Mining Frequent Itemsets in Distorted Databases with Granular Computing

Jinlong Wang
School of Computer Engineering, Qingdao Technological University
Qingdao, Shandong 266033, China
wangjinlong@gmail.com

Congfu Xu*
Institute of Artificial Intelligence, Zhejiang University
Hangzhou, Zhejiang 310027, China
xucongfu@cs.zju.edu.cn

Gang Li
School of Engineering and Information Technology, Deakin University,
221 Burwood Highway, Vic 3125, Australia
gang.li@deakin.edu.au

Data perturbation is one popular method to achieve privacy-preserving data mining. However, distorted databases bring enormous overheads to mining algorithms as compared to original databases. In this paper, we present the GrC-FIM algorithm to address the efficiency problem in mining frequent itemsets from distorted databases. Two measures are introduced to overcome the weakness in existing work: firstly, the concept of independent granule is introduced, and granule inference is used to distinguish between non-independent itemsets and independent itemsets. We further prove that the support counts of non-independent itemsets can be directly derived from sub-itemsets, so that the error-prone reconstruction process can be avoided. This could improve the efficiency of the algorithm, and bring more accurate results; secondly, through the granular-bitmap representation, the support counts can be calculated in an efficient way. The empirical results on representative synthetic and real-world databases indicate that the proposed GrC-FIM algorithm outperforms the popular EMASK algorithm in both the efficiency and the support count reconstruction accuracy.

Keywords: data mining; granular computing; frequent itemset; granule inference

1. Introduction

The rapid development of networking, data collection, storage and analysis technologies has resulted in global concerns about privacy of personal information. Data Mining, with its capability to discover valuable, and hidden knowledge from large databases, is particularly vulnerable to abuse. These concerns have recently
lead to the concept of privacy-preserving data mining (PPDM)\(^1\), which promises to discover the hidden information without violating privacy.

Data perturbation is one of the most popular approaches to privacy-preserving data mining. It focuses on individual privacy, and randomizes the values in individual records of the original data, before sending the distorted data to the data mining process\(^1,3,4\). For basket analysis or association rule discovery, the transaction records are distorted by adding some new items and discarding some existing items. From the distorted database, it is possible to discover frequent itemsets in the original database, by statistical estimation of supports in the original databases, based on the supports in the distorted databases\(^3,5\). While this privacy-preserving data mining approach is encouraging, it brings expensive overheads into the mining process. According to Agrawal et al.\(^5\), “privacy-preserving mining can take as much as two to three orders of magnitude more time as compared to direct mining.”

Our work focuses on the efficiency issue of mining the distorted data, in particular, we will address the problem of how to efficiently discover frequent itemsets from the distorted database. The work is carried out in the framework of MASK/EMASK (Efficient Mining Associations with Secrecy Konstraints)\(^3,5\), which aims to reconstruct the support count of a \(k\)-itemset from the distorted database. The process typically needs to keep track of the counts of the \(2^k\) combinations of \(k\)-itemset\(^3\). In 2004, Agrawal et al. extended their original method by using set operations to eliminate the extra overhead of counting all combinations generated from a \(k\)-itemset. While their EMASK algorithm performs well on sparse data sets, it still remains very slow for dense data sets such as census data, in which a number of long frequent itemsets exist. Moreover, its reconstruction accuracy also drops significantly for frequent itemsets longer than 5, as will be demonstrated in the experiment section.

In this paper, we present a novel algorithm GrC-FIM (Granular Computing based Frequent Itemset Mining in distorted databases) based on the granular computing (GrC) approach and the bitmap representation. Two measures are adopted in our algorithm to further improve the efficiency and the accuracy of the MASK/EMASK algorithms:

- firstly, we introduce the granule computing into the context of PPDM, and define a new concept of independent granule. The candidate itemsets can be divided into non-independent itemsets and independent itemsets, according to the features of their corresponding granule. We prove that, for non-independent itemsets, their support counts can be directly derived from sub-itemsets, rather than reconstructed using the formula proposed in MASK/EMASK. This could not only improve the efficiency of the algorithm, but also bring more accurate results;
- secondly, we adopt the bitmap representation for granules. One of the pioneers’ work in bitmap representation for association rule mining is from Lin et al.\(^28\), and in this paper, we further bring this approach into the
mining of frequent itemsets from distorted databases. With the bitmap representation, the support count can be calculated in a very efficient way with the AND and the COUNT bit operations. The empirical results indicate that the proposed GrC-FIM algorithm provides significant improvements over the EMASK algorithm, for both the dense data sets, and the sparse data sets.

The rest of the paper is organized as follows: An introduction to the basic notation is given in Section 2, followed by a recap of related work. Section 3 discusses the granular computing in the context of frequent itemsets mining, and introduces the concept of independent granule, then shows how the granule inference could be used to calculate the support of itemsets. The GrC-FIM algorithm is presented in Section 4, and the experiment design and empirical results analysis are given in Section 5. Section 6 concludes the paper with some discussions.

