Efficient Remining of Generalized Multi-supported Association Rules under Support Update

WEN-YANG LIN [1] and MING-CHENG TSENG [2]
[1] Dept. of Information Management, [2] Institute of Information Engineering
I-Shou University
1, Section 1, Hsueh-Cheng Rd., Ta-Hsu Hsiang, Kaohsiung 84041
TAIWAN

Abstract: Mining generalized association rules among items in the presence of taxonomy and with nonuniform minimum support has been recognized as an important model in data mining. In our previous work, we have investigated this problem and proposed two algorithms, MMS_Cumulate and MMS_Stratify. In real applications, however, the work of discovering interesting association rules is an iterative process. The analysts need to repeatedly adjust the constraint of minimum support and/or minimum confidence to discover real informative rules. How to reduce the response time for each remining process thus becomes a crucial issue. In this paper, we examined the problem of maintaining the discovered multi-supported generalized association rules when the multiple minimum support constraint is updated. Empirical evaluation showed that the proposed RM_MMS algorithm is very efficient and has good linear scale-up characteristic.

Key-Words: Data remining, generalized association rules, multiple minimum supports, taxonomy.

1 Introduction

Mining association rules from a large database of business data, such as transaction records, has been a popular topic within the area of data mining [1, 2, 6]. This problem is originally motivated by application known as market basket analysis to find correlations among items purchased by customers.

An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are sets of items. Such a rule reveals that transactions in the database containing items in X tend to contain items in Y, and the probability, measured as the fraction of transactions containing X also containing Y, is called the *confidence* of the rule. The *support* of the rule is the fraction of the transactions that contain all items in both X and Y. For an association rule to be valid, the rule should satisfy a user-specified minimum support, called *ms*, and minimum confidence, called *minconf*, respectively. The problem of mining association rules is to discover all association rules that satisfy *ms* and *minconf*. For example, an association rule, Desktop $\Rightarrow$ Ink-jet (*sup* = 30%, *conf* = 60%), says that 30% (support) of customers purchase both Desktop PC and Ink-jet printer together, and 60% (confidence) of customers who purchase Desktop PC also purchase Ink-jet printer. Such information is useful in many aspects of market management, such as store layout planning, target marketing, understanding customers' purchasing behavior, etc.

In many applications, there are taxonomies (hierarchies), explicitly or implicitly, over the items. In some applications, it may be more useful to find associations at different levels of the taxonomy than only at the primitive concept level [3, 7]. For example, consider Figure 1, the taxonomy of items from which the previous association rule derived. It is likely to happen that the association rule, Desktop $\Rightarrow$ Ink-jet (*sup* = 30%, *conf* = 60%), does not hold when the minimum support is set to 40%, but the following association rule may be valid, PC $\Rightarrow$ Printer.

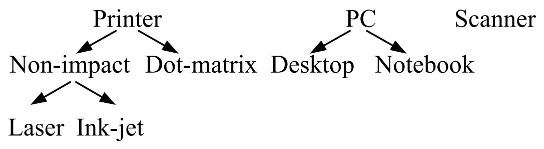

Fig. 1 An example of taxonomy *T*.

In our previous work, we have investigated the problem of mining generalized association rules across different levels of taxonomy with multiple minimum supports [8]. We proposed two efficient algorithms, MMS_Cumulate and MMS_Stratify, which not only can discover associations that span different hierarchy levels but also have high potential to produce rare but informative item rules. In real applications, however, the work of discovering interesting association rules is an iterative process. The analysts need to repeatedly adjust the constraint of minimum support and/or minimum confidence to discover real informative rules. How to reduce the response time for each remining process thus become a crucial issue.

One way for dealing with this issue is to adopt the current mining approach to re-mine the database from scratch, which, however, has the following disadvantages:

1. It is not cost-effective because the discovered frequent itemsets are not utilized.
2. This is not acceptable in general since generating frequent itemsets is time-consuming.

To be more realistic and cost-effective, it is better to perform the association mining algorithms to generate the initial association rules, and when update to the multiple minimum supports occurs, then apply an updating method to re-build the discovered rules. The challenge falls into deploying an efficient updating algorithm to facilitate the whole mining process. This problem is nontrivial because update may invalidate some of the discovered association rules, and turn previous weak rules into strong ones.

