

# A Post-Analysis Framework for Mining Generalized Association Rules with Multiple Minimum Supports<sup>†</sup>

Fu-lai Chung and Chung-leung Lui

Department of Computing  
The Hong Kong Polytechnic University  
Hung Hom, Kowloon, Hong Kong.

{cskchung, cscllui}@comp.polyu.edu.hk

## ABSTRACT

Mining association rules at multiple concept levels leads to the discovery of more specific and concrete knowledge from data. Nevertheless, setting an appropriate minimum support for cross-level itemsets is still a non-trivial task. There exists a dilemma that if minimum support is set too high, low-level itemsets do not have enough support and may not be embedded by the high-level rules. On the other hand, if the minimum support is set too low, too many rules among which many are too general to the users will be mined. To address the problem, a post-analysis framework for mining generalized association rules with multiple minimum supports is proposed. A post-processing algorithm is applied to Srikant and Agrawal's generalized association rules (cumulate algorithm) so that low-level rules can have enough minimum support while the number of high-level rules is prevented from combinatorial explosion. Encouraging simulation results on a synthetic transaction database were observed and reported.

## Keywords

Generalized association rules, multiple minimum supports, generality.

## 1. INTRODUCTION

In recent years, the rapid growth in size and number of databases far exceeds human abilities to effectively and efficiently analyze such data. Therefore, it is necessary to devise an automated mechanism for better understanding and characterization of the huge amount of data on hand, in particular, using known phenomena and trends as well as discovering new and interesting knowledge. Knowledge discovery in databases (KDD) is generally considered as an emerging field of databases and machine learning. It is "the nontrivial extraction of implicit, previously unknown and potentially useful information in data" [11]. An interesting sub-field of it is the problem of discovering association rules. Generally, an association rule can be viewed as a relation over attributes and has the form  $X \rightarrow Y$ , where  $X$  and  $Y$  are set of items, called itemsets. Given a transaction database,

where each transaction is a set of items, the intuitive meaning of an association rule is that data of the database which contain the items in  $X$  tend to also contain the items in  $Y$ . Each rule is quantified by support and confidence. Support is the percentage of all transactions containing all items from  $X \cup Y$  and confidence is the percentage of all transactions containing all items from  $X$  also contain all items from  $Y$ . The problem of mining association rules is to find all rules that satisfy a user-specified minimum support (*minsup*) and minimum confidence (*minconf*) [1]. One of the limitations in mining association rules is that of granularity [3,10]. Rules at lower level may not have *minsup*. For example, given a set of supermarket transactions, while an association rule  $\{\text{Beer}\} \rightarrow \{\text{Nuts}\}$  achieves *minsup* and *minconf*, this rule may not be found, if the data is represented at a different granularity. The database may contain items such as brand X beer, brand Y beer, and brand Z nuts etc. At this finer level of granularity, there may not have any association (e.g.,  $\{\text{brand X beer}\} \rightarrow \{\text{brand Z nuts}\}$ ) that achieves *minsup* or *minconf* and consequently the  $\{\text{Beer}\} \rightarrow \{\text{Nuts}\}$  association will remain undiscovered.

Generalized association rules [10] and multiple level association rules [4,5] were proposed to address the above problem by assuming a taxonomy (is-a hierarchy) over the attributes [4,10]. Associations can then be mined from the items and nodes of the taxonomy. For example, different brands of beer and nuts could be grouped into a generalized term, called beer and nuts in taxonomy respectively. Consequently, the  $\{\text{Beer}\} \rightarrow \{\text{Nuts}\}$  could be mined as a generalized or multiple level association rule. Both methods facilitate the representation of a rule at all concept levels. Unfortunately, support in taxonomy is upward closed. Owing to this property, there exists a dilemma when setting *minsup* in generalized association rules. First, if *minsup* is set too high, low-level rules may not have *minsup* and they will not appear in high levels. Second, if *minsup* is set too low, this will cause combinatorial explosion, producing too many rules [8]. In fact, multiple level association rules [5] can set different *minsup* at each level, but the values have to be set manually and the way to do so lacks theoretical support. More importantly, it is a difficult task to set different *minsup* for each cross-level itemset.