2. Background

Frequent itemsets mining is an essential process in market basket analysis, association discovery, sequential patterns mining, episodes, structured pattern mining, and many other important data mining tasks. Mining frequent itemsets can be formally stated as follows:

Let $At = \{a_1, a_2, \ldots, a_m\}$ be the set of all items, and $T = \langle t_1, t_2, \ldots, t_n \rangle$ be the set of transactions, where each transaction $t_i$ ($i \in [1, n]$) contains a subset of items from $At$. An itemset is a collection of zero or more items from $At$. If an itemset contains $k$ items, it is called a $k$-itemset. The support count of an itemset $C$ is the number of transactions containing $C$ in $T$. If an itemset $C$'s support count is no less than a predefined minimum threshold, it is called a frequent itemset. The frequent itemset mining is then performed to find all frequent itemsets from the transaction database $T$, with respect to a user-provided threshold.

In the context of privacy-preserving data mining under the MASK/EMASK framework, the transaction database $T$ is usually represented in a binary format where an item is treated as a binary variable with the value of 1 if the item is presented in a transaction, and with the value 0 otherwise. In 2002, Rizvi et al proposed the MASK algorithm, which uses a simple distortion process: each item value (1 or 0) in the original database $T$ either keeps the same with probability $p$ or flips with probability $1 - p$. All the transactions are distorted in this fashion, resulting in the distorted transaction database $D$ which will be supplied to the data mining process. In the MASK distortion scheme, a $k$-itemset in the original transaction database may be randomized to any of $2^k$ combinations. For example, a “11” may be distorted to “00”, “01”, “10” or “11”. In order to accurately reconstruct the support count of a 1-itemset in the original database $T$, we need to keep track of the counts of both itemsets “0” and “1” in the distorted database $D$. In principle, the MASK algorithm has to keep track of $2^k$ counts in order to calculate the support count of a $k$-itemset in the original database $T$, and this results in significant
computational cost as compared with direct mining from the original database. Although the MASK algorithm took the approach of counting optimization to reduce the costs down to \( k - 1 \) counts, its runtime efficiency remains a serious problem.

The EMASK algorithm extended the MASK method by introducing a symbol-specific distortion: ‘1’s are flipped with probability \((1 - p)\), while ‘0’s are flipped with a different probability \((1 - q)\). For a 1-itemset \( a_i \), the support count in the original database \( T \) can be estimated using equation 1.

\[
\begin{bmatrix}
c_T^1 \\
c_T^0 
\end{bmatrix} = \begin{bmatrix}
p & 1 - q \\
1 - p & q 
\end{bmatrix}^{-1} \begin{bmatrix}
c_D^1 \\
c_D^0 
\end{bmatrix}
\]

where \( c_T^1 \) and \( c_T^0 \) represent the number of ‘1’s and ‘0’s of the item \( a_i \) in the original database \( T \), while \( c_D^1 \) and \( c_D^0 \) represent the number of ‘1’s and ‘0’s of the item \( a_i \) in the distorted database \( D \). Another measure taken in the EMASK algorithm is the utilization of set operations to further reduce computational cost: Given itemsets \( A \) and \( B \), we have \(|A \cap B| = |B| - |A \cap B|\), where \(|·|\) is the set cardinality, which also determines the support count of the itemset in the database. This formula can be generalized to

\[
|A_1' A_2' \cdots A_m'B_1B_2 \cdots B_n| \\
= |B_1B_2 \cdots B_n| + \sum_{k=1}^{m} \sum_{\{x_1, \ldots, x_k\} \subseteq \{1, \ldots, m\}} (-1)^k |A_{x_1} A_{x_2} \cdots A_{x_k} B_1B_2 \cdots B_n|
\]

This generalized formula can be used to calculate the counts of all combinations from a \( k \)-itemset from the count of itself and the counts of its subsets which are available from previous passes over the distorted database. Hence, the extra overhead of counting all combinations generated by the data perturbation can be eliminated.

However, the MASK/EMASK algorithms need to maintain a full list of the support counts of frequent itemsets in the distorted database, and for any new generated candidate itemset, it will reconstruct their support counts in the original database. This process is still time consuming, especially when the itemset length is large. Moreover, it is not memory efficient since all the support counts of frequent itemsets in the distorted database have to be stored.

3. Granular Computing and Granule Inference

Granular Computing is an emerging conceptual and computing paradigm of information processing. This field has been gaining popularity in the past ten years. Simply speaking, granular computing is inspired by the ways in which humans granulate information and reason with coarse-grained information, and motivated by the urgent need for intelligent processing of empirical data that

\(^*\)here we assume a binary representation of the transaction database.
is commonly available in vast quantities, into humanly manageable abstract knowledge\textsuperscript{14}. A granule is “a clump of objects drawn together by indistinguishability, similarity, proximity or functionality”\textsuperscript{15}. In general, a larger granule may be divided into smaller granules, and smaller granules can join into a larger granule. Once a granulation structure is constructed, it is possible to establish relationships and connections between a class of objects\textsuperscript{21,22,23}.

3.1. Bitmap Representation of Granules in a Transaction Database

Information table is a popular method to construct and interpret granules\textsuperscript{24,25}: formally, an information table is a quadruple $S = (T, At, \{V_a | a \in At\}, \{I_a | a \in At\})$, where $T$ is a finite nonempty set of objects, $At$ is a finite nonempty set of attributes, $V_a$ is a nonempty set of values for $a \in At$, and $I_a : T \rightarrow V_a$ is an information function, which maps an object of $T$ to exactly one value in $V_a$.