In this paper, we consider the updating approach and propose an algorithm, called the RM_MMS (ReMining under Multiple Minimum Supports Update). Our algorithm, by utilizing the discovered frequent itemsets, can significantly reduce the number of candidate itemsets and database rescanning. Empirical evaluation showed that our algorithm is very efficient and has linear scalability.

The remaining of this paper is organized as follows. A review of related work is given in Section 2. The problem statement is formalized in Section 3. In Section 4, we explain how to remine generalized association rules under multiple minimum supports update and propose an algorithm. The evaluation of the proposed algorithm is described in Section 5. Finally, our conclusion is stated in Section 6.

2 Related Work

The problem of mining association rules in presence of taxonomy information is first addressed in [4, 7], independently. In [7], they aims at finding associations among items at any level of the taxonomy under the *ms* and *minconf* constraint. In [4], they primarily devote to mining associations level-by-level in a fixed hierarchy. But they have generalized the uniform minimum support constraint to a form of level-wise assignment.

Another form of association rule model with multiple minimum supports has been proposed in [5]. Their method allows users to specify different minimum support to different item and can find rules involving both frequent and rare items. However, their model considers no taxonomy at all, and hence fails to find generalized association rules.

3 Problem Formulation

3.1 Mining generalized association rules with multiple minimum supports

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and $DB = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions, where each transaction $t_i = \langle tid, A\rangle$ has a unique identifier *tid* and a set of items A ($A \subseteq I$). We assume that the taxonomy of items, T, is available and is denoted as a directed acyclic graph on $I \cup J$, where $J = \{j_1, j_2, \ldots, j_p\}$ represents the set of generalized items derived from I. An edge in T denotes an *is-a* relationship. That is, if there is an edge from j to i, we call j a parent (generalization) of i and i a child of j. For example, in Figure 1, I = {Laser printer, Ink-jet printer, Dot matrix printer, Desktop PC, Notebook, Scanner} and J = {Non-impact printer, Printer, Personal computer}.

Definition 1 Given a transaction $t = \langle tid, A\rangle$, we say an itemset B is in t if every item in B is in A or is an ancestor of some item in A. An itemset B has support s, denoted as $s = sup(B)$, in the transaction set DB if s% of transactions in DB contain B.

Definition 2 Given a set of transactions DB and a taxonomy T, a generalized association rule is an implication of the form $A \Rightarrow B$, where $A, B \subset I \cup J$, $A \cap B = \varnothing$, and no item in B is an ancestor of any item in A. The support of this rule, $sup(A \Rightarrow B)$, is equal to the support of $A \cup B$. The confidence of the rule, $conf(A \Rightarrow B)$, is the ratio of $sup(A \cup B)$ and $sup(A)$.

The condition in Definition 2 that no item in B is an ancestor of any item in A is essential; otherwise, a rule of the form, $a \Rightarrow ancestor(a)$, always has 100% confidence and is trivial.

Definition 3 Let $ms(a)$ denote the minimum support of an item a in $I \cup J$. An itemset $A = \{a_1, a_2, \ldots, a_k\}$, where $a_i \in I \cup J$, is *frequent* if the support of A is larger than the lowest value of minimum support of items in A, i.e., $sup(A) \geq \min_{a_i \in A} ms(a_i)$.

Definition 4 A multi-supported, generalized association rule $A \Rightarrow B$ is *strong* if

$$sup(A \Rightarrow B) \geq \min_{a_i \in A \cup B} ms(a_i)$$

and

$$conf(A \Rightarrow B) \geq minconf.$$

Definition 5 Given a set of transactions DB, a taxonomy T, the user-specified minimum supports for all items in T, $\{ms(a_1), ms(a_2), \ldots, ms(a_n)\}$, and the *minconf*, the problem of mining multi-supported, generalized association rules is to find all association rules that are strong.

