International Journal of Grid and Distributed Computing
Vol. 11, No. 9 (2018), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2018.11.9.01

Hierarchical Level Based Frequent Pattern Mining

Jeong Hee Hwang[1], Hyeok Kim[2] and Jeong Hee Chi[3*]

[1]*Department of Computer Science, Namseoul University, Korea*
[2,3]*Department of Software, Konkuk University, Korea*
[1]*jhhwang@nsu.ac.kr,* [2,3]*{gkdl1599, jhchi}@konkuk.ac.kr*

Abstract

Temporal data mining which considers time attributes, centers on discovering the association of items that occur over a period or discovering items that occur sequentially and frequently. However, there are cases in which an item occurs frequently during a certain period even though it has not occurred frequently during the entire period. In this paper, we propose a TD-HTIP (Temporal Data-Hierarchical Time Interval Period) algorithm that searches for items that occur frequently within a specific time range by constructing a hierarchical time interval period that considers a certain time interval for temporal data. The proposed algorithm constructs a matrix of transaction information containing items that occur over time, and searches for frequent items based on the matrix. Experimental results show that the proposed algorithm enables searching more frequent items compared to the existing algorithms for execution time.

Keywords: *Temporal data mining, Data mining, Association rule, Frequent pattern*

1. Introduction

Due to the development of information technology, the amount and speed of data generated by the development of computer technology, and related technologies are increasing [1]. Data mining is a systematic and automatic finding of statistical rules and patterns within a large amount of stored data. Frequent patterns are item sets that occur frequently in a data set. Finding frequent patterns has an essential role in exploring associations, correlations, and interesting items between data. Frequent pattern mining is a way to find association rules or to determine the relevance between data, and the most commonly used association rule defines association between concurrent events [2-8]. Also, sequence pattern [9-11] is similar to association rule except for the time factor in finding the association with the events occurring over a certain period of time. In general, there are many kinds of knowledge related to time, and it is difficult to classify them clearly, but some types can be categorized according to the type of temporal pattern included in temporal knowledge. A cyclic pattern is a pattern that occurs periodically as time changes. For example, it is a pattern associated with cycles such as 'weekly', and 'yearly'. A sequential pattern is a sequence of successive occurrences of a particular event. An event has a causal relationship that causes other events to occur. In addition, a trend pattern is a pattern where the value changes over time that can predict future fluctuations. It can make predictions such as "Rise in stock A will lead to a decline in stock B". As above, analyzing the association of event changes in consideration of time can be applied in many fields [12-14].

General association rule mining [15-16] is based on items that occur frequently in transactions, and it focuses on finding the associated items occurring in a certain transaction or window for stream data. Knowledge discovered through data mining need

Received (January 15, 2018), Review Result (March 14, 2018), Accepted (July 13, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

to be provided as useful information through analysis of the data. What is important here is trend analysis of data that changes over time. In other words, human interest, drug types in the medical field, and financial data change over time, and the basis for analyzing these changes is the exploration of frequent patterns that occur over time. This study proposes an algorithm to find association between items considering a constant time interval period for temporal data. The proposed method constructs a basic time interval period of a certain unit of time to explore frequent items in transactions constituting each time unit. Then, basic time interval is grouped to explore frequent items for an extended time interval of a hierarchical level structure. Since the grouping unit has a hierarchical level structure, frequent items can be explored for the entire temporal database if the grouping is repeated and extended, and there is an advantage that it is possible to find a frequent item in a specific period or to detect a frequent item change with time. The algorithm proposed in this paper can be applied to discovering how the associated items changes with time, such as financial data with time information, medical data, and personal interest patterns.

The remainder of the paper is organized as follows. Section 2 reviews researches related to the proposed algorithm, and Section 3 defines basic concepts for mining and explains TD-HTIP algorithm for discovering frequent items. Section 4 analyzes the efficiency of the proposed algorithm through comparison experiments with existing algorithms. We conclude the work in Section 5.

