Contemporary Engineering Sciences, Vol. 7, 2014, no. 9, 405 - 417
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.428

An Effective Approach for Mass Transit Routing and Optimization

Daning Chen

Department of Mathematics and Statistical Sciences
Jackson State University
Jackson, MS, 39217, USA

Xiaoping Shen and Arihant Karnawat

Department of Mathematics, Ohio University
Athens, OH, 45701, USA

Alagu Sundaram Muthiah

Microsoft India R&D Pvt. Ltd.
Hyderabad-500 046, AP, India

Saif Farooqui

Hexaware Technologies
Chennai-600 017, TN, India

Abstract

The need for efficient transport management based applications has witnessed a tremendous growth due to the explosive burst in population and the corresponding demand for transport services. In this paper we propose a novel Mass Transit Routing

and Optimization Tool (MTRO), which uses Data Mining (DM) techniques to extract information from the commuter database and assist the transport authorities in optimizing routes. Additionally, the application mines the optimal route for a given destination. Association Rule mining technique identifies the frequently used routes. The technique of Data Classification is employed to categorize the paths based on cost and time and suggest the optimal path to a destination. The proposed algorithms give better performance over traditional Frequent Pattern (FP) Growth based approaches with significant reduction in run time and memory usage.

Keywords: Transport management, mass transit routing, optimization, Data Mining (DM), Association Rule, Frequent Pattern (FP)

1. Introduction

In our times, there is an ever-growing demand for transport services. With cities expanding and populations growing, a large number of people need transport facilities to satisfy their needs. It is neither economical nor advisable for every individual to acquire independent means of transport, here, enters the necessity for mass transit systems. Having established the need for mass transit systems (buses, subways, etc.) we need to keep in mind that these services have to be based on existing infrastructure (roads, railways, bus stops, etc.), unless a complete remodeling of the system is done, which is not a viable solution. Thus, we need to optimize the usage of resources available.

Data mining is the non-trivial process of extraction of previously unknown, hidden and potentially useful information from large databases. Association Rule mining technique is used to identify relationships between routes that are frequently traveled together and hence identify the congestion in each path. The Classification and Regression Tree (CART) algorithm is used to segregate the good and bad routes from the various choices available to reach a given destination.

2. Preliminaries

In this section, we begin with recalling some notations and definitions which are related to the subsequent discussions. Curious readers can easily find references in [1]-[3]. A brief introduction to the database structure is given by the end of the section.

2.1 Definitions and notations

Definition 1. Congestion Index of a route R (CI_R). CI_R is a parameter used to measure the inability of a route R to meet the commuter demands, defined by [4]

$$CI_R = a_R / b_R, \quad (1)$$

where a_R is the number of people using the bus route R and b_R is the capacity of the buses operating in route R. We omit the subscript R when the distinction of routes is not important in the context.

Definition 2. Normalized Congestion Index (NCI). It is a measure to determine the feasibility of a route after a bus has been removed. It is calculated as the ratio of number of people traveling in the route by the capacity of the bus after removing a bus from the route.

Definition 3. Candidate Radius (CR). CR is defined as the radial distance from the source stop of the congested. The parameter is used to determine candidate routes which will undergo rerouting. Fig. 1 illustrates a sample bus network with n bus stops, where {Si | Si represents ith bus stop in the network, i = 1, 2, 3……n} and {Si Sj | Si Sj represents the sequential order of the bus stops. The bus has to travel from the Stop S_i to Stop S_j through the network, I,j = 1, 2, 3…..n.

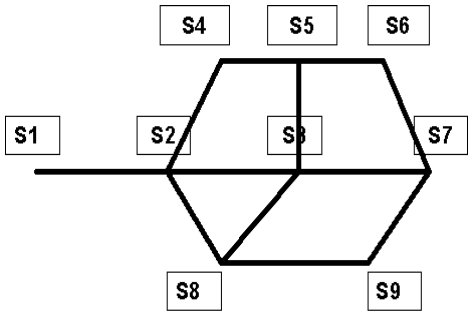

Fig. 1. Sample Bus network.

2.2 Database Schema

A brief outline of the database structure is presented here to help readers to understand how the tool uses the data to perform route optimization. Details about each bus stop are provided in Table 1. Paths taken by each bus route and the order of occurrence of the paths are maintained in Table 2. The number of buses plying in each route is recorded in Table 3.The journey details of every passenger are furnished in Table 4.