With respect to an attribute $a \in At$, objects with the same value on $a$ cannot be differentiated based solely on their values on attribute $a$. These objects can form an elementary granule\textsuperscript{24,25}: for a value $v \in V_a$, the elementary granule corresponding to the atomic formula $(a, v)$ can be obtained: $G(a, v) = \{x \in T | I_a(x) = v\}$. This granule consists of all objects which take the same value $v$ on attribute $a$. This construction can be generalized to granules with more than one attribute\textsuperscript{24}: for a pair of attributes $a, b \in At$ and two values $v_a \in V_a$, $v_b \in V_b$, the granule corresponding to $(a, v_a) \land (b, v_b)$ can be obtained as $G((a, v_a) \land (b, v_b)) = G(a, v_a) \cap G(b, v_b)$.

Machine oriented data model is a model which uses these granules as attribute values, which are encoded with machine semantics\textsuperscript{26}. With such a model, many data mining processes can be reduced to some processes of set operations on granules. Moreover, the granule can be represented using bitmaps which offer efficient storage and extremely fast bit operations like AND and COUNT.

In the context of data mining, each item in the transaction database can be considered as an attribute in the information table, and the elementary granule is the set of transactions which contains the corresponding item. For a transaction database, the possible value domain $V_a$ of an attribute (or item) $a$ is $\{0, 1\}$, then the elementary granule for $G(a, 1)$ is the set of all transactions which contain item $a$. Considering the fact that in transaction database, the value 0 is not as important as the value 1, we use $G(\{a\})$ to simplify $G(a, 1)$ in this paper. More general granules like $G(\{a, b\})$ is the set of all transactions which contains item $a$ and $b$, and we have $G(\{a, b\}) = G(\{a\}) \cap G(\{b\})$. Moreover, the support count of an itemset can then be reduced to the cardinality of its corresponding granule.

Using the bitmap technique, the granule can be represented as a bit vector of $|T|$ dimensions\textsuperscript{b}, where each bit position corresponds to one transaction, and the value of 1 or 0 in each position represents whether or not the granule contains the

\textsuperscript{b}$|T|$ is the number of transactions in transaction database $T$
corresponding transaction. Under this representation, the cardinality of the granule is the number of 1 in the bit vector, hence the \textit{support count} of the corresponding itemset can also be calculated extremely fast using \textit{bit COUNT} operation. In this way, mining association rules can be more efficient \cite{27,28}.

As a running example, we consider the transaction database in Table 1(a): We have five transactions (also known as objects in the context of granular computing) \( T = \{T_1, T_2, T_3, T_4, T_5\} \), and five items (also known as attributes in the context of granular computing) \( \mathcal{A} = \{a, b, c, d, e\} \). The transaction database can be represented as binary format in the framework of MASK/EMASK. However, here we are talking about another level of bitmap representation: the bitmap representation of granules. In this transaction database, we have 5 different elementary granules, each corresponding to one collection of transactions. For example, \( G(\{a\}) \) is the set of all the transactions which contain the item \( a \). The bitmap representation of granules is given in Table 1(b). It is clear to see that each granule can be represented in both the list and bitmap form, and it is also easy to calculate that \( G(\{a,b\}) = G(\{a\}) \cap G(\{b\}) = \{T_1\} = 10000 \). The intersection and cardinality operations of bitmap representation of elementary granules offers high efficiency in the calculation of \textit{support count} of the corresponding itemsets.

3.2. Granule Inference

A granule represents a transaction set such that each transaction in the set contains the corresponding itemset. Under this interpretation, mining frequent itemsets may be regarded as finding the granules of which the cardinality is higher than a user-determined threshold. Here, we show how the granule inference can be used to

<table>
<thead>
<tr>
<th>Granules</th>
<th>List representation</th>
<th>Bitmap representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G({a}) )</td>
<td>( {T_1, T_3, T_4} )</td>
<td>10110</td>
</tr>
<tr>
<td>( G({b}) )</td>
<td>( {T_1, T_5} )</td>
<td>10001</td>
</tr>
<tr>
<td>( G({c}) )</td>
<td>( {T_2, T_5} )</td>
<td>01001</td>
</tr>
<tr>
<td>( G({d}) )</td>
<td>( {T_2, T_3} )</td>
<td>01100</td>
</tr>
<tr>
<td>( G({e}) )</td>
<td>( {T_2, T_5} )</td>
<td>01001</td>
</tr>
</tbody>
</table>
support this process.

Let \( B \subseteq A^t \) denote an itemset, and let \( a \in B \) be one item in \( B \), and \( G(B) \) is the granule corresponding to the itemset \( B \). Using the concept of attribute independence in rough set theory, we can give the following definitions for granule inference:

**Definition 1.** The item \( a \) is dispensable in itemset \( B \) if \( G(B) = G(B - \{a\}) \), otherwise the item \( a \) is indispensable in itemset \( B \).

