Interactive Mining of Frequent Itemsets over Arbitrary Time Intervals in a Data Stream

Ming-Yen Lin[1] **Sue-Chen Hsueh[2]** **Sheng-Kun Hwang[1]**

[1]Department of Information Engineering and Computer Science, Feng Chia University, Taiwan
[2]Department of Information Management, Chaoyang University of Technology, Taiwan

linmy@fcu.edu.tw schsueh@cyut.edu.tw m9305966@fcu.edu.tw

Abstract

Mining frequent patterns in a data stream is very challenging for the high complexity of managing patterns with bounded memory against the unbounded data. While many approaches assume a fixed support threshold, a changeable threshold is more realistic, considering the rapid updating of the streaming transactions in practice. Additionally, mining of itemsets over various time granularities rather than over the entire stream may provide more flexibility for many applications. Therefore, we propose a interactive mechanism to perform the mining of frequent itemsets over arbitrary time intervals in the data stream, allowing a changeable support threshold. A synopsis vector having tilted-time tables is devised for maintaining statistics of past transactions for support computation over user-specified time periods. The extensive experiments over various parameter settings demonstrate that our approach is efficient and capable of mining frequent itemsets in the data stream interactively, with variable support thresholds.

Keywords: data stream, frequent pattern, interactive mining, tilted synopsis vector, time-granularity

1 Introduction

Data stream is a new data model with many applications that attracts many researchers to investigate efficient and effective mechanisms in recent years (Babcock 2002). Many distributed, data-intensive applications, like telecommunications, web services, and sensor networks, continuously generate an unbounded sequence of data items at a high rate in real time nowadays. These transient data streams cannot be modelled as persistent relations so that traditional database management systems are becoming inadequate in supporting the functionalities of modelling this new class of data. Conventional approaches for information queries and knowledge discovery, such as association rule mining, now demand new mechanisms to handle the rapid and time-changing data streams.

The discovery of frequent items and frequent itemsets has been studied extensively in the data mining community, with many algorithms proposed and implemented. However, the unbounded nature of data streams disallows the holding of the entire stream in the memory, and often incurs a high call-back cost even if the past data can be stored in external media. Stream mining algorithms would generally be restricted to scan the data items only once and can present merely approximate results rather than accurate results because some data items will be inevitably discarded. The 'one-pass' constraint, however, inhibits the direct application of traditional mining algorithms on data streams. Overcoming the difficulties of frequent pattern mining in data streams hence become a very challenging research issue.

The mining of frequent items (Charikar 2002) or frequent itemsets in a data stream has been addressed recently (Jiang 2006). Three models including landmark window (Manku 2002), sliding window (Chang 2005, Chi 2004) and time-fading window (damped window) (Giannella 2002) are employed to characterize the time span of the transactions in the mining. The entire history of the data stream is considered in the landmark window. The weights of past transactions are decreased to emphasize the importance of recent history in the time-fading window. In sliding window model, only transactions in the latest window (a fixed period of time span) are considered.

Many data stream association mining algorithms are developed (Chen 2002, Li 2006). The CountSketch algorithm (Charikar 2002) finds frequent items in a stream of items by estimating the item-frequency. A 'Group Testing' algorithm using a small space structure (Cormode 2003) and a hash-based algorithm are presented for finding frequent items in the data stream (Jin 2003). The Buffer-Trie-SetGen is used to mine frequent itemsets in a transactional data stream in the algorithm (Manku 2002). The (itemset, count, possible_missed_count) triplets are kept in an on-disk trie with batched updating of incoming transactions. The DSM-FI algorithm (Li 2004) uses a FP-tree (Han 1999) like forest and estimated supports to mine frequent itemsets in a landmark window model. The estWin (Chang 2005) is a sliding window algorithm which finds recently frequent itemsets adaptively. In addition, the Moment algorithm (Chi 2004) employs a 'closed enumeration tree' for fast discovery of closed frequent itemsets in a data stream. The FP-stream algorithm (Giannella 2002) incrementally maintains tilted-time windows for frequent itemsets at multiple time

granularities and effectively discovers frequent patterns. Specifically, the FP-stream may discover patterns in a specified time period, not necessary the entire time period.

