban traffic network, data mining, vehicle navigation, traffic information

**Abstract.** Along with the development of Intelligent Transportation System, traffic detectors collect numerous transportation state data in information databases and accumulate. Such data is greatly meaningful to the vehicle navigation. In this paper, we propose a noble two-stage algorithm about vehicle navigation by using data mining methods on the historical and current transportation dataset. This algorithm begins with picking sensitive data about start and end point in an urban traffic network, and data from related (or nearest) road fragments. Referring to current time and season, the algorithm gives an evaluation to every related road fragments and outputs a most reasonable route between start and end point. The experimental and theoretical analyzes show that this algorithm can form an efficient and effective route in reasonable time.

## Introduction

Urban traffic navigation algorithms are greatly based on traffic information datasets, which can be get from detectors of ITS (Intelligent Transformation System). But, after the ITS occurred, a huge number of detectors were spread into traffic network to monitor the partial or total transportation system situation. In such situation today, mass data is formed more and more quickly than before, on which introduces new problem into traffic information analysis.

According to such situation, analysis methods *cannot* just be the regular ones, such as association rules mining, data fusion, and data classification, in order to simulate the nearest future about traffic condition. These methods have inherently functionally insufficient problem. *For example*, association rules mining may evaluates single road segments in a route separately, without considering the structure or substructure of the urban traffic network. When a road segment between start and end point is used frequently, it might have highly related with the new route. Obviously, this segment happens to have nothing to do with the new route when it is a one-way street, or a recently broken one.

Focusing on graph datasets, which records data and relationships between data at the same time, graph mining is a meaningful branch of data mining to solve such problems, especially in vehicle navigation. For a single graph data, it has two parts, which are vertices and edges. Vertices represent data, and edges stand for relationships. *For example*, in chemical dataset, vertices are atoms, and edges are compound keys between atoms. In urban traffic network, vertices are separate street segments, and edges are traffic crosses. When analysis applies on urban traffic network graph dataset, the results bring out some interesting knowledge behind the mass dataset.

Unfortunately, graph mining algorithms are the high time-complexity class. The algorithms on such problems have to bare the complexity because the subgraph isomorphism is NP-complete [1,2]. Today, there are several algorithms proposed to solve graph mining problems. On chemical dataset, the algorithms are based on depth first visiting method to mining frequent substructures to achieve high performance [3-5]. But, in traffic datasets, it is very different from chemical datasets, where the size of graphs is much bigger than chemical datasets. When the algorithms in chemical dataset graph mining are introduced into traffic datasets, they should not work well, or have limited efficiency.

When a vehicle navigation algorithm combined with the thinking of graph mining, it considers not only the frequency of road segments, but also the substructures of traffic network. Most existing navigation algorithms are founded on shortest path in a directed graph. Shortest path in a directed

