Knowledge-Based Systems 122 (2017) 75–89

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Efficient algorithms for mining colossal patterns in high dimensional databases

Thanh-Long Nguyen[a,b], Bay Vo[c,d,*], Vaclav Snasel[e]

[a] Division of Data Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[b] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[c] Faculty of Information Technology, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
[d] College of Electronics and Information Engineering, Sejong University, Seoul, South Korea
[e] Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava, 17. listopadu 15/2172, 708 33 Ostrava - Poruba, Czech Republic

ARTICLE INFO

*Article history:*
Received 2 May 2016
Revised 22 January 2017
Accepted 23 January 2017
Available online 25 January 2017

*Keywords:*
Bottom up
Colossal patterns
Data mining
High dimensional databases

ABSTRACT

Mining association rules plays an important role in decision support systems. To mine strong association rules, it is necessary to mine frequent patterns. There are many algorithms that have been developed to efficiently mine frequent patterns, such as Apriori, Eclat, FP-Growth, PrePost, and FIN. However, these are only efficient with a small number of items in the database. When a database has a large number of items (from thousands to hundreds of thousands) but the number of transactions is small, these algorithms cannot run when the minimum support threshold is also small (because the search space is huge). This thus causes the problem of mining colossal patterns in high dimensional databases. In 2012, Sohrabi and Barforoush proposed the BVBUC algorithm for mining colossal patterns based on a bottom-up scheme. However, this needs more time to check subsets and supersets, because it generates a lot of candidates and consumes more memory to store these. In this paper we propose new, efficient algorithms for mining colossal patterns. Firstly, the CP (Colossal Pattern)-tree is designed. Next, we develop two theorems to rapidly compute patterns of nodes and prune nodes without the loss of information in colossal patterns. Based on the CP-tree and these theorems, an algorithm (named CP-Miner) is proposed to solve the problem of mining colossal patterns. A sorting strategy for efficiently mining colossal patterns is thus developed. This strategy helps to reduce the number of significant candidates and the time needed to check subsets and supersets. The PCP-Miner algorithm, which uses this strategy, is then proposed, and we also conduct experiments to show the efficiency of these algorithms.

## 1. Introduction

Since the problem of mining association rules was first set out in 1993 [1], many algorithms for mining frequent (closed) patterns have been proposed, such as Apriori-based algorithms [2,8,14], FP (frequent pattern)-tree based algorithms [9,10,15], IT (itemset tidset)-tree based algorithms [19,20,23,24,25], bit vectors based algorithms [8,18,19], N-lists and Nodesets based algorithms [4,5,6,7,11,21]. Moreover, mining frequent patterns in uncertain databases [12,13] has also been examined. These algorithms are based on items (i.e., they are based on 1-items to generate 2-itemsets, based on 2-itemsets to generate 3-itemsets, and so on) for mining frequent (closed) patterns. The main purpose of these is to improve the mining time and/or memory usage for mining frequent (closed) itemsets. However, they are only efficient when the number of items that satisfy the minimum support threshold (minSup, in this paper, and we use this to mean the minSup count) in the database is small. When the number of items that satisfy the minSup is large, this leads to a huge search space and these algorithms are then inefficient, even though they cannot run [26] due to resource limitations. To solve this problem, Zhu et al. [26] proposed a method for mining colossal patterns called the Pattern-Fusion algorithm. To overcome the huge search space, Pattern-Fusion uses an approximation approach to mine $K$ good frequent patterns. This means that Pattern-Fusion may not mine all colossal patterns that satisfy Definition 4 (below). In 2010 Dabbiru and Shashi [3] proposed the CMP (Colossal Pattern Miner) algorithm for mining such patterns. Next, Sohrabi and Barforoush [16] proposed a method for mining colossal patterns in high dimensional databases in 2012. The authors also proposed the BVBUC algorithm, which uses a bottom-up strategy based on transactions. The BVBUC

* Corresponding author.
*E-mail addresses:* nguyenthanhlong1@tdt.edu.vn (T.-L. Nguyen), bayvodinh@gmail.com, vd.bay@hutech.edu.vn (B. Vo), vaclav.snasel@vsb.cz (V. Snasel).

http://dx.doi.org/10.1016/j.knosys.2017.01.034
0950-7051/© 2017 Elsevier B.V. All rights reserved.

joins 1-transactions together to generate 2-transactions, and so on, until the number of transactions reaches minSup (which means that the patterns in a set of transactions are frequent). Moreover, the authors also proposed a formula to prune branches that cannot expand to reach minSup to reduce the search space.

Although BVBUC is faster than CMP and Pattern-Fusion it has some limitations, as follows:

1. The patterns of a set of transactions are computed many times, and this makes BVBUC inefficient.
2. BVBUC uses the downward closure property to prune items that do not satisfy minSup, but it does not remove transactions (after removing infrequent items, some transactions do not contain any items).
3. BVBUC generates a lot of duplications, and thus more time is needed to check these.
4. BVBUC finds patterns based on a set of transactions (tidset) by computing the intersections among them. In fact, two tidsets only differ by one transaction if they have parent-child relationship. For example: tidsets {1, 2, 3} and {1, 2, 3, 4} only differ in transaction 4. In this case, if we have pattern *X* of tidset {1, 2, 3}, we can get pattern of tidset {1, 2, 3, 4} by computing the intersection between *X* and pattern of transaction 4.
5. In high dimensional databases, using bit vectors consumes more memory and time to join them.

