International Journal of Computer Science and Informatics

Volume 2 | Issue 4 Article 2

April 2013

Frequent Itemsets Used in Mining of Train Delays

D. Kishore Babu
BVCITS, Amalapuram, domalakishore@gmail.com

Y Naga Satish
BVCITS, Amalapuram, nagasatishyalla@gmail.com

G. L. N. V. S. Kumar
BVCITS, Amalapuram, kumar4248@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcsi

 Part of the Computer Engineering Commons, Information Security Commons, and the Systems and Communications Commons

Recommended Citation

Babu, D. Kishore; Satish, Y Naga; and Kumar, G. L. N. V. S. (2013) "Frequent Itemsets Used in Mining of Train Delays," *International Journal of Computer Science and Informatics*: Vol. 2 : Iss. 4 , Article 2.
DOI: 10.47893/IJCSI.2013.1097
Available at: https://www.interscience.in/ijcsi/vol2/iss4/2

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Frequent Itemsets Used in Mining of Train Delays

D. Kishore Babu, Y Naga Satish & G. L. N. V. S. Kumar

BVCITS, Amalapuram

E-mail : domalakishore@gmail.com, nagasatishyalla@gmail.com, kumar4248@gmail.com

Abstract - The Indian railway network has a high traffic density with Vijayawada as its gravity center. The star-shape of the network implies heavily loaded bifurcations in which knock-on delays are likely to occur. Knock-on delays should be minimized to improve the total punctuality in the network. Based on experience, the most critical junctions in the traffic flow are known, but others might be hidden. To reveal the hidden patterns of trains passing delays to each other, we study, adapt and apply the state-of-the-art techniques for mining frequent episodes to this specific problem.

Key words - *Train delays, Data Analysis, Pattern mining, frequent itemsets, Hidden trains.*

I. INTRODUCTION

The Indian railway network, as shown in Figure 1, is very complex because of the numerous bifurcations and stations at relatively short distances. It belongs to the group of the most dense railway networks. Moreover, its star-shaped structure creates a huge bottleneck in its center, Vijayawada, as approximately 40% of the daily Indian trains pass through the Vijayawada North-South junction.

During the past three years, the punctuality of the Indian trains has gradually decreased towards a worrisome level. Meanwhile, the number of passengers, and therefore also the number of trains necessary to transport those passengers, has increased. Even though the infrastructure capacity is also slightly increasing by doubling the number of tracks on the main lines around Vijayawada, the punctuality is still decreasing. To solve the decreasing punctuality problem, its main causes should be discovered, but because of the complexity of the network, it is hard to trace their true origin. It may happen that a structural delay in a particular part of the network seems to be caused by busy traffic, although in reality this might be caused by a traffic operator in a seemingly unrelated place, who makes a bad decision every day, unaware of the consequences of his decision

We study the application of data mining techniques in order to discover related train delays in this data. Whereas, Flier et al. [1] try to discover patterns underlying dependencies of the delays, this paper considers the patterns in the delays themselves but we search for patterns in the delays themselves. Mirabadi and Sharafian [5] use association mining to analyze the causes in accident data sets, whereas we consider the so called frequent pattern mining methods. Prototypical examples of these methods can be found in a supermarket retail setting, where the marketer is interested in all sets of products being bought together by customers. A well known result being discovered by a retail store in the early days of frequent pattern mining is that “70% of all customers who buy computer also buy software”. Such patterns could then be used for targeted advertising, product placement, or other cross-selling studies. In general, the identification of sets of items, products, symptoms, characteristics, and so forth that often occur together in the given database can be seen as one of the most basic tasks in Data Mining [7, 3].

Here we use a database of containing the times of trains passing through characteristic points in the railway network is being used. In order to discover hidden patterns of trains passing delays to each other, our first goal is to find frequently occurring sets of train delays. More specifically, we try to find all delays that frequently occur within a certain time window, counted over several days or months of data [5]. In this study, we take into account the train (departure) delays larger than or equal to 3 or 6 minutes. For example, we consider patterns such as: Trains A, B, and C, with C departing before A and B, are often delayed at a specific location, approximately at the same time. Computing such patterns, however, is intractable in practice [4] as the number of such potentially interesting patterns grows exponentially with the

number of trains. Fortunately, efficient pattern mining techniques have recently been developed, making the discovery of such patterns possible. A remaining challenge is still to distinguish the interesting patterns from the irrelevant ones.