Table 1. Bus Stop

Attribute	**Description**
E_ID	Unique identifier to each bus stop.
C_X	X-coordinate of the bus stop.
C_Y	Y-coordinate of the bus stop.
E_NAME	Name of the stop.

Table 2. Bus Route

Attribute	Description
R_ID	Unique identifier to each bus route.
R_NO	Sequence number of the path in the route.
R_PATH	Path taken by the bus.

Table 3. Bus Route Count

Attribute	Description
R_ID	Unique identifier to each bus route.
COUNT	Number of buses plying for the route.

Table 4. Passenger

Attribute	Description
PASS_ID	Unique identifier for every passenger.
SRC	Passenger's source bus stop.
DEST	Passenger's destination bus stop.
R_ID	The bus route that the passenger traveled by.
DOJ	Date of journey.
TOJ	Time of journey.

3. Mtro Framework

In this section, we describe the MTRO tool and its associated algorithms. The MTRO Framework (Fig. 2) consists mainly of 5 components: (1) Association Rule Mining, (2) Congestion Analysis, (3) Rerouting, (4) Trend Prediction and (5) CART Classification.
The Association Rule Mining Component is further sub-divided into (i) FP-Tree Construction and (ii) Enhanced FP-Growth sub-components. The Rerouting component is made up of (i) Direct Rerouting and (ii) Relative Rerouting.

3.1 Association Rule Mining

The association rule mining technique comprises of two steps - Frequent Pattern Tree (FP-Tree) Construction and Enhanced FP-Growth Algorithm [5]-[9].

3.1.1 FP-Tree Construction Algorithm

The working of the algorithm is elucidated by using the following example which follows the bus network given in Fig. 1. The FP-Tree (See Fig 3.) is the output of the

FP-Tree Construction Algorithm with Table 5 as the input. Notice that S10 and S11 have the same items, which implies that there were 2 passengers/commuters starting from the same source and ending at the same destination.

3.1.2 Enhanced Frequent Pattern Growth Algorithm

The Enhanced FP Growth algorithm is used to generate the Association rules after the construction of the FP-Tree. The algorithm starts by initializing a temporary variable TEMP to null. Now a scan is done of the tree in a depth-first manner,
starting with the root node. The algorithm proceeds to the first child of root which is S2S3 and searches for it in the sub tree of the current TEMP node, and the counts for that are added. It is discovered that S2S3 occurs only once with count 4, the node along with its count is concatenated with TEMP (currently null), thus giving our first rule as just S2S3. The algorithm then moves onto to the child of S2S3

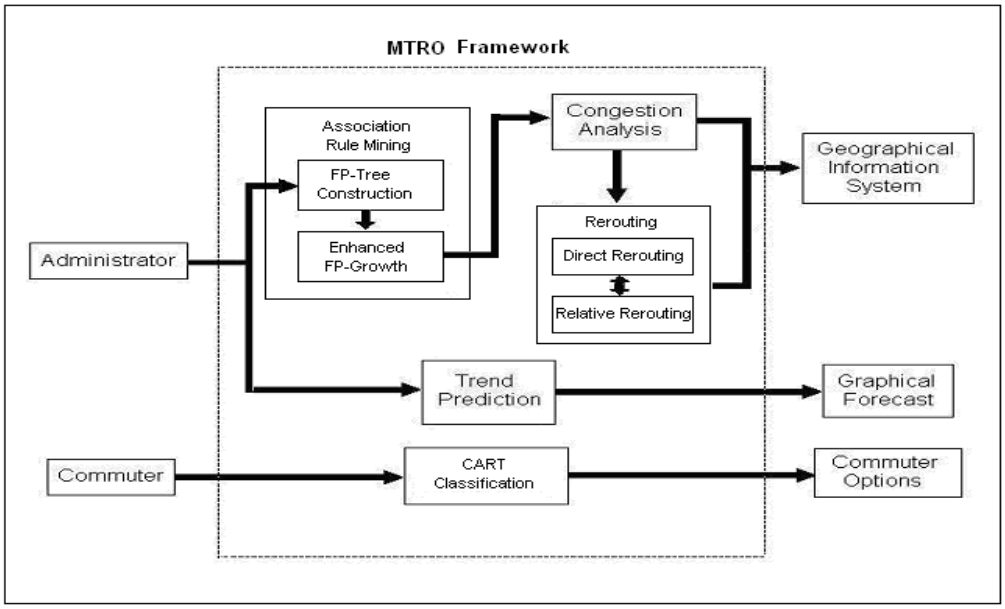

Fig. 2. MTRO Framework

which is the S3S7 node and searches for it in the sub tree marked. It is found that it occurs only once with count 4. So now S3S7 is concatenated with TEMP (S2S3:4) resulting in the rule S2S3, S3S7 along with the count of S3S7 which is 4. Next we come to the child of S3S7 which is the S7S9 node and search for it in the sub tree marked. We find it occurs once more with count 1. So the net count of S7S9 is 2 (1+1). Finally as before, S7S9 is concatenated with TEMP (S2S3, S2S7: 4) and then the rule formed is S2S3, S3S7, S7S9: 2 with the count of S7S9 which is 2. The rules generated by the Enhanced FP Growth algorithm are shown in Table 6.