**Definition 2.** The itemset \( B \) is independent if all its items are indispensable.

**Definition 3.** For itemset \( B \),

1. \( G(B) \) is an independent granule iff \( \forall a \in B, G(B) \neq G(B - \{a\}) \).
2. \( G(B) \) is a dependent granule iff \( \exists a \in B, G(B) = G(B - \{a\}) \).

For example, in the running example shown in Table 1(b), the granule \( G(\{c, e\}) \) is a dependent granule, since \( \exists e \in \{c, e\} \) and \( G(\{c, e\}) = G(\{c, e\} - \{e\}) = G(\{c\}) \).

**Definition 4.** For an itemset \( B \subseteq A^t \), if its granule \( G(B) \) is an independent granule, we call \( B \) an independent itemset; otherwise, it is called a non-independent itemset.

**Lemma 1.** For any two itemsets \( B \) and \( C \), if \( B \subseteq C \), then \( G(C) \subseteq G(B) \).

**Proof.** \( G(C) = G(B \cup (C - B)) = G(B) \cap G(C - B) \subseteq G(B) \).

**Theorem 1.** For any two itemsets \( B \) and \( C \), when \( B \subseteq C \), if an item \( a \in B \) is dispensable in \( B \), then \( a \) is dispensable in \( C \), i.e., \( G(C - \{a\}) = G(C) \).

**Proof.** Since \( C - \{a\} \subseteq C \), from Lemma 1, we have
\[
G(C) \subseteq G(C - \{a\})
\] (2)
On the other hand, since \( a \) is dispensable in \( B \), by Definition 1 we have
\[
G(B - \{a\}) = G(B)
\] (3)
Since \( \{a\} \subseteq B \), by Lemma 1 we have
\[
G(B) \subseteq G(a)
\] (4)
then from eq. 3 and eq. 4, we also have
\[
G(B - \{a\}) \subseteq G(a)
\] (5)
since \( G(C) = G(C - \{a\} \cup \{a\}) = G(C - \{a\}) \cap G(\{a\}) \), from eq. 5 we can have
\[
G(C) \supseteq G(C - \{a\}) \cap G(B - \{a\})
\] (6)

\(^c\)From now on, we will use attribute and item, attribute set and itemset intermediately in this paper.
Considering the fact that \((B - \{a\}) \subseteq (C - \{a\})\), by Lemma 1, we have
\[
G(C - \{a\}) \subseteq G(B - \{a\})
\] (7)
then from eq. 6 and eq. 7, we have
\[
G(C) \supseteq G(C - \{a\})
\] (8)
Finally from eq. 2 and eq. 8, we prove that \(G(C - \{a\}) = G(C)\).

As an example, in the Table 1(b), the item e is dispensable in itemset \(\{c, e\}\), and since \(\{c, e\} \subseteq \{c, d, e\}\), from Theorem 1, the item e is also dispensable in \(\{c, d, e\}\).

From Theorem 1 and Definition 3, we can obtain the following corollary.

**Corollary 1.** Let \(B \subseteq C\): if \(G(B)\) is a dependent granule, then \(G(C)\) is a dependent granule.

**Proof.** From Definition 3, if \(G(B)\) is a dependent granule, then
\[
\exists a \in B, G(B) = G(B - \{a\})
\] (9)
which shows that \(a\) is dispensable in \(B\). According to Theorem 1, \(a\) is dispensable in \(C\), so \(G(C)\) is a dependent granule.

**Corollary 2.** Let \(B \subseteq C\): if \(G(C)\) is an independent granule, then \(G(B)\) is an independent granule.

**Proof.** This theorem can be proven by contradiction with Corollary 1.

The Corollary 1 and Corollary 2 show that the independent granule satisfies the anti-monotone property, which is important for us to quickly identify dependent granules.

### 3.3. Support counts computation based on granule inference

As mentioned in subsection 3.1, the support count of an itemset can be calculated through the cardinality of the corresponding granule: for an itemset \(B\), we have \(\text{sup}(B) = |G(B)|\), where the \(\text{sup}(B)\) is the support count of the itemset \(B\).

**Theorem 2.** For \(A, B \subseteq At\), if \(G(A) = G(B)\), then \(|G(A)| = |G(B)|\).

**Proof.** Refer to the definition and properties of granule.

**Corollary 3.** If \(G(C)\) is a dependent granule, then \(|G(C)| = \min\{|G(B)|\mid B \subseteq C \text{ and } |B| = |C| - 1\}\), where \(\min\{\cdot\}\) denotes the least one among the element number in \(|\cdot|\).
Proof. Since the support count of an itemset is a monotonous decreasing function, the “≤” relation is obvious,

\[ |G(C)| \leq \min \{ |G(B)||B \subseteq C \subseteq \text{At and } |B| = |C| - 1 \} \]  

(10)

For the “≥” part, if \( G(C) \) is a dependent granule, then by Definition 3

\[ \exists a \in C, G(C) = G(C - \{a\}) \]  

(11)

So,

\[ |G(C)| = |G(C - \{a\})| \geq \min \{ |G(B)||B \subseteq C \subseteq \text{At and } |B| = |C| - 1 \} \]  

(12)

The corollary can be proven. □

The concept of key itemset was used in Ref. 30, similar as our independent itemset, however in this paper we use the granule inference in GrC to introduce and interpret the concept, and this is different from the work in Ref. 30.

