I J C T A, 8(5), 2015, pp. 2275-2284

A Novel Frequent Pattern Mining Approach with OTSP

Y. Jeya Sheela* and S. H. Krishnaveni**

Abstract: An efficient frequent pattern mining algorithm must overthrow time, space and computational complexity. However, trade-off may be observed with respect to time and space. Current research studies strive to develop a technique which can balance these two parameters. Most of the existing algorithms employ tree structures, which are observed to be inefficient, owing to the usage of pointers. Each and every node of the tree is associated with the pointer and this leads to computational, time and space complexity. This paper presents a novel approach for frequent pattern mining namely One Time Scan plus Pyramid (OTSP), which relies on single scanning process and pyramid data structure. The input database is scanned only once by OTSP and the resultant table is mounted onto the pyramid data structure. The pyramid data structure compartmentalizes the data items with respect to the degree of combination and stores it along with the occurrence frequency rate. This way of data mounting saves execution time and memory along with the reduced computational complexity. The performance of OTSP is analysed with respect to execution time, memory consumption and scalability and the OTSP yields good results.

Keywords: Frequent pattern mining, scanning, pyramid data structure.

1. INTRODUCTION

Frequent pattern mining is a branch of association rule mining, which aims at retrieving frequent data patterns from voluminous data. Detection of frequent data patterns is necessary to gain knowledge from the database, which paves way for predicting the future scenario and acts as the best solution to meet up any scenario. The scenario can either be favourable or unfavourable. This is applicable for all the domains such as healthcare, marketing, finance, retails, forecasting etc. The detection of frequent data patterns can effectively figure out the best and worst cases of the historical data, through which the expected outcome can be predicted.

In general, the pattern mining is the process that takes a database as input and the pattern is a group of entities in a database. The frequent pattern is determined by the support threshold of the pattern. Support of a pattern can be defined as the set of transactions which involves that particular pattern. Apriori and Frequent Pattern growth (FP-growth) are the most famous algorithms of frequent pattern mining [1-6]. Many enhancements have been done to apriori algorithms, in order to improve the efficiency. However, most of the apriori based algorithms experience time and space complexity.

Association rule generation is a twin step process. Initially, the frequent patterns are generated, which depends on the support value. This is followed by the generation of association rules with respect to the confidence value. Confidence is computed by the count of transactions that includes item 1 along with item 2, in a given database. An efficient pattern mining algorithm must involve least computational, time and space complexity. The time complexity can be reduced by the reduced count of scans. The computational and space complexity can be eliminated by the minimized usage of pointers.

* Assistant Professor, Department of Information Technology, *Email: minisheela@ymail.com*

** Noorul Islam University, Kumaracoil, *Email: shkrishnaveni@gmail.com*

In this paper we propose a new method for frequent pattern mining namely OTSP, which throws out the time, space and computational complexity, by incorporating the pyramid data structure. To the best of our knowledge, we are the first to implement pyramid data structure for frequent pattern mining. This work scans the database only once and this is beneficial to the computing environment. The main contributions of this paper are listed below.

- The incorporation of layer based pyramid data structure provides the highest degree of data organization.
- One time scanning ensures least computational, time and resource overhead.
- The frequent data items can easily be retrieved, owing to the highest degree of data organisation.
- The employment of pyramid data structure introduces multiple benefits to the system, which are layered data organization, efficient data retrieval, reduced time, space and resource overhead.

The performance of the OTSP is evaluated in terms of execution time, memory usage and scalability. The efficiency of the proposed work outperforms the existing techniques by means of the employment of pointers and one time scanning process. The experimental results of the proposed work are satisfactory in terms of all the performance metrics being employed.

The rest of the paper is organised as follows. Section 2 is loaded with the review of literature. The proposed approach is presented in section 3 and the performance of the proposed approach is evaluated in section 4. Finally, the concluding remarks are presented in section 5.

2. REVIEW OF LITERATURE

This section intends to present a short review of existing literature with respect to the data structures for frequent pattern mining.

2.1. Association rule mining

The base for all the association rule mining algorithms are Apriori and FP-growth [5,6]. Apriori algorithm is based on the candidate approach, which relies on the Apriori property. The usage of this property minimizes the search space however, the total count of database scans is maximal and this introduces the memory overhead. The drawback of apriori's multiple scan is overcome by FP-growth algorithm. FP-growth algorithm follows tree structure called FP-tree. The FP-tree consists of a header table to store the item name, support, links and a tree to hold the items of the database. This algorithm divides the database into several databases for the given pattern. The FP-growth algorithm scans the database only twice.