Table 5. Sample Transaction Database DB

S.No.	Items	S. No.	Items
		6	S8S9, S9S7
1	S2S3, S3S7, S7S9	7	S8S2, S2S1
2	S1S2, S2S3, S3S7	8	S2S3, S3S7
3	S1S2, S2S3, S3S7, S7S9	9	S3S2, S2S1
4	S1S2, S2S8, S8S9	10	S7S3, S3S2, S2S1
5	S8S9, S9S7	11	S7S3, S3S2, S2S1

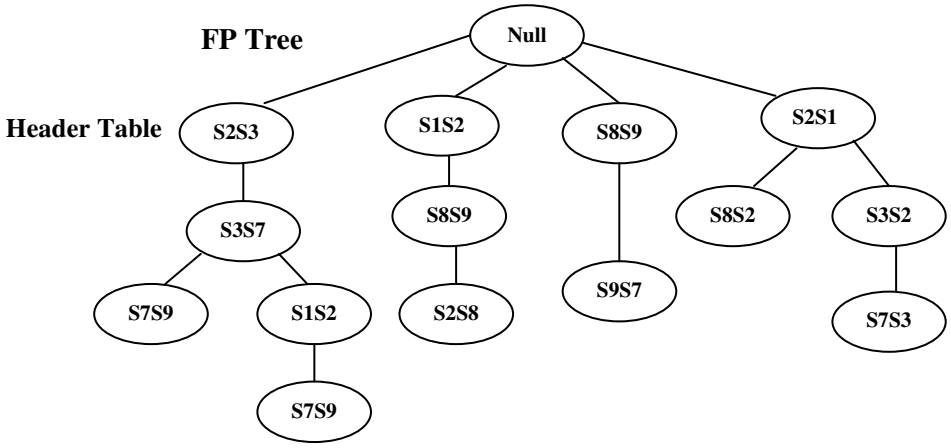

Fig. 3. FP-Tree Construction

3.2 Re-routing Algorithm

The re-routing algorithm is implemented after having identified the congested routes and their corresponding congestion indices and is used to optimize the transport network, by allowing redirection of buses from under-utilized routes to those which are heavily congested. This algorithm functions iteratively and continues to suggest redirections until it reaches a point where further redirection would cause congestion to occur at the previously under-utilized routes. At this point, the algorithm provides suggestions for new buses to be introduced to alleviate the situation. The algorithm is carried out in two phases.

3.2.1 Direct Rerouting

Arrange the routes in descending order based on congestion index. Consider the bus route with the highest congestion index (CI), as the route whose congestion has to be reduced. Identify the bus routes (if available) with the least congestion index i.e. routes with extra buses available. This is done by calculating the Normalized congestion index (NCI) checking if the NCI is still below one. Remove the buses obtained (i.e.) the buses with the least congestion index and employ them on the congested route. Recalculate the congestion indices of the routes operated upon. This is repeated till there is no non-congested bus identifiable.

3.2.2 Relative Rerouting

The congestion indexes of all the routes are recalculated and the routes arranged in descending order of congestion index. A flag is associated with all the routes, called 'Possible' and 'Not Possible'.

Algorithm Enhanced FP-Growth (EFP-Growth)
Input: FP-tree
Output: The complete set of frequent patterns.
Method: call EFP-growth (root, null)

```
Procedure EFP-Growth (ROOT Q, FPprefix α)
{
   for each item a_i in Q, in top down order
   {              // Mining multipath FP-tree
                  sum all count of a_i under Q into a_i.support
                  if a_i .support >= ξ
                  then
                  {
                     generate pattern β = αUa_i with support=a_i.support;
                     if a_i has child
                     then call EFP-Growth (a_i, β);
                  } }}
```