Using the definitions and properties in this section, we have the following equivalence notions: itemset \( B \) is independent ↔ Granule \( G(B) \) is an independent granule ↔ itemset \( B \) is an independent itemset. With the candidate set generation-and-test approach ⁷, we can differentiate between non-independent candidate itemsets and independent candidate itemsets, so the properties of granule in this section can be utilized in frequent itemset mining, as we will see in the next section.

4. The GrC-FIM Algorithm

In this section, we present the GrC-FIM algorithm. First, the formal problem of mining frequent itemsets from distorted databases is defined, followed by a description of the application of granule inference to improve the efficiency of the mining process. Then, the pseudo-code of the algorithm is provided with explanations.

4.1. Mining Frequent Itemsets from Distorted Databases

In the context of privacy-preserving data mining, data perturbation is one popular method. Under the framework of MASK/EMASK ³, ⁵, the original transaction database \( T \) is randomized by a symbolic-specific distortion process, in which ‘1’ and ‘0’ in the original database are respectively flipped with \((1 - p)\) and \((1 - q)\). Then, the distorted database \( D \) is supplied to the data mining algorithm, with an aim to protect the privacy information in individual transactions.

For any itemset \( C \), let \( \text{sup}^D(C) \) be the support count of \( C \) in the distorted database \( D \), and let \( \text{sup}^T(C) \) be the support count of \( C \) in the original database \( T \). The aim of the frequent itemsets mining algorithm is then: Given the distortion parameters \( p \) and \( q \), and the distorted database \( D \), to reconstruct the support count of itemsets and to identify all frequent itemsets in the original database \( T \).

The MASK/EMASK algorithms ³, ⁵ achieve this goal using an Apriori style process: generate the candidate \( k \)-itemsets by combining two frequent \((k - 1)\)-itemsets that have the first \( k - 2 \) items which are the same, and the last item
different; then for each candidate $k$-itemset $C_i^k \in C_k$, it reconstructs the support count $sup^T(C_i^k)$, if it is above the user-specified threshold, the itemset is a frequent $k$-itemset; otherwise, it is removed. As the process of reconstructing the support count of itemsets in the original database needs to scan the distorted database followed by an extensive set operations, the process is still very time consuming, especially when $k$ is bigger.

It is possible to improve the efficiency of this mining task by two measures:

- firstly, for a candidate itemset $C_i^k$, if its corresponding granule $G_T(C_i^k)$ in the original database $T$ is a dependent granule, by the Corollary 3, its support count $sup^T(C_i^k)$ can be inferred. Hence, the reconstruction process can be avoided;
- secondly, the bitmap representation for granules, both the identification of dependent granules and the cardinality of a granule, can be done in an extremely efficient way by the AND and the COUNT bit operations. Moreover, the bitmap representation enables us to horizontally partition the database into disjointed subsets, so that for a huge database, the memory restriction could be overcome.

The critical step in this method is to determine whether a granule is a dependent granule in the original database $T$: by Corollary 1, we know that if a granule $G_T(C_i^k)$ is a dependent granule in $T$, for any $(k+1)$-itemset $C_{k+1}^i$ containing $C_i^k$, its granule $G_T(C_{k+1}^i)$ is also a dependent granule in $T$. Hence, the support count $sup^T(C_{k+1}^i)$ can be directly derived by Corollary 3.

On the other hand, when the $(k+1)$-itemset $C_{k+1}^i$ was merged from two frequent $k$-itemsets $L_1^k$ and $L_2^k$ ($L_k$ represents the frequent $k$-itemsets), and both of them correspond to independent granules, we can determine whether $G_T(C_{k+1}^i)$ is an independent granule or not using the following three steps:

- Firstly, we assume that $G_T(C_{k+1}^i)$ is an independent granule in $T$, and predict its support count $sup^T(C_{k+1}^i)$ in $T$, as the minimum of $sup^T(L_k^1)$ and $sup^T(L_k^2)$.
- Secondly, we calculate the support count $sup^D(C_{k+1}^i)$ in the distorted database $D$ using the bit AND and bit COUNT operations from the bitmap representation of the elementary granules on the distorted database. Then, the support count $sup^T(C_{k+1}^i)$ of $C_{k+1}^i$ in the original database $T$ can be reconstructed using the formula in the EMASK algorithm.
- Finally, if $sup^T(C_{k+1}^i)$ equals to the reconstructed $sup^T(C_{k+1}^i)$, by Corollary 3, we can determine that $G_T(C_{k+1}^i)$ is a dependent granule; otherwise, it is an independent granule.