Note that the above approaches, including FP-stream (Giannella 2002), for mining frequent itemsets over data streams accept only one minimum support in the mining. The minimum support cannot be changed during the mining for these approaches. For example, if the minimum support is changed to a smaller value, these algorithms cannot obtain the statistics of the discarded transactions so that the result is incorrect. Such a mining with only one unchangeable minimum support is called *fixed support mining*. In reality, the minimum support is not a fixed value for the entire stream of transactions. The user may specify a threshold in the beginning, adjust the threshold after evaluating the discovered result, or change the threshold after a period of time after receiving volumes of transactions. The minimum support threshold therefore should be variable to suit the interactive need of the user. In contrast to frequent itemset mining with a fixed support, the mining with respect to a changeable support is referred to as *variable support mining or interactive mining*. The algorithm in (Lin 2006) addressed this problem but the method can be used only in the landmark window spanning over the entire period. Although online association rule mining (Hidber 1999) and interactive mining (Lin 2004) may have changeable support thresholds, both algorithms are inapplicable to the stream data because a scanning of entire transactions is required.

In this paper, we formulate the problem of interactively mining frequent patterns in a data stream with respect to arbitrary time period, and with variable support threshold. An algorithm called VSMTP (Variable Support Mining of Time-sensitive Patterns) is proposed to support various mining requirements. For example, discovering the frequent itemsets in the period from t1 to t2, answering the periods when itemset (a, b) is frequent, giving alerts if the support of certain itemset changes dramatically in the period from t3 to t4, and so on. Such queries are named time-sensitive queries. In addition, the specified minimum support can be any value, instead of a fixed value.

The VSMTP algorithm uses a compact structure to store the summary information of stream transactions across time so that time-sensitive queries can be answered. The summary structure, called tilted synopsis vector, is designed to approximate past transactions with a flexible similarity threshold. To speed up the mining of frequent queries in common discovery concerning patterns about the entire history, a compact structure is used to maintain the set of potential frequent itemsets. The comprehensive experiments conducted show that VSMTP is highly efficient and linearly scalable.

The rest of the paper is organized as follows. Section 2 gives a formal description of the variable support mining problem. Section 3 describes the proposed VSMTP algorithm in detail. In Section 4, the results of comprehensive experiments are presented. Finally, Section 5 concludes our study.

2 Problem Statement

Let $\Psi = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ be a set of literals, called items. An itemset e having *k* items, represented by $(\beta_1, \beta_2, \ldots, \beta_k)$, is called a *k*-itemset. The length of a *k*-itemset *e*, written as $|e|$, is *k*. Without loss of generality, the items in an itemset are assumed to be in lexicographic order. A data stream $DS = \{\tau_1, \tau_2, \ldots, \tau_c, \ldots\}$ is an infinite sequence of incoming transactions, where each transaction τ_i is an itemset associated with a unique transaction identifier. Let τ_c be the latest incoming transaction, called current transaction. Without loss of generality, a number of transactions are processed in a batch. Let $tt_0, tt_1, \ldots, tt_n$ be successive non-overlapping time windows each representing a batch in the data stream, where tt_0 is the oldest batch and tt_n contains current transaction. The number of transactions in tt_i is called the window size of tt_i, denoted by $|tt_i|$. The data stream model is the same as previous data stream models: some (older) transactions might be discarded in the mining process because the memory cannot hold the infinite number of transactions.

A transaction τ_i contains an itemset *e* if $e \subseteq \tau_i$. The support of an itemset *e* in a specified time interval t1 to t2, denoted by $sup_{t1:t2}(e)$, is the number of transactions containing *e* divided by the number of transactions in DS from time t1 to t2. The user specified a minimum support threshold $ms \in (0, 1]$ in the beginning of the data stream. At any point of time, along with the incoming of transactions, the user may change the minimum support threshold so that the thresholds form a series of minimum supports, named *support sequence*. Let ms_c, called current minimum support, be the minimum support when we saw transaction τ_c. An itemset *e* is called a frequent itemset in time period t1 to t2 if $sup_{t1:t2}(e) \geq ms_c$. Time t1 and t2 can be mapped to the time window tt_α and tt_β, respectively. That is, itemset *e* is frequent in period t1 to t2 if its approximate frequency, namely support count, is greater than or equal to $ms_c*(|tt_\alpha|+|tt_{\alpha+1}+\ldots+|tt_\beta|)$. The objective is to determine the frequent itemsets for any specified time period, with respect to any specified minimum support.