2. Related Works

Sequential pattern mining is a technique for exploring patterns in which a certain set of items occurs sequentially among transactions composed of a set of items, and it adds a temporal relation to the association rule that explores the association between items occurring within a transaction. An existing research, frequent episode exploration [16], proposed a technique for finding episodes that occur frequently from a series of event sequence data. Here, an episode is defined as a frequent occurrence of a specific sequence of events, and represent events closely related to each other. The method of exploration finds all frequent episodes that the ratio of the window that include the episodes occurred in the window size specified by the user satisfies the given threshold. For example, when the sequence of events occurring every second is {a, b, d, a, c, b, d, d, a}, assuming that the time window size is 3 seconds and the minimum frequency is 50%, (a, b) is an episode that satisfies the minimum frequency because the frequency of (a, b) is 2 for three windows. In addition, time-series analysis is a technique for predicting the future by capturing trends from time series data, such as stocks, prices, and sales volume, which are arranged in time base. Mining for this temporal database explores useful association knowledge from time-related data. Through time-based data mining, a variety of knowledge can be found through data with time and meaning associations such as patient history, product purchase history, and web logs. For example, an association rule that has time information may be explored, such as "People who purchases item A during summer season every year also purchases item B".

Association rule mining finds a frequent item set satisfying a predefined minimum support in a transaction and finds an association rule that reflects the relation between these frequent item sets. Apriori [2] is the most widely used algorithm. However, the Apriori algorithm has the problem of continuously scanning large volumes of transaction databases for each frequent item set exploration. An existing study [6] proposed a mining algorithm based on hierarchical time granule and applied Apriori algorithm to find association rules. Various algorithms have been proposed to solve the problems of Apriori algorithm and to search for frequent items more efficiently. Methods to reduce the number of candidate items to reduce the number of database scans and improve the efficiency of the Apriori algorithm such as DIC (Dynamic Itemset Counting) [19], DHP

(Direct Hashing and Pruning) [20], Partition [21], and Sampling [22] algorithms have been proposed. Although these algorithms reduced the number of database scans compared to Apriori, it is still difficult to determine frequent items for each candidate item due to the need of generating many candidate item sets. In order to solve these problems, methods of discovering frequent items [23] have recently been proposed by scanning a database, displaying items on a row, and displaying TID in a column to generate a matrix. A matrix is used to calculate the support of candidate items in a frequent item set and once it is created, it is not necessary to scan the database again. In addition, the time required to generate the candidate items is less than that of the conventional algorithm. A representative algorithm using matrix is the ECLAT algorithm [23]. There are horizontal data format and vertical data format in the methods for generating a data structure in the ECLAT algorithm. The horizontal data format is expressed as TID: itemset and consists of the transaction number (TID) containing the itemset. The vertical data format consists of Item-TID_set and is expressed as Item: TID_set.

3. Temporal Data Mining

3.1. Problem Definition

Temporal data with time information consists of items of transaction. The items in the transaction are reconstituted from the original items occurring over time to the items of interest for mining. The initial data is generated by filtering the items of interest from the raw data or data items satisfying the threshold specified by the user. In this paper, our algorithm is to mine frequent item sets based on a time interval period containing a certain number of transactions. The basic definitions for this are as follows.

Definition 3.1 Itemset

The set of items that occur in temporal data is defined as Itemset(I), represented as $I = \{I_1, I_2, I_3,..I_i...,I_n\}$, and in the Itemset, the i^{th} item is denoted by I_i.

Definition 3.2 Temporal Transaction

A temporal transaction is a transaction that contains temporal data that occurs over time. It is denoted by Tr, and serial numbers are assigned to transactions that are generated over time. In $Tr = \{Tr_1, Tr_2, Tr_3,..,Tr_i,..Tr_n\}$, Tr_i represents the i^{th} transaction.

Definition 3.3 Time Interval Period

A single time interval period (TP) includes a certain number of transactions, and the size of the TP is denoted by $|W|$. For example, if the number of transactions contained in one TP is 4, $|W|=4$.

Definition 3.4 Time Interval Period Grouping

A TP containing a certain number of transactions has a level and sequence and is expressed in $TP_{v.s}$. Here, v is the hierarchical temporal level that groups the TP and s is the serial number of the TP grouped at the same level. Thus, two TP is grouped to generate a hierarchical level TP. When grouping TP, it is grouped with the next sequence of the same level. Figure 1 shows an example of a TP grouping scheme. The expression of grouping is represented by $TP_{2.1}=TP_{1,1} \cup TP_{1.2}$, $TP_{2.2}=TP_{1,2} \cup TP_{1.3}$.