Using above mentioned measures, it is clear that the algorithm will only maintain the support count of independent frequent itemsets in the distorted database for the reconstruction process, and all the computation about the non-independent frequent itemsets can be derived using granule inference. Hence the computation and memory costs are reduced.
4.2. **The Process of the GrC-FIM Algorithm**

Based on above ideas, we propose the GrC-FIM (Granular Computing based Frequent Itemset Mining in distorted databases) algorithm, and the pseudo-code of it is given in Algorithm 1.
**Algorithm 1: GrC-FIM Algorithm**

**Data:** The distorted database $D$, distortion parameters $p$ and $q$, minimum support $s$.

**Result:** Frequent itemsets $L$ in the original database $T$.

**begin**

**Step 1.** Compute $L_1$ and transform $D$;

for each $a \in At$ do

$\text{sup}^D(a) = 0$;

$\text{sup}^d_i(a) = 0$ ($i \in \{1, 2, \ldots, w\}$);

for $i = 1; i \leq w; i++$ do

$\text{TrGrBit}(d_i)$;

for each $a \in At$ do

$\text{sup}^D(a) += \text{sup}^d_i(a)$;

Save GrBit($d_i$) into hard disk;

for $\forall a \in At$ do

Reconstruct the support count $\text{sup}^T(a)$ with $p, q$;

Generate $L_1$;

**Step 2.** Determine the independent itemsets in $L_1$.

for $\forall L_1^i \in L_1$ do

if $L_1^i.\text{sup} == 100\%$ then

$L_1^i.\text{independent} = \text{false}$

else

$L_1^i.\text{independent} = \text{true}$;

Save the distorted support count $L_1^i.\text{disup} (\text{sup}^D(L_1^i))$;

**Step 3.** Compute frequent itemsets $L_k (k \geq 2)$;

for $k = 2; L_{k-1} \neq \emptyset; k++$ do

**Step 3.1** Generate candidate sets $C_k$

$C_k = \text{gen.candidate}(L_{k-1})$;

**Step 3.2** Compute the support of candidate frequent itemsets

for $\forall C_k^i \in C_k$ do

if $C_k^i.\text{independent} == \text{false}$ then

$\text{sup}^T(C_k^i) = \text{sup}^T(C_k)$

if $C_k^i.\text{independent} == \text{true}$ then

Reconstruct the support $\text{sup}^T(C_k^i)$ of itemset $C_k^i$;

if $\text{sup}^T(C_k^i) == \text{sup}^T(C_k^i)$ then

$C_k^i.\text{independent} = \text{false}$

Generate $L_k$;

Save the distorted support counts of the independent itemsets in $L_k$;

**end**
The algorithm starts with a partitioning of the distorted database into the bitmap representation of elementary granules through the function $T_{GrBit}()$. For huge databases like those used in our experiment, each elementary granule $GrBit(d_i^*)$ can be saved into a hard disk. Suppose the distorted database $D$ is horizontally partitioned into $w$ partitions $D = d_1 \cup d_2 \cup \cdots \cup d_w$, where $w$ is determined with the constraints of memory amount. The support count $sup^D(C)$ of an itemset $C$ in the distorted database $D$ can be calculated as $sup^D(C) = \sum_{i=1}^{w} sup^{d_i}(C)$. Then, with the support reconstruction process, the frequent 1-itemsets can be obtained.

In the second step, the algorithm estimates an initial set of dependent elementary granules in the original database $T$. If an item appears in every transaction in the original database $T$, it is labeled as a dependent granule, so here for most databases, we start with an empty set of dependent granules. For independent granules, its support count $sup^D(L_i)$ in the distorted database $D$ needs to be saved, for the reconstruction of support counts in $T$ in the following steps.

Step 3 is iterative: It firstly generates candidate $k$-itemsets from frequent $(k-1)$-itemsets. For each candidate $k$-itemset $C_k^i$, if any of its merging itemsets is non-independent, the granule of $C_k^i$ is then dependent, and its support count can be determined using Corollary 3; otherwise, the support count $sup^T(C_k^i)$ will be predicted from the support counts of its merging itemsets, and then using the EMASK formula to reconstruct the support count $sup^T(C_k^i)$. Then, if the reconstructed support count equals to the predicted one, we label the $k$-itemset $C_k^i$ as non-independent. The process iterates until no candidate itemsets can be generated.

5. Performance Evaluation

The GrC-FIM algorithm adopts two measures to improve the performance of frequent itemset mining from a distorted database. In order to evaluate the effect of these two measures, we implement a slim-down version of the GrC-FIM algorithm, called the GB-FIM, which implements the EMASK algorithm with bitmap representation, but no granule inference was used; while the GrC-FIM algorithm can be considered as the EMASK algorithm with bitmap representation, and with granule inference.

In this section, these two algorithms are compared with the EMASK algorithm implemented with the source code provided by the authors. All algorithms were implemented in C++ and complied under Cygwin using gcc2.9.5 with the −O option. The comparison experiment was conducted on a 3GHz Intel Pentium PC with 512MB memory and Windows XP Service Pack 2 installed.

Four massive transaction databases were used in our experiment, and their characteristics are shown in Table 2. Among them, $T25.I4.D1M.N1K$ ($T25$) and $T40.I10.D1M.N1K$ ($T40$) were synthetic database generated from the IBM synthetic market-basket data generator, which mimics the transactions in a retailing environment. The database $pumsb^*$ is a real-world census data. It contains very
dense transaction records, and produces many long frequent itemsets even with very high support thresholds. The database *accidents* is another real-world data, which contains the traffic accident records in Flanders (Belgium) collected by NIS (National Institute of Statistics) from 1991 to 2000. These two real-world databases are different from the synthetic ones in two aspects: firstly, they are more dense while the synthetic ones are generally sparse; secondly, these two real-world databases contain a much larger number of long frequent itemsets than the synthetic ones, even though the average itemsets lengths are similar to both.