The problem statement relaxes the restrictions of common frequent pattern discovery in a data stream. The user can specify any time period, instead of the entire history of data streams. Moreover, the specified minimum support can be changed anytime. Although FP-stream (Giannella 2002) may find out frequent patterns at multiple time granularities, the support threshold cannot be changed. In contrast, at any point of time, a new minimum support can be specified for our problem definition.

3 VSMTP: Variable Support Mining of Time-sensitive Patterns in Data Streams

In this section, we describe the proposed algorithm, called VSMTP (**V**ariable **S**upport **M**ining of **T**ime-sensitive **P**atterns), for variable support mining of frequent itemsets in a data stream with respect to arbitrary time period.

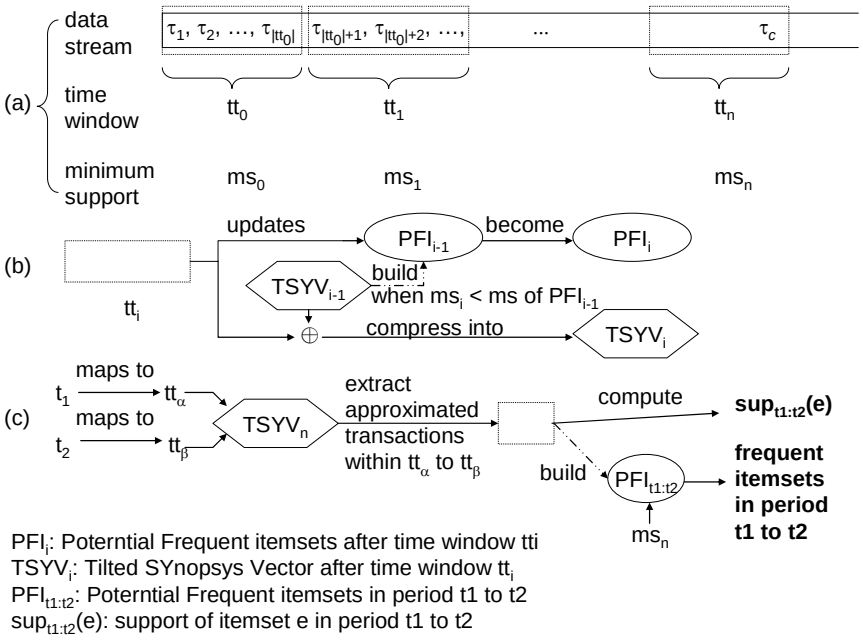

Figure 1: The overall concept of the VSMTP algorithm (a) transactions in the time window and the support sequence (b) the two operations on a time window tti (c) compute the support of itemset *e* and frequent itemsets in a period t1 to t2.

Section 3.1 gives an overview of the VSMTP algorithm. The structures used in VSMTP are delineated in Section 3.2. Section 3.3 presents the VSMTP algorithm in detail.

3.1 An overview of the VSMTP Algorithm

An algorithm called VSMDS (Lin 2006) was proposed for mining frequent itemsets over data streams with changeable support threshold. However, VSMDS may answer queries for the up-to-date patterns, from the beginning of the stream, but not the patterns within arbitrary time interval. Since the patterns are meaningful with respect to certain time interval. This kind of patterns is referred to as time-sensitive patterns. The framework of VSMDS is applied in the proposed algorithm, called VSMTP, to mine time-sensitive patterns with approximate support guarantee. A tilted synopsis vector is constructed for maintaining the statistics of past transactions in each period. Figure 1 presents an overview of the proposed VSMSD algorithm.

On seeing a new time window tt_i of a batch of transactions, VSMTP applies two operations on the batch, updating the PFI_{i-1} (**P**otential **F**requent **I**temsets at tt_{i-1}) into PFI_i and compressing tt_i with $TSYV_{i-1}$ into $TSYV_i$ (**T**ilted **S**ynopsis **V**ector at tt_i). The PFI_i is used to quickly return the result for the most common query: mining the frequent patterns over the entire history. The PFI can be maintained using any compact structure for storing itemsets, such as the tree in (Manku 2002). The $TSYV_i$ keeps the summary of transactions in time windows from tt_0 up to current window, in a tilted way, as described in Section 3.2.