Definition 3.5 Item Support

For every transaction included in the grouping with $TP_{v.s}$ having the same serial number of the same level, the number of transactions(Num) including the item is defined as item support and is denoted as Supp(I_i).

$$Supp(I_i) = \frac{\sum Num(I_i \in TP_{v.s})}{|W| \times V_{of} TP_{v.s}} \quad (1)$$

Definition 3.6 Minimum Support(min_sup)

Minimum support refers to the minimum value that the ratio (%) of the number of transactions containing the item must satisfy the total number of transactions included in $TP_{v.s}$ grouped at each hierarchical time level. It is used as a threshold to search for frequent association items.

Definition 3.7 Frequent Itemset

Frequent ItemSet (*FIS*) is an itemset that satisfies a minimum satisfaction degree (min_sup) in each $TP_{v.s}$, that filters items that meet the minimum frequency specified by the user from the temporal database (*TDB*). Here, the minimum frequency can be specified by the user to eliminate an item in which the occurrence frequency of the item is too infrequent.

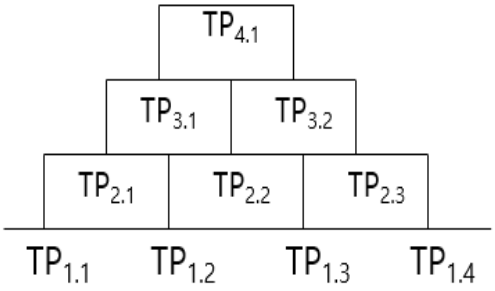

Figure 1. Structure of TP Grouping

The method of discovering frequent itemsets is to extract frequent itemsets satisfying the minimum support in each basic time interval period of 1-level and construct 2-level *TP* through 1-level *TP* grouping as showing in Figure 1. The union of the frequent items extracted from the 1-level *TP* becomes a 2-level *TP* item, and those items that do not satisfy min_sup are removed. In other words, common items are searched in the time interval period of the grouped *TP*. In this way, 2-level *TP* are grouped to explore frequent items in 3-level *TP*.

3.2. TD_HTIP Algorithm

TP consists of a certain number of transactions and has a serial number that reflects the time information. Also, the items that satisfy the least frequent items in the original item included in each *TP* are filtered and sorted in alphabetical order. If $|W|$ is 4 that a *TP* contains 4 transactions, it searches for frequent items satisfying the minimum support from items of transactions in each *TP*. To explore frequent items, we used the Eclat algorithm [19], which performs mining by constructing a matrix to reduce candidate item creation time. The items included in the entire *TP* are expressed in columns, and the matrix including the transaction including the item and the *TID* is constituted by rows. By using this matrix information, the frequent items considering the minimum frequency in each *TP* are filtered, and the filtered items are basic itemsets of 1-level *TP*. Table 1 shows an example of the items that make up the 1-level *TP*.

Table 1. 1-level TP Itemset

Time Interval Period	TID	Filtered Items
$TP_{1.1}$	T1	d
	T2	c, d
	T3	c
	T4	d
$TP_{1.2}$	T5	a, c, d
	T6	a, b, c, d
	T7	b, c, d
	T8	a, d
$TP_{1.3}$	T9	b
	T10	a, c
	T11	a, b, c
	T12	b, c
$TP_{1.4}$	T13	b, d
	T14	b, c, d
	T15	b
	T16	b, c, d

By grouping *TP*s composed of frequent items of 1-level, frequent items in the grouped *TP* are explored. For example, if the size of the *TP* is $|W|=4$, the frequent items are found in all eight transactions included in $TP_{1.1}$ and $TP_{1.2}$, and it is stored in $TP_{2.1}$. Thus, frequent items are unioned in $TP_{1.1}$ and $TP_{1.2}$ and whether it satisfies min_sup is checked, to confirm the presence of frequent items. Next, common frequent items are extracted from $TP_{1.2}$ and $TP_{1.3}$ and stored in $TP_{2.2}$. In this way, frequent items are found in 1-level $TP_{1.s}$ and $TP_{1.s+1}$ ($1 \leq s$ and $s \geq n-1$) and stored in $TP_{2.s}$(2-level, s sequence). Then, the *TP* sequences of the 2-level frequent items are grouped to generate a 3-level $TP_{3.s}$ sequence. In this method, a hierarchical *TP* level sequence is constructed by gradually extending the time interval period, and as a result, the result of exploring the frequent items for the transactions included in the entire *TDB* is generated.