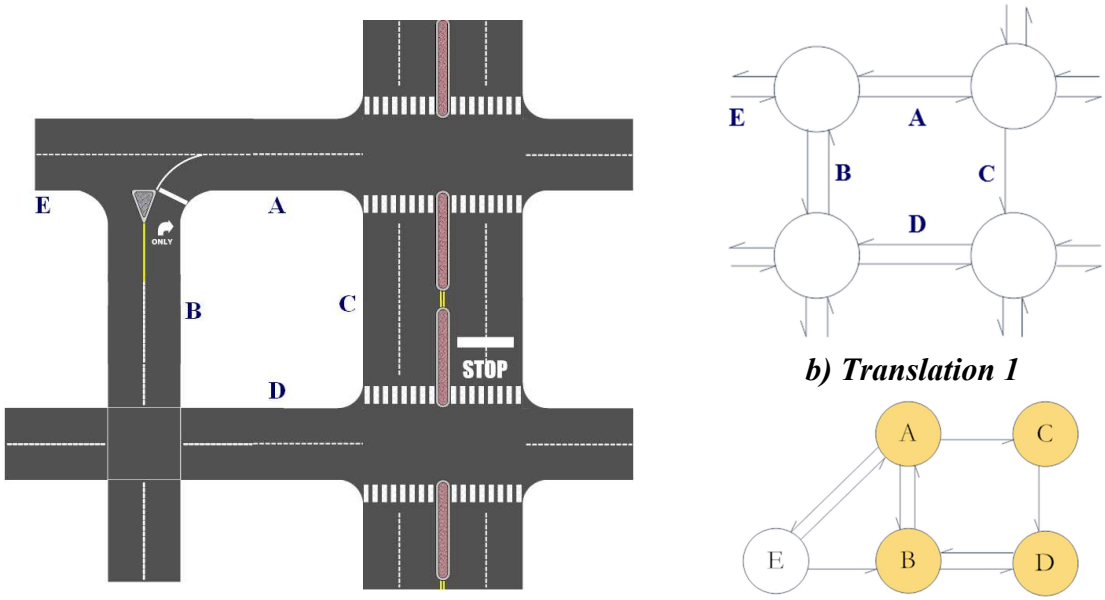
This article is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license
(https://creativecommons.org/licenses/by/4.0)

graph [6] is an effective algorithm to find out whether vertex *B* is reachable from vertex *A*, though it does not consider the usage of an edge between them, or the situations of substructures of a traffic graph. In paper [6], H. Gonzalez proposed an algorithm based on urban traffic graph to find out the efficient route between start and end points. This algorithm sets transportation data as references to calculate the route, but it is works only on current transportation dataset.

In this paper, a noble urban traffic navigation algorithm is introduced, which is based on graph mining and association rules mining. It has two stages to fulfill the target of data analysis when the user picks start and end vertices in the traffic graph. In the first stage, association rules mining algorithm is implemented to produce a set of frequent edges in the graph. After this, it gives the function to evaluate every single edge. In the second stage, graph mining method carries out to form frequent subgraph based on these edges according to the historical traffic network graph and current transportation dataset. In Evaluation and Results part, this algorithm shows that it can give a more efficient and effective route than exit ones.

## Methodology

Transportation network can be translated into graph data, which vertices are road segments and edges are road crosses. This translation keeps the traffic signals and road conditions more greatly than other methods. In figure 1, it is a part of an urban traffic network a), and it is translated in the way above c). Also, it can be translated by another method, which is that the vertices are road crossed and edges are road segments (c). It is clear that this translation loses some information in the original traffic network.




*a) Traffic Network* *b) Translation 1* *c) Translation 2*

**Figure 1. The mapping between traffic network and directed graph.**

The transportation dataset can be divided into two parts. The first part is historical records, nearest or furthest. And the second part is current records. In this paper, the algorithm is founded on such historical and current data. For the same urban traffic network, it has many snapshots in the dataset for different time stamp.

*Definition 1 (Traffic Graph Data)*. A traffic graph data is a directed graph, which is a 6-tuple, $G = \{V, E, \Sigma, L, D, \omega\}$. Here, $V$ is vertex set, $E$ is edge set, $\Sigma$ is label set of vertices and edges. $L$ is a mapping between $V$ and $E$ to $\Sigma$, which represents $V \rightarrow \Sigma$ and $E \rightarrow \Sigma$. $D$ is another mapping which represents the edges directions between two vertices, $D$: $v_1 \rightarrow v_2$. $\omega$ is a weighting function which calculates the traffic conditions affected by other elements.

$\omega$ is a set which contains several elements about environment, such as whether, rush hours or not, special days, safety, and others. By such set, a definition can be made to weight the vertices and edges in a traffic graph data. Different interests make different functions. This should be defined by users.

In this algorithm, we focus on frequent edges and frequent substructures in urban traffic graph. For this target, the algorithm can be divided into two stages. In the first stage, it calculates the frequent edges between start and end points in the graph. It discovers the frequent substructures above these frequent edges in the second stage. The growth of frequent substructures accords to the depth first method. When one of those substructures is large enough to cover the start and end vertices, it will contain the efficient route of this trip. After all, the algorithm deploys another depth first visiting to find out such route.