Table 2. Transaction database characteristics

<table>
<thead>
<tr>
<th>Database</th>
<th>#Transactions</th>
<th>Items</th>
<th>Max. Length</th>
<th>Avg. Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>T25.I4.D1M.N1K</td>
<td>1000000</td>
<td>972</td>
<td>53</td>
<td>24.9</td>
</tr>
<tr>
<td>T40.I10.D1M.N1K</td>
<td>1000000</td>
<td>991</td>
<td>81</td>
<td>39.6</td>
</tr>
<tr>
<td>pumsb*</td>
<td>49046</td>
<td>7117</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>accidents</td>
<td>340183</td>
<td>468</td>
<td>51</td>
<td>33.8</td>
</tr>
</tbody>
</table>

In our experiments, the data sets were distorted with parameters $p = 0.5$ and $q = 0.97$ as in EMASK 5. For a fair comparison on algorithms and scalability, the GrC-FIM and the GB-FIM algorithms were run with only $5K$ transactions loaded into the main memory at each time.

5.1. Efficiency analysis

The execution time of three compared algorithms on different data sets is shown in Fig. 1 and Fig. 2. In each figure, the $x$-axis is the minimum support threshold, and the $y$-axis is the CPU time (in sec). As shown in the figures, both the GrC-FIM and the GB-FIM outperform the EMASK, and the margin grows as the *minimum support threshold* decreases.

For synthetic databases ($T25$ and $T40$), the execution time of both GrC-FIM and GB-FIM increase slowly as the *minimum support threshold* decreases. Out of the GrC-FIM and the GB-FIM algorithm, the GrC-FIM slightly outperforms the GB-FIM. This comes from the fact that few *dependent* granules exist in these two databases, accordingly limited efficiency gain can be got from granule inference.

For the database $pumsb^*$, as the *minimum support threshold* decreases, the time required by all the algorithms increases. However, for low thresholds, the improvement of efficiency in the algorithms GrC-FIM and GB-FIM is evident, and the GrC-FIM algorithm outperforms the GB-FIM.

For the database *Accidents*, both the algorithm of GrC-FIM and GB-FIM outperform EMASK, and the performance of GrC-FIM and GB-FIM algorithm is

---

It is important to note that GrC-FIM can load more transactions into the main memory, and this will result in a further improvement in the efficiency.
From these results, we can make conclusions that:

1. the measure of bitmap representation of granules dramatically improves the efficiency of the algorithms;
2. the measure of granule inference can also further improve the efficiency of frequent itemset mining algorithms, and this improvement is especially evident on databases in which many long frequent itemsets exist, and the granule inference can be applied to avoid the support count reconstruction process.

5.2. Accuracy analysis

Since the difference between the GB-FIM algorithm and the EMASK algorithm is at the bitmap representation, and all other operations are the same, these two algorithms achieve the same results. Hence, in this section we compare the GrC-
FIM algorithm with the EMASK algorithm on the accuracy of the mining results. Following the evaluation method proposed in MASK/EMASK, the evaluation is based on the support error and the identify error.

5.2.1. Comparison on Support Error

The support error reflects the average differences between the reconstructed support count and the original support count over correctly identified frequent itemsets, and it is defined as:

$$
\rho = \frac{1}{|F|} \sum_{f} \frac{|rec_{sup_f} - act_{sup_f}|}{act_{sup_f}} \times 100.
$$  \hspace{1cm} (13)

In our experiment, we calculate the ratio of the support error from our GrC-FIM algorithm to the support error from the EMASK algorithm: If the ratio is less than 1, the GrC-FIM algorithm achieves a smaller support error; on the other hand, if the ratio is greater than 1, the EMASK achieves better mining results.

Fig. 3(a) and Fig.3(b) show the ratio calculated on the synthetic data sets T25 and T40. In these figures, the x-axis represents the k, which is the length of the frequent itemset, while the y-axis represents the ratio. Since the data is sparse, there are less patterns, and support count can be inferred.

![Fig. 3. comparison of support error on real data sets](image_url)

Fig. 4(a) shows the ratio calculated on the database *pumsb*, and Fig. 4(b) shows the ratio calculated on the database *accident*. Table 3 gives one example of the support errors comparison between two algorithms on the database *pumsb*, with a minimum support threshold $s = 40\%$. $|F|$ indicates the number of frequent itemsets at each level ($k$).

It is evident that for short frequent itemsets (like $k < 5$), both algorithms have similar support errors under different threshold values; while for longer itemsets
with $k > 5$, the GrC-FIM algorithm dramatically improves the accuracy of the reconstructed support count. This improvement in accuracy comes from the direct inference of the support counts of non-independent itemsets, rather than the error-prone reconstruction step.

