IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-6, June 2019 E-ISSN: 2347-2693

Mining High Utility Pattern from Sequential Database

A. A. Tale[1*], N. R. Wankhade[2]

[1, 2]Late G. N. Sapkal College of Engineering, Nashik, India

[*]*Corresponding Author: ankittale@hotmail.com Tel.: +91-7798-495602*

DOI: https://doi.org/10.26438/ijcse/v7i6.529533 | Available online at: www.ijcseonline.org

Accepted: 08/Jun/2019, Published: 30/Jun/2019

Abstract**—** Now-a-days, finding an interesting pattern from the given dataset is an emerging trend to learn more about user behaviour and patterns of interest. Prior work on this problem many pattern mining approaches use two-phase pattern mining with one exception that are however inefficient and scalable to mine high utility sequential pattern mining. The way mention above suffers scalability issue for numerous candidates and growing sequence. This paper proposes an approach to apply tight upper bound for pruning patterns. Whereas, the freshness lies in the implemented algorithm that helps to prune tight sequence utility. The applied data structure helps us to maintain sequence patterns whose values are greater than applied thresholds. Extensive experiments on real datasets show that the defined algorithm is able to mine high utility sequential pattern incrementally.

Keywords**—**Data-mining, High Utility Patterns, Sequential Pattern Mining, Pattern Mining, Pruning, Itemset share framework.

I. Introduction

Discovery of an interesting pattern, and sequential pattern has been keyed tasks of data mining, which have an assortment of application such as analysis of genome in sequential patterns, analyzed web-access log and inventory management prediction where interestingness is measured.

Most of the utility mining algorithm employs candidate generations for and normal, or sequential patterns. A high utility sequential pattern is described as the arrangement of product in such an interesting pattern that crop maximum profit from an implementation. Whereas, in a habit, a sequence database is incrementally updated over time. For example, in a mobile subscriber database, a series of the new transaction are made by a new customer will create a new sequence infused in subscriber's database and those transactions are made to affix in the database corresponding to a sequence already in the present database.

In this paper, we address both of the above challenges by introducing new mining algorithm with novel data structure to prune out efficient data from a database sequence.

1. Essentially, we propose a tight utility upper bound of a sequence to improve pattern mining expertly by removing more sequences that was not able to complete thresholds with an implemented algorithm.

2. To aid, incremental high utility sequential pattern mining, we propose a data structure that would generate a candidate pattern tree to maintain each sequence whose transaction is greater than the threshold value.

3. Our algorithm helps us to incrementally mine out high sequential patterns based on their sequential utility. An experiment is carried out to evaluate the performance of a proposed algorithm, a result shows that it outperforms state another algorithm to mine sequential patterns.

The rest of the paper is organized as follows. Section 2 shows information about the literature survey. Section 3 show the propped system and discuss it in briefly with implementation. Section 4 defines final result and analysis observed which mining patterns. Section 5 defines conclusion. Section 6 describe future implementation, and conclude it.

II. Related Work

A. *Frequent Pattern Mining*

Frequent pattern mining is interrelated to High Utility Pattern Mining, including constraint-based mining. In upcoming segments, we will briefly review work in utility mining and on frequent pattern mining.
Initially, we will review frequent pattern mining initially proposed by Agarwal, where he proposed an efficient algorithm that generates all significant associations rules between interrelated item & database. Where an algorithm contains buffer management and a fictitious pruning

approach to obtained desired results, it employs an anti-monotonicity property, where the support of a superset of the pattern at most the support of pattern. Apriori from Agarwal & Shrikant et.al[7] is renowned bread-first search algorithm for mining frequent pattern is the disk-based database as far as frequent pattern length. FP-growth by Han is another renowned depth-first search algorithm which compresses a database based on FP-Tree in main memory. Eclat is another algorithm proposed by Zaki et.al is the famous hybrid database, that keeps a database or a database partition in a vertical tid-list in either breath-first list and depth-first search and vice-versa. Also, Elact explores such a breadth-first search strategy. Whereas depth-first search strategy is less memory intensive as compared to breadth-first search.

B. Constraint Based Pattern Mining

Constraint-based mining is a milestone derived from frequently based mining mainly focus on how to push constraint on Frequent Patterns Mining Algorithm.

Pei et.al[xx] discussed the constraints based on weighted support and observed the property is known as convertible anti-monotonicity, by arranging in descending weight orders. They also demonstrate how to push constraint into FP-growth algorithms. Bucila et.al considered the pattern that satisfies partnership of anti-monotone and monotone constraint and proposed an algorithm, DualMiner which prune search space constraints.

