# Mining High Utility Itemsets in Large High Dimensional Data

Guangzhu Yu
Information and Technology College, Donghua University
Shanghai, China
ygz@mail.dhu.edu.cn

Keqing Li
Computer Technology College, Yangtze University
Jingzhou, China
likq03@126.com

Shihuang Shao
Information and Technology College, Donghua University
Shanghai, China
guang216@126.com

## Abstract

*Existing algorithms for utility mining are inadequate on datasets with high dimensions or long patterns. This paper proposes a hybrid method, which is composed of a row enumeration algorithm (i.e., Inter-transaction) and a column enumeration algorithm (i.e., Two-phase), to discover high utility itemsets from two directions: Two-phase seeks short high utility itemsets from the bottom, while Inter-transaction seeks long high utility itemsets from the top. In addition, optimization technique is adopted to improve the performance of computing the intersection of transactions. Experiments on synthetic data show that the hybrid method achieves high performance in large high dimensional datasets.*

## 1. Introduction

The traditional association rule mining (ARM) assumes that the important must be frequent, and thus aims at discovering frequent itemsets. But in real world, the frequent is not necessarily important; some infrequent itemsets may have high utility values and thus may be important to users. For this reason, utility-based ARM was proposed to discover all itemsets in a database with utility values higher than a user specified threshold. Since the downward closure property, which states if an itemset is frequent by support, then all its nonempty subsets must also be frequent by support, doesn't apply to utility mining [1] [2], most algorithms for frequent pattern mining can't be applied to utility mining. By now, to the best of our knowledge, only UMining [3] and Two-phase [4] can be used to mine all high utility itemsets. However, both of them are Apriori-like, which employ a bottom-up, breadth-first search, and thus are inadequate on datasets with long patterns or high dimensions. To handle datasets with long patterns efficiently, [5] proposed a hybrid model and a row enumeration algorithm, i.e., Inter-transaction. But Inter-transaction aims to identify only long high utility itemsets, another algorithm has to be used to identify short high utility itemsets. Due to the limitation of pages, [5] didn't study the overall performance of the hybrid model, didn't discuss the effect of parameters such as the length threshold minlen and the size of partitions, and so on. In this paper, we use Two-phase and Inter-transaction to build a hybrid method to perform the mining task: Two-phase searches short high utility itemsets from the bottom, while Inter-transaction searches long high utility itemsets from the top. The reason why we choose Two-phase is that we are certain of that it's the best approach for utility mining by now. We also propose an optimization technique to speedup the intersection of transactions.

This paper is a further study of the former work in [5]. We assume the readers are familiar with the terminologies and notations in [5] and [6], such as the utility of an itemset, length threshold minlen, utility threshold minutil, set of transaction identifiers (denoted as tidlist), intersection transaction T(tidlist), current support, partition, local long high utility itemset, and so on. To make the paper self-contained, we introduce Inter-transaction algorithm in section 2.

## 2. Inter-transaction algorithm

Any a high utility itemset must be in a closed itemset, this means that if we can identify all closed itemsets, and then mine them (here mining a closed itemset means verifying the utilities of all subsets contained in the closed itemset. For more detail, refer to the subroutine mine_single_trans in [5]), we can identify all high utility itemsets.

According to lamma 3.2 in [7], an intersection transaction T(tidlist) must be a closed itemset. Suppose X1=T(tidlist1) be a closed itemset, X2 be a high utility itemset. If $X2 \subset X1$, could we discover X2 as a high utility itemset through mining closed itemset X1? According to the subroutine mine_single_trans introduced in [5], maybe we can't (for example, X1 is not a high utility itemset). In this situation, we can infer that the support of X2 must be larger than that of X1, and there must exists a closed itemset X3 such that $X2 \subset X3 \subset X1$, or X2 is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*e-Forensics 2008*, January 21-23, 2008, Adelaide, Australia.
© 2008 ICST 978-963-9799-19-6.

another closed itemset. If X2 is another closed itemset, it can be found as a high utility itemset through mining X2; if X2 is not a closed itemset and it is contained in another closed itemset X3=T(tidlist3), it can also be found as a high utility itemset through mining X3. Note that X3 must be a high utility itemset, because the utility of X3 is larger than that of X2 under the same support (according to (9) in [5]). In this way, the problem of identifying high utility itemsets is transposed into the problem of identifying closed itemsets.