In [7], the enhancement of apriori algorithm namely UT-Miner is proposed for sparse data. The UT-Miner utilizes the array structure for data storage. Though, this algorithm could not achieve good performance in terms of time and space complexity. The optimized version of FP-growth, named as FP-growth-goethals is presented in [8]. Another improved FP-growth named as FP-growth-tiny is presented in [9]. This algorithm tends to improve the efficiency by producing conditional FP-trees with the help of conditional patterns.

CT-PRO is another algorithm which eliminated the process of recursion [10]. This is made possible by the process of appending the count array to the FP-tree nodes. The array entities denote the occurrence frequency of the item set. In [11], IFP-growth is proposed by utilizing the FP-tree+. This tree employs an address table to the FP-tree, which eliminates the need for conditional FP-trees. This results in the faster mining capability. The drawbacks of this algorithm are prior knowledge about addressing is needed and the memory overhead is not dealt with.

MAFIA-FI [12] is the algorithm that saves data in a bitmap, which in turn reduces the tree searching procedure. The 2D bitmap represents the items and transactions respectively. Thus, this algorithm is proved to be efficient in terms of pruning and tree traversals. On the other hand, this algorithm failed to address the memory overhead. FP-growth* is another algorithm that has shown better pruning efficiency by means of FP-array [13]. Though, the tree size is not attempted to be minimized by this algorithm. The works from [14-16] consider utility as another parameter along with support and confidence. However, this parameter can make sense only for retail domain and not for our work. Besides this, these works narrow down the scope of the work and in order to widen the applicability to all domains, this work doesn't consider any such special parameters. However, the proposed work is compared with these recent works.

Motivated by the above mentioned works, this paper intends to present a new method which overthrows the time, space and computational complexity, by means of pyramid structure.

3. FREQUENT PATTERN MINING BASED ON PYRAMID STRUCTURE

In this section, the detailed description of the proposed work is presented. The working principle of the proposed work is presented in the forthcoming section. The beauty of the proposed work is that this structure is applicable for all domains.

3.1. Preliminaries

Consider a transaction database *TD*, which consists of data items $I = \{i_1, i_2, \ldots i_n\}$. A transaction comprises one or more data items and every transaction is associated with a transaction id. Let p be the pattern which includes several data items.

$$p = \{i_a, i_b, \ldots i_k; a \leq 1, k \leq n\} \tag{1}$$

where 1 and n are the first and last items in *TD*.

The support of a pattern p can be computed by the occurrence frequency of p in all the transactions present in the *TD*. The term confidence is defined as the occurrence of data item c along with b. The support and confidence can be given by

$$S(A \Rightarrow B) = S(A \cup B) \tag{2}$$

$$C(A \Rightarrow B) = \frac{S(A \cup B)}{S(A)} \tag{3}$$

A pattern is considered to be frequent when the support of that pattern is greater than the support threshold. The frequent pattern mining aims at detecting all the frequently occurring patterns from the database. This work assumes that the transactions in the database are sorted.

Consider a database with number of transactions and the sample transactions are {*Tn*1; *a, b, c*}, {*Tn*2; *a, b*}, {*Tn*3; *b, c, d, e*}. On scanning the transactions, the list of items can be figured out as {*i*: *a, b, c, d, e*}. Assume that the support threshold is fixed as 60%. On analysing the transactions, it is evident that {*a, b*}and {*b, c*} appears twice out of three transactions. Thus, the support values of {*a, b*}, {*b, c*} are greater than the support threshold value. On the other hand, {*a, c*} and {*c, d*} are the least occurred items and the support value of these pairs are lower than the support threshold.

3.2. Proposed approach – OTSP

The proposed approach takes a database as input and scans it only once and arrives at different patterns. Initially, the database is scanned in a serial fashion. All the possible combinations are framed and their

occurrence frequency rates are figured out for every combination by scanning all the transactions. On reaching the end of the database, the proposed approach completes figuring out all the possible combinations being present in the transactions along with their occurrence frequency rate. This idea conserves much more time and computational complexity.

