Improvement of Eclat Algorithm Based on Support in Frequent Itemset Mining

Xiaomei Yu, Hong Wang
School of Information Science and Engineering,
Shandong Provincial Key Laboratory for Distributed Computer software Novel Technology,
Shandong Normal University,Jinan, Shandong, China,
Email: yxm0708@126.com

***Abstract*—Finding frequent itemsets is computationally the most expensive step in association rules mining, and most of the research attention has been focused on it. With the observation that support plays an important role in frequent item mining, in this paper, a conjecture on support count is proved and improvements of traditional Eclat algorithm are presented. The new Bi-Eclat algorithm sorted on support: Items sort in descending order according to the frequencies in transaction cache while itemsets use ascending order of support during support count. Compared with traditional Eclat algorithm, the results of experiments show that the Bi-Eclat algorithm gains better performance on several public databases given. Furthermore, the Bi-Eclat algorithm is applied in analyzing combination principles of prescriptions for Hepatitis B in Traditional Chinese Medicine, which shows its efficiency and effectiveness in practical usefulness.**

***Index Terms*—frequent itemset, support count, equivalence class, Eclat algorithm**

I. Introduction

Nowadays, finding frequent itemsets is the major task in association rule mining and it determines the overall performance of mining association rules. Once the frequent itemsets are found, the other step of mining association rules is not much costly to generate rules which represent interesting relationships between the mined frequent itemsets.

There have been many different algorithms proposed for the field of frequent itemsets mining[1][2][3][4]. Goethals[1] analyzed and compared the most successful algorithms, then tested them with experiments. [2] reviewed the efficient algorithms and recent developments in sequential frequent itemsets mining by considering additional dimension of time. By relaxing the support definition, [3] presented two new algorithms for finding approximate frequent itemsets in imperfect transaction database. [4] summarized the most representative algorithms in mining frequent itemsets over uncertain databases and proposed a novel algorithm based on some new findings. All these algorithms can be categorized as variants or extensions of one of three different base algorithms, Apriori[5], FP-growth[6] and Eclat[7]. The Apriori algorithm works by eliminating most large candidate sets in the way of looking first at small sets and recognizing the large sets which can not be frequent unless all its subsets are. Various improvements to the basic Apriori algorithm concentrate on reducing the number of scans over the database and reducing the size of the candidate k-itemsets. The well-known FP-growth algorithm compresses the database of frequent items into a frequent pattern tree, which retains useful itemset association information. It divides the compressed database into a set of conditional database; each associated with one frequent pattern fragment, and then mines each database separately. The improvements of basic FP-tree concentrate on decreasing excess overhead in constructing many conditional FP-Trees, because its weakness reduces the performance of FP-Growth algorithm as the patterns get longer and/or the support level gets lower. The Eclat algorithm is an acronym of equivalence class clustering and bottom up lattice traversal. In Eclat, frequent items are generated by intersecting the tid-lists of all distinct pairs of atoms and checking the cardinality of resulting tid-list. A recursive procedure is called when those itemsets at the current level are found to be frequent. This process is repeated until all frequent itemsets have been enumerated. Without much pruning in Apriori or FP-growth algorithm, the run time of Eclat increases exponentially with a larger number of transactions which drops off the performance of Eclat algorithm eventually.

Based on previous work[8][9][10][11][12], in this paper, we propose some improvements to the traditional Eclat algorithm. During the transaction caching, a special tree structure is used according to the descending order of support in order to keep storage efficient, while during the candidates generation phrase, each class (sub-lattice) is processed corresponding to a ascending order of support, with the aim to reduce the redundant steps as much as possible. Based on these ideas, we have designed a new algorithm for mining frequent itemsets called bidirectional process strategy for Eclat algorithm. Experiments have been conducted to study the performance of Bi-Eclat and compared it with traditional Eclat algorithm. Our new bidirectional process strategy is found to outperform on the given public databases.

The rest of the paper is organized as follows. In next section, we define relevant terms and review several principles. In Section III, we summarize the drawbacks in traditional Eclat algorithm and describe the main idea of our new strategy. In Section IV, the improved algorithm

doi:10.4304/jcp.9.9.2116-2123

on Eclat is presented and described. We then experimentally validate our approach on real and synthetic datasets in Section V and section VI. At the end of the paper, conclusions and our future work are presented.

II. CONCEPTION

Items and Support Count Let $I=\{i_1,i_2,...,i_{max}\}$ be the set of all items. Any subset $X \in P(I)$ of I is called an itemset. Let $\Gamma \subseteq P(I)$ be a multiset of itemsets, called transaction database, and its elements $T=\{t_1,t_2,...,t_n\} \in \Gamma$ is called transactions. Each transaction t_i contains a subset of items chosen from I. An itemset consisting of k items is called a k-itemset. A useful property of itemsets is the support count, which is the number of transactions containing a special itemset. The support count Sup(X) of an itemset X can be expressed mathematically as follows[13]:

$$\sup(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}|.$$

Support determines how often a rule is applicable to a given data set and we say an itemset is frequent if its support count is equal to or bigger than the given integer called minimum support threshold (min_sup).

lexicographic order Most frequent itemset mining algorithms as Apriori[5] and Eclat[7] use a lexicographical order on the items I and the itemsets P(I) to prevent candidate item sets be checked two times for support count. With itemsets X, Y ∈ P(I), the lexicographic order on I uniquely determines a total order on P(I), which is stated mathematically as X<Y: ⇔ min(X\prefix(X,Y))<min(Y\prefix(X,Y)). Moreover, an itemset $Y \in P(I)$ with $X \subset Y$ and $X < Y$ is called an extension of X. An extension Y of X with $Y=X \cup \{y\}$ (y>maxX) is called an 1-item-extension of X[14].