Fig. 1 : The Indian Railway Network

Typically a frequent pattern mining algorithm will find an enormous amount of patterns amongst which many can be ruled out by irrelevance. For example, two local trains which have no common characteristic points in their route could, however appear as a pattern if they are both frequently delayed, and their common occurrence can be explained already by statistical independence. In the next Section, we explain the studied pattern mining techniques. In Section 3, we discuss the collected data at characteristic points and report on preliminary experiments, showing promising results, and we conclude the paper with suggestions for future work

II. PATTERN MINING

2.1 Itemsets

The simplest possible patterns are itemsets [7]. Typically, we look for items (or events) that often occur together, where the user, by setting a frequency threshold, decides what is meant by 'often'.

Formally, let $\mathbf{I} = \{x_1, x_2, \ldots x_n\}$ be a set of items. A set $X \subseteq \mathbf{I}$ is called an *itemset*. The database D consists of a set of transactions, where each transaction is of the form (t, X) where t is a unique transaction identifier, and X is an itemset. itemset.Given a database D, the *support* of an item set Y in D is defined as the number of transactions in D that contain Y , or

$$Sup(Y, D) = |\{t | (t, X) \in D \text{ and } Y \subseteq X\}|.$$

Y is said to be frequent in D if $sup(Y, D) \geq$ minsup, where minsup is a user defined minimum support threshold (often referred to as the frequency threshold).Support has a very interesting property - it is *downward-closed*. This means that the support of an itemset is always smaller than or equal to the support of any of its subsets. This observation is crucial for many frequent pattern algorithms, and can also be used to reduce the output size by leaving out everything that can be deduced from the patterns left in the output.

A typical transaction database can be seen in Table 1

TID	Items Bought
1	{monitor,keyboard,mouse}
2	{mobile,battery,charger}
3	{ computer, software, mouse}
4	{Milk, Diapers, Bread, Butter}

As can be seen in Table 2 the database does not consist of transactions of this type

In order to mine frequent itemsets in the traditional way, the database would need to be transformed. Therefore a transaction database can be created, in which each transaction consists of train IDs of trains that were late within a given period of time. Table 3 shows a part of the data from Table 2 transformed into a transaction database with each transaction consisting of trains delayed within a period of five minutes. Each transaction represents one such period. Mining frequent itemsets would then result in obtaining sets of train IDs that are often late 'together'.

Time stamp	Train ID
06:05 15/08/2011	A
06:07 15/08/2011	B
06:09 15/08/2011	C
06:35 15/08/2011	D

Table 2: A simplified example of a train delay database

In the example given in Table 3, assuming a support threshold of 2, the frequent itemsets are

{A}, {B}, {C}, {A, B} and {B, C}

TID	Delayed trains
06:00 15/02/2011 - 06:04 15/08/2011	{ }
06:01 15/02/2011- 06:05 15/08/2011	{A}
06:02 15/02/2011 - 06:06 15/08/2011	{A}
06:03 15/02/2011 - 06:07 15/08/2011	{A,B}

06:04 15/02/2011 - 06:08 15/08/2011	{A,B}
06:05 15/02/2011 - 06:09 15/08/2011	{A,B,C}
06:06 15/02/2011 - 06:10 15/08/2011	{B,C}
06:07 15/02/2011 - 06:11 15/08/2011	{B,C}
06:08 15/02/2011 - 06:12 15/08/2011	{C}
06:09 15/02/2011 - 06:13 15/08/2011	{C}
06:10 15/02/2011 - 06:14 15/08 /2011	{ }

Table 3: Transformed version of an extract of the data in Table 2.

Table 3 illustrates why the frequent itemset method is not the most intuitive to tackle our problem. First of all, it requires a lot of preprocessing work in order to transform the data into the necessary format. Second, the transformation results in a dataset full of redundant information, as there are many empty or identical transactions. This problem is further multiplied when we refine our time units to seconds instead of minutes. Finally, as will be shown later, itemsets are quite limited and other methods allow us to find much better patterns.