Initial scanning process

Input: Database
Output: Combinations of data items
Begin
l = last record of the database;
b = first record of the database;
for(i = 1; i ≤ l; i++) do
add unique data items into the scan table;
if (dataitem ≥ count 2)
compute the combinations being present in the transaction;
if(unique combination)
append in the scan table;
else
discard;
End;

Consider a sample database with seven transactions. Each transaction is represented by {*Tn*1, *Tn*2,…*Tn*7}. The available data items in the database are {*o*, *p*, *q*, *r*, *s*}. The process of scanning is depicted in figure 1.

Figure 1 depicts the sample scanning process of the proposed approach. As the proposed approach follows serial scanning, one transaction is read at a time. Initially, after scanning *T*1, the data items present in *T*1 are enrolled into the table with value 1. This is followed by the read process of T2, which contains {*O*, *P*, *R*, *S*}. Now, the possible two and three pair combinations, which are present both in *T*1 and *T*2 are found out. The two pair combinations being present in both *T*1 and *T*2 are *OP*, *PS*, *OS* and the repeated

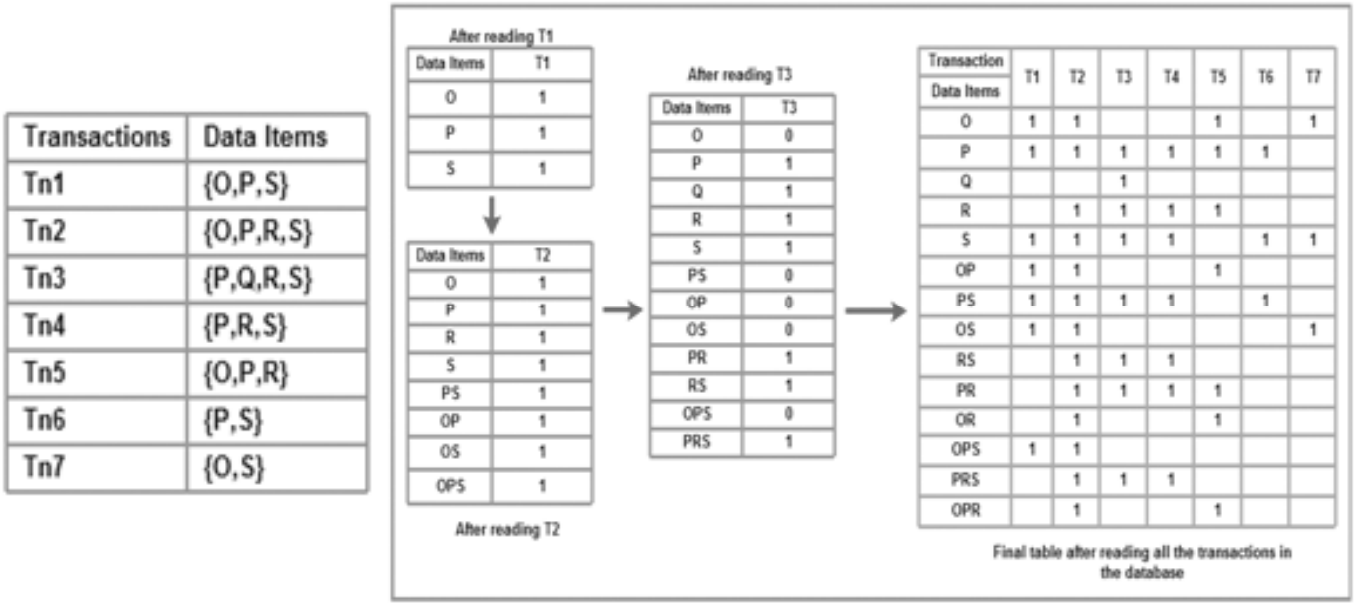

Figure 1: Process of sample scanning w.r.t the sample transactions

three pair combination is *OPS*. Similarly, this process is repeated for all the transactions and the resultant table is returned.

On keen observation, it could be noted that no combinations are generated if at all, the combination is present twice in a database. This makes sense that no combined data items can have null values on all the columns. Otherwise, this can be stated as during the process of scanning, no combinations are formed at the first occurrence. Instead, the combinations are formed only if the same combination has been scanned earlier. This reduces the memory overhead. The resultant table is loaded into the special pyramid data structure, which adds in some more advantages to the system and is presented in the forthcoming section.