Prefix and equivalence class The prefix is item sets added to the beginning of all frequent itemsets in order to generate new candidate itemsets. Mathematically, with itemsets X, Y ∈ P(I), the prefix is stated as prefix(X,Y):={{x ∈ X|x ≤ z}| maximal z∈ X∩Y:{x∈ X|x≤z}={y∈ Y|y≤z}}. Two itemsets are in the same class if they share a common k length prefix. And we call θ_k a prefix-based equivalence relation. Each equivalence class $[X]_{\theta_k}$ induced by the equivalence relation θ_k is a sub-lattice of P.

Apriori property All nonempty subsets of a frequent itemset must also be frequent. As a corollary, we get that all supersets of an infrequent itemset are infrequent[15].

In order to improve the efficiency of generating frequent itemsets, the Apriori property is usually used to prune the search space.

III. THE IMPROVEMENT ON TRADITIONAL ECLAT

A vertical tid-list of database format was created by Zaki in Eclat algorithm[7], where all frequent itemsets can be enumerated via simple tid-list intersection. Furthermore, the lattice theory and prefix-based approach are introduced to split the original search space (lattic) into small sub-lattice. Finally, the bottom-up search strategy is applied to enumerate the frequent itemsets within each sub-lattice corresponds to a reverse lexicographic order.

A. The Problem of the Eclat Algorithm

In Eclat algorithm, there is not clear boundary between the process of support count and candidate generation. Some possible drawbacks can be seen during the implementation:

- In Eclat algorithm, the computation is done by intersection of the Tid sets of the corresponding k-itemsets. However, the Tid sets may be quite long, which takes substantial memory space as well as much computation time for intersecting the long sets, thus stress the available main memory with increasing of length in frequent itemsets.
- In Eclat algorithm, the classes are solved from bottom to top, according to a reverse lexicographic order. Here, the information on support count is available and may play an important role in pruning, but Eclat algorithm does not make full use of the information in sorting to reduce the useless operation on itemsets which will be deleted in the support-based pruning latter.
- Eclat algorithm does not take full advantage of Apriori property to reduce the number of candidate itemsets explored during frequent itemset generation. Therefore, the number of candidate itemsets generated in Eclat algorithm is much greater than that in Apriori algorithm. The situation gets worse for item sets with many dense and long patterns.

B. A Property on Support Count

Conjecture A subset with high support has more probability to become a part of the long frequent itemset; while a subset with low support count has less probability to become a part of the long frequent itemset.

Proof Let $X \in P(I)$ be the itemset which is composed of some items in I. The items in X can be sorted in different order: $X=\{x_{max},x_{max-1},...,x_1\}$ means items in X sorted in descending order of support, while $X=\{x_1,x_2,...,x_{max}\}(max \geq 2)$ means items in X sorted in ascending order of support.

For n=1: We consider $\sup(\{x_1\}) \leq \sup(\{x_2\}) \leq \sup(\{x_{max-1}\}) ... \leq \sup(\{x_{max}\})$ and X is a frequent itemset. Based on the Apriori or anti-monotone principle, we get the comparison on support as follows:

$\sup(\{x_{max}\} \cap \{x_{max-1}\}) \leq \min(\sup\{x_{max}\}, \sup\{x_{max-1}\}) \leq \sup(\{x_{max-1}\})$

$\sup(\{x_1\} \cap \{x_2\}) \leq \min(\sup\{x_1\}, \sup\{x_2\}) \leq \sup(\{x_1\})$

Given $\sup(\{x_1\}) \leq min\_sup \leq \sup(\{x_{max-1}\})$, item x_{max-1} is a frequent item while item x_1 is infrequent and should be pruned. For $\sup(\{x_1\}) \leq \sup(\{x_{max-1}\})$, it is likely that $\sup(\{x_{max-1}\})$ has more chance to satisfy the inequity of $\sup(\{x_{max-1}\}) \geq min\_sup$. That is to say, item x_{max-1} has more chance to be a frequent item and used to generate

the longer frequent itemsets. Thus with 1-item-extension of X, prefix x_{max-1} has more chance of becoming a part of the long frequent itemset.