2.2. Temporal pattern

An episode is a temporal pattern that can be represented as a directed acyclic graph, or DAG. In such a graph, each node represents an event (an item, or a symbol), and each directed edge from event x to event y implies that x must take place before y. Clearly, if such a graph contained cycles, this would be contradictory, and could never occur in a database. Note that both itemsets and sequences can be represented as DAGs. An itemset is simply a DAG with no edges (events can then occur in any order), and a sequence is a DAG where the events are fullyordere(for example, a sequence s_1 s_2 … s_k corresponds to graph ($s_1 \rightarrow s_2 \rightarrow \cdots s_k$) .However, we can now find more general patterns, such as the one given in Figure 2. The pattern depicted here tells us that *A* always occurs before *B* and *C*, while *B* and *C* both occur before *D*, but the order in which *B* and *C* occur may vary.

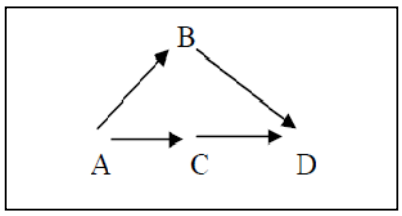

Fig. 2.A: General Temporal Pattern

2.3. Closed Pattern

Another problem that we have already touched upon is the size of the output. Often, much of the output can be left out, as a lot of patterns can be inferred from a certain smaller set of patterns. We have already mentioned that for each discovered frequent pattern (in our case, episode), we also know that all its sub patterns must be frequent. However, should we leave out all these sub episodes, the only thing we would know about them is that they are frequent, but we would be unable to tell how frequent. If we wish to rank episodes, and we do, we cannot remove any information about the frequency from the output.

Another way to reduce output is to generate only closed patterns [7]. In general, a pattern is considered closed, if it has no super pattern with the same support. This holds for episodes, too.

As an example, consider a sequence of delayed trains ABC X Y Z ABC. Assume the time stamps to be consecutive. Given a sliding window of size 3 minutes, and a support threshold of 2, we find that the episode (A→B→C), meaning that train A is delayed before B, and B before C, has frequency 2, but so do all its sub episodes of size 3, such as(A→ B, C), (A, B→ C) or (A, B, C).These episodes can thus safely be left out of the output, without any loss of information.Thus, if episode (A→B) is in the output, and episode (A, B) is not, we can safely conclude that the support of episode (A, B) is equal to the support of episode (A→B) Furthermore, we can conclude that if these two trains are both late, then A will always depart/arrive first. If, however, episode (A, B) can be found in the output, and neither (A→B) nor (B→A) are frequent, we can conclude that these two trains are often late together, but not necessarily in any particular order. If both (A, B) and (A→B) are found in the output, then the support of (A, B) must be higher than the support of (A→B), and we can conclude that the two trains are often late together, and A mostly arrives/departs earlier than B

III. EXPERIMENTS

In our experiments we have used the latest implementation of an algorithm, CloseEpi, for generating closed episodes, as described in [8].

3.1. Data Pre-processing

If we look at all data in the database as one long sequence of late trains coupled with time stamps, we will find patterns consisting of trains that never even cross paths.

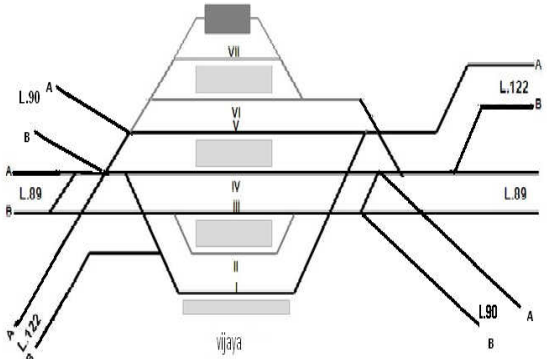

Fig. 3: The schematic station layout of Vijayawada

To avoid this, we generate one long sequence for each spatial reference point. In this way, we find trains that are late at approximately the same time, in the same place.

3.2 Example

To test the CloseEpi algorithm the decision was made to focus on delays at departure of 3 or more minutes and 6 or more minutes. These choices relate respectively to the blocking time and the official threshold for a train in delay.

For our experiments, we have chosen a window size of 30 minutes (or 1800 seconds). Although the support of a pattern does not immediately translate into the number of days in which it occurs, this can be easily estimated or even simply counted in the original dataset. More specifically, the lower bound of the number of days the pattern occurs is the support of the pattern divided by 1800, rounding the number upwards, and the upper bound is given by the minimum of the upper bounds of all its sub-patterns.