Example 1 Suppose that a shopping transaction database *DB* in Table 1 consists of items *I* = {Laser printer, Ink-jet printer, Dot matrix printer, Desktop PC, Notebook, Scanner} and taxonomy *T* as shown in Figure 1. Let the minimum confidence (*minconf*) be 60% and the minimum support (*ms*) associated with each item in the taxonomy be as follows: *ms*(Printer) = 80%, *ms*(Non-impact) = 65%, *ms*(Laser) =25%, *ms*(Dot matrix) = 70%, *ms*(Ink-jet) = 60%, *ms*(PC) = 35%, *ms*(Desktop) = 25%, *ms*(Notebook) = 25%, *ms*(Scanner) = 15%. The following generalized association rule, PC, Laser ⇒ Dot matrix (*sup* = 16.7%, *conf* = 50%), fails because its support is less than min{*ms*(PC), *ms*(Laser), *ms*(Dot matrix)} = 25%. But another rule, PC ⇒ Laser (*sup* = 33.3%, *conf* = 66.7%), holds because both its support and confidence are larger than min{*ms*(PC), *ms*(Laser)} = 25% and *minconf*, respectively. Table 2 lists the frequent itemsets and the resulting strong rules for this example.

Table 1 A transaction database (*DB*).

TID	Items Purchased
11	Notebook, Laser printer
12	Scanner, Dot matrix printer
13	Dot matrix printer, Ink-jet printer
14	Notebook, Dot matrix printer, Laser printer
15	Scanner
16	Desktop computer

Table 2 Frequent itemsets and resulting association rules for Example 1.

Frequent Itemsets	min *ms* (%)	*sup* (%)
{Scanner}	15	33.3
{Notebook}	25	33.3
{Laser}	25	33.3
{PC}	35	50.0
{Scanner, Printer}	15	16.7
{Scanner, Dot matrix}	15	16.7
{Laser, PC}	25	33.3
{Notebook, Printer}	25	33.3
{Notebook, Non-impact}	25	33.3
{Notebook, Laser}	25	33.3

Rules
PC ⇒ Laser (*sup* = 33.3%, *conf* = 66.7%)
Laser ⇒ PC (*sup* = 33.3%, *conf* = 100%)
Notebook ⇒ Printer (*sup* = 33.3%, *conf* = 100%)
Notebook ⇒ Non-impact (*sup* = 33.3%, *conf* = 100%)
Notebook ⇒ Laser (*sup* = 33.3%, *conf* = 100%)
Laser ⇒ Notebook (*sup* = 33.3%, *conf* = 100%)

As founded in [1], the work of association rules mining can be decomposed into two phases:

1. Frequent itemsets generation: find all itemsets that sufficiently exceed the *ms*.
2. Rules construction: from the frequent itemsets generate all association rules having confidence higher than the *minconf*.

Since the second phase is straightforward and less expensive, we concentrate only on the first phase of finding all frequent itemsets.

3.2 Remining generalized multi-supported association rules under support update

As stated in previous section, the analysts need to repeatedly adjust the constraint of minimum support to discover real informative rules. This implies that if the constraint of minimum support is adjusted, the previous discovered associations might be invalid and some undiscovered associations should be generated. That is, the discovered association rules must be updated to reflect the new circumstance. Analogous to the associations mining, this problem can be reduced to the problem of updating the frequent itemsets.

Example 2 Consider Example 1 again. The *ms* for each items is updated as follows: *ms*(Printer) = 60%, *ms*(Non-impact) = 55%, *ms*(Laser) = 10%, *ms*(Dot matrix) = 50%, *ms*(Ink-jet) = 20%, *ms*(PC) = 55%, *ms*(Desktop) = 25%, *ms*(Notebook) = 35%, *ms*(Scanner) = 15%. Table 3 lists the frequent itemsets and the resulting strong rules.

Table 3 Frequent itemsets and resulting association rules for Example 2.

Frequent Itemsets	min *ms* (%)	*sup* (%)
{Laser}	10	33.3
{Scanner}	15	33.3
{Dot matrix}	50	50.0
{Printer}	60	66.7
{Laser, Notebook}	10	33.3
{Laser, Dot matrix}	10	16.7
{Laser, PC}	10	33.3
{Scanner, Printer}	15	16.7
{Scanner, Dot matrix}	15	16.7
{Laser, Notebook, Dot matrix}	10	16.7
{Laser, Dot matrix, PC}	10	16.7

Rules
Laser ⇒ PC (*sup* = 33.3%, *conf* = 100%)
Laser ⇒ Notebook (*sup* = 33.3%, *conf* = 100%)
Laser, Dot matrix ⇒ PC (*sup* = 16.7%, *conf* = 100%)
Laser, Dot matrix ⇒ Notebook (*sup* = 16.7%, *conf* = 100%)

Comparing Table 3 to Table 2, we observe that four old frequent itemsets in Table 2 {Notebook}, {PC}, {Notebook, Printer}, and {Notebook, Non-impact} are discarded, while five new frequent itemsets, {Dot matrix}, {Printer}, {Laser, Dot matrix}, {Laser, Notebook, Dot matrix } and {Laser, Dot matrix, PC}, are added into Table 3.