To verify the inductive step, we assume the truth of n=k. X is a frequent (k-1)-itemset and y is a frequent item, then the items in X can be sorted in different order: $X=\{x_{k-1},x_{k-2},...,x_1\}$ means items in X sorted in descending order of support, while $X=\{x_1,x_2,...,x_{k-1}\}$ means items in X sorted in ascending order of support. We get the k-itemset with 1-item-extension of X: $Y_1 = \{x_k\} \cup X = \{x_k\} \cup \{x_{k-1},x_{k-2},...,x_1\}$, $Y_2 = \{y_1\} \cup X = \{y_1\} \cup \{x_1,x_2,...,x_{k-1}\}$, thus we have the inequities of $sup(\{y_1\}) \le sup(\{x_1\}) \le sup(\{x_{k-1}\}) \le sup(\{x_k\})$ and $sup(\{y_1\} \cup X) \le sup(\{x_k\} \cup X)$.

This assumption leads us to the following: with $sup(\{x_1\}) \le sup(\{x_2\}) \le ... \le sup(\{x_{k-1}\}) \le sup(\{x_k\}) \le sup(\{x_{k+1}\})$, we consider the (k+1)-itemset with 1-item-extension of X: $Z_1 = \{x_{k+1}\} \cup \{x_k,x_{k-1},x_{k-2},...,x_1\} = \{x_{k+1}\} \cup Y_1$; $Z_2 = \{z_1\} \cup \{y_1,x_1,x_2,...,x_{k-1}\} = \{z_1\} \cup Y_2$, where $sup(\{z_1\}) \le sup(\{y_1\}) \le ... \le sup(\{x_k\}) \le sup(\{x_{k+1}\})$. Based on the Apriori or anti-monotone principle, we get the comparison on support as follows:

$sup((\{x_{k+1}\} \cup Y_1) \cap (\{x_k\} \cup Y_1)) \le \min(sup(\{x_{k+1}\} \cup Y_1), sup(\{x_k\} \cup Y_1)) \le sup(\{x_k\} \cup Y_1) \le sup(\{x_k\} \cup X)$

$sup((\{z_1\} \cup Y_2) \cap (\{y_1\} \cup Y_2)) \le \min(sup(\{z_1\} \cup Y_2)), sup(\{y_1\} \cup Y_2)) \le sup(\{z_1\} \cup Y_2) \le sup(\{y_1\} \cup Y_2) \le sup(\{y_1\} \cup X)$

Considering the assumption of n=k, we have the inequity of $sup(\{y_1\} \cup X) \le sup(\{x_k\} \cup X)$. Now given the min_sup which satisfies $sup(\{y_1\} \cup X) \le min\_sup \le sup(\{x_k\} \cup X)$, we can draw the conclusion that itemset $\{x_k\} \cup X$ has the chance to be a frequent itemset while $\{y_1\} \cup X$ has not. For $sup(\{y_1\} \cup X) \le sup(\{x_k\} \cup X)$, it is likely that $sup(\{x_k\} \cup X)$ has more chance to satisfy the inequity of $sup(\{x_k\} \cup X) \ge min\_sup$. That is to say, as a frequent item, x_{k+1} has more chance to be used to generate the longer candidate itemsets. Thus with 1-item-extension of Y, prefix x_{k+1} has more chance of becoming a part of the long frequent itemset.

Consequently, we now know from the Principle of Mathematical Induction that the conjecture holds for 1-item-extension of frequent itemset.

With the similar proof, we can prove the same conclusion on generation of frequent k-itemset with the intersection of any two (k-1) subsets. Therefore, we get a property on support:

Property: A subset with high support has more probability to become a part of the long frequent itemset; while a subset with low support count has less probability to become a part of the long frequent itemset.

C. Support Counting

During support counting, in order to decide which candidates are contained in the given transactions, a part of the equivalence lattices have to be traversed. To reduce the number of unnecessarily visited atoms(items) to a low level, firstly we have to check if the transaction contains the least frequent item since it is likely to provide the strongest filtering among the candidate items, i.e., this is the item that is contained in the least amount of transactions. Next, the second least frequent item is advised to check, then the third, and so on. We expect the least amount of redundant checks and thus the best run-time, so we choose the order of items in the ascending order according to their supports.

Considering the influence of different orders on generation of frequent itemsets, based on the support property proved above, we know that frequent itemsets generation prefers ascending order of support to reduce candidate itemsets or intersecting operations. Therefore we use ascending order of support instead of lexicographic order in traditional Eclat during support counting. And the experiments proved its effectiveness in section V.

Our improvement benefits traditional Elcat algorithm at least in two ways: The first, based on the Apriori principle, our improvement on Eclat find infrequent itemsets as early as possible and prunes or deletes them since infrequent items can not become a part of any frequent itemset. Furthermore, with the ascending order of support in atoms of the sets, frequent subsets with low support are checked firstly, then we can cease the intersecting operation at the infrequent point and the rest redundant intersections are avoid, because the resulting itemset can not be frequent since part of the intersection is testified incompetent.