We tested the algorithm on the data collected in Vijayawada, a medium-sized station in the south-north of India. Vijayawada was chosen as it has an intelligible infrastructure, as shown on Figure 3. The total number of trains leaving Vijayawada station in the month of August 2011 is 241. There are 396 trains with a departure delay at Vijayawada of more than 3 minutes and 180 trains have a delay at departure which is equal or larger than 6 minutes. The delays are mainly situated during peak hours. Because the number of trains with a delay passing through Vijayawada is relatively small, the output can be manually evaluated. The two lines intersecting at Vijayawada are: line 89 connecting Rajahmundry with Tenali and line 122 connecting Repalle-tenali–Gnt-Hyb. The station layout of Vijayawada (Figure 3) line 89 is situated horizontally on the scheme and line 122 goes diagonally from lower left corner to the upper right corner. This intersection creates potential conflict situations which add to the station's complexity. Moreover, the station must also handle many connections, which can also cause the transmission of delays.

The trains passing through Vijayawada are categorized as local trains (numbered as the 1 series and the 18 series), a city rail (22 series) going to and coming from Vijayawada, an intercity connection (23 series) with fewer stops than a city rail or local train, and the peak hour trains (89 series).

The output of the ClosEpi algorithm is a rough text file of closed episodes with a support larger than the predefined threshold. An episode is represented by a graph of size (n, k) where n is the number of nodes and k is the number of edges. Note that a graph of size (n, 0) is an itemset. We aimed to discover the top 20 episodes of size 1 and 2, and the top 5 episodes of size 3 and 4, so we varied the support threshold accordingly. In Tables 5–7 some of the episodes which were detected in the top 20 most frequently appearing patterns are listed. For example, the local train no. 67239 from Vijayawada to Guntur is discovered as being 3 or more minutes late at departure on 15 days, and 6 or more minutes on 8 days in the month of January 2011.

Train ID	Route	Sup Delay≥3'	pport Delay≥6' ≥≥66'
67239	Viujayawada–Guntur	27000	14400
12764	Chirala–Vijayawada	28800	18000
4266	Chirala – Tenali	27000	14400
565	Repalle-tenali–Gnt-Hyb	2520	12600

Table 5: Episodes of size (1, 0) representing the delay at departure in station Vijayawada during evening peak hour (16h – 19h) for January 2011.

Fig. 4 : Station layout of Vijayawada

A paired pattern can be a graph of size (2, 0), meaning the trains appear together but without a specific order, or of size (2, 1), where there is an order of appearance for the two trains. For example, train no. 67239 and train no.12764 appear together as being 3 or more minutes late on at least 9 days and at most on 15 days in January 2011. The pattern trains no.12764 and 67239 have a delay at departure of 3 or more minutes, and train no. 12764 leaves before 67239 appears on at least 8 days and at most 15 days in January 2011

Among the top 20 patterns with pairs of trains (Table 6), it can be noticed that the pattern 67239→ 565 was only discovered in the search for 6 or more Minutes delay at departure. This means that the pattern will also appear while searching for 3 or more minutes of delay at departure but the support of this pattern is not high enough to appear in the top 20 output. The patterns which include lots of information are to be found in the output of episodes of size 3 and up, as can be seen in Tables 6 and 7. But to discover the episodes of sizes (3, k) and (4, k) the threshold had to be lowered to 5500 which corresponds to a minimal appearance of the pattern on 4 days. The question remains if this really is an interesting pattern.

In the example the peak-hour train no. 12764 often departs from the station with a delay of 3 minutes with a support of 28800 and a support of 18000 for a delay of 6 minutes (see Table 5). In real-time the peak-hour train no. 4266 follows train no. 12764 on the same trajectory, 4 minutes later. This can also be detected by looking at the occupation of the tracks in Figure 4. It is, therefore, obvious that whenever no. 12764 has a delay, the 4266 will also have a delay. Trains nos. 67239 and 565 both offer a connection to nos. 12764 and 4266. So, if train 12764 has a delay, it will be transmitted to trains 67239 and 565. This is also stated in Table 7, which shows an episode of size four, found by the ClosEpi algorithm, where trains no. 12764, 67239, 4266, and 565 are all late at departure and 12764 departs before the other three trains.