Like CARPENTER [7] and TD-CLOSET [8], Inter-transaction algorithm is based on row enumeration. But there still exist two problems. One is that since the downward closure property doesn't apply to utility mining, the pruning strategies used in CARPENTER and TD-CLOSET can't be used in our Inter-transaction. The other problem is that when the number of rows grows, the performance of row enumeration algorithms decreases dramatically.

To solve the first problem, all the short (intersection) transactions are pruned out as early as possible in Inter-transaction. The rationale behind the method is that short (intersection) transactions have no effect on the support and utility of long itemsets and Inter-transaction aims to find only long high utility itemsets. Now that the intersection of long transactions is usually very short, large amounts of short itemsets can be pruned out. To solve the second problem, Inter-transaction adopts a partition method to divide a database into multiple partitions, with each partition containing a fitting amount of transactions. In the first scan of a database, Inter-transaction finds all local high utility itemsets from every partition, and then these local high utility itemsets are merged to generate a set of potential high utility itemsets. In the second scan of the database, the actual utility and support for these itemsets are computed and global high utility itemsets are identified. The whole process is just like that described in [6]. The correctness of the partition method is guaranteed by following theorem:

Suppose D is a transaction database, I is a set of items, $P=P_1, P_2, \ldots, P_j$ is a set of partitions of D, i.e., $\sum_{i=1}^{j} P_i = D, P_m \cap P_n = \Phi, m \neq n$. If $X \subseteq I$ is a high utility itemset, it will appear as a local high utility itemset in at least one of the partitions.

Ref. [5] also suggest the partition size should be between $5/a$ and $10/a$. Coefficient $a$ is the minimum acceptable ratio of the utility value of an itemset to the total utility value in the database. For more detail about Inter-transaction, refer to [5].

## 3. Data layout alternatives

Conceptually, a database is a two-dimensional matrix. In a horizontal algorithm, the data can usually be implemented in the following two ways [9] [10]:

Horizontal item-vector (HIV): the database is organized as a set of rows with each row storing a transaction identifier (TID) and a bit-vector of 0's and 1's to represent for each of the items, its presence or absence, respectively, in the transaction;

Horizontal item-list (HIL): this is similar to HIV, except that each row stores an ordered list of item-identifiers (IID), representing only the items that actually occur in the transaction.

Vertical tid-vector (VTV) and Vertical tid-list (VTL) methods are used in vertical algorithms [9][10]. Each aforementioned format can be used to express the data independently. If we express each transaction in HIV format, an intersection transaction (a closed itemset) can be obtained from the intersection of two bit-vectors. Although the bitwise and operation is well supported by computer hardware and very efficient, the overall performance of the intersection of two transactions decreases dramatically with the increase of the number of items. For example, if the number of items is 8k, we have to use 1k bytes (8k bits) to express each transaction. In order to perform the intersection of two transactions, 8k bit operations are needed and this is intolerable. If two transactions are represented only in HIL format, the benefit of bitwise logical operations couldn't be shared. When the length of data in HIL format grows, the performance decreases dramatically.

Some optimization techniques such as VIPER [9] and DIFFSET [10] have been proposed to enhance the performance of the intersection of two bit-vectors in vertical mining algorithms. VIPER uses a compressed bit-vector called "snakes" to improve the performance of an intersection operation. Although VIPER can reduce the need for memories and improve the speed of an intersection operation of two TID-lists, compressing or uncompressing a bit-vector is still costly. DIFFSET only keeps track of differences in the tids of a candidate pattern from its generating frequent patterns. When the differences are large enough, databases in DIFFSET format may occupy more space. So DIFFSET is only suitable for dense databases [10]. Generally speaking, existing optimization techniques have limited improvement on the speed of the intersection of long transactions, the reason may be that most of them aim at reducing the memory requirement of algorithms, and thus adopt various compressed formats to store a database. Because transaction intersection can't be performed directly in these compressed formats, extra format converting can't be avoided. In our algorithm, we not only refrain from compressing each row in main

memory, but we also use redundant information to reduce the amount of bitwise logical operations.