D. Transaction Caching

I/O and string to integer parsing costs are reduced if the transactions are stored in the main memory instead of disk. This technique is used in Eclat as well as other algorithms. In the implementation of traditional Eclat, the lexicographic order on items is usually used. However, based on the support property and the Apriori property, we learn that a subset with high support count has more probability to become a part of the long frequent itemset. Thus the intersecting operation on atoms with high support may be more than that on the atoms with low support. That is to say, atoms with high support may be visited more frequently than the atoms with low support in transaction cache. Therefore, we suppose that transaction cache requires descending order according to the frequencies in order to keep storage efficient. In this way can we get the fewer amount of comparisons and then the better running time.

In summary, descending order is good for the compactness and efficient store, while ascending order results in a fewer redundant in support counting. Therefore, transaction caching uses descending order according to the frequencies while ascending order of support is introduced during support count in our improvement. These two requirements can be satisfied at the same time. The items are recoded according to ascending order of supports, but the items are stored descending in tid-list after being inserted into the cache. Since the efficient support count requires the items of the transaction to be stored with ascending order, we simply reverse each transaction when it is retrieved from the cache.

IV. BI-ECLAT ALGORITHM

In this section, we describe the Bi-Eclat algorithm which is an improved algorithm based on traditional Eclat algorithm, and then we provide an example to illuminate the main idea of Bi-Eclat.

A.Bi-Eclat Algorithm

The Bi-Eclat algorithm consists of three steps as follows:

1) Identify each frequent item and sort the tid-list for atoms in descending order according to support count: After a transformation from horizontal to vertical on-the-fly, we scan the tid-list of items successively and incrementing the item's support for each entry. After the computation of support and comparison with the minimum support, we get the 1-frequent itemsets sorted in descending order of support.

2) Construct the equivalence class lattice and generate candidate itemsets: Begin with the set of atoms of class sorted in ascending order according to their supports, we construct the equivalence class lattice. Firstly, an itemset is generated as a union of subsets of the sets of atoms, and secondly its support is computed by intersection of tid-lists of atoms. In fact, by intersecting every two subsets with the length of k-1 in tid-list(Fig.1), we get the supports of all k-itemsets in each class.

3) Prune the candidate itemsets and mine frequent itemsets: We take advantage of the ascending order of support to delete infrequent items as well as reduce redundant intersection. Then our Bi-Eclat algorithm mines all frequent itemsets in the recursive procedure. With a breadth-first search, all frequent itemsets can be obtained in this way.

The main pseudo code for Bi-Eclat algorithm is given as follows:

Input: vertical tid-list database

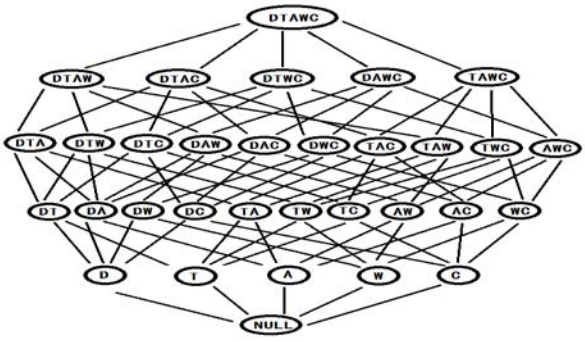

Figure 1. Computing Support of Itemsets in Equivalence Classes

Output: 1-frequent itemsets sorted in descending order of support

Procedure find_frequent_1-itemsets
For all atoms $A_i \in S$ do
For all transactions $T_j \in A_i$ do
$|tid\text{-}list(A_i)|=|tid\text{-}list(A_i)|+1$;
End For
For all $A_i \in S$, with $|tid\text{-}list(A_i)| \geq min\_sup$ do
Sorting A_i in the descending order of support;
End For
$S=S \cup \{R\}$; $T_1=T_1 \cup \{A_i\}$;
End For

Input: atom set S sorted in ascending order of support
Output: frequent itemsets
Procedure candidate_frequent_itemsets( )
For all atoms $A_i \in S$ do
$T_i=\Phi$
For all atoms $A_j \in S$, with $sup(A_j)>sup(A_i)$ do
$R=A_i \cup A_j$;
$tid\text{-}list(R)=tid\text{-}list(A_i) \cap tid\text{-}list(A_j)$;
If $|tid\text{-}list(R)| \geq min\_sup$
$S=S \cup \{R\}$; $T_i=T_i \cup \{R\}$;
End If
End For
While $T_i \neq \Phi$ do candidate_frequent_itemsets (T_i)
End For

B. An Example of Bi-Eclat

The Bi-Eclat algorithm is illustrated by the following example.

Consider an example of bookstore sales database shown on the left in Fig.2. There are several different items {A,C,D,F,H,K,T,W}, and the database consists of six customers who bought books by these authors. The right in Fig.2 shows the tid-list for the atoms in our example database, which is a result of horizontal to vertical transformation and sorted in descending order according to frequencies.