Bonchi et.al [10] introduced and Ex-Ante property that states if any transaction which does not satisfy for given monotone constraints should be removed or dropped from transactions. For a further, extends Bonchi & Goethals et.al [11] applied an Ex-Ante property on FP-growth algorithm for pruning search space. Bonchi and Lucchese et.al [12] conclude the reduction technique to a unified framework. De Readt et.al. inspected how standard constraints programming techniques can be applied to constraint-based problems that are antimonotone, monotone and convertible. Bayardo and Agarwal et.al [9], Morishita and Sese et.al applied a technique for pruning search space based on upper bound when constraints are not monotone & anti-monotone or convertible. Based on this implementation we are applying the technique to prune search space based on tight upper bound in given implementations.

III. METHODOLOGY

Every mining approach uses two-phase candidate generation approach, further our architecture proposed incremental mining high utility sequential pattern approach as the initial phase, and incremental phase. In the initial phase, we propose and utility pattern miner to generate transaction utility sequence and prefix extension utility to mine out high utility sequential pattern. To obtain better achievement in the incremental phase we propose data structure that helps to generate the candidate tree for mining high utility sequential patterns.

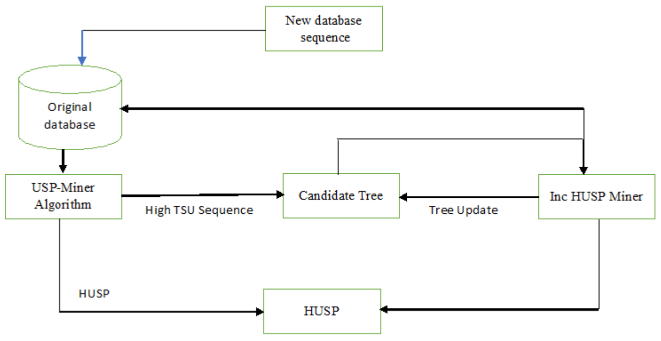

Fig. 3.1. Implementation

When a new sequence is inserted into the database (D), our proposed algorithm for incremental high utility sequential pattern is employed to mine the new high sequential utility from original database D. The key idea of our proposed algorithm is to Incremental High Utility Sequential Pattern is to traverse each node of tree T is based on depth first search approach and skip that node which does not show up in the updated sequence.

The implementation of USP-Miner algorithm and Inc-USP Miner are defined in below sections.

A. Efficient Utility Calculation

In the above algorithm, we calculate efficient utility calculation, where s and t are two sequences where t is a subset of s. Innocent way to obtain the utility of each occurrence and obtained the maximum utility of all instances. This motivates us to design efficient way to calculate maximum utility.

Assumption 1:
Given two arrangements s and t where t is a subset of s, suppose that s has multiple instances of t and let the set of the distinct extension positions of sequence t in s be P = {p1, p2..., pq}. The utility oft in s (i.e., u (t, s)) equals to max {u (t, pi, s) |∀pi ∈P}.

Theorem 1: Given a sequence t, one t’s i-extension sequence t', and a sequence s in D where t0 v s, the maximum utility of t' in s at extension position p (i.e., u(t', p, s)) can be derived by u(t0,p,s) = u(t, p, s)+u(i0,p,s) where item i' is the extension item of t'.

Theorem 2: Given a sequence t, one t’s s-extension sequence t', and a sequence s in D where t' subset of s, let item i' be the extension item of t' and p be an extension position of t'. Also let P = {p1, p2... pq}be the set of extension positions oft in s.

The maximum utility of t' in s at extension position p can be derived from u (t', p, s) = max {u (t, pi, s) |∀ pi ∈P and pi < p} +u (i0, p, s).

Theorem 1 and 2 moreover provides efficient way of mining utility patterns. But according to assumption 1, it is efficient to derive utility from each instance of node and transaction. Whereas, a candidate tree is an expansion of lexicographic tree buffering of set D for incremental high utility sequential pattern. TSU is having a tight upper bound of utilities. Hence the transaction sequence utility guide to the construction of candidate tree T in the first phase.

B. The Intial Phase

In initial phase we define our USP-miner algorithm that help us to mine out candidate tree. Efficient utility calculation, where s and t are two sequences where t is a subset of s. An innocent way to obtain the utility of each occurrence and obtained the maximum utility of all instances. This motivates us to design efficient way to calculate maximum utility.