Besides the HIV format, we also store each transaction in HIL format. Although this method will waste lots of memory, the cost is affordable because the partition method can save lots of memory, and its benefit is tremendous. HIV format is used to perform the intersection of bit-vectors, while HIL format can guide us to choose only necessary bits in a bit-vector to perform bitwise and operation. Let S1 be the length of transaction $T_1$ in HIL format, S2 be the length of transaction $T_2$, if S1>S2, we choose T2 (the shorter transaction) as the benchmark to determine on what bits the bitwise and operation should be performed. The method is: if the k-th bit in $T_2$ is 1, bitwise and operation should be performed on the bit, all other bits should be set 0 in the result of intersection operation. For example, as shown in table 1, there are three transactions T1, T2 and T3, If we want to get T(1,2,3), we can first get T(1,2) by intersecting $T_1$ with $T_2$, then we get T(1,2,3) by intersecting T(1,2) with $T_3$. In order to get T(1,2), we choose the shorter transaction (in HIL format) $T_2$ as the benchmark and decide bitwise and operations should be performed only on the first bit, the eighteenth bit and the nineteenth bit. Other bits should be set at zero. As an intermediate result, we get T(1,2)={1,19}( in HIL format). In the subsequent process of intersecting T(1,2) with $T_3$, we choose T(1,2) as the benchmark and decide that bitwise and operations should be performed only on the first bit and the last bit. We get the final result T(1,2,3)= {1,19}. In the whole process, only five bit operations are needed.

Table1. The process of computing T(1,2,3)

| | HIV format | HIL format |
|---|---|---|
| T1 | 1011,1100,1100,0110,001 | 1,3,4,5,6,9,10,14,15,19 |
| T2 | 1000,0000,0000,0000,011 | 1,18,19 |
| T3 | 1011,1100,1101,0110,001 | 1,3,4,5,6,9,10,12,14,15,19 |
| T1 ∩ T2 | 1000,0000,0000,0000,001 | 1,19 |
| T1∩T2∩T3 | 1000,0000,0000,0000,001 | 1,19 |

In this way, the amount of bit operations linearly depends only on the length of the short transaction in HIL format. Experiments show the method outperforms VIPER and DIFFSET with respect to the running time, especially in the context of long bit-vector.

## 4. Mining task assigning

According to our hybrid method, all the long itemsets with more than minlen items should be identified by Inter-transaction, whereas all the short itemsets with less than minlen items should be identified by Two-phase. So minlen actually assigns mining tasks between Inter-transaction and Two-phase. The larger the minlen, the fewer the tasks assigned to Inter-transaction, and the shorter the execution time needed for Inter-transaction. Although a large minlen always means a short execution time for Inter-transactions, more tasks will be left for Two-phase, and the overall performance isn't necessarily high. On the other hand, a too small minlen doesn't benefit the overall performance of the hybrid method either. So a proper minlen is a key parameter for the hybrid method. In our implement of Two-phase, only short itemsets (with less than minlen items) are generated and tested.

## 5. Experimental results

All the experiments were performed on a 2GHz Legend server with 4GB of memory, running windows 2003. The program was coded in Delphi 7.

Three datasets were used in our experiments; all were generated by IBM quest data generator [11]. The first two are T40.I30.D8000K with 1k items and 4k items respectively, and the last is T20I6D8000K with 4k items, where T# stands for the average length of transactions, I# for the average length of maximal potentially large itemsets and D# for the number of transactions. The method of generating utility values for the items in each transaction is just the same as that described in [5].

Figure 1 shows that the size of partitions is very important for Inter-transaction. Just as we have mentioned in [5], a partition that is too small or too large will degrade the performance of the algorithm. In fact, there exists a relationship between the size of partitions and minimum utility threshold minutil. When minutil is divided into half of the initial value, that is, minutil/2 , we can keep the local utility threshold unchanged by doubling the size of partitions. This is why our method works well under a small minutil.

Figure 2 shows the effect of minlen on the total running time of the hybrid method. The total running time of the hybrid method is equal to the running time of Inter-transaction (used to find long high utility itemsets) plus the running time of Two-phase (used to find short high utility itemsets). As minlen decreases, Inter-transaction will finish more and more mining task, and the ratio of running time of Inter-transaction to total running time increases.

Figures 3 and 4 show that the hybrid method is not suited for datasets with only short patterns, this is because Inter-transaction can't take obvious effect on these datasets. As for those datasets with lots of long patterns, Two-phase has to extend short itemsets step by step to obtain long itemsets, whereas Inter-transaction obtains long itemsets directly by

intersecting relevant transactions. In this situation, the hybrid method has great advantages over the Two-phase algorithm. In figure3, minlen is set at 5; in figure4, minlen is set at 3.