3.2.1. Pyramid data structure

The resultant table is stored in the pyramid data structure, which paves way effective retrieval of data, faster execution time and least memory overhead. The pyramid data structure follows the layered architecture, which provides efficient data organization. The process of mounting the available scanned table into the pyramid data structure is depicted in figure 2.

As per the above diagram, the transactional data are carefully mounted onto the layered pyramid data structure. As the sample database contains only seven transactions the maximum possible combinations are three. Thus, the pyramid data structure has got three layers. Each layer is exclusively dedicated for different range of combinations of data items and are stored in arrays. In this case, the first layer consists of {(O, 4), (P, 6), (Q,1), (R, 4), (S, 6), the mid layer is loaded with the twin combinations such as {(OP, 3), (PS, 5), (OS, 3), (RS, 3), (PR, 4), (OR, 2)} and the final layer is mounted with triplet combinations of item sets with their occurrence frequency, which are the following {(OPS, 2), (PRS, 3), (OPR, 2)}.

This way of data organization yields multiple merits to the proposed approach. This method of data mounting shows least overhead in terms of space, time and computational complexity, whereas the degree of ease-of-use is very high. The concept of pyramid data structure does not use pointers, which improves the efficiency of the work.

Besides this, the analyst can navigate through the data, based on the degree of combination of data items. This in turn improves the performance of the system, as it can be judged the highly sold single item, a pair combination and triplet combination of data item (in case of a retail database). On the other hand, the symptoms of diseases can also be done with the same procedure. The decision can be made

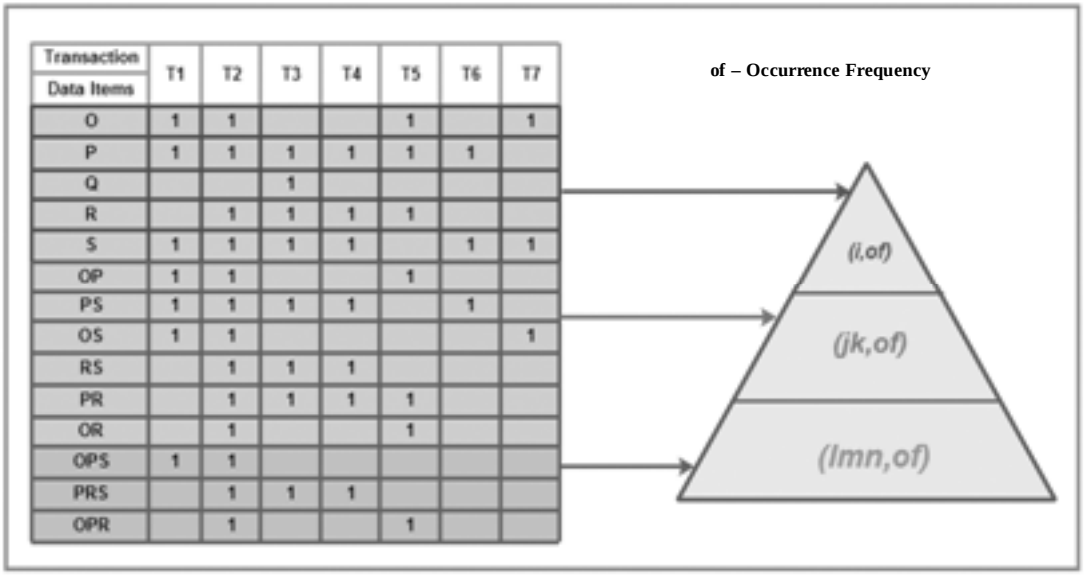

Figure 2: Data mounting in pyramid data structure

accordingly, with respect to the knowledge being gained from the database. In retail database, the marketing demand of all the products can be recognized in a single fly. This makes the decision process very easier but efficient.

The association rules can easily be framed, as the data structure possesses the data items with their occurrence frequency rate in the database. With this information, the support and confidence are easily computed and so the association rules. The single, twin and triplet or more combinations can be concerned in different categories, which idea makes the process hassle-free.