Table 6. Association Rules

Rules	Count	Rules	Count
S2S3	4	S8S9	3
S2S3, S3S7	4	S8S9, S9S7	2
S2S3, S3S7, S7S9	2	S2S1	4
S1S2, S2S3, S3S7	2	S8S2,S2S1	1
S1S2, S2S3, S3S7, S7S9	1	S3S2, S2S1	3
S1S2	3	S7S3, S3S2, S2S1	2
S1S2, S2S8, S8S9	1		

By default the flag is set to 'Possible'. Consider the route with the highest congestion index. Re-routing involves a new bus getting redirected from the start of the congested path, right through to the end, at which point it rejoins its original route. Less congested routes are identified within a given radius of the source stop of the congested path; these form the candidates for re-routing. Check the candidates to see if re-routing a bus from that route would lead to congestion on the route. If so, discard the route else keep it. Select the candidate with the least congestion index for re-routing. If no such candidate exists then set the "Not Possible" flag for the congested route. Recalculate the congestion indices of the routes operated upon and sort them. This is repeated till all congested routes have the "Not Possible" flag set, indicating that only a new bus can be introduced to improve the situation.

Algorithm. Direct Rerouting
Input: Congestion Index (ci) for all direct routes, computed using the association rules
Output: Direct Rerouted Routes
Method: call reroute_direct (route[ ], ci[ ])

```
Procedure reroute_direct (route[ ], ci[ ])
{   A:
    for each rte in route[i]
    {
                    if ci[i]>1
                    then add rte to RequiredSet
                    else
                    {
                        add rte to AvailableSet
                        compute nci for rte
                    }
    }
    if (RequiredSet != null) and (AvailableSet != null)
    then
    {
                    lrte = min(AvailableSet, nci)
                    grte = max(RequiredSet, ci)
                    reroute lrte to grte
                    compute ci for all i in route[i]
                    goto A
    }
}
```

3.3 CART Classification

We make use of classification technique to segregate the good and bad routes from the various choices available to reach a given destination. We make use of classification when the user specifies his criteria to be both cost and time, so as to strike an optimal balance between them. If the user only wants the route suggestion to be based on cost or time, then Djikstra's algorithm is applied. When classification is needed, we make use of a decision tree model, which is created using the CART (Classification And Regression Tress) algorithm. The decision tree made use of is given in Fig 4.

An initial pass is made through the database to determine the average cost and time values for that particular destination. The last two stages of the decision tree are used to compare the cost and time of a particular route with the respective average values. When checking for the time constraint, an additional check is made with the bus schedule database to ensure that the passenger waiting time does not exceed a pre-defined value. After classifying, the routes belonging to the best class are displayed, based on their optimality.

Algorithm Relative Routing
Input: Congestion Index (ci) for all routes, computed from reroute_direct
Output: Rerouted Routes
Method: all reroute_relative (route[ ], ci[ ])

```
Procedure reroute_relative (route[ ], ci[ ])
{
    A:
    for each rte in route[i]
    { set possibility = true }
    mrte = max(route, ci)
    src = source(mrte)
    dest = destination(mrte)

  /*Returns a set of potentially reroutable
     Candidate*/
 srcSet=call rerouteCdt(src,candidateRadius )
 destSet=call rerouteCdt(dest,candidateRadius )

 /*Finds the least congested route with the
  source in srcSet and destination in destSet.*/
    brte = findBestRoute( )
    Reroute mrte to brte
    compute ci for all i in route[i]
    if route[] is not empty
    then goto A }
```

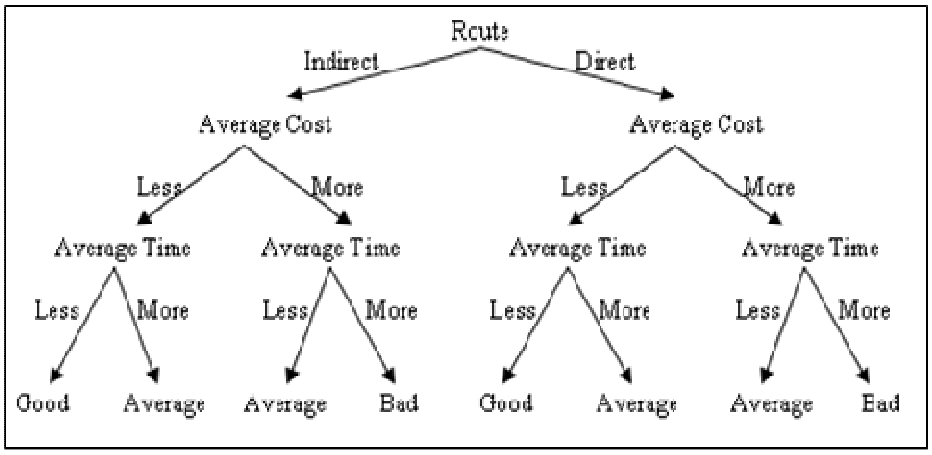

Fig. 4. CART Classification

4. Experimental Results

In this section we perfume simulations to evaluate the performance of the MTRO. The experimental results show that the proposed Enhanced FP Growth algorithm is more efficient than the traditional FP Growth and the QFP Growth algorithms.