A candidate tree is an expansion of lexicographic tree buffering of set D for incremental high utility sequential pattern. We also apply tight upper bound to prune only desired sequence which are able to generates the sequential patterns also, TSU is having a tight upper bound of utilities. Below case needs to consider:

Case 1: t is a high TSU sequence in D and D'.
Case 2: t is a low TSU sequence in D and becomes a high TSU sequence in D'.
Consider below algorithm for generation of candidate tree node.

Algorithm:

```
Input:
A sequence database, minimum thresholds, and an empty set of high utility sequential pattern, node sequence.
Output: Tree Structure of Candidate Pattern

Begin
1. For each tree node, check if prefix utility pattern is greater than user minimum utility.
2. Buffered that node into generating a tree, else remove the node if it does not satisfy minimum thresholds.
3. Recursively sort out each transaction to mine candidate tree.
End
```

C. The Incremental Phase

The updated database D obtained from an initial phase is are partition into a disjoint set. After the database D is updated with the new sequence, we need to adjust the candidate tree T to support multiple database updates. Following algorithm is used to incrementally mine out high utility transaction

Algorithm:

```
Input: A sequence database D', minimum thresholds, a candidate pattern tree T from Initial Phase and node sequence.
Output: High Utility Sequential Pattern.

Begin:
1. If all sequence in the database in transaction t is empty sequence skip those sequence.
2. Calculate Transaction Sequence Utility of each extension sequence t in database D.
3. Based on Case 1, the algorithm always updates the node inserted into the database to generate a HUSP Tree.
4. Based on Case 2, our algorithm builds the nodes insert into them and recursively calls USP miner to expands and generate HUSP tree.
End
```

In this implementation, the utility of a sequence is buffered in the tree T has to be updated, to obtain the up-to-date utility. We can skip those sequence which are not able to presence in provided updated sequence.

IV. Results and Discussion

We compare our algorithm with the best state-of-art-algorithm on varying data characteristics, including different utility distribution changing data sized, different average length of transactions.

First, we generate efficient utility calculation from input transactions. Below graph shows the running time of the algorithm on the respective data-set. The running time with efficient utility anti-proportional to supports. For every utility that was generated, our developed algorithm takes less amount of time as compared to state-of-the-art-algorithm.
After that, we conduct a comparative experiment with the number of items ranging from 1K to 5K. However, the running time of other algorithm increases as there is an increase in the candidate.

Finally, we evaluate our algorithm. Based on transaction length by comparing results. As depicted the time increase with the average length of the transaction since the average length, and number of high utility sequence also increases, that why we see a gasp among the time interval for running an algorithm.

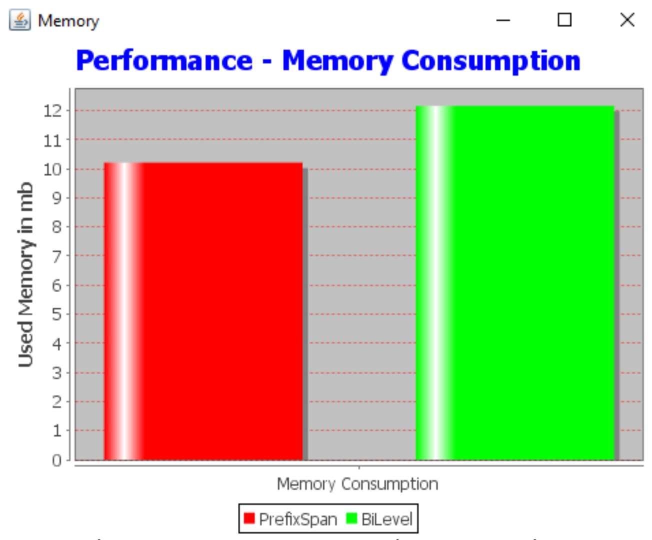

Fig: 4.1 Memory Consumption Comparison

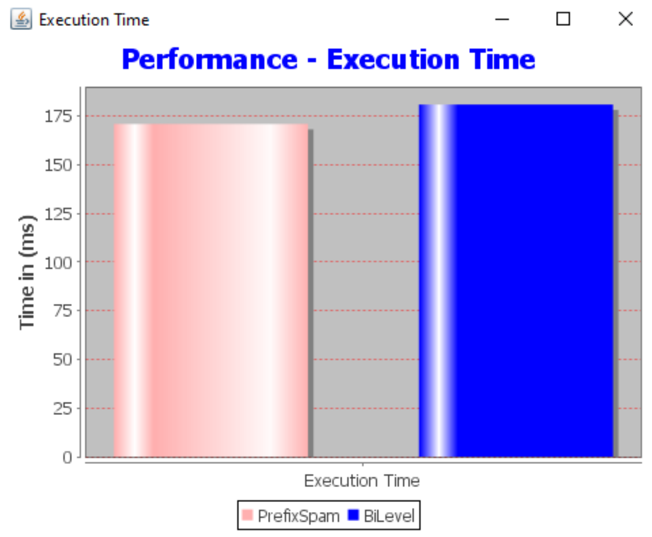

Fig: 4.2 Execution Time Comparison

From above figure 4.1 and 4.2 we can conclude that the implemented system requires less amount of memory transaction as compared to an existing transaction, also we required less amount of time to generate sequential pattern and which can be said less prone to scalability as contrasted to the present system based on the improvement in database.

