Knowledge-Based Systems 104 (2016) 106–122

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

An efficient algorithm for mining the top-*k* high utility itemsets, using novel threshold raising and pruning strategies

Quang-Huy Duong[a,*], Bo Liao[a], Philippe Fournier-Viger[b], Thu-Lan Dam[a,c]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[b] School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, Guangdong, 518055, China
[c] Faculty of Information Technology, Hanoi University of Industry, Hanoi, Vietnam

ARTICLE INFO

Article history:
Received 3 December 2015
Revised 15 April 2016
Accepted 16 April 2016
Available online 19 April 2016

Keywords:
High utility itemset mining
Top-k mining
Threshold raising strategies
Co-occurrence pruning
Transitive extension pruning
Coverage

ABSTRACT

Top-*k* high utility itemset mining is the process of discovering the *k* itemsets having the highest utilities in a transactional database. In recent years, several algorithms have been proposed for this task. However, it remains very expensive both in terms of runtime and memory consumption. The reason is that current algorithms often generate a huge amount of candidate itemsets and are unable to prune the search space effectively. In this paper, we address this issue by proposing a novel algorithm named kHMC to discover the top-*k* high utility itemsets more efficiently. Unlike several algorithms for top-*k* high utility itemset mining, kHMC discovers high utility itemsets using a single phase. Furthermore, it employs three strategies named RIU, CUD, and COV to raise its internal minimum utility threshold effectively, and thus reduce the search space. The COV strategy introduces a novel concept of *coverage*. The concept of coverage can be employed to prune the search space in high utility itemset mining, or to raise the threshold in top-*k* high utility itemset mining, as proposed in this paper. Furthermore, kHMC relies on a novel co-occurrence pruning technique named EUCPT to avoid performing costly join operations for calculating the utilities of itemsets. Moreover, a novel pruning strategy named TEP is proposed for reducing the search space. To evaluate the performance of the proposed algorithm, extensive experiments have been conducted on six datasets having various characteristics. Results show that the proposed algorithm outperforms the state-of-the-art TKO and REPT algorithms for top-*k* high utility itemset mining both in terms of memory consumption and runtime.

1. Introduction

Frequent itemset mining (FIM) [1–4] is a fundamental research topic in data mining. It consists of discovering the frequent itemsets appearing in a transactional database. A frequent itemset is a group of items having a support (occurrence frequency) that is no less than a user-specified minimum support threshold. Numerous algorithms have been developed to discover frequent itemsets efficiently [5–12]. Most of them are based on two well-known representative algorithms: Apriori [1] and FP-Growth [4]. The Apriori algorithm employs a level-wise candidate generation-and-test approach. A drawback of Apriori-based algorithms is that they scan the database multiple times and can generate a huge amount of candidates. This greatly reduces the performance of Apriori-based algorithms. The FP-Growth algorithm relies on an efficient tree structure named FP-Tree, which is a compact representation of the database. To discover frequent itemsets, FP-Growth first builds an FP-Tree, and then uses a pattern-growth approach to discover the frequent itemsets directly from the FP-tree, without generating candidates. FP-Growth-based algorithms generally outperform Apriori-based algorithms. In FIM, the downward closure property [1,4] is a well-known property, used to reduce the search space. This property states that if an itemset is infrequent, all its supersets are also infrequent. It is used by almost all FIM algorithms to prune the search space. FIM has a wide range of applications, and has been applied to various types of databases such as transactional databases [13], streaming databases [14], uncertain databases [15], and time-series databases [16]. Some representative applications of FIM are web analysis [17] and bioinformatics [18].

An important limitation of FIM is that in real-world applications, specifying the minimum support threshold required by FIM algorithms is not an easy task for users, since users often do not know what threshold values are the best for their requirements [19]. But choosing an appropriate threshold value is crucial

* Corresponding author. Tel.: +84936240829.
E-mail addresses: huydqyb@gmail.com (Q.-H. Duong), boliao@yeah.net (B. Liao), philfv@hitsz.edu.cn (P. Fournier-Viger), lanfict@gmail.com (T.-L. Dam).

http://dx.doi.org/10.1016/j.knosys.2016.04.016
0950-7051/

since it affects the number of frequent itemsets found by FIM algorithms. If the threshold is set too high, few frequent itemsets will be found, and thus many interesting itemsets will be missed. But if the threshold is set too low, a huge number of frequent itemsets will be obtained, which is both time and space consuming, and users may find it difficult to analyze a large amount of frequent itemsets. To find an appropriate threshold value, a user may thus need to run a FIM algorithm several times. To solve this problem, a new research direction was proposed named top-*k* FIM [6,19–23]. In top-*k* FIM, instead of specifying a minimum threshold value, the user must specify a parameter *k* indicating the number of itemsets to be found. A top-*k* FIM algorithm then returns the *k* itemsets having the highest supports.

A second important limitation of FIM is that it assumes that all items have the same importance (e.g., unit profit or weight) and that items may not appear more than once in each transaction. But these assumptions often do not hold in real applications. For example, items in a transactional database may have different unit profits, and items may have non binary purchase quantities in transactions. Besides, in real-life, retailers are often more interested in finding the itemsets that yield a high profit than those bought frequently. Thus, top-*k* FIM does not satisfy the requirement of users who want to discover itemsets having high utilities (e.g., generating a high profit).