When a user asks the support of an itemset *e* in period t1 to t2 $sup_{t1:t2}(e)$, time t1 and t2 are mapped to the corresponding time windows tt_α and tt_β, respectively. The transactions in time windows from tt and tt are then extracted from $TSYV_i$ and a corresponding PFI tree is constructed for the support computation. Consequently, the frequent itemsets in any periods, with respect to any support value, can be discovered. In fact, any frequent itemset mining algorithm such as FP-growth algorithm can be applied to mining the extracted set of transactions.

In addition, the $TSYV_{i-1}$ is invoked to build PFI_i only when the PFI_i cannot provide the up-to-date results. We keep all the itemsets having supports at least the support threshold in PFI_{i-1} during the process. Therefore, the $TSYV_{i-1}$ is required only when a newly specified support is smaller than the support threshold in PFI_i. Querying frequent itemsets of higher supports can be retrieved from PFI_i without the participation of $TSYV_{i-1}$ and $TSYV_i$.

3.2 Tilted Synopsis Vector

VSMTP algorithm effectively compresses the (potentially) discarded transactions into a summary structure. The summary structure can approximate all the transactions in the data stream for each time window. Consequently, we may use the summary structure to extract the transactions for any specified period. The idea of transaction compression in VSMTP is similar to Proximus (Koyuturk 2005) or VSMDS (Lin 2006). However, a structure updating is carried out in place of the re-mining in Proximus for association mining (Koyuturk 2005). The summary structure for the data stream, referred to as a *synopsis vector* (abbreviated as *SYV*), in VSMDS (Lin 2006) can provide merely the approximation regarding the entire history. In order to offer the desired approximation for any period, the summary structure needs to be re-designed.

A naïve extension may allocate as many SYVs as the number of time windows. Nevertheless, such an allocation would consume the available memory very fast. The new structure should confine its space allocation to deal with the unbounded nature of data stream transactions. Therefore, we propose a tilted synopsis vector (abbreviated as TSYV) to maintain a tilted-time table for each approximated transaction.

Tilted-time window is based on the fact that people are often interested in recent changes at a fine granularity, but long term changes at a coarse granularity (Chen 2002). The summary thus may have different granularities. The tilted-time window frame can also be constructed based on a logarithmic time. In this way, with one year of data and the finest precision at quarter, we will need log2(365*24*4)+1 ≈ 17 units of time instead of 366*24*4=35,136 units. The logarithmic tilted-time window schema hence is very space-efficient.

The TSYV (**T**ilted **S**ynopsis **V**ector) in VSMTP can be considered as a list of (delegate, cardinality, tilted-table) triplets. The cardinality indicates the number of occurrences of the delegate; the delegate represents a group of *approximated* itemsets (transactions). The tilted table records the tilted-time summary of the delegate for all the time windows. A delegate *dg* is said to *approximate* an itemset *e* if the difference between *dg* and *e* is no more than certain difference threshold (abbreviated as *dh*). If the *dh* is 0, the delegate is exactly the itemset itself. If *dh* is 1, delegate (a, c, e) may approximate (represent) itemset (a, c, f) or (b, c, e). If there is more than one delegate satisfying *dh*, the one with the minimum distance is selected. The distance between delegate *dg* and itemset *e*, denoted by *dist*(*dg*, *e*), is currently defined as the total number of different items between *dg* and *e* divided by the sum of the lengths of dg and *e*. The one with the largest cardinality will be selected if the both the difference and the distance for some delegates is the same.

Figure 2 shows the structure of the TSYV. Figure 2(a) depicts the concept of logarithmic time scale: using 5 units to record 16 time slots, where the oldest 8 slots, 4 slots, 2 slots are recorded each by one unit. To record the units of the time slots coming in a later time, a successive merging of previous units might be invoked. Such a scheme has been adopted in (Giannella 2002). Figure 2(b) illustrates a TSYV of three delegates representing 488 transactions in total. The cardinality for (*a*, *c*, *e*) is 132, indicating that delegate (*a*, *c*, *e*) occurred 132 times from the beginning of the stream. The tilted table for (a, c, e) states that 92 transactions are approximated in the most recent time window, 32 in the last time window, etc. The tt_3 of value 75 means that 75 transactions merged from the oldest four time windows are approximated. Note that a larger difference threshold often generates a smaller TSYV, which has fewer delegates but larger cardinalities.