The presence of occurrence frequency in the data structure helps the system to make decision on the go and is very much useful for the analysis purpose. Besides this, the support and confidence computation processes are made easier. Intense attention can be paid over different combinations of items, which maximizes the business profit and also to withstand the marketing traits. The performance of the proposed work is evaluated and the experimental results are presented in the forthcoming section.

4. PERFORMANCE EVALUATION

The performance of the proposed approach is evaluated with respect to three important metrics such as memory consumption, execution time and scalability. The proposed approach is compared with FHM [14], CHUD [15] and UP-Growth [16].

4.1. Testing environment and datasets

The proposed approach is implemented in Java programming language and got executed in windows 7 operating system and intel core i5 processor. This work employs three different datasets for evaluating the performance against the existing works. The datasets being utilized are chess, retail and pumsb. A dataset can either be dense or sparse. Dense dataset makes sense that every transaction contains lesser count of items and mostly, the transactions resemble each other. On the other hand, the transaction of sparse datasets possesses different data items and the total count of items would be more.

The chess dataset is dense and it contains information about the chessboards. Retail is a sparse dataset and this dataset has the product sales data of retail stores. Census data of US is presented in pumsb, which is dense dataset with large amount of data. All the aforementioned datasets can be downloaded from [17]. Table 1 presents the transaction and item details of the datasets.

Table 1
Details of datasets

Datasets *Details*	*Chess*	*Retail*	*Pumsb*
File size (Mb)	0.4	4.07	16.29
Total transaction	3156	88162	49046
Total items	84	21387	7116

The performance of the proposed one-time scan with the pyramid data structure is analysed for the above mentioned datasets and the graphical results are presented below.

• Execution time analysis

The execution time of any proposed technique must be as minimal as possible. The minimal execution time paves way for several advantages such as effective resource utilization and this reduces the cost. The