4.1 Simulations.

In our simulations, Java programming language is used for implementing these algorithms. The experiments run on the same compiler, JDK 1.4 on a 3.0 GHz, Pentium IV processor equipped with a 512 MB Ram. The operating system used is Windows XP Professional. From the figure 5 and 6 we can comprehend that Enhanced FP Growth is more efficient than its counterpart.

In the Figures 5 and 6 we have compared the time and space complexity of the 3 algorithms. This is done with respect to the support threshold. The support (s) of an association rule is the ratio (in percent) of the records that contain XUY to the total number of records in the database, where X and Y belong to the Item set. Here we can clearly see that Enhanced FP growth has reduced the memory usage and runtime over its other counterparts.

4.2 Evaluation.

4.2.1 Comparing with FP and QFP. A direct comparison between the Enhanced FP-Growth algorithm and traditional Growth algorithm and the QFP algorithm revealed that the EFP algorithm is more efficient in terms of average run time. We organize the result in Table 7 below. It shows that the that Enhanced FP-Growth

algorithm runs 2.17 times faster than the traditional FP-Growth algorithm and 1.71 times faster than QFP-Growth under the above testing conditions. The efficiency is taken over an average number of runs. This analysis is done from the Fig. 5.

4.2.2. Comparing with QFP. To comparing the Enhanced FP-Growth algorithm with the QFP algorithm, we consider the relative difference by comparing both these methods with the traditional FP-Growth algorithm. This can be described numerical by using the following ratios:

$$R_1 = \left| \frac{T_{QFP} - T_{FP}}{T_{FP}} \right| \text{ and } R_2 = \left| \frac{T_{EFP} - T_{GA}}{T_{GA}} \right|, \quad (2)$$

where T_{QFP}, T_{EFP} and T_{FP} are run time for methods QFP, EFP and FP, respectively. Similarly, we define ratios to relate memory usage these methods

$$R_3 = \left| \frac{M_{QFP} - M_{FP}}{M_{FP}} \right| \text{ and } R_4 = \left| \frac{M_{EFP} - M_{FP}}{M_{FP}} \right| \quad (3)$$

We illustrate (2) and (3) in Fig 7, left and right panels, respectively.

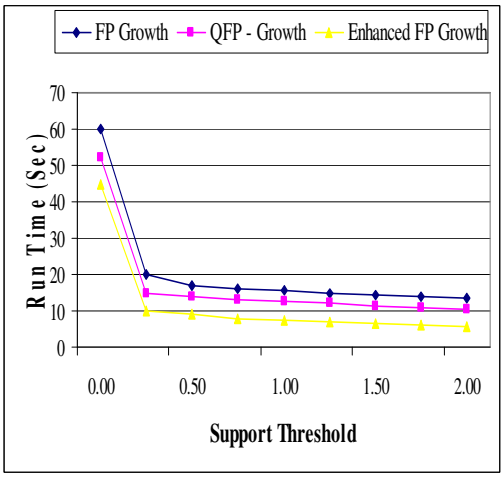

Fig. 5. Scalability of QFP-Growth With Respect To Support with Single Run

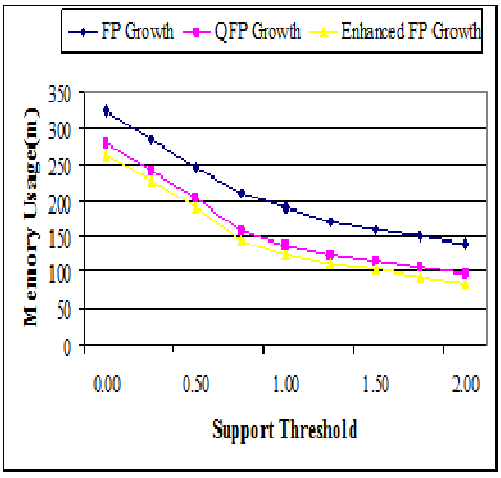

Fig. 6. Memory Comparison.