.

4 The Proposed Method

4.1 Algorithm basics

The primary challenge of devising effective association rule remining algorithm is how to reuse the discovered frequent itemsets and avoid the possibility of re-scanning the database.

Let a k-itemset denote an itemset with k items. The basic process of our generalized association rules remining algorithm under multiple minimum supports update follows the level-wise approach adopted by most Apriori-like algorithms. However, the well-known Apriori pruning technique based on the concept of *downward closure* does not work for multiple support specification. For example, consider four items a, b, c, and d that have minimum supports specified as $ms(a) = 15\%$, $ms(b) = 20\%$, $ms(c) = 4\%$, and $ms(d) = 6\%$. Clearly, a 2-itemset $\{a, b\}$ with 10% of support is discarded for $10\% < \min\{ms(a), ms(b)\}$. According to the downward closure, the 3-itemsets $\{a, b, c\}$ and $\{a, b, d\}$ will be pruned even though their supports may be larger than $ms(c)$ and $ms(d)$, respectively. To solve this problem, we have adopted the *sorted closure property* [5] in our previous work for mining generalized association rules with multiple minimum supports. Hereafter, to distinguish from the traditional itemset, a sorted k-itemset denoted as $\langle a_1, a_2, \ldots, a_k\rangle$ is used.

Lemma 1 If a sorted k-itemset $\langle a_1, a_2, \ldots, a_k\rangle$, for $k \geq 2$ and $ms(a_1) \leq ms(a_2) \leq \ldots \leq ms(a_k)$, is frequent, then all of its sorted subsets with $k-1$ items are frequent, except the subset $\langle a_2, a_3, \ldots, a_k\rangle$.

Lemma 2 For $k = 2$, the procedure apriori-gen(L_1) fails to generate all candidate 2-itemsets in C_2.

For example, consider a sorted candidate 2-itemset $\langle a, b\rangle$. It is easy to find if we want to generate this itemset from L_1, both items a and b should be included in L_1; that is, each one should be occurring more frequently than the corresponding minimum support $ms(a)$ and $ms(b)$. Clearly, the case $ms(a) \leq sup(b) < ms(b)$ fails to generate $\langle a, b\rangle$ in C_2 even $sup(\langle a, b\rangle) \geq ms(a)$.

Lemma 3 For $k \geq 3$, any k-itemset $A = \langle a_1, a_2, \ldots, a_k\rangle$ generated by procedure apriori-gen(L_{k-1}) can be pruned if there exists one ($k-1$) subset of A, say $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_{k-1}}\rangle$, such that $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_{k-1}}\rangle \notin L_{k-1}$ and $a_{i_1} = a_1$ or $ms(a_{i_1}) = ms(a_{i_2})$.

For more details, please refer to [8].

4.2 Algorithm RM_MMS

The basic process of the RM_MMS algorithm is as follows. First, count all 1-itemsets which is infrequent correspondently to the old multiple minimum supports (ms_{old}) setting. Second, create the new frequent 1-itemsets L_1^{new} according to the new multiple minimum supports (ms_{new}) setting. Next, create the frontier set F and use it to generate candidate 2-itemsets C_2. Then, generate the frequent 2-itemsets L_2^{new} by scanning some part of C_2 in DB. Finally, for $k \geq 3$, repeatedly generate candidate k-itemsets C_k from L_{k-1} and create frequent k-itemsets L_k^{new} until no frequent itemsets. In each iteration k for C_k generation, an additional pruning technique with enhancements applied in [8] are performed.

Lemma 4 An itemset is frequent with respect to ms_{new} if it is frequent with respect to ms_{old} and $ms_{new} \leq ms_{old}$.

Lemma 5 An itemset is infrequent with respect to ms_{new} if it is infrequent with respect to ms_{old} and $ms_{new} \geq ms_{old}$.

Lemma 6 An itemset is uncertain of frequency with respect to ms_{new} if it is frequent with respect to ms_{old} and $ms_{new} > ms_{old}$, or if it is infrequent with respect to ms_{old} and $ms_{new} < ms_{old}$.