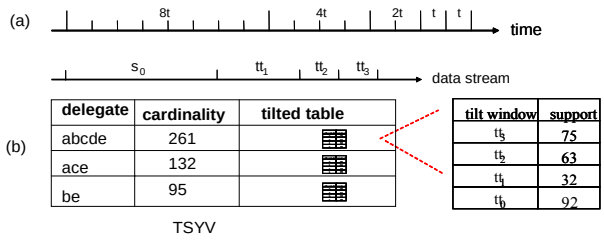

Figure 2: The TSYV Structure

3.3 The VSMTP Algorithm

Figure 3 outlines the VSMTP algorithm and the subroutines used in VSMTP. When a new window of transactions (tt_i) enters the data stream, VSMTP determines whether the synopsis vector is to be used by comparing the minimum support in the PFI (ms_{PFI}) with current minimum support (ms_c). If the ms_c is no less than the ms_{PFI}, VSMTP simply uses ms_{PFI} as the threshold to update the PFI by tt_i. VSMTP would prune the PFI with respect to ms_c to reclaim the memory if necessary. If the ms_c is smaller than the ms_{PFI}, VSMTP reconstructs the PFI with the synopsis vector and tt_i. The desired patterns can be retrieved by a traversal on the PFI. The batch is compressed into the synopsis vector at last.

The *t_update* subroutine maintains itemsets having supports no less than ms_{PFI} in the PFI for fast retrieval of frequent patterns. A tree similar to (Lin 2006) is used for storing the PFI for quick support updating of itemsets of variable length. Similarly, the *build* subroutine utilizes the lexicographic property of consecutive item-comparisons for fast mining of potential frequent itemsets.

To compress the transactions in tt_i, the *t_compress* subroutine aims to find a delegate which is closest to the transaction. The difference threshold, described in Section 3.2, is used to restrict the degree of dissimilarity of the delegate. If no delegate satisfies the requirement, the transaction itself becomes a new delegate. The difference threshold is a trade-off between the compression ratio of the transactions and the accuracy of the tilted synopsis vector. The value of the threshold can be dynamically controlled upon the available memory. That is, VSMTP increases *dh* and compresses TSYV once the available memory is not enough.

4 Experimental Results

We have conducted extensive experiments to evaluate the algorithm. The experiments were performed on an AMD Sempron 2400+ PC with 1GB memory, running the Windows XP, using data-sets generated from (Agrawal 1994). Various parameters were used to generate different distributions of the data sets in the experiments. Here, we report the results on dataset T10.I5.D1000k, which has one million transactions of average size 10 and the average size of maximally frequent itemsets is 5. Please refer to (Agrawal 1994) for the details of the parameters. The experiments on other datasets were consistent.

Algorithm VSMTP

Input: tt_i: a new incoming batch in the data stream DS; ms_i: minimum support; PFI_{i-1}: potential frequent itemset tree for $tt_0 \ldots tt_{i-1}$; $TSYV_{i-1}$: synopsis vector for $tt_1 \ldots tt_{i-1}$
Output: F_i: frequent itemsets; PFI_i: potential frequent itemset for $tt_1 \ldots tt_i$; $TSYV_i$: tilted synopsis vector for $tt_1 \ldots tt_i$

// Initially, PFI_0 and $TSYV_0$ are empty, ms_{PFI} is 1
if ($ms_i \geq ms_{PFI}$)
  PFI_i = t_update(PFI_{i-1}, B_i) ; // update PFI by B_i w.r.t. ms_{PFI}
else
  $ms_{PFI} = ms_i$; // update ms_{PFI}
  PFI_{i-1} = build($TSYV_{i-1}$, ms_{PFI}) ; // build $TSYV_{i-1}$ w.r.t. ms_{PFI}
  PFI_i = t_update(PFI_{i-1}, tt_i) ; // update PFI by tti w.r.t. ms_{PFI}
  output $F_i = \{e \mid e \in PFI_i$ and sup(e) $\geq ms_i\}$; // frequent patterns w.r.t. ms_i
end-if
$TSYV_i$= compress(tt_i, $TSYV_{i-1}$) ; // compress tt_i into $TSYV$