In this work, rather than developing a sophisticated mining algorithm, a post-analysis framework for finding frequent itemsets with multiple minimum supports is proposed. The *minsup* of each itemset can be different, depending on its level of concept. Hence, low-level itemsets have smaller *minsup*, while high-level itemsets have larger *minsup*. This property leads to mining both low-level

---

<sup>†</sup> This work is supported by The Hong Kong Polytechnic University Research Studentship, project no. G-V722

and high-level rules in a simple and efficient manner. The paper is organized into five sections. The generalized association rules and upward closure property are introduced in Section 2. Section 3 presents the post-analysis framework for mining frequent itemsets with multiple minimum supports. The experimental results are reported in Section 4 and the final section concludes the paper.

## 2. GENERALIZED ASSOCIATION RULES AND UPWARD CLOSURE PROPERTY

Given a set of transactions  $D$ , a set of taxonomies  $\tau$  and a user-specified minimum interest (called *min-interest*), the problem of mining generalized association rules is to find all interesting association rules that have support and confidence greater than the user-specified *minsup* and *minconf* respectively. The problem of mining generalized association rules can be decomposed in three major steps [10]:

1. Find all sets of items (itemsets) whose support is greater than the user-specified *minsup*. Itemsets with *minsup* are called frequent itemsets.
2. Use the frequent itemsets to generate the desired rules. The general idea is that, if say, ABCD and AB are frequent itemsets, then we can determine if the rule  $AB \rightarrow CD$  holds by computing the ratio  $conf = \text{support}(ABCD)/\text{support}(AB)$ . If  $conf \geq \text{minconf}$ , then the rule holds.
3. Prune all uninteresting rules from this set.

Among the above processes, we are focusing on how to generate frequent itemsets with multiple minimum supports at step one. An efficient algorithm [10] for finding all frequent itemsets where the items can be from any level of the taxonomy was proposed. However, the *minsup* of all itemsets are the same. Single *minsup* implicitly assumes that all items in different level of taxonomy are of the same nature or have similar frequencies in the database. Unfortunately, this assumption is definitely invalid in practice. Support is *upward closed* over taxonomy. If an item satisfies the *minsup*, all its ancestors also satisfy the *minsup*. Owing to this *upward closure property*, the following dilemma exists:

1. If *minsup* is set too high, we may not discover rules at lower levels, because those rules may not have *minsup*. Moreover, we cannot make sure that those rules have been represented at higher levels, because they may not have *minconf*.
2. If *minsup* is set too low, this will cause combinatorial explosion, producing too many rules [8]. Most frequent items (especially for the high level ones) will be associated with one another in all possible ways and many of them are too general and ambiguous to the users.

For example, let us consider the following taxonomy. Item *Clothes* is the root of the taxonomy, and it has two children. They are *Jackets* and *Shirts*. The support of itemsets  $\{Jackets, Boots\}^1$  and  $\{Shirts, Boots\}$  are 5% and 10% respectively. Meanwhile, the support of  $\{Clothes, Boots\}$  must be  $\geq 10\%$ , may be 20%. If there are some rules like  $\{Clothes\} \rightarrow \{Boots\}$ ,  $\{Jackets\} \rightarrow \{Boots\}$ , and  $\{Shirts\} \rightarrow \{Boots\}$  with confidence 50%, 80%, and 90% respectively. Now, we want to find associations with *minsup* 15%,

and *minconf* 60%. There is no rule being discovered, because low-level rules do not have *minsup* while high-level rules do not have *minconf*. If we change *minsup* to 5%, we find two rules instead. Thus, the number of discovered rules grows exponentially, e.g., a rule  $\{Clothes\} \rightarrow \{Food\}$  with support = 6% and confidence = 80% is also discovered, but it is not an interesting one.