As shown in Fig.1, using the output of the first step, items with support below the min_sup in tid-list are skipped and atoms in the set are sorted in increasing order of support {D,T,A,W,C}(min_sup=2). We combine DT and DA to obtain the frequent itemset DTA. Then we extend it with the next pair DTW, to get DTAW.

sup({DTA})=sup({DT}∩{DA}∩{TA})=sup({56}∩{456}∩{1356})=sup({56}=2≥min_sup

sup({DTW})=sup({DT}∩{DW}∩{TW})=sup({56}∩{245}∩{135})

The computation terminated before the intersecting operation with TW because sup({56} ∩ {245})=1<min_sup and DTW is infrequent. By contrast, we choose WATD (i.e. the same itemset as DTAW) as an opposite order and compute the supports.

sup({WAT})=sup({WA} ∩ {WT} ∩ {AT})= sup({1345} ∩ {135} ∩ {1356}) =sup({135})=3>=min_sup

sup({WAD})=sup({WA} ∩ {WD} ∩ {AD})= sup({1345}∩{245}∩{456})=sup({45})=2>=min_sup

sup({ATD})=sup({AT} ∩ {AD} ∩ {TD})=sup({1356}∩{456}∩{56})=sup({56}=2≥min_sup

sup({WTD})=sup({WT}∩{WD}∩{TD})=sup({135}∩{245}∩{56})The computation terminated before the following intersecting operation because sup({135} ∩ {245})=1<min_sup and {WTD} is infrequent. As we have seen, in the given four steps, at least two more redundant steps on intersection and comparison are saved by using ascending order according to support in the second step.

V. EXPERIMENTS

In this section, we evaluate our Bi-Eclat algorithm by applying it on public databases. A. Data Sets Chosen in the Experiments

All of the experiments we performed ran on several databases with different characteristics. These databases are: the mushroom database contains different attributes of various species of mushrooms; the database of chess lists chess end game positions for king vs. King and rook; the accidents database contains each traffic accident that occurs with injured or deadly wounded casualties on a public road in Belgium for the period 1991-2000; the dataset of T40I10D100K and T10I4D100K were generated using the generator from the IBM Almaden Quest research group and the web click stream database of BMS-Webview-1 is from an e-commerce web site which contains several months worth of clicks stream data. All of the databases are publicly available at the Frequent Itemset Mining Implementations Repository[16] and some of their characters are listed below(showed in Table 1.).

TABLE 1.
SOME CHARACTERS OF THE DATABASE IN OUR EXPERIMENTS

Database	Items	Avg.lenth	Transaction
Mushroom	119	23	8,124
Chess	75	37	3196
Connect	129	43	67,557
Accidents	468	33.8	340,183
Retail	16469	10.3	88,126
T40I10D100k	943	40.61	100,000
T10I4D100k	871	11.1	100,000

We selected different min_sup values and used them in each database. Results from different databases sometimes reveal similar performance on the algorithms, so only the most typical ones are shown in the figures below.

B. Enviroments of the Experiments

Tests were run on a PC with a 2.5 GHz Intel(R) core(TM) i5-2520M processor and 4 Gbytes of RAM. The operating system was Window 7 and ubuntu Linux 12.04.2.

Based on the present Eclat algorithm[17], we designed the improved Bi-Eclat algorithm and obtained the results, then described them with mapping software.

In the experiments, we used 3 different algorithms to test the effects of sorting on the performance. They are

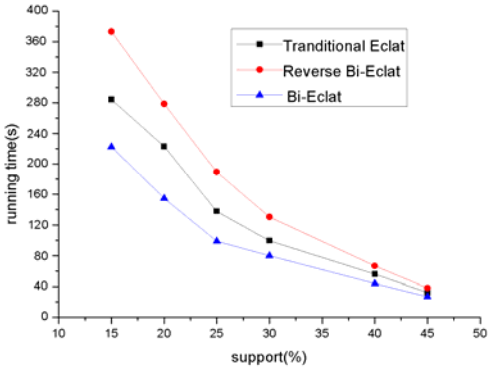

Figure 3. Support Comparison on Database of Connect

support order in Bi-Eclat algorithm, reverse support order in Bi-Eclat algorithm and the lexicographic order in traditional Eclat algorithm.

C. Experiments on Dense Database

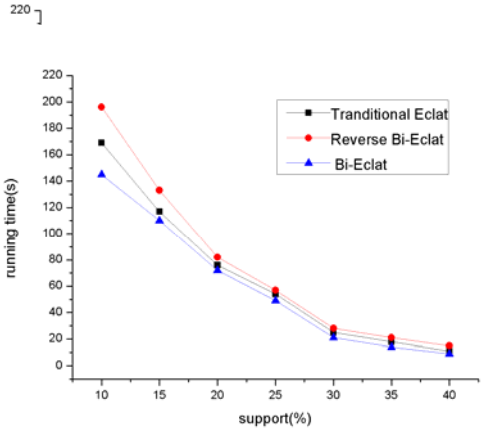

Figure 4. Support Comparison on Database of Mushroom

At first, we used 3 different orders to test the effects on performance in the dense databases such as mushroom, chess and connect. The results are shown in Fig.3 and Fig.4. With little difference in memory occupation, experimental results show that running time is dramatically affected by the sort used in algorithms: support count order in Bi-Eclat algorithm leads to the smallest running time, while reverse support count order in Bi-Eclat algorithm got the highest one. It conforms to our expectations at the beginning: sorted in descending order of support count is good for the compactness and efficient store; while ascending order results in fewer redundant steps in support counting.