Table 2. 1-level Frequent Itemsets(FIS)

1-level	seq.	TID	Freq. Items	FIS(≥length(2)
TP_1	$TP_{1.1}$	T1, T2, T3, T4	c, d	
	$TP_{1.2}$	T5, T6, T7, T8	a, b, c, d	ac, ad, bc, cd, bd, acd, bcd
	$TP_{1.3}$	T9, T10, T11, T12	a, b, c	ac, bc
	$TP_{1.4}$	T13, T14, T15, T16	b, c, d	bd, cd, bcd

We explain the mining process with the example of Table 1, assuming *min_sup*= 0.5. As an example of the support for some items of $TP_{1.2}$, *Supp(a)* = 3/4, *Supp(c)* = 3/4, *Supp(ac)* = 2/4, *Supp(bd)* = 2/4, which satisfies *min_sup* that means a, c, ac, bd are frequent items, and Table 2 shows the most frequent items for each 1-level *TP*. 1-level *TP* sequence generates a 2-level *TP* sequence through grouping. For example, if a frequent item in a 1-level *TP* sequence is unioned to group $TP_{1.1}$ and $TP_{1.2}$ to generate 2-level $TP_{2.1}$, $FIS(TP_{1.1}) \cup FIS(TP_{1.2}) = \{a, b, c, d\}$, and these items become candidate items. Then $Supp\ (I_i)$ is calculated to check the satisfaction of *min_sup*, and it is decided whether or

not the frequent item is satisfied. When frequent items are determined, they become items of *FIS* ($TP_{2.1}$). In this way, the *TP* sequences belonging to the time interval period are grouped, and the frequent items are determined while increasing the level.

The algorithm for this is described as follows algorithm 1.

Algorithm 1 : TD-HTIP Algorithm

Input : Temporal Database(*TDB*), *min_sup*
Output : Frequent itemsets(*FIS*) at each time interval period(*TP*) with level and sequence

Step 1 : Find *FIS* at *TP* on 1_level
1 : Scan *TDB*
2 : Initialize *TP* by the number of transaction(|*W*|)
3 : Sort the items in *TP* according to alphabetical order
4 : Call *TECLAP Function* to find frequent items at 1_level
5 : Insert frequent itemsets in each *TP* into $FIS(TP_{l,s})$ //1_level TP sequence

STEP 2 : Group *TP* sequences at each level to search for frequent items
// Grouping of *TP* sequences of the same level
6 : for i= 1 to COUNT(TP_{level})-1 do
7 : Candidate items = $FIS(TP_{level,i}) \cup FIS(TP_{level,i+1})$
8 : if $Supp(candidate\ item) \geq min\_sup$
9 : Insert frequent itemsets into $TP_{level+1,i}$
10 : end for

Algorithm 2 : TECLAT function

Input: *TP* in Database , *min_sup*
Output: Frequent Itemsets at 1-level

1 : Create Vertical Data Matrix(*VDM*) by item in each *TP*,
2 : Create Pattern tree with a root of null
3 : for i = 1 to Count(item) in *VDM* do
4 : if Supp(*VMD[i].TranID_sets*) ≥ *min_sup* then
5 : add itemsets in *VMD[i]* to Pattern tree
6 : end for
7 : Find all frequent itemsets(*FIS*) from Pattern tree

The *TP* sequence of the next level is generated through the grouping process of STEP 2, and the process of STEP 2 is repeated for every *TP* sequence generated at each level. This process is repeated until there are no more *TPs* to be grouped.

4. Experiments

We evaluate the performance of the proposed TD_HTIP algorithm comparing with the previous TP-HTAR algorithm in [5]. In the experimental results, the TD_HTIP algorithm is denoted by A1 and the TP-HTAR algorithm is denoted by A2. All experiment is performed in R and conducted in a machine with a 3.30GHz CPU and 8GB RAM running Window 10.