(a) Algorithm VSMTP

Subroutine t_update(PFI_{i-1}, tt_i)
Input: PFI_{i-1}: potential frequent itemset for $tt_1 \ldots tt_{i-1}$; tt_i: a new incoming batch in DS
Output: PFI_i: potential frequent itemset for $B_1 \ldots B_i$

$F_{Bi} = \{e \mid e \in B_i$ and sup(e) in $B_i \geq ms_{PFI}\}$;
for each itemset e in F_{Bi} do
  if ($e \in PFI_{i-1}$) // itemset e exists in PFI_{i-1}
    increase the support count of e in PFI_{i-1} by the support count of e in tt_i ;
  else
    insert (e, support count of e in tt_i) into PFI_{i-1} ;
  end-if
end-for
return PFI_{i-1} ;

Subroutine build($TSYV_{i-1}$, ms)
Input: $TSYV_{i-1}$: synopsis vector for $tt_1 \ldots tt_{i-1}$; ms: the minimum support threshold
Output: PFI_i: potential frequent itemset tree for $TSYV_{i-1}$

PFI_i = empty tree ;
$F_{SYV} = \{e \mid e \in TSYV_{i-1}$ and sup(e) in $TSYV_{i-1} \geq ms\}$;
for each itemset e in F_{SYV} do
  insert (e, sup(e) in F_{SYV})) into PFI_i
end-for
return PFI_i ;

Subroutine t_compress(tt_i, $TSYV_{i-1}$)
Input: tt_i: a new incoming batch in ds; $TSYV_{i-1}$: tilted synopsis vector for $tt_1 \ldots tt_{i-1}$
Output: $TSYV_i$: synopsis vector for $tt_1 \ldots tt_i$;

for each transaction t in B_i do // Let dh be the distance threshold
  $dg = \{e' \mid e' \in TSYV_{i-1}$ and dist(t, e') = 0$\}$;
  if ($dg = \varnothing$) $dg = \{e'' \mid e'' \in TSYV_{i-1}$ and $\forall\, e' \in TSYV_{i-1}$ dist(t, e') $\geq$ dist(t, e'')$\}$;
  if (dist(t, dg) > dh) // the distance between the best delegate and t is too large
    insert (t, 1) into $TSYV_i$ // t is a new delegate with cardinality 1 in $TSYV_i$
  else
    increase the cardinality of dg by 1
  end-if
end-for

(b) Subroutines used in Algorithm VSMTP

Figure 3. (a) Algorithm VSMTP (b) the subroutines.

The difference threshold was 10 and 10000 transactions were in a time window. Figure 4 shows the performance of VSMTP algorithm with respect to a support sequence of random values ranging from 1.1% to 2%. The breakdown of execution time is shown in Figure 5. The Cr in the figures is the dynamic compression ratio simulating the amount of available memory, representing the fraction of TSYV to the total number of transactions. The performance with respect to various time window sizes is shown in Figure 6, and the working memory sizes for the experiments are depicted in Figure 7. Let the compression ratio be the size of the tilted synopsis vector divided by the total number of transactions, simulating the amount of available memory. Figure 8 indicates that VSMTP algorithm scales up sub-linearly with respect to the dataset size, from one million to 5 million.

5 Conclusion

In this paper, we propose the VSMTP algorithm for mining frequent patterns over arbitrary time intervals in data streams with changeable support threshold. To the best of our knowledge, VSMTP is the first algorithm that provides users with the ability to change the support and mine patterns with respect to any designated time interval in data streams. VSMTP utilizes the lexicographically structured PFI for fast updating and mining of patterns. The proposed tilted synopsis vector is effective for maintaining statistics of past and incoming transactions across different time intervals in the stream. The extensive experiments confirm that VSMTP efficiently mines frequent patterns with respect to variable supports, and has sub-linear scalability.

