Performance Analysis of Tree-Based Algorithms for Incremental High Utility Pattern Mining

Heungmo Ryang and Unil Yun(✉)

Department of Computer Engineering, Sejong University, Seoul, Korea
ryang@sju.ac.kr, yunei@sejong.ac.kr

Abstract. To overcome drawbacks of traditional pattern mining such as difficulty reflecting characteristics of real-world databases to pattern mining, high utility pattern mining has been proposed and researched. Since database sizes become larger incrementally in many real-world applications, there is a need of appropriate methods to deal with such databases for discovering useful information from them efficiently. For this purpose, various approaches have been suggested. In this paper, we compare and analyze algorithms for high utility pattern mining from dynamic databases by considering characteristics of incremental databases and utilizing tree-based data structures. Moreover, we study their characteristics and direction of improvements based on experimental results of performance evaluation.

Keywords: Data mining · Pattern mining · Utility mining · Incremental mining · High utility patterns · Tree-Based algorithms

1 Introduction

Pattern mining is one of data mining techniques to discover useful information from huge databases, and it extracts such information in pattern forms. Although traditional pattern mining [1, 3] has played an significant role in the data mining field, this approach not only treats all items in databases with the same importance but also represents item occurrence as a binary form, i.e., 0 or 1. In addition, it cannot reflect characteristics of real-world databases to pattern mining completely. High utility pattern mining has been proposed and researched to overcome the limitations. Furthermore, database sizes become larger incrementally in various real-world applications and previous general static methods are not suitable for finding meaningful information efficiently from such databases. Incremental high utility pattern mining [2, 4, 6] has been studied to mine essential information from dynamic databases by considering real characteristics of them.

The remainder of this paper is organized as follows. In Sect. 2, we describe influential researches related to tree-based incremental high utility pattern mining. In Sect. 3, we analyze characteristics of recent tree-based methods for incremental high utility pattern mining through experiments for performance evaluation with real datasets and study direction of improvements based on experimental results. Lastly, in Sect. 4, we summarize contributions of this paper.

J.J. (Jong Hyuk) Park et al. (eds.), *Advances in Computer Science and Ubiquitous Computing*, Lecture Notes in Electrical Engineering 421, DOI 10.1007/978-981-10-3023-9_20

2 Related Work

In this section, we describe two types of related studies: traditional frequent pattern mining and high utility pattern mining for static databases.

2.1 Frequent Pattern Mining

Apriori [1] and FP-Growth [3] are well-known BFS and DFS frequent pattern mining algorithms respectively. The former utilizes a candidate generation-and-test approach, which causes the performance degradation since this method makes a large number of candidates and requires multiple database scans. The latter conducts a series of mining processes through only two database scans without candidate generation by employing a divide-and-conquer manner. In general, FP-Growth-like algorithms outperform Apriori-based ones. In pattern mining, meanwhile, the anti-monotone property [1] is used for efficient mining. This suggests that if a pattern is not valid due to its smaller frequency than a given threshold, then its super patterns are not also valid. When an invalid pattern is found in mining processes, the pattern's search space is can be eliminated, which improves mining efficiency. Therefore, satisfying this property is an essential criterion in pattern mining.

2.2 High Utility Pattern Mining

In utility mining, maintaining the anti-monotone property is not easy and Two-Phase [5] is the first algorithm that satisfies the property by applying the overestimation concept, called transaction weighted utilization (twu), to mining processes. This model generates candidates with no smaller overestimation values than a given threshold in the first phase and then identifies actual high utility patterns by computing their utility values through an additional database scan. Since the twu concept was suggested, although various static algorithms with the overestimation model have been developed, they are not appropriate for handling dynamic databases.

3 Tree-Based Incremental High Utility Pattern Mining

In contrast to the previous static approaches where the whole mining processes have to be conducted whenever a target database is increased, incremental high utility pattern mining methods simply reflect new information to the current data structures for dealing with the dynamic database. In incremental high utility pattern mining, algorithms build global data structures with the original databases, generate candidate patterns, and identify actual high utility patterns from the candidates. After that, when new transaction information is added to the original databases, the incremental approaches update the data structures, optimize them, and conduct mining processes. Figure 1 shows a simple overall process of incremental high utility pattern mining. In the figure, there are two types of data, the original and incremented ones. As explained

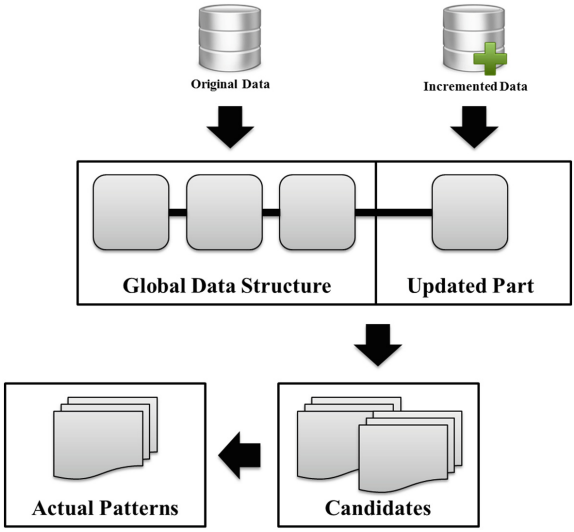

Fig. 1. Simple overall process of incremental high utility pattern mining

above, a set of high utility patterns is mined from the original data and then updated pattern results are extracted by handling only the incremented data.