4.1. Experiment Environment

The experimental dataset use data generated by R data generator. Table 3 illustrates the parameters used for data generation. The average number of items per transaction was set to 10, and 10K and 100K transactions were created. Also, in order to evaluate the performance according to the number of items, the data set was created by increasing the total number of items from 100 to 4000. In addition, to test the change of number of frequent items according to density of data, 10K and 100K data sets were created for the

same number of items. Therefore, T10I1000D10K and T10I1000D100K data sets refer to 10K dataset and 100K dataset consisting of 1000 items.

Table 3. Parameters for the Synthetic Data

Parameter	Description	Value
T	The average length of items per transaction	10
I	The total number of items	100 ~ 4000
D	The total number of transactions	10K, 100K

Table 4 shows the experimental parameters. To analyze the number of frequent items generated according to minimum support, *min_sup* was varied from 0.01 to 0.08. Also, the number of transactions included in one window was adjusted by changing the window size from 625 to 25000. That is, when the window size is 625, 16 basic periods are generated in the 10K data set. We conducted the experiment by varying the basic period from 4 to 32.

Table 4. Simulation Parameters

Parameters	Value
Minimum Support (*min_supp*)	0.01 ~ 0.08
Window Size(*ws*)	625 ~ 25000
Basic Periods(*p*)	4, 8, 16, 32
A1	TD_HTIP algorithm
A2	TP-HTAR algorithm
A2-RS	A2 with Relative-Support

4.2. Experimental Evaluation

The algorithm proposed in this paper is an approach to search for frequent items occurring within a certain time period, and we compared the number of frequent items generated by each algorithm for performance comparison with the existing TP-HTAR algorithm.

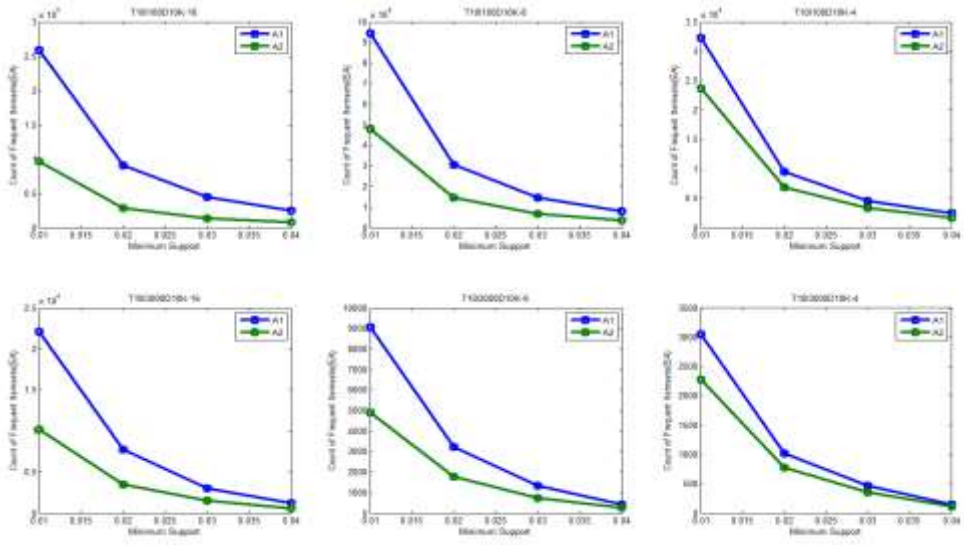

Figure 2. Number of Frequent Items According to Minimum Support

First, we compared the number of frequent items generated according to minimum support. Figure 2 shows the basic periods 4, 8, and 16 in the dataset T10I100D10K and T10I3000D10K, and the results of varying minimum support from 0.01 to 0.04. These datasets were used to compare the performance of each algorithm with a relatively small number of transactions, when there were few or many items. As shown in the experimental results, it can be seen that the A1 algorithm proposed in this paper generates more frequent items than the A2 algorithm in all the results. In particular, if the T10I100D10K dataset has a minimum support of 0.04, A1 algorithm showed 25,172 frequent items, or 3.26 times more than the 7,717 generated by A2 algorithm, and even in the worst case, 0.025 of T10I3000D10K, the performance of A2 was 116 and A1 was 149, which was 1.28 times higher than that of A2.