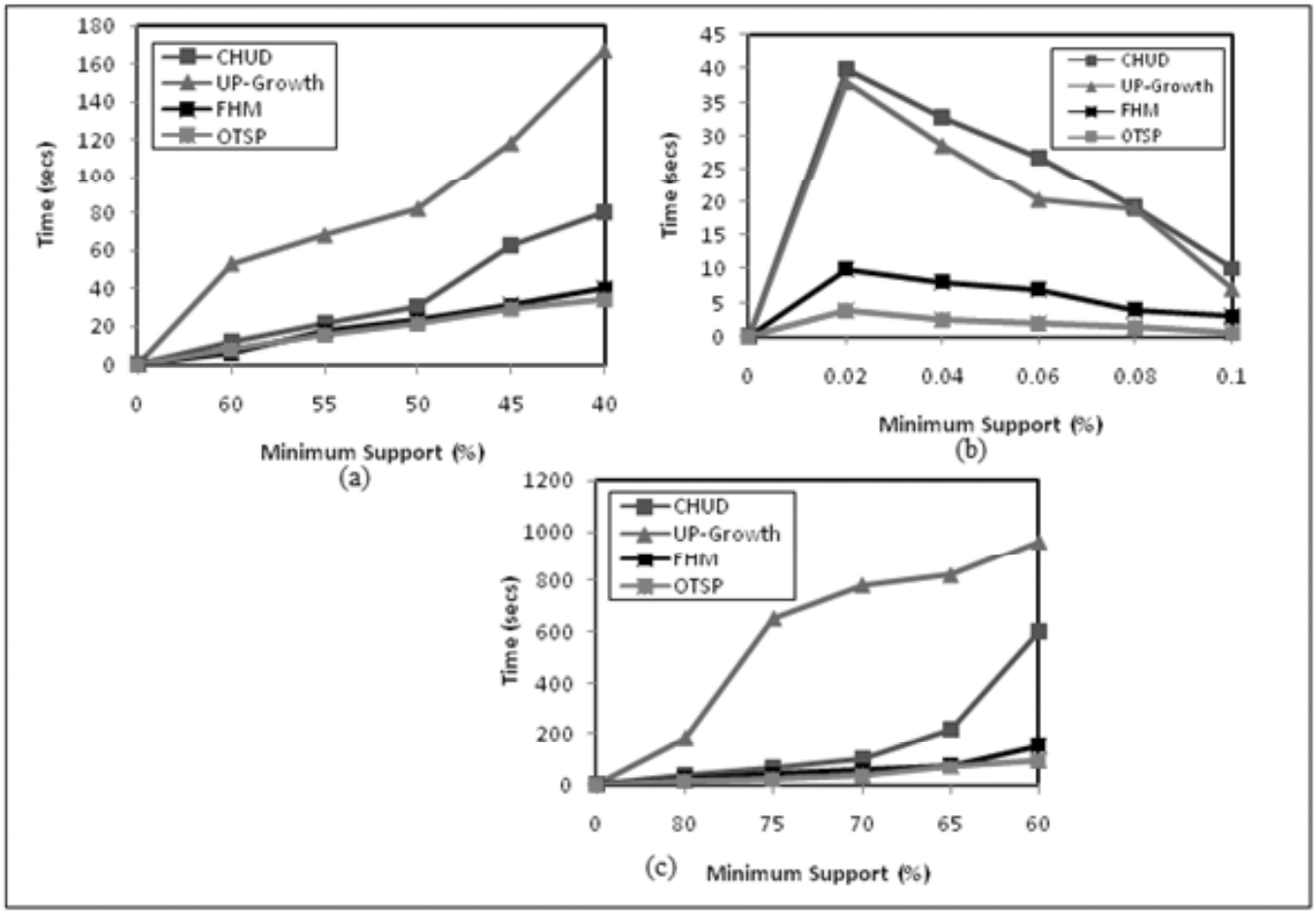

Figure 3: (a) Execution time analysis on chess dataset (b) Execution time analysis on Retail dataset (c) Execution time analysis on Pumsb dataset

techniques are evaluated on all the three datasets and the experimental results are presented below through figures 3 a, b, c.

On analysing the experimental results of the execution time, the proposed OTSP consumes lesser time for execution. This saves the resource and enhances the performance of the system. In all the three datasets, the OTSP consumes the maximum of 97 seconds for the pumsb dataset and the least time is 8 seconds for the chess dataset. The lesser execution time is achieved by the single scanning process and the layered arrangement of data items in the pyramid data structure. Additionally, the presence of occurrence frequency rate in the data structure expedites the entire process even more. Thus, the execution time of the proposed work is reasonable and the performance is satisfactory.

• Memory consumption analysis

The lesser the consumption of memory, the greater is the efficiency of the system. Least memory consumption paves way for many benefits such as reduced cost and faster execution. The memory consumption of the proposed OTSP is tested against the existing techniques with respect to chess, retail and pumsb datasets are presented from figure 4 a, b and c.

The memory consumption of the proposed OTSP is found to be minimal because of the following reasons. OTSP involves single scanning process which considerably reduces the memory overhead. Besides this, the organized structure of pyramid data structure utilizes the memory effectively, which further improves the memory conservation. On comparative analysis, OTSP outperforms the existing techniques by means of lesser memory consumption.

• Scalability analysis

The scalability of the OTSP is tested with respect to the different counts of data items. The experimental results are compared with the existing techniques and the graph is presented in figure 5.

The OTSP proves its scalability also with respect to different items. On the whole, the proposed work shows better performance by following the single scan process and the incorporation of the pyramid