**Frequent Directed Edges Mining.** There are many graphs in an urban traffic dataset. It can be drawn from one urban traffic network at different timestamps. For all these induced graphs, the first thing to do is to give the beginning vertex $A$ and ending vertex $B$ of a trip. Then it begins to form the frequent directed edge set in the graphs according to the Edge Picking Algorithm (EPA) bellow.

When the beginning vertex $A$ and ending vertex $B$ are given, EPA analyzes current transportation records to collect related information. Then it launches to query the historical records to collect graphs in urban traffic dataset which have the nearest situation to current one, which named $G_h$. Afterwards, EPA carries out a frequent edges mining step to discover the most frequently used edges $e_f$ between $A$ and $B$. As the result, the frequent edge set $E_f$ is returned to the up layer of the algorithm to form the final route.

Here, the most frequently used edge $e_f$ is such edge. It is used mostly in the same situation like current. It considers not only transportation system effectives, but also other conditions like weather, rush hours, and so on. Different user should define different functions about thus procedure. When a trip is highly related about safety like chemical transportation, it should adjust the weight of an edge more leaning to safe. In the evaluation and results part, we will show an example about the adjustment of such function.

**Enlargement of Frequent Edges.** After the algorithm EPA returns the frequent edge set $E_f$, it evolves into the second step of mining.

The second stage is a graph mining algorithm indeed which is called Edge eXpands Algorithm (EXA).

In EXA, it first sorts the frequent edges in $E_f$ by descending order. Then, it picks out the most first edge in $E_f$ as current edge $e_c$. EXA queries frequent edges in $G_h$ to add it into $e_c$'s vertices to form a larger path or substructure. The query method is depth first visiting. After EXA adds edges to every edge in $E_f$, it checks whether these paths or substructures can joint together to cover vertices $A$ and $B$. If it does, EXA returns that path or substructure and ends running. If it does not, EXA adds new frequent edges in $G_f$ to those paths or substructures in $E_f$ until it satisfied the ending condition.

When EXA forms a path or a substructure that covers vertices $A$ and $B$, it visiting the traffic network graph by such path or a path in the substructure to check out whether it is an answer. After all, if there are more than one answers returned, EXA should give out the evaluation of all these paths to the user.

## Evaluation and Results

The evaluation environment is Window 7 Ultimate running on i7 CPU, which has 4GB RAM and a 160SSD hard disk. All primary code was writing in gcc 3.4.4 under CodeBlocks.

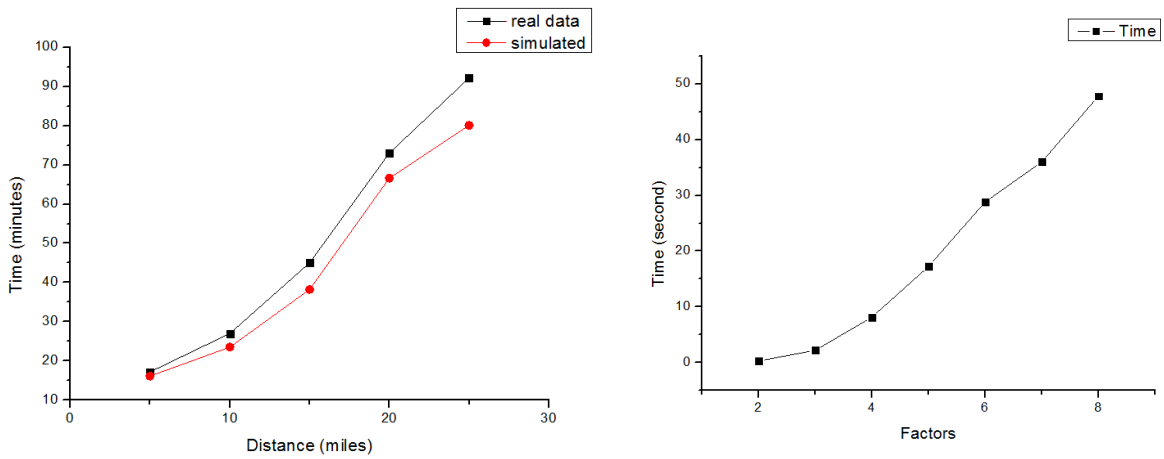
We used real road maps from areas of the United States in all of our experiments. The map we used is San Francisco Bay area (90 by 125 miles) with 175,343 nodes, and 223,606 edges. We simulated different traffic conditions using the Network-based Generator of Moving Objects by Thomas Brinkhoff [8], which is a well-known traffic simulator.