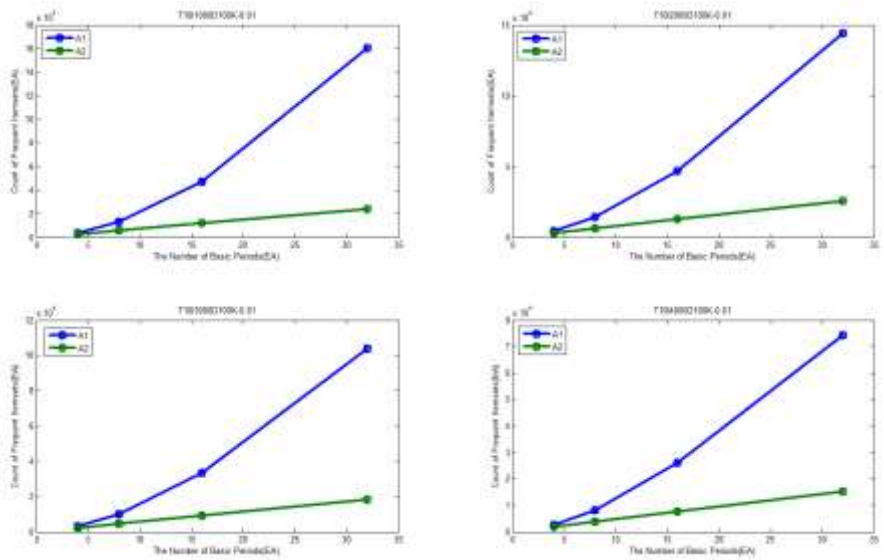

Figure 3. Number of Frequent Items According to basic Periods

We also compared and analyzed the changes in the number of frequent items according to the number of basic time interval periods. Figure 3 shows the results of varying basic periods from 4 to 32. Each graph shows the results for a minimum support of 0.01 for dataset from T10I1000D100K to T10I4000D100K. Experimental results show that the number of frequent items increases as the number of basic periods increases in both A1 and A2 algorithms. Especially, when the number of items is 1000 and there are 32 periods, it can be seen that A1 algorithm finds 6.7 times more frequent items than A2. Also, when the number of items is 4000 and there are 4 periods, it shows the fewest frequent items, but also in this case it was shown that A1 generated 1.38 times as many frequent items as A2. Through this study it was shown that A1 provides more frequent items at 3.25 times than A2 at minimum support 0.01 for 100K dataset.

Next, we compared the execution time of each algorithm. Figure 4 shows runtime results based on minimum support, and it shows the experiment results of 16 periods of T10I100D10K and 32 periods of T10I4000D100K. Experimental results show that as the minimum support increases, the execution times of A1 and A2 algorithms become similar, and this is because both algorithms result in fewer frequent items being generated as the minimum support is increased.