## 3. MINING FREQUENT ITEMSETS WITH MULTIPLE MINIMUM SUPPORTS

Over a taxonomy (or a concept hierarchy), high-level item is implicitly having higher support than its lower level items in accordance with the upward closure property. Now, suppose item  $x'$  is an ancestor of item  $x$ , such that the *minsup* of an itemset  $\{x'\}$  should be larger than that of itemset  $\{x\}$ . If  $\text{minsup}(\{x'\})$  and  $\text{minsup}(\{x\})$  denote the minimum supports of itemsets  $\{x'\}$  and  $\{x\}$  respectively, then  $\text{minsup}(\{x'\}) > \text{minsup}(\{x\})$ . Thus, we define  $\text{minsup}(t)$  for the independent *minsup* of an itemset  $t$ , where  $t$  is a variable. The notation *minsup* (without the bracketed information) holds the same meaning in [10], but not for all itemsets. It is just for the itemset whose items are all at the leaf-level of taxonomy. Hence,  $\text{minsup}(t) \geq \text{minsup}$ .

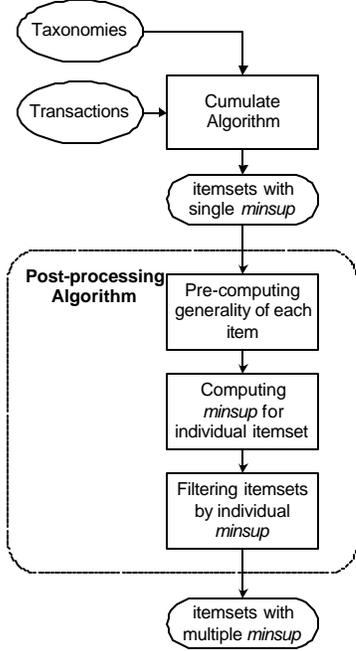
In [10], efficient algorithms for finding large itemsets of generalized association rules are based on level-wise search. Let  $k$ -itemset denote an itemset with  $k$  items. All the potential large itemsets at level  $k$  are generated from large itemsets at level  $k-1$ . However, under the concept of multiple minimum supports, there has a problem in the existing algorithms. For example, consider the following itemsets  $\{1\}$ ,  $\{2\}$ , and  $\{1, 2\}$  and their minimum supports are:

$$\begin{aligned} \text{minsup}(\{1\}) &= 10\% \\ \text{minsup}(\{2\}) &= 20\% \\ \text{minsup}(\{1, 2\}) &= 10\% \end{aligned}$$

If we find that itemsets  $\{1\}$  and  $\{1, 2\}$  have enough *minsup* but itemset  $\{2\}$  has not, then itemset  $\{1, 2\}$  cannot be generated using the existing algorithms, even though  $\{1, 2\}$  is a large itemset. Thus, it is necessary to propose a new method for solving the above problem. Now, we have two directions to explore efficient mining of generalized association rules with multiple minimum supports. One choice is making of a derivative algorithm from the existing algorithms in [10], but it may lead to the destruction of level-wise characteristic in the algorithms. Another choice is to apply a post-processing method to extract out the large itemsets  $t$  with  $\text{minsup}(t)$  after generating the itemsets with single *minsup*, where  $\text{minsup} \leq \text{minsup}(t)$ , using the algorithms in [10].

It is apparently that the second choice is simple and direct for finding frequent itemsets with multiple minimum supports. It is to first generate all itemsets  $S$  with single *minsup* using the Cumulate algorithm proposed in [10], and then apply a post-processing step to extract the set of itemsets where each itemset in has independent  $\text{minsup}(\ )$ . Note here that all  $\text{minsup}(\ ) \geq \text{minsup}$ , so  $\subseteq S$ . Now, the problem is how to determine the minimum support of each itemset independently and dynamically. Here, we propose a post-processing algorithm based on the concept of generality for solving the problem, and sketch the whole mining process for discovering frequent itemsets with multiple minimum supports.