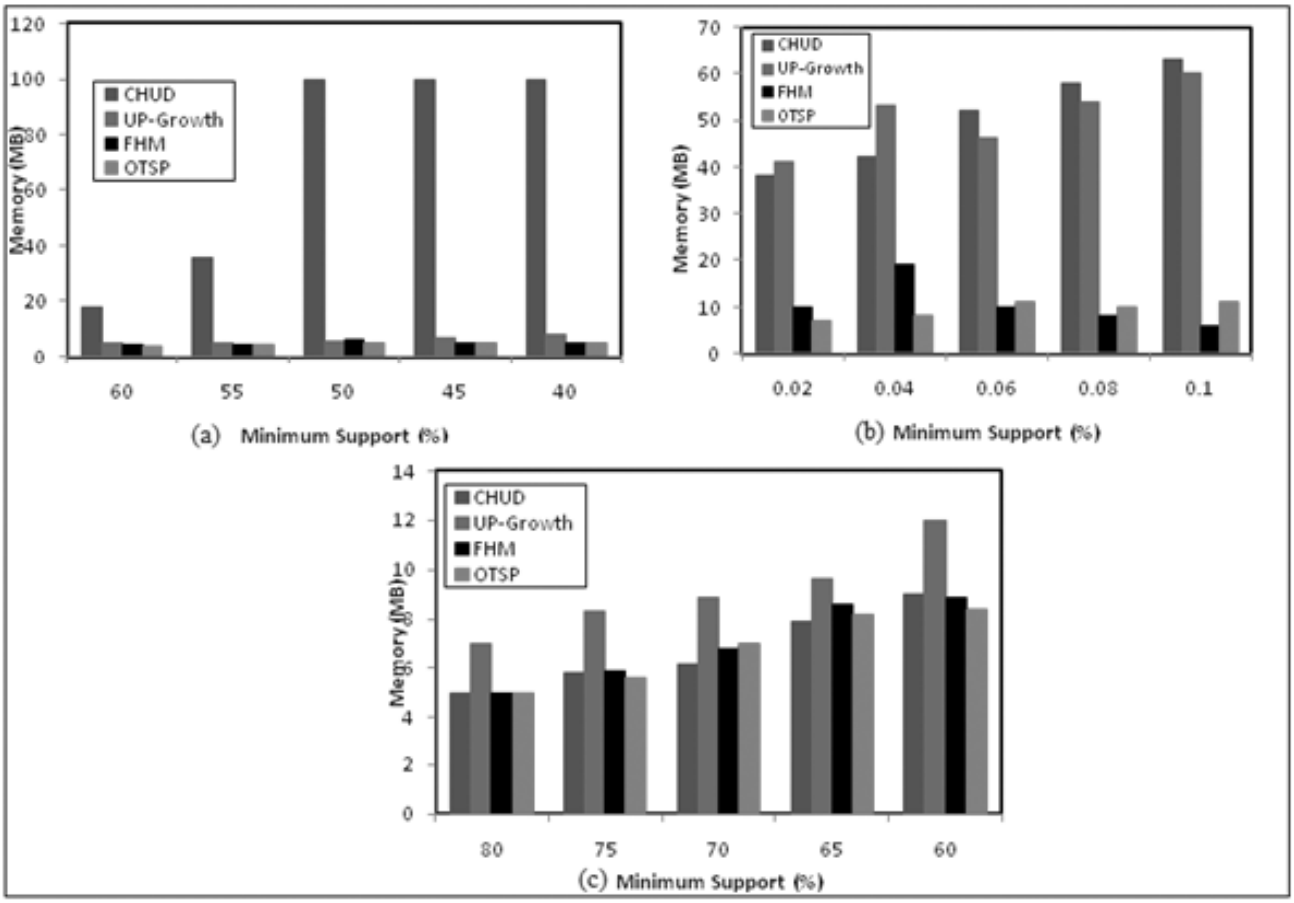

Figure 4: (a) Memory consumption analysis on chess dataset, (b) retail dataset, (c) pumsb dataset

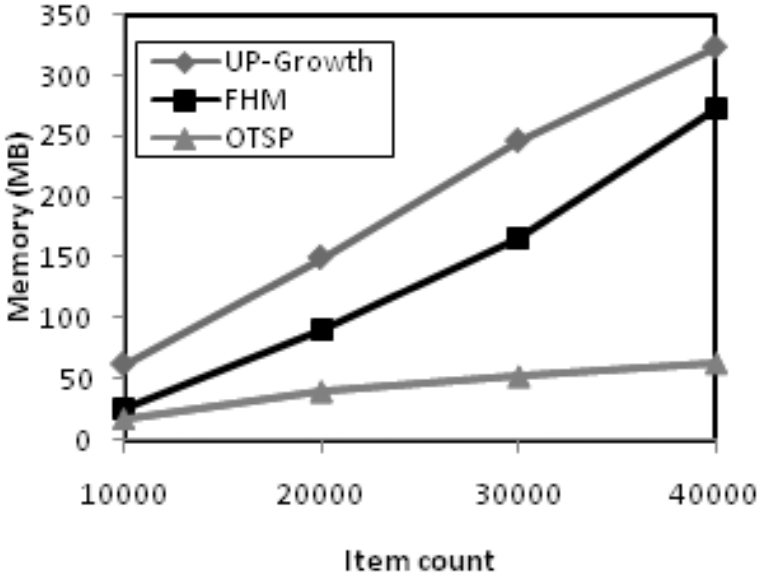

Figure 5: Scalability testing

data structure. The layered pyramid structure provides room to all the data items, with respect to the increasing degree of combination. This way of data organization makes all the associated processes such as support and confidence computation very easier. This in turn saves the execution time and memory along with scalability. Thus, the main objective to overthrow time, space and computational complexity is achieved by OTSP.