Based on some of the limitations of BVBUC, in this work we propose a new method for mining colossal patterns. The main contributions of this paper are as follows:

1. We develop a method for computing the pattern of a set of transactions once.
2. Based on the downward closure property, we remove 1-items for which their supports do not satisfy minSup. After that, we remove transactions that do not contain any item to reduce the number of transactions that need to be traversed.
3. We use patterns at the parent level to compute patterns at the child levels to reduce the number of patterns that need to compute the intersections and reduce the duplications.
4. We develop theorems to prune non-colossal patterns early in the process.
5. We use dynamic bit vectors [18] instead of bit vectors to save memory and computational time.

The rest of this paper is organized as follows. Section 2 presents some works related to frequent (closed) pattern mining, mining colossal patterns and an overview of the BVBUC algorithm. Section 3 presents a theorem to compute the pattern from two sets of transactions and a theorem to quickly prune candidates, and proposes the CP-Miner algorithm for mining colossal patterns based on this. We also present a strategy for early pruning of the items and transactions based on minSup to reduce the search space. Section 4 presents a sorting strategy and pruning technique based on the parent-child relationships in the CP-tree. The PCP-Miner algorithm for efficiently mining colossal patterns is also developed. In Section 5 we compare the proposed algorithms with BVBUC with regard to runtime and number of nodes in the search trees, and discuss the results. Section 6 then gives our conclusions and some future research directions.

## 2. Related works

The problem of mining frequent patterns was first proposed by Agrawal et al. in 1993 [1], and this is the main problem of association rule mining. In 1994, Agrawal and Srikant developed the downward closure property to prune candidates that do not satisfy minSup [2]. The Apriori algorithm, a level-wise approach, was also proposed. Using the downward closure property, Apriori generates candidate (k+ 1)–itemsets from frequent k-itemsets and also uses this property to prune candidates. In 1997, Zaki et al. developed the Eclat algorithm for mining frequent itemsets using the IT-tree (Itemset Tidset-tree) [23]. Eclat applies an in-depth first search scheme and vertical data format to mine frequent itemsets. The advantages of this approach are that it only scans the database once and rapidly compute the supports of patterns based on the intersections of tidsets. Because tidsets consume more memory in dense databases, in 2003 Gouda and Zaki proposed using diffsets instead to reduce memory usage and time [24]. The IT-tree was also used in CHARM [25] to mine frequent closed patterns. Early in the process CHARM uses subset checking to omit patterns that cannot be closed, and checks whether a candidate is closed or not using hash table. Diffsets are also used in another study [25]. In 2000, Han et al. proposed the FP-tree structure and used it for mining frequent patterns [10]. In this, the FP-tree compresses the database in a prefix tree and then uses projections in this to mine frequent patterns. The authors also used an FP-tree to mine frequent closed patterns [22]. Grahne and Zhu used an FP-array to reduce the number of traverses and projections in an FP-tree, and applied an FP-array to mine frequent (closed) patterns [9]. Bit vector based algorithms were then also developed [8,17,18]. In 2007, Dong and Han proposed the BitTableFI algorithm for mining frequent patterns [8]. This is based on Apriori, but uses bit vectors to store tidsets of patterns and computes the supports of these by computing the intersections of bit vectors. The support of a pattern is the number of bits 1 in a bit vector. By using bit vectors, BitTableFI only scans the database once (Apriori scans the database k times, where k is the longest pattern). In 2008, Song et al. improved BitTableFI by using the subsume concept to quickly determine the support of subsumed patterns, and thus the supports do not need to be computed for these patterns [17]. However, BitTableFI and its improved algorithm (Index-BitTableFI) use fixed bit vectors, which means that the number of bits in each bit vector does not change and is the number of transactions in the database. When the number of transactions in the database is large, bit vectors consume more memory to store and time to compute their intersections. In 2012, Vo et al. proposed the dynamic bit vector (DBV) concept, and used it to mine frequent closed patterns [18]. A dynamic bit vector is also a bit vector, but it removes zero bits at the beginning and end. The authors also proposed an algorithm for computing the intersection between two DBVs, and the way to determine whether a pattern is not frequent in the process of computing the intersections of DBVs. In 2012, Deng and Lv proposed a method for mining frequent patterns using the N-list [4]. N-list has a structure like an FP-tree (adding Pre and Post for each node, and not storing a header table as FP-tree does). Unlike FP-Growth, an N-list based algorithm (PrePost) uses a vertical data format to mine frequent patterns. An improved PrePost algorithm using the subsume concept to reduce the search space has also been proposed [21], while N-lists have been used to mine frequent closed patterns [11]. Deng and Lv then proposed the Nodesets structure. Nodesets only use Pre (or Post) to compress the database, and the FIN algorithm uses this structure to mine frequent patterns. DiffNodesets, a different Nodesets strategy that can reduce both memory usage and computing time, was proposed in 2016 [7]. Yun et al. also proposed an LP-tree structure for mining frequent itemsets [15].

The above algorithms use item-extension to mine frequent (closed) patterns, but are only suitable when the number of frequent 1-items is small. When the number of frequent 1-items is large the search space is huge, making these algorithms inefficient. The concept of colossal patterns was thus developed to solve the problem of high dimensional databases (which cause the number of frequent 1-items to be large when minSup is small) [26]. The