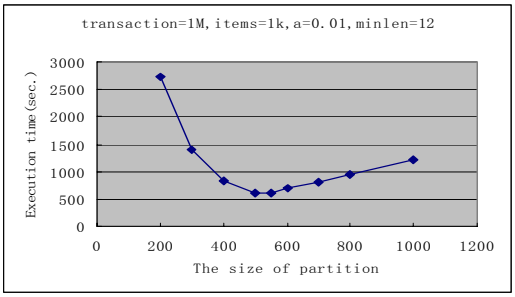


**Figure 1. The effect of size of partitions on performance**

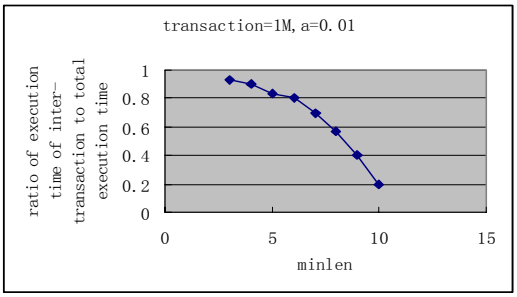


**Figure 2. The effect of minlen on total running time**

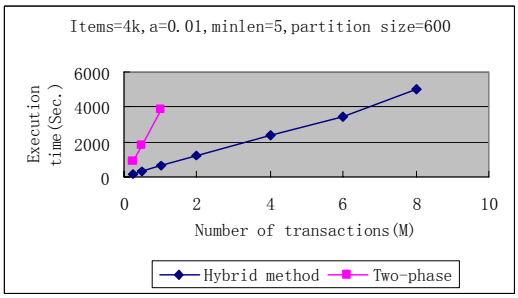


**Figure 3. The execution time for T40I30D8000**

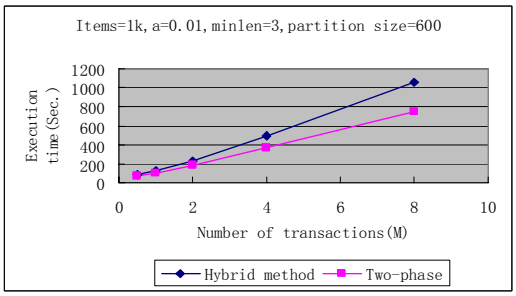


**Figure 4. The execution time for T20I6D8000**

## 6. Conclusions

The paper proposes a hybrid method to find high utility itemsets in two directions. In stead of using a single algorithm to solve a complex problem, the hybrid method decomposes a complex problem into two parts and then uses different algorithms to solve them separately, according to different aims and data characteristics. In addition, redundant information is used to speedup the intersection of transactions. Without the optimization technique, Inter-transaction can't work well under long transactions environment. Experiments show the hybrid method is suitable for large high dimensional datasets with long patterns.

## References


[1] Yao H. and Hamilton H.J., "A unified framework for utility based measures for mining itemsets", Proc. of the Int. Workshop on Utility-Based Data Mining, Philadelphia, PA, 2006, pp. 28-37

[2] Shen Y. D., Zhang Z. and Yang Q, "Objective-oriented utility-based association mining", Proc. of ICDM' 02, Japan, pp. 426-433

[3] Yao H. and Hamilton, H.J., "Mining itemset utilities from transaction databases", Data & Knowledge Engineering, 59 , 2006, pp. 603 – 626

[4] Liu Y., Wei-keng L. and Choudhary A., "A fast high utility itemsets mining algorithm", Proc. of 1st Int. Workshop on Utiliy-based Data Mining, 2005, pp. 90-99

[5] Guangzhu Y., Shihuan S., Daoqing S and Bin L., "Mining long high utility itemsets in transaction databases", Proc. of the 7th WSEAS Int. Conf. on Simulation, Modelling and Optimization, Beijing, 2007, pp. 326-331

[6] Savasere A, Omiecinsky E. and Navathe S., "An efficient algorithm for mining association rules in large databases", 21st Int'l Conf. on Very Large Databases, Zurich, Switzerland, 1995, 432-444.

[7] Feng P., Gao. C.and et al., " CARPENTER: Finding closed patterns in long biological database", Proc. of SIGKDD, Washington, 2003, pp. 413-419

[8] Hongyan Liu, Jiawei Han and et al, "Mining frequent Patterns from Very High Dimensional Data: A Top-down Row Enumeration Approach", Proc. of the Sixth SIAM Int. Conf. on Data Mining, Bethesda, Maryland, 2006, pp. 20-22

[9] Shenoy P, Haritsa J. R. and et al., "Turbo-charging vertical mining of large databases", Proc. of ACM SIGMOD Int. Conf. on Management of Data, Dallas, Texas, 2000, pp. 22-33.

[10] Zaki M J, Gouda K., "Fast vertical mining using diffsets", In Proc. of ACM SIGKDD, Washington DC, 2003, PP. 326-335.