To address these important issues, the problem of top-*k* FIM was recently redefined as the problem of top-*k* high utility itemset mining. Mining high utility itemsets (HUIs) [24–30] is an important research problem in data mining, which has a wide range of applications. It consists of discovering itemsets having utilities (e.g., profit) no less than a user-specified minimum utility threshold, that is high utility itemsets. The problem of top-*k* high utility itemset mining consists of finding the *k* itemsets having the highest utilities in a transactional database. Top-*k* high utility itemset mining is an important data mining task that is useful in many domains. For example, it can be used to find the *k* sets of products that are the most profitable when sold together, in retail stores. Top-*k* high utility itemset mining is harder than top-*k* FIM, since the downward closure property used in FIM to prune the search space, does not hold in high utility itemset mining. To circumvent this issue, overestimation methods [24,28] such as the transaction weighted utility (TWU) have been proposed.

Although several algorithms have been designed for top-*k* high utility itemset mining, it remains a very expensive task in terms of runtime and memory consumption. It is thus an important research problem to design more efficient algorithms. Despite that top-*k* high utility itemset mining is inspired by the traditional problem of top-*k* FIM, the methods used in top-*k* FIM cannot be directly applied to solve the problem of top-*k* high utility itemset mining. To design an efficient top-*k* high utility itemset mining algorithm, additional issues need to be considered such as designing effective search space pruning techniques by considering information about the utilities of itemsets, and also how to raise the internal minimum utility threshold effectively to reduce the number of candidates.

To address the need for a more efficient top-*k* high utility itemset mining algorithm, this paper proposes a novel algorithm, called top-*k* High utility itemset Mining using Co-occurrence pruning (kHMC). It introduces several novel ideas to discover top-*k* high utility itemsets efficiently. The contributions of this paper are summarized as follows:

1. An efficient algorithm is proposed for mining the top-*k* high utility itemsets in transactional databases. The proposed algorithm relies on two efficient strategies for pruning the search space. The first strategy is an improved co-occurrence pruning strategy that eliminates a large number of join operations, and thus prunes a large part of the search space. This strategy is implemented using a novel co-occurrence pruning structure with threshold (EUCST). The second strategy is named transitive extension pruning (TEP). It reduces the search space using a novel upper-bound on the utilities of itemsets.
2. The proposed algorithm relies on the utility-list structure [31]. To further improve the performance of discovering high utility itemsets using utility-lists, a low complexity utility-list construction procedure is proposed. Moreover, the proposed algorithm utilizes a strategy for abandoning utility-list construction early.
3. As previously mentioned, a key challenge in top-*k* high utility itemset mining is how to raise the internal minimum utility threshold effectively while searching for the top-*k* high utility itemsets, to reduce the search space. To address this challenge, the proposed algorithm utilizes several strategies, named RIU, COV, and CUD, for initializing and dynamically adjusting its internal minimum utility threshold. The COV strategy is based on a novel concept of coverage. The concept of coverage can be employed to prune the search space in high utility itemset mining, or to raise the threshold as introduced in this paper. This article proves that by using these strategies, no top-*k* HUIs will be missed, and demonstrates that these strategies are effective at raising the internal minimum utility threshold close to the optimal value.
4. Extensive experimental evaluations are conducted on both real and synthetic datasets to evaluate the proposed techniques. Results show that the proposed algorithm is faster and consumes less memory than the state-of-the-art TKU, REPT, and TKO algorithms for top-*k* high utility itemset mining.

The paper is organized as follows. Section 2 briefly reviews related work on (top-*k*) high utility itemset mining. Section 3 presents preliminaries and formally defines the problem of top-*k* high utility itemset mining. Section 4 proposes an improved strategy for pruning the search space based on item co-occurrence information. Section 5 presents two techniques for improving the efficiency of utility-list construction. Section 6 introduces a strategy for reducing the search space using a novel concept of transitive extension upper-bound utility. Section 7 proposes the concept of coverage. Section 8 presents the three threshold raising strategies used in the designed algorithm. Then, Section 9 describes the proposed kHMC algorithm, to mine the top-*k* high utility itemsets, which incorporates all the techniques presented in previous sections. Section 10 presents an extensive experimental evaluation. Finally, Section 11 draws the conclusion and discusses future work.

2. Related work

This section briefly reviews studies related to high utility itemset mining and top-*k* high utility itemset mining.

2.1. High utility itemset mining

Several algorithms have been proposed for high utility itemset mining. Two-Phase [26], an Apriori-based algorithm, employs an upper-bound called the Transaction Weighted Utilization (TWU) to restore the downward closure property for mining high utility itemsets. According to the TWU model, an itemset having a TWU lower than the minimum utility threshold is not a high utility itemset, as well as all its supersets. The TWU model allows to reduce the search space. However, a disadvantage of using the TWU model is that the TWU is a loose upper-bound on the utilities of itemsets and thus a huge amount of candidates still need to be considered to discover the high utility itemsets. The IIDS [30]