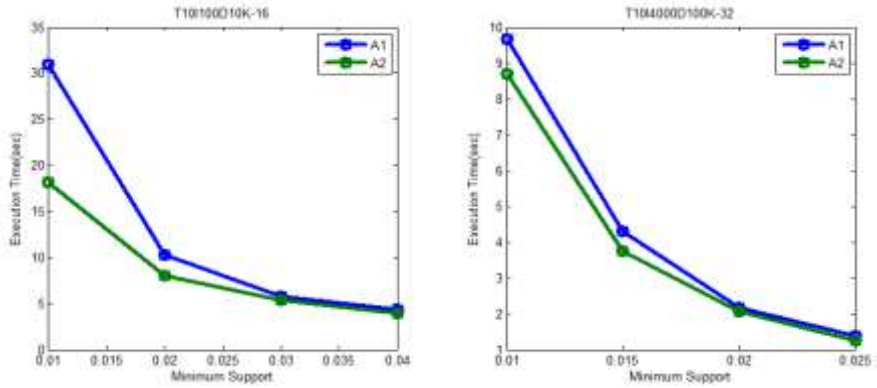

Figure 4. Execution Time According to Minimum Support

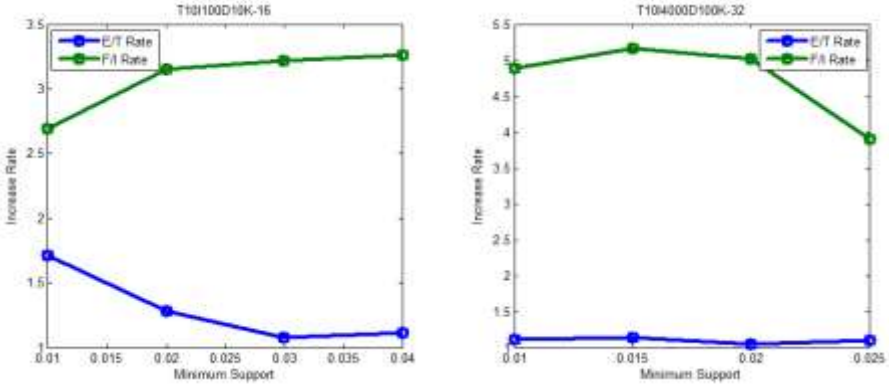

Figure 5. Increase Rate of Frequent Items and Execution Time

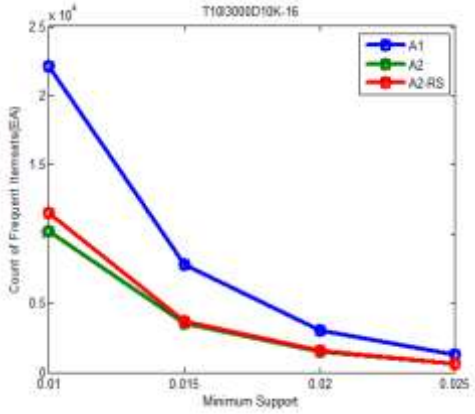

Figure 6. Number of Frequent Items According to Relative Support and Minimum Support

In order to analyze the increase rate of execution time according to the number of frequent items, we experimented with increasing rate of frequent items and execution time according to minimum support as shown in Figure 5. As a result, in the number of frequent items according support, while A1 technique increased by 3.08 times and 4.74 times on average compared with A2, execution time showed an average increase of 1.29 times and 1.10 times. Therefore, although the A1 method proposed in this paper takes more time than the A2 method, it shows that the number of frequent items generated by this method is about 3 times more.

Also, we experimented with changes in the number of frequent items based on relative support. Figure 6 shows the change in the number of frequent items according to the relative support of the A1 and A2 techniques for the 16 periods of the T10I3000D10K. Experimental results show that, when the relative support is applied, minimum support 0.01 of A2 method generated frequent items which increased by 13%, and in case of 0.025, 4% of frequent items were additionally increased. It can be seen that the A1 technique always generates a larger number of frequent items even if relative support is applied to A2.

5. Conclusions

In this paper, we proposed a TD-HTIP algorithm that finds frequent items in time interval period based hierarchical level, which contains a fixed size transaction in temporal database and uses TP sequence grouping. Also, we compared the performance of the proposed algorithm with the performance of the existing algorithm, and experimental results show that the proposed algorithm takes about 1.2 times more time than the conventional algorithm, but it finds 3 times more extracted frequent items. Therefore, the proposed algorithm is effective for searching for frequent item changes over time and can be applied to frequent item extraction for specific time domain. For example, it can be efficiently applied to applications to detect changes in data, such as continuously incoming data consisting of protocol and data exchange traffic associated with network components such as networking-related routers, traffic volume control, fault handling, and intrusion detection.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Seoul Accord Vitalization Program(IITP-2018-2012-1-00593) supervised by the IITP(Institute for Information & communications Technology Promotion).