5. CONCLUSION

This paper presents a novel approach for effective frequent pattern mining namely OTSP. The root of the proposed work relies on the pyramid data structure and the single scanning process. These two flavours improve the performance of OTSP. The single scanning process of database ensures lesser execution time along with memory conservation. The memory is conserved because any combination of data items is considered, only if it occurs twice. The incorporation of pyramid data structure introduces layered room for the data items with respect to the degree of data item combination, along with the occurrence frequency rate of the combination. This simplifies the computation of support and confidence and so the association rules can be framed very faster. Besides this, OTSP is not restrained by any particular domain, which widens the scope of this work.

References

[1] H.F. Li, S. Lee, Mining top-K path traversal patterns over streaming web click sequences, Journal of Information Science and Engineering, vol. 25, no. 4, pp. 1121-1133, 2009.

[2] Y. Chen, W. Peng, S. Lee, CEMiner – an efficient algorithm for mining closed patterns from time interval-based data, in: International Conference on Data Mining (ICDM), pp. 121–130, 2011.

[3] M. Hamada, K. Tsuda, T. Kudo, T. Kin, K. Asai, Mining frequent stem patterns from unaligned RNA sequences, Bioinformatics, vol. 22, no. 20, pp. 2480–2487, 2006.

[4] S. Ruggieri, Frequent regular itemset mining, Knowledge Discovery and Data Mining (KDD), pp. 263– 272, 2010

[5] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, Very Large Data Bases(VLDB), pp. 487–499, 1994.

[6] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent pattern tree approach, Data Mining and Knowledge Discovery (DMKD), vol. 8, no. 1, pp. 53-87, 2004.

[7] F.Y. Ye, J.D. Wang, B.L. Shao, New algorithm for mining frequent itemsets in sparse database, in: Proc. the Fourth International Conference on Machine Learning and Cybernetics, pp. 1554–1558, 2005.

[8] B. Goethals. <http://adrem.ua.ac.be/~goethals/software/>.

[9] E. Ozkural, C. Aykanat, A space optimization for FP-growth, in: FIMI '04 Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, November 2004.

[10] Y.G. Sucahyo, R.P. Gopalan, CT-PRO: a bottom-up non recursive frequent itemset mining algorithm using compressed FP-tree data structure, in: FIMI '04 Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, November 2004.

[11] K. Lin, I. Liao, Z. Chen, An improved frequent pattern growth method for mining association rules, Expert System with Applications (ESWA), vol. 38, no. 5, pp. 5154-5161, 2011.

[12] D. Burdick, M. Calimlim, J. Flanick, J. gehrke, T. Yiu, MAFIA: a maximal frequent itemset algorithm, Transactions on Knowledge and Data Engineering (TKDE), vol. 17, no. 11, pp. 1490-1503, 2005.

[13] G. Grahne, J. Zhu, Fast algorithms for frequent itemset mining using FP-trees, Transactions on Knowledge and Data Engineering (TKDE), vol. 17, no. 10, pp. 1347-1362, 2005.

[14] Fournier-Viger, P., Gomariz, A., Soltani, A., Gueniche, T., SPMF: Opensource data mining library. Available at <http://www.philippe-fournierviger.com/spmf/>, 2014

[15] Wu, C. W., Fournier-Viger, P., Yu, P. S., & Tseng, V. S. Efficient algorithms for mining the concise and lossless representation of closed+ high utility itemsets. IEEE Transactions on Knowledge and Data Engineering, 99(PrePrints), 1, 2014

[16] Tseng, V. S., Shie, B.-E., Wu, C.-W., & Yu, P. S. Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering, 25(8), 1772–1786,2013.

[17] http://fimi.ua.ac.be/data/

[18] S.H. Krishnaveni, K.L. Shunmuganathan, L. Padma Suresh. A Novel NSCT based Illuminant Invariant Extraction with Optimized Edge Detection Technique for Face Recognition.International Journal of Computer Applications, vol 70, no. 3,0975-8887, 2013.