V. CONCLUSION AND FUTURE SCOPE

This paper proposed a new algorithm to mine out incrementing high utility sequence patterns, with tight upper bound pruning which help us to generate sequential patterns. Our contribution includes a data structure to generate a candidate tree to maintain each sequence whose TSU is greater than the minimum thresholds, an incremental algorithm HUSP for mining patterns based on depth-first search approach. We also applied tight upper bound pruning to increase efficiency by eliminating more sequence that is not able to high utility compared to an existing high utility sequence pattern. In the future, we can work on high utility sequential pattern mining for parallel and distributed database and their application in big data.

REFERENCES

[1] J.Liu, Ke Wang, Benjamin C.M.Fung, "Mining High Utility Patterns in One Phase without Generating Candidates", IEEE Trans.Knowl. Data Eng., vol. 28, no.5, pp-1245-1247, May 2016.

[2] S. Dawar and V. Goyal, "UP-Hist tree: An efficient data structure for mining high utility patterns from transaction databases," in Proc. 19th Int. Database Eng. Appl. Symp., 2015, pp. 56–61.

[3] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," IEEE Trans. Knowl. Data Eng., vol. 25, no. 8, pp. 1772–1786, Aug. 2013.

[4] A. Erwin, R. P. Gopalan, and N. R. Achuthan, "Efficient mining of high utility itemsets from large datasets," in Proc. 12th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2008, pp. 554–561.

[5] H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction databases," Data Knowl. Eng., vol. 59, no. 3, pp. 603–626, 2006.

[6] R. Agarwal, C. Aggarwal, and V. Prasad, "Depth first generation of long patterns," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2000, pp. 108–118

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. 20th Int. Conf. Very Large Databases, 1994, pp. 487–499.

[8] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Trans. Knowl. Data Eng., vol. 21, no. 12, pp. 1708– 1721, Dec. 2009.

[9] R. Bayardo and R. Agrawal, "Mining the most interesting rules," in Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 1999, pp. 145–154.

[10] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "ExAnte: A preprocessing method for frequent-pattern mining," IEEE Intell. Syst., vol. 20, no. 3, pp. 25–31, May/Jun. 2005.

[11] F. Bonchi and B. Goethals, "FP-Bonsai: The art of growing and pruning small FP-trees," in Proc. 8th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2004, pp. 155–160.

[12] F. Bonchi and C. Lucchese, "Extending the state-of-the-art of constraint-based pattern discovery," Data Knowl. Eng., vol. 60, no. 2, pp. 377–399, 2007.

[13] T. De Bie, "Maximum entropy models and subjective interestingness: An application to tiles in binary databases," Data Mining Knowl. Discovery, vol. 23, no. 3, pp. 407–446, 2011

[14] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility cooccurrence pruning," in Proc. 21st Int. Symp. Found. Intell. Syst., 2014, pp. 83–92.

[15] Y.-C. Li, J.-S. Yeh, and C.-C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," Data Knowl. Eng., vol. 64, no. 1, pp. 198–217, 2008.

Authors Profile

Mr. Ankit A Tale pursed Bachelor of Computer Engineering from University of Pune, India in 2014. He is currently pursuing Master of Computer Engineering from University of Pune, India and currently working as Developer in Aress Software Pvt Ltd ,since 2018. He has published one research papers in reputed international journals and it’s also available online. His main research work focuses on Algorithms, Machine Learning, Artifical Intellegence, Big Data Analytics, Data Mining, IoT based education. He has 2+ years of software development experience.

Mr N R Wankhade pursed Bachelor of Computer Engineering from University of Amravati in year 2009. He is currently pursuing Ph.D. and currently working as Professor in Department of Computer Engineering, University of Pune, India since 2005. He is a member of IEEE & IEEE computer society since 2013. He has published more than 20 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it’s also available online. His main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy, IoT and Computational Intelligence based education. He has 20 years of teaching experience and 4 years of Research Experience.