For itemsets satisfying Lemma 4, there is no need to rescan the database DB to determine whether they are frequent. In Lemma 6, one case is that the itemsets which are frequent with respect to ms_{old} and $ms_{new} > ms_{old}$ are not required to rescan the database DB, but they are required to be calculated to determine whether they are frequent, and the other case is that the itemsets which are infrequent with respect to ms_{old} and $ms_{new} < ms_{old}$ need to rescan the database DB.

The main steps of RM_MMS Algorithm are presented as follows:

Inputs: (1) DB: the database; (2) the old multiple minimum supports (ms_{old}) setting; (3) the new multiple minimum supports (ms_{new}) setting; (4) T: the item taxonomy; (5) $L^{old} = \bigcup_k L_k^{old}$ the set of old frequent itemsets.

Output: $L^{new} = \bigcup_k L_k^{new}$: the set of new frequent itemsets corresponding to ms_{new}.

Steps:

1. Divide the set of 1-itemsets C_1 into two parts: one (X_1) consists of items in L_1^{old}, and the other (Y_1) contains those not in L_1^{old}.
2. Add generalized items in T into DB as DB'.
3. Count the supports of Y_1 in DB'.
4. Sort C_1 in increasing order of their ms and create frontier set F and L_1^{new}.
5. Generate the set of candidate 2-itemsets C_2 from F.
6. Divide C_2 into two parts: the itemsets in L_2^{old} and those not in L_2^{old}. For the itemsets in L_2^{old}, further divide them into two parts: X_{2a} for $ms_{new} \leq ms_{old}$ and X_{2b} for $ms_{new} > ms_{old}$. For those not in L_2^{old}, further divide them into two parts: Y_{2a} for $ms_{new} < ms_{old}$ and Y_{2b} for $ms_{new} \geq ms_{old}$.
7. Count the supports of itemsets in Y_{2a} over DB'.
8. Create L_2^{new} by combining X_{2a} and those itemsets which are frequent in X_{2b} and Y_{2a}.
9. Generate candidates C_3 from L_2.
10. Repeat Steps 6-9 for new candidates C_k until no frequent k-itemsets L_k^{new} created.

For more details of generating frontier set F, please refer to [8]. An example illustrates the RM_MMS algorithm is provided in Appendix, where item "A" stands for "Printer", "B" for "Non-impact printer", "C" for "Laser printer", "D" for "Dot matrix printer", "E" for "Ink-jet printer", "F" for "Personal computer", "G" for "Desktop PC", "H" for "Notebook", and "I" for "Scanner".

5 Experiments

In this section, we evaluate the performance of algorithm, RM_MMS, using the synthetic dataset generated by IBM data generator [2]. The parameter settings are shown in Table 4. The experiments were performed on an Intel Pentium-IV 2.80GHz with 2096MB RAM, running on Windows 2000.

We conducted an experiment to compare the efficiency of these three algorithms under various sizes of database. The minimum supports were specified to items randomly, ranging from 1.0% to 6.0%. Here, we adopt the ordinary case that the minimum support of an item a is no larger than any of its ancestors $\hat{a}$, i.e., $ms(a) \leq ms(\hat{a})$. We implemented the experiment in two cases: case 1 (the worst one) for $ms_{new} < ms_{old1}$ and case 2 (the best one) for $ms_{new} \geq ms_{old2}$. The result is depicted in Figure 2. RM_MMS outperforms MMS_Stratify and MMS_Cumulate both in execution time and scalability.

Table 4. Parameter settings for synthetic data

Parameter		Default
$\|DB\|$	Number of transactions	100,000
$\|t\|$	Average size of transactions	5
N	Number of items	200
R	Number of groups	30
L	Number of levels	3
F	Fanout	5

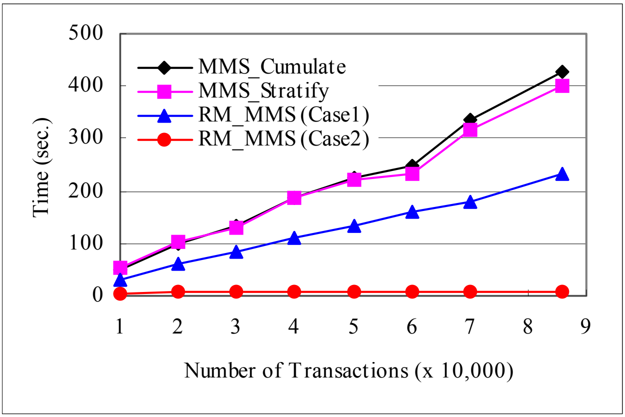

Fig. 2 Execution time for various sizes of transactions.