<sup>1</sup> Boots is an item in another taxonomy

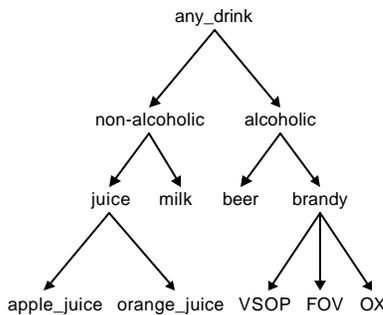


**Figure 1. A post-analysis framework for mining itemsets with multiple minsup**

In accordance with the framework in figure 1, the post-processing algorithm can be divided into three parts. The first step is to generate a list that pre-computes the generality of each item (no matter it is interior node or leaf node item). Second, each itemset discovered from the Cumulate algorithm [10] is assigned an individual minimum support using the itemset's generality. Finally, it is a simple task to extract out those itemsets that have individual minimum support in the last step.

### 3.1 Generality

Generality of an item indicates its level of concept over the taxonomy. Before proceeding to introduce generality, let us review the background theory of concept generalization. Concept generalization of an item means forming a general concept description (non-leaf node items) of a class of objects (leaf node items) given a set of concept trees (taxonomies) [6,7,9]. Intuitively, very different items generalize to an expression that is very far from each of them, while identical items generalize to themselves. That calls for the notion of generality of an item. For instance, Figure 2 depicts a taxonomy that describes the concept hierarchy of *drink*.



**Figure 2. Taxonomy of drink**

Under the above theory, each item (no matter it is non-leaf node or leaf node) is associated to a predicate that describes both concepts and objects within the same class. If two items are under the same predicate, we call they have common predicate [7]. Given items with the following description, (*predicate object*), we can determine their generalized form and generality. Now, let us consider the following four items with reference to the concept tree in Figure 2. For example, the items  $m_i$  are in the following form

- $m1: (drink\ apple\_juice)$
- $m2: (drink\ milk)$
- $m3: (drink\ orange\_juice)$
- $m4: (drink\ beer)$

They are similar, as characterized by the same predicate *drink*. Since each of these four items is characterized by the same predicate, they can be generalized into general terms at higher concept level. Let us also consider the following generalizations  $G(m_i, m_j)$  of two items  $m_i$  and  $m_j$ :

- $G(m1, m3): (drink\ juice)$
- $G(m1, m2): (drink\ non-alcoholic)$
- $G(m1, m4): (drink\ any\_drink)$

Note here that *juice*, *non-alcoholic*, and *any\_drink* are all values from the structured domain in Figure 2, and that *juice* is less general than *non-alcoholic* and *non-alcoholic* is less general than *any\_drink*. While *orange\_juice* is an instance, *juice* is a generalization with 2 instances and *any\_drink* is a generalization with 7 instances. Thus, we could define the degree of generality of an argument  $g(a)$ , as the ratio of the number of argument's instances to the total number of instances from argument's domain [7], that is:

$$g(a) = \frac{\text{number of instance of "a"}}{\text{number of instance of the domain of "a"}} \quad (1)$$

For example,

$$\begin{aligned} g(juice) &= 2/7 \\ g(non-alcoholic) &= 3/7 \\ g(any\_drink) &= 7/7 \end{aligned}$$

In fact, mining generalized association rules from the items and nodes of the taxonomy is a form of using generalized concept to represent a rule. In particular, any item in the non-leaf nodes of taxonomy is a generalized form of its descendants. This generalized form (non-leaf node item) can be measured by its generality to indicate its level of concept over the taxonomy. This definition applies also to an itemset. The generality of an itemset is computed by taking the minimum value of the generalities of all its items, because number of count of an itemset is restricted by its item that is the less frequent one in the database. Let  $t$  be an itemset, and  $x_i$  be an item in  $t$ . The generality of  $t$  is defined as:

$$g(t) = \min[g(x_1), g(x_2), \dots, g(x_n)] \quad (2)$$

For example, the generality of itemset  $t1$  and  $t2$ :

$$\begin{aligned} t1: (drink\ apple\_juice) (drink\ alcoholic) \\ g(t1) &= \min(0, 4/7) = 0 \\ t2: (drink\ juice) (drink\ non-alcoholic) \\ g(t2) &= \min(2/7, 3/7) = 2/7 \end{aligned}$$

Note also that the predicate of each item  $x_i$  can be different.