D. Experiments on Moderate Database

We also performed several experiments on a moderate database of accidents. As can be seen from the figure (Fig.5), Bi-Eclat algorithm gains much advantage in running time, though it is slighter than that in the databases of mushroom and connects. The same as before,

reverse Bi-Eclat algorithm showed the worst performance in the three algorithms.

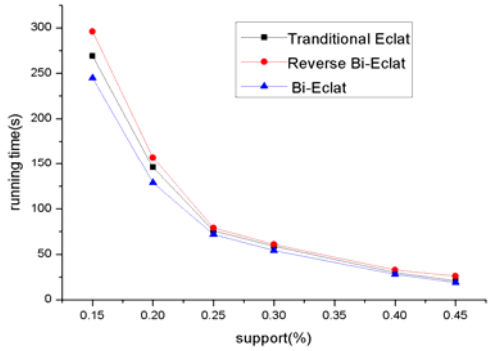

Figure 5. Support Comparison on Database of Accident

From the results gained above, we can draw some possible reasons from the experiments. In the processing of support count, the ascending order is used to all frequent itemsets and the prefix, so the items at the beginning are frequent ones with lower supports than the items following. Therefore, the intersecting operation is done on (k-1) itemsets which have less probability to generate k frequent itemsets. After the least check steps, the support count is terminated as early as possible. On the contrary, when descending order of support is selected, we have to check many redundant items, at the same time further search is needed in finding frequent itemsets. Then the search will not be terminated as immediately as in the case of the ascending order and the redundant steps are generated.

E. Experiments on Sparse Database

On the sparse database of retail, T40I10D100K and T10I4D100K, the experimental results we got were dramatically different. As we have seen in Fig.6 and Fig.7, on the dataset of T40I10D100K, our Bi-Eclat algorithm gains perfect performance in several points of support. However, on the dataset of T10I4D100K, the Bi-Eclat algorithm loses the competition with reverse Bi-Eclat algorithm at one or two points of support count. It may indicate that the Bi- Eclat algorithm is not stable in all of the sparse databases and it may not be suitable for special sparse database, which is the future research we will concentrate on.

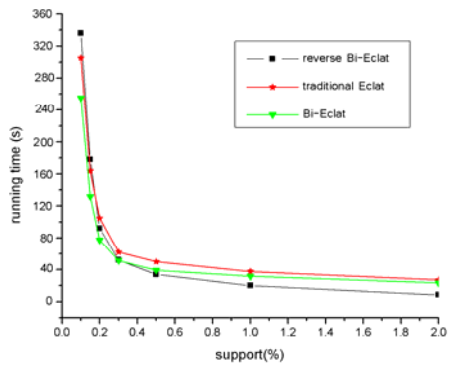

Figure 6. Support Comparison on Database of T10I4D100K

VI. Application to Prescription Mining in TCM

To demonstrate the practical usefulness of the Bi-Eclat algorithm, we applied it in research work of Traditional Chinese Medicine(TCM) and analyzed the composing principles of prescriptions for Hepatitis B by frequent itemset mining, with the purpose to discover commonly used herbs, combination principles and core combinations of herds in prescriptions for treating Hepatitis B.

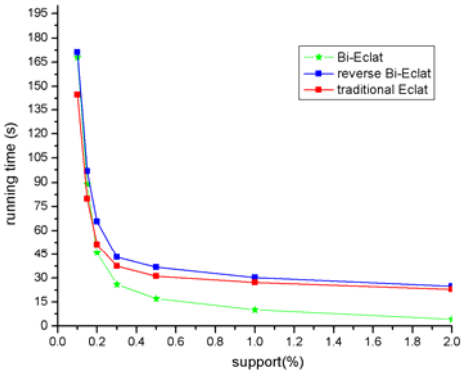

Figure 7. Support Comparison on Databases of T40I10D100K

The Hepatitis B prescription database, there are 342 prescriptions selected from the historical clinical data as well as from the Dictionary of Chinese Medicine Prescription. In each prescription, some kinds of Chinese herbal medicine are listed according to TCM syndrome and TCM clinical diagnosis. In order to discover novel medicine pairs and find a group of medicines which are often used together in the same kind of prescription, we applied our Bi-Eclat algorithm. In order to do so, we converted each prescription into a transaction, such that each kind of Chinese herbal medicine is an item in the transaction. To tackle problems of representation inconsistency, all items of Chinese herbal medicine are standardized and identified by their IDs. This resulted in a transaction database with 342 transactions and each transaction contains several items.Results on Running Time

In the TCM research, commercial software such as SPSS Clementine is prevailing in which the classic Apriori algorithm is used to data mining. As a comparison, we ran our Bi-Eclat algorithm on PC with the same environment shown above. Meanwhile, the same different min_sup values are selected in the two algorithms according to the experience on the syndrome and clinical diagnosis.

Without any doubt, the Bi-Eclat algorithm wins with overwhelming advantages in the performance of running time(shown in Fig.8). In shorter time, it successfully analyzed 342 prescriptions, the frequency of each herb and association rules among the herbs were computed, 17 core combinations and 5 new prescriptions were mined out from the database.