6 Conclusion

We have investigated in this paper the problem of remining association rules under multiple minimum supports update and presented an efficient algorithm, RM_MMS. Empirical evaluation showed that the algorithm is very effective and have good linear scale-up characteristic, compared with applying our previously proposed mining algorithms, MMS_Cumulate and MMS_Stratify, to complete the remining process under multiple minimum supports update.

References:

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 487-499.

[3] D.W. Cheung, S.D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," *Proc. DASFAA'97*, 1997, pp. 185-194.

[4] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *Proc.*

21st Int. Conf. Very Large Data Bases, pp. 420-431, Zurich, Swizerland, 1995.

[5] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," *Proc. 5th Int. Conf. Knowledge Discovery and Data Mining*, 1999, pp. 337-341.

[6] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases," *Proc. 21st Int. Conf. Very Large Data Bases*, 1995. pp. 432-444.

[7] R. Srikant and R. Agrawal, "Mining generalized association rules," *Proc. 21st Int. Conf. Very Large Data Bases*, 1995, pp. 407-419.

[8] M.C. Tseng and W.Y. Lin, "Mining generalized association rules with multiple minimum support," *Proc. Int. Conf. Data Warehousing and Knowledge Discovery*, pp. 11-20, 2001.

Appendix

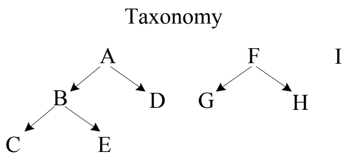

Table 5 An example for illustration of RM_MMS.

Database (*DB*)	
TID	Items Purchased
11	H, C
12	I, D
13	D, E
14	H, D, C
15	I
16	G

ms %		
Item	ms_{old}	ms_{new}
A	80	60
B	65	55
C	25	10
D	70	50
E	60	20
F	35	55
G	25	25
H	25	35
I	15	15

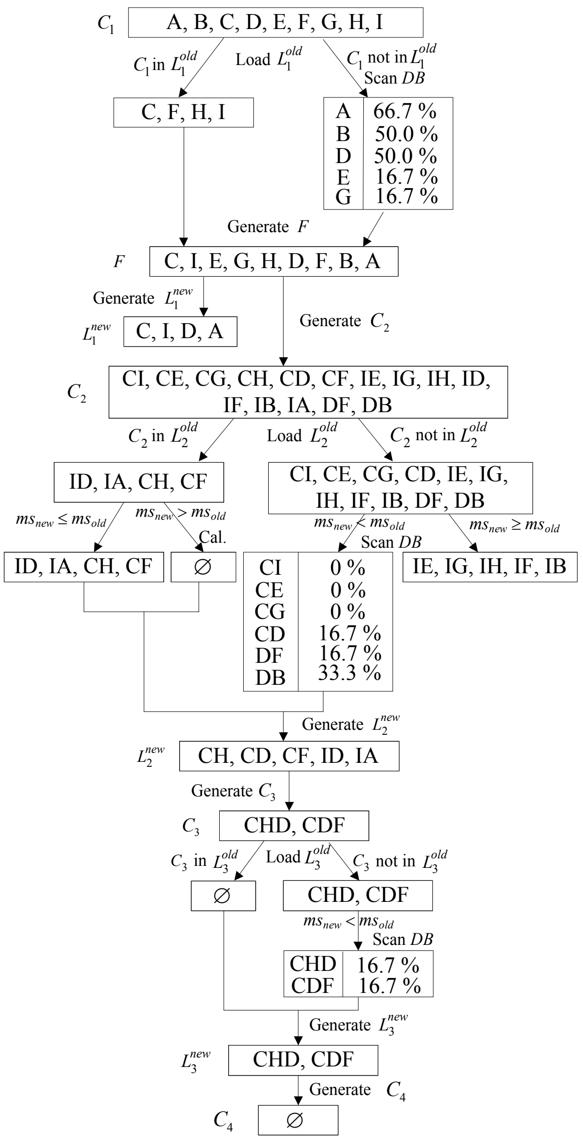

Fig. 3 Illustration of algorithm RM_MMS