### 3.2 Multiple Minimum Supports

Here, we propose a method to calculate multiple minimum supports for different itemsets dynamically from their generalities. Recall from previous sub-section that we can determine the generality of each itemset separately by traversing the taxonomy graph. Nevertheless, in order to increase the efficiency, we can pre-compute the generality of each item and the generality of an itemset can be deduced from generalities of all its items. Details are described below.

#### 3.2.1 Pre-computing generality of each item

Rather than calculating each itemset's generality by traversing the taxonomy graph, pre-computing the generality of each item can speed up the algorithm significantly. We first generate a list of item's generality. It is a simple task to compute the generality of each item (both non-leaf and leaf node). For example:

$$\begin{aligned} g(\text{apple\_juice}) &= 0/7 = 0 \\ g(\text{juice}) &= 2/7 \\ g(\text{non-alcoholic}) &= 3/7 \\ g(\text{any\_drink}) &= 7/7 = 1 \end{aligned}$$

and so on.

#### 3.2.2 Computing minimum support for individual itemset

Before computing the individual  $minsup(t)$ , the first task is to compute the generality of an itemset. Generality of each itemset can be determined from the pre-computed generalities of all its items. According to the item's generality list, any combination in itemset can be deduced from its generality, even though its items are coming from different taxonomies. For example, we have the following itemset:

$$\begin{aligned} t1: (\text{drink apple\_juice}) (\text{drink alcoholic}) \\ g(t1) &= \min(0, 4/7) = 0 \end{aligned}$$

This generality value indicates the level of concept of the itemset. The value of generality is ranging from zero to one, where values zero and one denote an itemset at the lowest and highest concept level respectively over the taxonomies. If the  $minsup(y)$  of an itemset  $y$  at the lowest concept level is  $\beta$ , we call this *lower-minimum support*. On the other hand, the  $minsup(x)$  of itemset  $x$  at the highest concept level must be greater than  $\beta$ , because higher minimum support for higher level itemsets can prevent combinatorial explosion. So, we define *lower-* and *upper-minimum support* for the lowest and highest concept levels itemsets respectively and they will be specified by the users.

**Definition 1:** The *lower-minimum support*  $\mathbf{b}$  is a user-defined parameter. This is  $minsup$  for the lowest concept level itemsets.

**Definition 2:** The *upper-minimum support*  $\mathbf{a}$  is a user-defined parameter. This is  $minsup$  for the highest concept level itemsets, and the value must be greater than *lower-minimum support*, i.e.,  $\alpha > \beta$ .

For an itemset  $t$ , its generality is varying from zero to one. Based upon  $\mathbf{a}$  and  $\mathbf{b}$ , the minimum support for each itemset  $t$  is computed independently as:

$$minsup(t) = \mathbf{b} + \{(\mathbf{a}-\mathbf{b}) * g(t)\} \quad (3)$$

Here,  $minsup(t)$  varies from *lower-minimum support* to *upper-minimum support*, depending on the generality of the itemset  $t$ . For example, if  $\mathbf{a} = 20$ ,  $\mathbf{b} = 10$ , and  $g(t) = 0.2$ , then  $minsup(t) = 12$ . Subsequently, the itemsets in  $S$  can be filtered by their own minimum support dynamically. Thus, each extracted itemset in  $S$  has its own  $minsup$  value.

## 4. SIMULATION RESULTS

In this section, we evaluate the performance of the proposed multiple  $minsup$  based generalized association rule algorithm against the original one with single  $minsup$  [10]. Measurements in terms of number of itemsets and number of rules will be reported.