The most important parameter of this algorithm is the weight function. Here, we use the Standard Deviation to reduce its complexity. If there are $N$ parameters in this function, the equation (1) is the weight function in the algorithm.

$$\Delta = \frac{1}{N}\sum_{k=1}^{N} \mu_k p_k \quad (1)$$

Here, $\mu_k$ means the $k^{th}$ parameter's ratio, and $p_k$ means the value of the $k^{th}$ parameter.

The results of experiments are shown in figure 2. Figure 2 a) shows the average internal between our algorithm and realistic data, which is a real ride in the dataset. In this part, we chose 5 couples of vertices in the traffic network as beginning and ending vertices. Each couple has 10 tests to generate the average data. It is clear that our algorithm is much efficient than real ride. Figure 2 b) shows the effective from our weighting function. As it shown in figure, the more complicated, the more time consumed.




***a) Algorithm and Realistic Data*** ***b) The Effectiveness of Algorithm***

**Figure 2. Experimental Results**

This two-stage data mining based navigation algorithm can give an efficient route in a specified time, with considering the traffic impact factors. Unless other navigation algorithms, it combines historical dataset and current dataset together to avoid the efficient route just in theory.

## Conclusion

In this paper, we propose a two-stage algorithm to calculate the route for a vehicle in an urban traffic network. Unlike existing navigation algorithms, this algorithm based not only on the traffic network map, but also considers the transprotation datasets. According to the two parts of transportation dataset, named as historical dataset and current dataset, this algorithm can be divided into two stage to generate the route, which is EPA and EXA. It combines association rules mining and graph mining into an integrated one to simulate the most efficient route between beginning and edge vertices at current situation. Evaluation and results shows that this algorithm can discover such route based on historical dataset and save time in travel.

## Acknowledgements

This research was sponsored in part by the Creative and Developing Foundation of Harbin Institute of Technology under grant HIT.NSRIF.2010036 and part supported by the 863 Foundation of China under grant No. SS2012AA112310. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the P.R.C. Government. The P.R.C. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

**References**

[1] M. Garey, D. Johnson. Computers and Intractability: A Guide to the Theory of Np-completeness. W. H. Freeman and Company, 1979.

[2] G. Yang. The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, WA, USA, 2004:344–353.

[3] X. Yan, J. Han. gSpan: Graph-based Substructure Pattern Mining. IEEE International Conference on Data Mining (ICDM). Maebashi City, Japan, 2002:548–551.

[4] J. Huan, W. Wang, J. Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. IEEE International Conference on Data Mining (ICDM). Melbourne, Florida, USA, 2003:549–552

[5] S. Yang, X. Yan, B. Zong, A. Khan. Towards Effective Partition Management for Large Graphs. SIGMOD'12 (Proc. 2012 Int. Conf. on Management of Data), Jun 2012.

[6] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, John Paul Sondag. Adaptive Fastest Path Computation on a Road Network: A Traffic Mining Approach. Very Large Databases 2007 (VLDB ’07), September 23-28: 794-805.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp. 595–601.

[8] T. Brinkhoff. Network-based generator of moving objects. Technical report, IAPG, http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/.