Table 7. Efficiency of Enhanced FP-Growth to its counterparts

Algorithm	**Efficiency of Enhanced FP-Growth (Average)**
FP-Growth	2.17
QFP-Growth	1.71

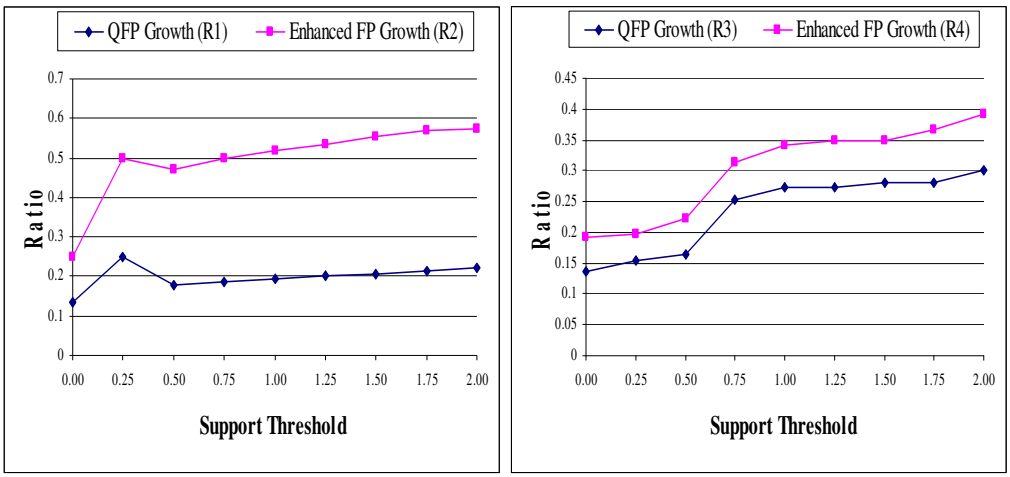

Fig. 7. Comparison between of EFP and QFP relative to FP-Growth Algorithm (memory usage).

5. Concluding Remarks

This tool was developed keeping in mind the growing needs of urban transport and its efficient management. The technique of association mining was apt for our requirements as it efficiently aided in recognizing the various routes adopted by the passengers and their consequent congestion indices. The tool is equipped with a user-friendly graphical interface, which allows the administrator of the tool to specify the bus network along with its inherent properties such as the distances between various stops, the paths traversed by the buses, etc. The tool also allows the administrator to customize several options such as the prediction coefficients, support counts, re-routing thresholds, etc. As an output the tool displays the congested areas and suggests possible re-routing. It also provides utilities to predict future congestion values and identifies seasonal trends and suggests optimal routes to a given destination based on cost and time factors. Experimental results show that it accurately detects the congested routes and offers good re-routing suggestions within the given parameters. The tool can be extended to support Global Positioning Systems (GPS) in buses to provide real time tracking which in turn would allow more accurate bus schedules. A prediction utility can also be constructed to forecast the future values of congestion index of a selected route over a user-specified time scale (E.g. monthly, yearly, etc). This prediction can be achieved by using non-linear prediction method or unsupervised learning techniques.

References

[1] D. H. Lee1, S.T. Jeng and P. Chandrasekar. Applying Data Mining Techniques for Traffic Incident Analysis, Journal of The Institution of Engineers, Singapore Vol. 44 (2004), 90-102.

[2] Intelligent Transportation Systems, IEEE Transactions on Vol. 2 (4), 2011, 1624 – 1639.

[3] C. Zheng, S. Chen, W. Wang, J. Lu, Using Principal Component Analysis to Solve a Class Imbalance Problem in Traffic Incident Detection, Mathematical Problems in Engineering, Vol. 2013 (2013), Article ID 524861, 8 pages.

[4] J. Han and M. Kamber. *Data Mining- Concepts & Techniques*. Morgan Kauffman Publishers, 2000.

[5] J. Han and Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases, Proc. of the 21nd International Conference on Very Large Databases, Zurich, Switzerland, 1997.

[6] Michael J. Demetsky. Development of Congestion Performance Measures In Virginia Transportation Research Council January 2003.

[7] N. Ramanujam, A. Karnawat, A. Sundaram and S. Farooqui. Classification and Association Rule Based Consultancy Minor Tool. Proc. of EISTA '04 at Florida, USA.

[8] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. Proc. of the Twentieth International Conference on Very Large Databases, Santiago, Chile, 1996.

Received: February 9, 2014