6 Acknowledgements

The authors thank the precious comments of the reviewers for improving the quality of the paper. The research was supported partially by National Science Council of R.O.C. under contract number NSC96-2221-E-035-093.

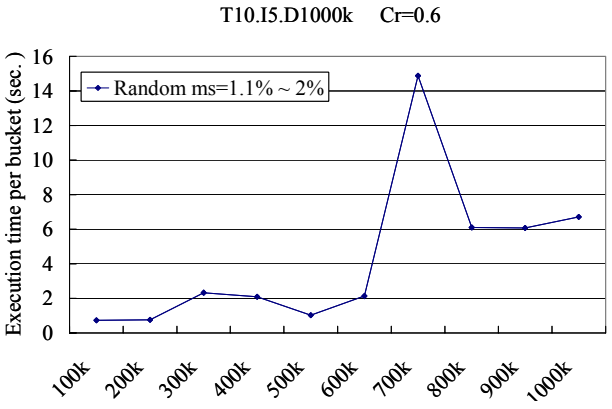

Figure 4. Mining the data stream with a support sequence of random thresholds

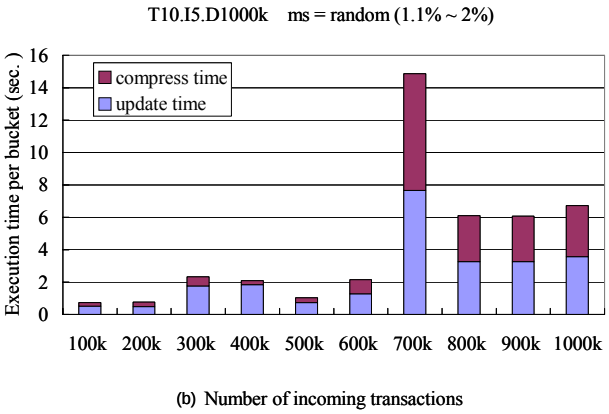

Figure 5. The breakdown of the processing time.

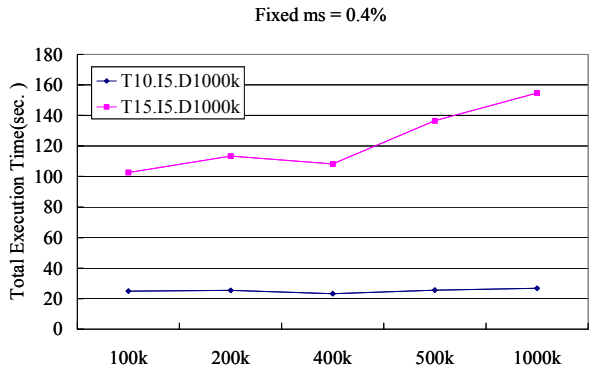

Figure 6. Effect on various time window sizes.

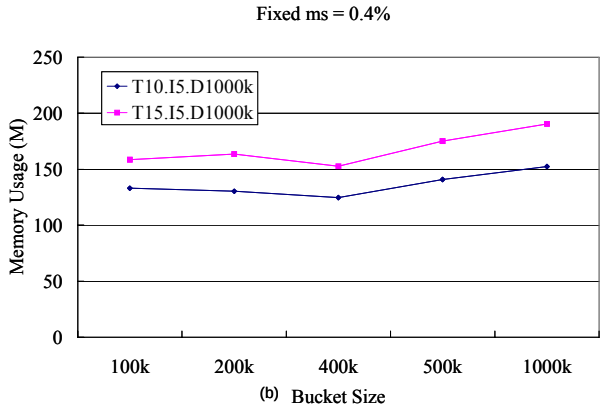

Figure 7. Working memory size w.r.t. window sizes.

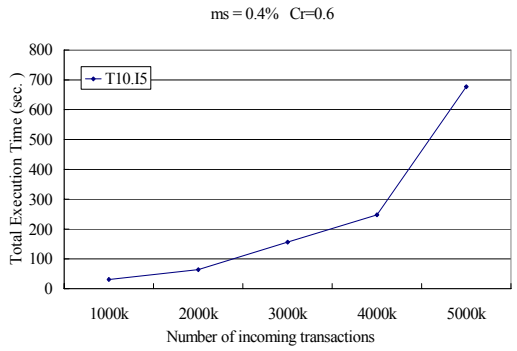

Figure 8. Scalability evaluation: 1000k to 5000k.