A. Mining Uncertain Data from Incomplete Prescriptions

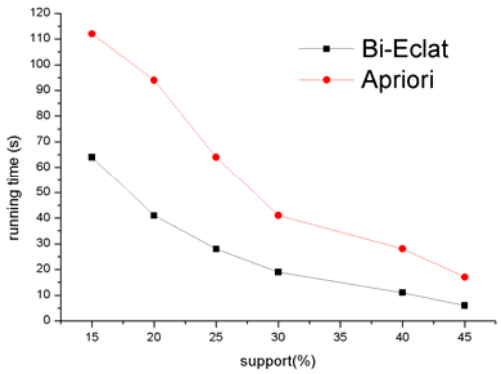

Figure 8. Running Time Comparison on Database of Presriptions

Considering the results of the two frequent itemset algorithms, both figure out merely 17 core combinations and 5 new prescriptions in 342 prescriptions. It is unacceptable for us to receive the distinct "sparse" frequent itemsets in an obvious dense database. Furthermore, it is not in lines with the currently known Chinese medical formula or the clinical treatment in practice. Taking into account potential error and loss in the historical prescriptions, it is difficult for existing traditional data tools such as SPSS Clementine to find out all the really existed combination principles and core combinations of herds in prescriptions because of their limitations on component organization.

In order to make a remedy on this defect, we introduced different weights on the potential frequent items in the transaction database which may be lost due to unknown reasons[18]. The weights determined by frequent itemsets mined and the experience gained in clinical diagnosis. By relaxing the support definition, we use Bi-Eclat algorithm for a fault-tolerant item set mining in incomplete transaction database of prescriptions[19]. The results show that 32 core combinations and 13 new prescriptions were mined out from the database.

So Bi-Eclat algorithm demonstrated its efficiency and effectiveness in the application of prescriptions analysis of TCM.

VII. Conclusion and Future Research

In this paper, with the observation that support count plays an important role in frequent item mining, we proposed a conjecture on support count and proved it on 1-item-extension of itemsets. Then we extended it to k-item-extension of itemsets with similar proof. Furthermore an improved algorithm is presented based on the traditional Eclat algorithm. The new Bi-Eclat algorithm sorts in descending order according to the frequencies in transaction cache while use ascending order of support during support count. Compared with Eclat algorithm, the results of experiments show that the improved algorithm gains better performance on several databases given. We applied the algorithm into composing principles of prescriptions for Hepatitis B, and proved its efficiency and effectiveness in practical usefulness.

In this article, we have assumed that transaction database and tid-list will fit in main memory. However, it is regrettable that the assumption does not hold in massive datasets[20] and online mining of frequent itemsets[21] which will stress the available main memory. In the future research, we will make effort to improve the frequent itemset algorithm in massive datasets. Then utilize and develop these studies into emerging applications such as frequent itemset mining in massive database of TCM.

Though much work has been done with related data mining software, we still face the problem that the transaction database to analyze is imperfect in many applications[22][23]. It is a challenge to perform service-objected frequent itemset mining in these applications. In the future research, we will also concentrate on fault-tolerant item set mining[24] , and explore new ways to improve service-objected frequent itemset mining in imperfect database.

Acknowledgment

We are grateful for the support of the Natural Science Foundation of China (61373149) and the Shandong Provincial Project for Science and Technology Development (2012GGB01058).

References

[1] B. Goethals, Frequent set mining, In O. Maimon, and L. Rokach, *Data Mining and Knowledge Discovery Handbook*, Second edition, Springer, pp. 321-338, 2010.

[2] J. Wang, "*Sequential patterns*," In encyclopedia of database systems, Springer, pp. 2621-2626, 2009.

[3] C. Borgelt, C. Braune, "*New algorithms for finding approximate frequent item sets*," Soft Computing, Springer-Verlag, Berlin, Germany, vol.16, no.5, pp.903-917, 2011. doi:10.1007/s00500-011-0776-2

[4] Y. Tong, L. Chen,Y. Cheng, "*Mining frequent itemsets over uncertain databases*," Proceedings of the VLDB Endowment, vol.5, no.11, pp.1650-1661, August, 2012.

[5] R. Agrawal and R. Srikant. "*Fast algorithms for mining association rules*," Proceedings 20th International Conference on Very Large Data Bases(VLDB'94), pp.487–499, September 12-15, 1994.

[6] J. Han, J. Pei, and Y. Yin, "*Mining frequent patterns without candidate generation*," Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ACM Press, pp. 1–12, 2000.

[7] M.J.Zaki. "*Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering*, " vol.3, pp.372–390, 2000. doi:10.1109/69.846291

[8] H. Zhou, "*An algorithm of frequent item sets mining based on transformation of the frequent item linked, Journal of Chinese Computer Systems*, " vol.7, pp.1254-1257, 2008.

[9] Y. Zhang, Zh. Xiong, X. Geng, et al, "*Analysis and improvement of ECLAT algorithm*," Computer Engineering, 36(23): 28-30, 2010.