FUP-HU [4] is an Apriori-based incremental high utility pattern mining algorithm and can reflect characteristics of real-world databases to pattern mining. However, this method requires a lot of database scans and operations. To address this issue, FP-Growth-based algorithms with tree data structures, IHUP [2] and HUPID [6], were proposed. IHUP applies the overestimation model of the Two-Phase algorithm [5] to its mining processes. For incremental high utility pattern mining, IHUP firstly constructs a tree data structure through a single database scan, extracts candidate patterns, and identifies actual high utility patterns from them with an additional scan. Although this algorithm mines valid patterns faster than the previous Apriori-based ones by utilizing the FP-Growth approach, it still generates a large number of candidates and consumes high computational time for the identification of actual patterns. To address this issue, HUPID extracts the fewer number of candidate patterns by reducing the overestimation values, through which it improves mining performance of tree-based incremental high utility pattern mining.

4 Performance Analysis

In this section, we evaluate mining performance of tree-based incremental high utility pattern mining algorithms, IHUP [2] and HUPID [6], in terms of runtime and the number of generated candidate patterns with the Chain-store dataset (http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html), where real utility information is included. All performance experiments were conducted on an experimental environment with a 4.0 GHz Intel processor and 32 GB memory, running the 64bit Windows 7 OS.

For the performance evaluation in an incremental environment, we first used only 20 % of the dataset, and then added each 20 % of the rest to the initial data. With the incremental data, we repeatedly constructed tree data structures of the algorithms, restructured them, and conducted mining processes.

Figure 2 shows experimental results of the performance evaluation for IHUP and HUPID with the incremental Chain-store dataset in terms of runtime and the number of candidates when a minimum utility threshold is 0.03 %. From the figure, we can observe that runtime is proportional to the number of extracted candidate patterns. In the results, moreover, HUPID mines the same number of high utility patterns faster than IHUP by generating the smaller number of candidates.

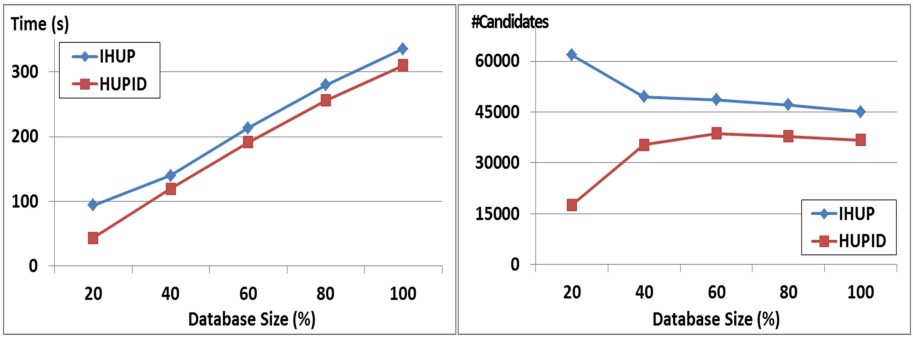

Fig. 2. Experimental results on chain-store

From the above experimental results, we can learn that performance of incremental high utility pattern mining algorithms depends on how many candidate patterns are extracted, and as a result developing an effective method for decreasing their number can be a good future work.

5 Conclusion

This paper studied characteristics of tree-based incremental high utility pattern mining algorithms and direction of their improvements based on experimental results for performance evaluation. As a result, we could learn that mining performance of an algorithm that generated the smaller number of candidates outperformed the other one and decreasing the number of extracted candidates can be a significant factor to improve performance of incremental high utility pattern mining.

Acknowledgments. This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF No. 20152062051 and NRF No. 20155054624).

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, Santiago de Chile, pp. 487–499 (1994)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. **21**(12), 1708–1721 (2009)
3. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. iData Min. Knowl. Disc. **8**(1), 53–87 (2004)
4. Lin, C.-W., Lan, G.-C., Hong, T.-P.: An incremental mining algorithm for high utility itemsets. Expert Syst. Appl. **39**(8), 7173–7180 (2012)
5. Liu, Y., Liao, W.-K., Choudhary, A.N.: A two-phase algorithm for fast discovery of high utility itemsets. In: Advances in Knowledge Discovery and Data Mining, Hanoi, pp. 689–695 (2005)
6. Yun, U., Ryang, H.: Incremental high utility pattern mining with static and dynamic databases. Appl. Intell. **42**(2), 323–352 (2015)