Train id	Relation	Train id	Delay ≥3'	Delay≥6'
67239		12764	15079	-
67239	←	12764	13557	-
67239		4266	18341	-
67239	←	4266	12995	-
67239		565	18828	8888
67239	→	565	-	5327
12764	→	4266	18608	9506
12764		565	18410	10391
12764	→	565	16838	8819
12764		565	20580	10608
12764	→	565	13325	5078
12764	←	565	-	5530

Table 6: Episodes of size (2, k) representing the delay at departure in station Vijayawada during evening peak hour (16h – 19h) for January 2011

Looking at the data for February 2011 (not included here) the pattern described in Table 7 is discovered for 3 or more minutes of delay with a support of 12126. In the cases of 6 or more minutes delay the pattern is discovered under the stronger form 12764 → 4266 →565 → 67239 with a support of 4604, meaning that if these trains have a delay at departure of 6 or more minutes, peak hour train no. 12769 departs before no. 4266, which leaves before no. 565, which in turn leaves before the local train no. 67239.

Episode			Support	
Train ID	Relation	Train ID	Delay≥3'	Delay ≥ 6'
12764	→	67239	10024	6104
	→	4266		
	→	565		

Table 7: Episode of size (4, k) representing the delay at departure in station Vijayawada during evening peak hour (16h – 19h) for January 2011

IV. CONCLUSION

There are still many opportunities for improvement, however. As we have studied the possibility of applying state-of-the-art pattern mining techniques to discover knock-on train delays in the Indian railway network using a database of data ,containing the times of trains passing through characteristic points in the network. Our experiments show that the ClosEpi algorithm is useful for detecting interesting patterns in the database. For example a good visualization of the discovered patterns would certainly help in identifying the most interesting patterns in the data more easily. Also, next to the support measure, other interestingness measures could also be considered. Selecting patterns solely based on the support measure still hides a lot of potentially interesting patterns, which could be found using other criteria. In order to avoid finding too many patterns consisting of trains that never even cross paths, we only considered trains passing in a single spatial reference point. As a result, we can not discover knock-on delays over the whole network. In order to tackle

this problem, the notion of a pattern needs to be redefined, but also the interestingness measures or other data pre-processing techniques need to be investigated.

REFERENCES

[1] Flier, H., Gelashivili, R., Graffagnino, T., and Nunkesser, M., "Mining Railway Delay Dependencies in Large-Scale Real-World Delay Data", Robust and Online Large- Scale Optimization, Lecture Notes in Computer Science, vol. 5868, 354–36, 2009.

[2] Agrawal,R.and Srikant,R.,"Mining sequential patterns", Proc. of the 11th International Conference on Data Engineering, vol. 0, 3–14, 1995.

[3] Goethe's,B.,"Frequent Set Mining", The Data Mining and Knowledge Discovery Handbook, chap. 17, 377–397, Springer, 2005.

[4] Gunopulis, D., Khardon, R., Labbuka, H., Saluja, S., Toivonen, H., and Sharma, R.S., "Discovering all most specific sentences", ACM Transactions on Database Systems, vol. 28(2), pp. 140–174, 2003.

[5] Mirabadi, A. and Sharifian, S., "Application of Association rules in Iranian Railways (RAI) accident data analysis", Safety Science, vol. 48, 1427–1435, 2010.

[6] Mannila, H., Toivonen, H., and Verkamo, A.I., "Discovery of Frequent Episodes in Event Sequences", Data Mining and Knowledge Discovery, vol. 1, 259–298, 1997.

[7] Tan, P.-N., Steinbach, M., and Kumar. Introduction to Data Mining, Pearson Addison Wesley, 2006.

[8] Tatti, N., and Cule, B., "Mining Closed Strict Episodes", Proc. of the IEEE International Conference on Data Mining, 2010.

[9] Wang, J. T.-L., Chirn, G.-W., Marr, T.G., Shapiro, B., Shasha, D., and Zhang, K., "Combinatorial pattern discovery for scientific data: some preliminary results", ACM SIGMOD Record, vol. 23, 115–125, 1994.

[10] Data Mining Concepts and Techniques - J.Han & amp; M.Kamber. ptpg_sit_2005

◈◈◈