### 4.1 Synthetic Data Generation

The synthetic data generation program was obtained from the IBM Almaden Research Center [http://www.almaden.ibm.com/cs/quest]. The essential idea behind the synthetic data generation program is to first build a taxonomy over the items. For any non-leaf node, the number of children is picked from a Poisson distribution with mean  $\mu$  equal to fanout  $F$ . The program assigns children to the roots, then to the nodes at the 2<sup>nd</sup> level, and so on, till run out of items. The program, on the other hand, generates a table of potentially frequent itemsets  $I$ , and then creates transactions by picking itemsets from  $I$  and inserting them in the transaction. Details of the program can be found in [2,10]. A set of sample parameter values for synthetic data generation is shown as follows:

**Table 1. Parameters for synthetic data generation with default values**

Parameter	Value	
$ D $	Number of transactions	100,000
$ T $	Average size of the transactions	5
$ I $	Average size of the maximal potentially frequency itemsets	4
N	Number of items	1,000
R	Number of roots	10
L	Number of levels	5
F	Fanout at each node	3

**Table 2. Parameters for running the experiments with default values**

Algorithm	Parameter	Value
Single $minsup$	Minimum support	5%
	Multiple $minsup$	Lower-minimum support
Multiple $minsup$	Upper-minimum support	5% - 15%
	Both	Minimum confidence
r-interest		1.1

### 4.2 Performance Analysis

The performance of the proposed method with multiple  $minsup$  is evaluated as follows. All parameters are set to the default values as shown in Tables 1 and 2. We first ran the algorithm with single  $minsup$  at 5% and 15%. Then, we fixed the *lower-minimum support*  $\mathbf{b}$  at 5% and varied the *upper-minimum support*  $\mathbf{a}$  from 5% to 15%. From the results plotted in Figures 3 and 4, the number of itemsets/rules found at  $\alpha = 5\%$  is exactly the same as

that of single *minsup* fixed at 5%. At  $\alpha = 15\%$ , the number of itemsets/rules is more than single *minsup* with 15%. It is because not all itemsets' *minsup* are 15%. In fact, their *minsup* vary from 5% to 15%.

From Figure 5, on the other hand, the percentage of itemsets being removed from single *minsup* = 5% is about 23% when  $\alpha = 15\%$ , while the percentage of rules being removed from single *minsup* = 5% is about 41% at  $\alpha = 15\%$ . This observation indicates that removing the itemsets with multiple *minsup* can effectively prevent combinatorial explosion in mining generalized association rules, because the number of itemsets was removed by 23% only while the number of rules being removed is 41%. This shows that the removed itemsets are the major sources in producing a lot of meaningless rules in the original algorithm.

Note here that the execution time of the multiple *minsup* algorithm is similar to the single *minsup* algorithm, when the *lower-minimum support* is equal to *minsup*. It is because the post-analysis step does not have to scan the database, and it only applies multiple *minsup* to remove itemsets from the domain of the original algorithm in [10]. In the above case, the execution of our approach is very similar to the single *minsup* algorithm with *minsup* = 5%, no matter how the *upper-minimum support* is changed. It is because the *lower-minimum support* was fixed.