[10] Zh. Xiong, P. Chen, Y. Zhuang, "I*mprovement of ECLAT algorithm for association rules based on hash boolean*

matrix," Application Research of Computers, vol.4, pp.1323-1325, 2010.
[11] M. J. Zaki, K. Gouda, "*Fast vertical mining using diffsets*," proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, D.C., ACM, pp.326-335, 2003. doi:10.1145/956750.956788
[12] T. Uno, M. Kiyomi, and H. Arimura, "*LCM ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining*," Proceedings of 1st Open Source Data Mining Workshop on Frequent Pattern Mining Implementations, ACM Press, New York, NY, USA, pp.77-86, 2005. doi:10.1145/1133905.1133916
[13] P. Tan, M. Steinbach, V. Kumar, "*Introduction to Data Mining*," Pearson Education Asia Ltd. pp. 201-223, 2011.
[14] S. Lars, "*Algorithmic Features of Eclat*," FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004.
[15] J. Han, M. Kamber, "*Mining Concept and Techniques*," Second Edition, Elsevier(Singapore)Pte Ltd., pp.146-160, 2011.
[16] http://fimi.ua.ac.be/, 2012.
[17] http://www-users.cs.umn.edu/~huix/link/fim.htm, 2012.
[18] C. Borgelt, "*Frequent item set mining*," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, J. Wiley & Sons, Chichester, United Kingdom, vol.2, No.6, pp.437-456, 2012. doi:10.1002/widm.1074wiley.com.
[19] B. Thomas, C. Reynold, C. W. David, "*Model-based probabilistic frequent itemset mining*," Knowl Inf Syst, vol.37, pp.181-217, 2013. doi:10.1007/s10115-012-051-2
[20] A. Rajaraman, J. D. Ullman, "*Mining of massive datasets*," Post &Telecom Press. pp.89-113, 2012.
[21] J. Cheng, Y. Ke, W. Ng, "A survey on algorithms for mining frequent itemsets over data streams," Knowl. Inf. Syst., vol.16, pp.1–27, 2008. doi:10.1007/s10115-007-0092-4
[22] P. M. David, C. L. Iván, B. Christian, "*Fuzzy frequent pattern mining in spike trains*," Advances in Intelligent Data Analysis XI: Proc. 11th Int. Symposium, Helsinki, Finland, Springer-Verlag, Berlin/Heidelberg, Germany, pp.289-300, 2012. doi:10.1007/978-3-642-34156-4_27
[23] G. L. Gerstein, E. R. Williamns, M. Diesmann, S. Gr¨un, and C. Trengove, "*Detecting synfire chains in parallel spike data*," Elsevier, Amsterdam, Netherlands, Journal of Neuroscience Methods, vol. 206, pp.54–64, 2012. doi:10.1016/j.jneumeth.2012.02.003
[24] T. Calders, C. Garboni, B. Goethals, "*Efficient pattern mining of uncertain data with sampling*," Proceedings of the 14th Pacific-Asia conference on knowledge discovery and data mining, PAKDD, Hyderabad, India, Springer, Berlin, vol. I, pp.480–487, 2010. doi: 10.1007/978-3-642-13657-3_51

Xiaomei Yu was born in Tai'an in 1972. She gained Master's and Bachelor's degree in management science and engineering of Shandong normal university in 2004. Her research interests are data mining and big data cloud.

She has been working in Shandong Normal University for more than 10 years. Currently she is an Associate Professor of Computer Education at School of Information Science and Engineering. She is a PhD student working under supervision of Dr.Hong Wang. Three books or published articles she written are as follows.

[1]Xiaomei Yu. Webpage Design and Production, the Shiyou University press in China, pp.302 – 332, 2009.

[2]Xiaomei Yu. A New Mechanism to Enhance Transfer Performance Over Wired-cum-Wireless Networks, 2009 ITME2009, pp.560 – 564, 2009.

[3]Xiaomei Yu. Using LEGO Mindstorms in the Undergraduate Curriculum of IT. IEEE International Symposium on IT in Medicine and Education. pp.270 – 273, 2012.

Hong Wang was born in Tianjin in 1966. She gained Master's and Bachelor's degree in computer software of Tianjin University in 1988, gained PHD of the Chinese academy of sciences in 2002. Her main research direction includes complex networks, workflow, mobile agent, social software and big data cloud.

She has been working in Shandong Normal University since 1991. Currently she is a professor and doctoral supervisor of Computer Science department at School of Information Science and Engineering. She writes and publishes over 60 academic articles, representative ones as follows.

[1] Hong Wang, Connecting Migration Method in a Mobile Agent System, Applied Artifical Intelligence, vol.8, pp.772 – 780, 2009.

[2] Hong Wang, Modeling and Evaluating of Relation between Acupuncture Points and Diseases Based on Multi-dimension Networks, ITME2012, pp.534 – 538, 2012.

[3] Hong Wang, Characteristics of the Relation between Diseases and Acupuncture Points in Human Body, ITME2011, pp.34 – 37, 2011.