7 References

Agrawal, R., Imielinski, T. and Swami, A. (1993): Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD International Conference on Management of Data*, Washington DC, USA, **22**:207-216, ACM Press.

Agrawal, R. and Srikant, R.(1994): Fast Algorithm for Mining Association Rules. *Proc. of the 20th International Conference on Very Large Databases*, 487-499.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002): Models and Issues in data stream systems. *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Popa L. (ed.), ACM Press.

Chang, J. H. and Lee, W. S. (2005): estWin: Online data stream stream mining of recent frequent itemsets by sliding window method. *Journal of Information Science*, **31**(2): 76-90.

Charikar, M., Chen, K., and Farach-Colton, M.(2002): Finding frequent items in data stream. *Proc. of the 29th International Colloquium on Automata, Languages and Programming*, 693-703, Malaga, Spain.

Chen, Y.; Dong, G.; Han, J.; Wah, B.W. and Wang, J (2002): Multidimensional regression analysis of time-series data streams. *Proc. of 2002 International Conference on Very Large Data Bases*, 323-334.

Chi, Y. and Wang, H.(2004): Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window. *Proc. of the Fourth IEEE International Conference on Data Mining*, 59-66, Brighton, United Kingdom, IEEE Computer Society Press.

Cormode, G., Muthukrishnan, S. (2003): What's hot and what's not: tracking most frequent items dynamically. *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 296-306.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S. (2002): Mining Frequent Patterns in Data Streams at Multiple Time Granularities. *Proc. of the NSF Workshop on Next Generation Data Mining*.

Golab, L. and Özsu, M. T.(2003): Issues in Data Stream Management. *ACM SIGMOD Record* **32**(2):5-14, ACM Press.

Hidber, C. (1999): Online association rule mining. *Proc. of the 1999 ACM SIGMOD International Conference on Management of Data*, 145-156, Delis A., Faloutsos C., Ghandeharizadeh S. (eds.), USA, ACM Press.

Han, J., Pei, J., and Yin, Y. (1999): Mining Frequent Patterns without Candidate Generation. *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data*, **9**(2): 1-12, ACM Press.

Jiang N. and Gruenwald L. (2006): Research issues in data stream association rule mining. *Proc. ACM SIGMOD Record* **35**(1):14-19, ACM Press.

Jin, C., Qian, W., Sha, C., Yu, J. X., and Zhou, A. (2003): Dynamically Maintaining Frequent Items over a Data Stream. *Proc. of the 12th ACM Conference on Information and Knowledge Management*, 287-294, USA, ACM Press.

Koyuturk, M., Grama, A., and Ramakrishnan, N. (2005): Compression, clustering and pattern discovery in very high dimensional discrete-attribute datasets. *IEEE Transactions on Knowledge and Data Engineering*, **17**(5):447-461.

Li, H. F., Lee, S. Y., and Shan, M. K.(2004): An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams. *Proc. of the First International Workshop on Knowledge Discovery in Data Streams*, 20-24.

Li, H.F., HO, C. C., Kuo F. F., and Lee, S. Y. (2006): A New Algorithm for Maintaining Closed Frequent Itemsets in Data Streams by Incremental Updates. *Workshops Proc. of the 6th IEEE International Conference on Data Mining*, 672-676, IEEE Computer Society Press.

Lin, M. Y. and Lee, S. Y. (2004): Interactive Sequence Discovery by Incremental Mining. *Journal of Information Sciences: An International Journal*, **165**(3-4): 187-205.

Lin, M. Y., Hsueh, S. C., and Hwang, S. K. (2006): Variable Support Mining of Frequent Itemsets over Data Streams Using Synopsis Vectors. *Lecture Notes in Artificial Intelligence,* **3918**:724-728, Ng W. K., Kitsuregawa M., Li J., and Chang K. (eds.), Springer.

Manku, G. S. and Motwani, R. (2002): Approximate Frequency Counts over Data Streams. *Proc. of the 28th VLDB Conference*, Hong Kong, China, 346-357, Morgan Kaufmann Press.