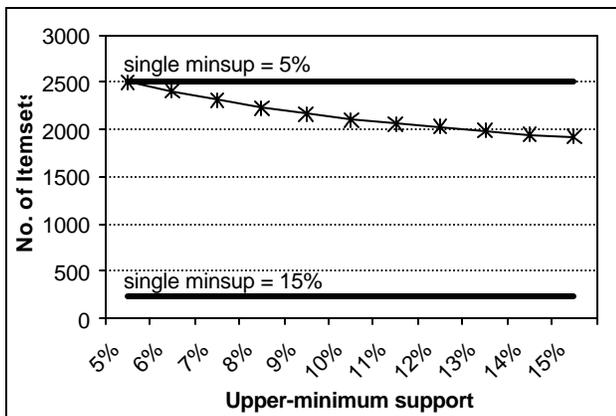


Figure 3. Number of itemsets discovered with different upper-minimum supports

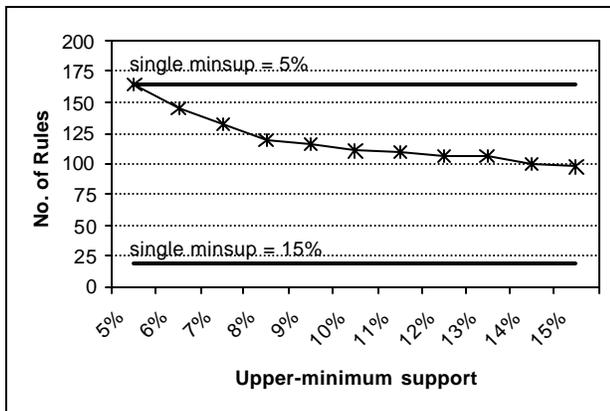


Figure 4. Number of rules discovered with different upper-minimum supports

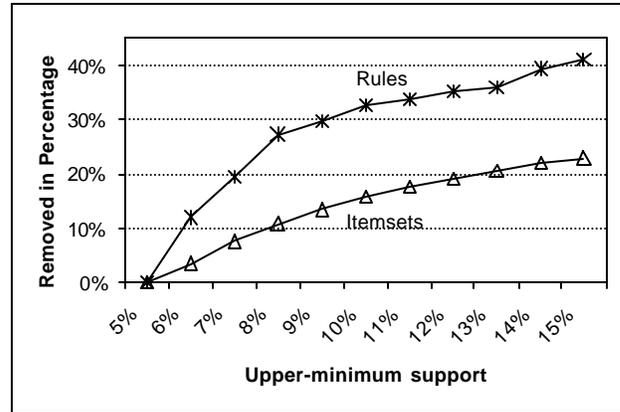


Figure 5. Percentage of itemsets and rules being removed from single *minsup* = 5%

## 5. CONCLUSIONS

We have introduced a post-analysis framework for finding frequent itemsets with multiple minimum supports from large transaction databases. The proposed approach eliminates some of the limitations of most existing generalized or multiple level association rules mining algorithms, e.g., they make use of single *minsup* or set the value at each level manually. In our approach, on the other hand, low-level rules can have enough *minsup*, while the number of high-level rules is prevented from combinatorial explosion.

## 6. REFERENCES

- [1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between sets of items in large databases," *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207-216, Washington, D.C., May, 1993.
- [2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the VLDB Conference*, Santiago, Chile, Sep, 1994.
- [3] A. Andrusiewicz and M. E. Orłowska, "On data granularity factors that affect data mining," *Proceedings of the 8<sup>th</sup> International Database Workshop*, 1997.
- [4] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," *Proceedings of the 21<sup>st</sup> VLDB Conference*, Zurich, Switzerland, 1995.
- [5] J. Han and Y. Fu, "Mining multiple-level association rules in large databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 5, 1999
- [6] Y. Kodratoff, and J. G. Ganascia, "Improving the generalization step in learning," *Machine Learning: An Artificial Intelligence Approach*, vol. 2, Los Angeles, pp. 215-244, 1986.
- [7] Y. Kodratoff, and G. Tecuci, "Learning Based on Conceptual Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 6, Nov 1988.

- [8] B. Liu, W. Hsu, and Y. Ma, "Mining Association Rules with Multiple Minimum Supports," *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 337-341, San Diego, CA, USA, August, 1999.
- [9] R. S. Michalski, and R. Stepp, "Learning by observation," *Machine Learning: An Artificial Intelligence Approach*, pp. 163-190, 1983.
- [10] R. Srikant and R. Agrawal, "Mining generalized association rules," *Proceedings of the 21<sup>st</sup> VLDB Conference*, Zurich, Switzerland, pp. 407-419, 1995.
- [11] J. M. Zytkow, "The KDD land of plenty," *AAAI Workshop Notes – Knowledge Discovery Databases*, Anaheim, CA, pp. iii-vi, July 14, 1991.