References

[1] M. Chouhan and V. K. Verma, “A Recent Study of Various Methods for Mining Pattern Using Time”, International Journal, vol. 2, no. 2, **(2017)**, pp. 21-24.
[2] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, The VLDB Conference, vol. 1215, **(1994)**, pp.487-499.
[3] C. Y. Chang, M. S. Chen and C. H. Lee, “Mining general temporal association rules for items with different exhibition periods”, The IEEE International Conference on Data Mining, **(2001)**, pp. 59-66.
[4] C. H. Lee, C. R. Lin and M. S. Chen, “On mining general temporal association rules in a publication database”, The IEEE International Conference on Data Mining, **(2001)**, pp. 337-344.
[5] J. M. Ale and G. H. Rossi, “An Approach to Discovering Temporal Association Rules”, Proceedings of the 2000 ACM symposium on Applied computing ACM, vol. 1, **(2000)**, pp. 294-300.
[6] T. P. Hong, G. C. Lan, J. H. Su, P. S. Wu and S. L. Wang, “Discovery of temporal association rules with hierarchical granular framework”, Applied Computing and informatics, vol. 12, no. 2, **(2016)**, pp. 134-141.
[7] S. Kuriakose and R. Nedunchezhian, “Efficient adaptive frequent pattern mining techniques for market analysis in sequential and parallel systems”, International Arab Journal of Information Technology, vol. 14, no. 2, **(2017)**, pp. 175-185.
[8] R. Karim, M. Cochez, O. Beyan, C. F. Ahmed and S. Decker, “Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach”, Information Sciences, vol. 432, **(2018)**, pp. 278-300.
[9] F. Wang and Y. H. Li, “An Improved Apriori Algorithm based on the matrix”, FBIE 2008, **(2008)**, pp.152-155.
[10] R. Agrawal and R. Srikant, “Mining Sequential Patterns”, Proceeding of International Conference on Data Engineering, **(1995)**, pp. 3-14.
[11] Y. Xue, T. Li, Z. Liu, C. Pang, M. Li, Z. Liao and X. Hu, “A new approach for the deep order preserving submatrix problem based on sequential pattern mining”, International Journal of Machine Learning and Cybernetics, vol. 9, no. 2, **(2018)**, pp. 263-279.

[12] R. Srikant and R. Agrawal, "Mining sequence patterns: generalizations and performance improvements", Proceeding of International Conference on Extending Databases Technology, **(1996)**, pp. 1-17.
[13] J. Pei, J. Han, B. M. Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal and M. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 11, **(2004)**, pp. 1424-1440.
[14] J. Han, G. Dong and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database", Proceeding of International Conference on Data Engineering, **(1999)**, pp. 106-115.
[15] J. F. Roddick, K. Hornsby and M. Spilipoulou, "Temporal Spatial and Spatio-Temporal Data mining and Knowledge Discovery Research Bibliography", ACM SIGKDD Exploration Newsletter, vol. 1, no. 1, **(2000)**.
[16] H. Mannila, H. Toivonen and A. I. Verkamo, "Discovery of frequent episodes in event sequence", Data mining and Knowledge Discovery, vol. 1, no. 3, **(1997)**, pp. 259-289.
[17] C. F. Ahmed, S. K. Tanbeer and B. S. Jeong, "Efficient Mining of Weighted Frequent Patterns Over Data Streams", 11th IEEE International Conference on High Performance Computing and Communications, **(2009)**, pp. 400-406.
[18] J. Chang and W. Lee, "A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams", Journal of Information Science and Engineering, vol. 20, no. 4, **(2004)**, pp. 753-762.
[19] S. Brin, R. Motwani, J. D. Ullman and S. Tsur, "Dynamic itemset counting and implication rules for market basket data", Proceeding of the ACM SIGMOD International Conference on Management of Data, **(1997)**, pp. 255-264.
[20] J. S. Park, M. S. Chen and P. S. Yu, "An Effective Hash-based Algorithm for Mining Association Rules", Proceeding of ACM SIGMOD Confidence on Management of Data(SIGMOD'95), **(1995)**, pp. 175-186.
[21] A. Savasers, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association rules in Large Databases", Proceeding of the 21st International Conference on Large Databases, **(1995)**, pp. 432-444.
[22] H. Toivonen, "Sampling Large Data-base for Association Rules", The VLDB Conference, vol. 96, **(1996)**, pp. 134-145.
[23] M. Zaki, "Scalable algorithms for association mining", IEEE Transaction on knowledge and Data Engineering, vol. 12, no. 3, **(2000)**, pp. 372-390.

Authors

Jeonghee Hwang, she received Ph.D degrees from Chungbuk National University in 2005 in computer science. In 2006 joined the faculty of Namseoul University, where she is now an assistant professor. Her research interests include data mining, semantic web, ubiquitous computing and big data processing.

Hyeok Kim, he is undergraduate student, Konkuk University, Korea. His research interests include data mining and big data mining.

Jeonghee Chi, she received Ph.D degrees from Chungbuk National University in 2006 in computer science. In 2007 she joined the faculty of Konkuk University, where she is now an assistant professor. Her research interests include IoT, stream data mining, machine learning and big data analysis.