### 5.2.2. Comparison on Identify Error

The *identity error* reflects the percentage error in identifying frequent itemsets. It includes the percentage of false positive ($\sigma^+$) and the percentage of false negatives ($\sigma^-$), where, $R$ denotes the reconstructed frequent itemsets, and $F$ denotes the correct frequent itemsets.

$$\sigma^+ = \frac{|R - F|}{|F|} \times 100$$  \hspace{1cm} (14)

$$\sigma^- = \frac{|F - R|}{|F|} \times 100$$  \hspace{1cm} (15)
On synthetic data sets (T25 and T40), the results of identify error from both algorithms are the same as shown in Fig. 5, where FP means false positive (σ+) and FN means false negative (σ−). The x-axis represents the k, which is the length of frequent itemset, while the y-axis represents the identify error of each algorithm.

![Graph comparison of identify error on synthetic data sets](image)

(a) T25(s=0.25%)

(b) T40(s=0.75%)

Fig. 5. comparison of identify error on synthetic data sets

Fig. 6 shows the identify error comparison on real data sets. Table 4 and Table 5 give examples of identify errors respectively on two real data sets.

<table>
<thead>
<tr>
<th>k</th>
<th>σ+(FP_EMASK)</th>
<th>σ−(FN_EMASK)</th>
<th>σ+(FP_GrC-FIM)</th>
<th>σ−(FN_GrC-FIM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.17391</td>
<td>0</td>
<td>2.17391</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3.19149</td>
<td>0.35461</td>
<td>3.19149</td>
<td>0.35461</td>
</tr>
<tr>
<td>3</td>
<td>3.49076</td>
<td>1.43737</td>
<td>3.49076</td>
<td>1.43737</td>
</tr>
<tr>
<td>4</td>
<td>2.69576</td>
<td>5.00642</td>
<td>2.69576</td>
<td>5.00642</td>
</tr>
<tr>
<td>5</td>
<td>2.3728</td>
<td>16.0959</td>
<td>2.3728</td>
<td>15.9247</td>
</tr>
<tr>
<td>6</td>
<td>1.88369</td>
<td>41.8435</td>
<td>1.95684</td>
<td>41.1851</td>
</tr>
<tr>
<td>7</td>
<td>1.06195</td>
<td>74.5487</td>
<td>1.11504</td>
<td>73.7699</td>
</tr>
<tr>
<td>8</td>
<td>0.247135</td>
<td>94.7203</td>
<td>0.247135</td>
<td>94.3608</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>99.7313</td>
<td>0</td>
<td>99.6929</td>
</tr>
</tbody>
</table>

It is evident that the result of the GrC-FIM algorithm is similar with EMASK. The difference is that: the false positive of the Grc-FIM is worse than that of EMASK; and the false negative is better than that of EMASK. The difference is negligible.

From Tables 4 and 5, it is also important to be aware that even though the results from subsection 5.2.1 show that the Grc-FIM algorithm can bring more
accurate support counts estimation, none of the compared algorithms can achieve satisfactory identify accuracy when $k$ is larger than 6. This actually indicates that the MASK/EMASK reconstruction process is imperfect, especially for those longer itemsets. How to further improve the accuracy of support counts reconstruction for independent itemsets is essential to the practical application of state-of-the-art PPDM algorithms.

6. Conclusions and future works

Mining frequent itemsets from distorted databases is time-consuming as compared to mining from original databases. In this paper, we have presented an effective algorithm to address the problem. The keys to the improved performance come from two aspects:

- firstly, by virtue of granule inference, the support counts of candidate non-independent itemsets can be directly derived from frequent independent itemsets obtained in previous mining step. This could improve the accuracy of the
support count because the reconstruction step can be avoided, i.e., the error in reconstructing the support count of a non-independent itemset only comes from the errors of its sub-itemsets, renewing reconstruction error is avoided. This improvement is more evident for long frequent itemsets in dense data sets. Moreover, it can also lead to fewer memory usage since only the support counts of those independent frequent itemsets need to be kept.

- secondly, with the bitmap representation of granules, all granular computing and inference can be done in an extremely fast way. This brings significant efficiency into the mining algorithms.

As a summary, our proposed GrC-FIM algorithm improves the popular PPDM algorithm EMASK in both the efficiency and the support count reconstruction accuracy. However, there is still a long way to go before PPDM’s performance can be at par with the traditional data mining performance, as shown in subsection 5.2.2. This work also indicates that the reconstruction step in the MASK/EMASK algorithms is not as reliable as expected, therefore, one of the important future works will be on how to further refine the support count reconstruction process so that support count calculation of independent itemsets can be more accurate.

Acknowledgements

We thank the anonymous reviewers for their detailed and extremely helpful comments. The preliminary work of this paper was published in Proc. of 19th ACS Australian Joint Conference on Artificial Intelligence, and the authors would like to thank Prof. Yunhe Pan, Mr. Hongwei Dan for their helpful discussion in the preliminary work of this research. The first author is grateful to Prof. T.Y. Lin.
for a fruitful discussion during his visiting in ZheJiang University on March the 29th, 2005. We would also like to thank Mr. Sam Schmidt for proofreading the manuscript, and thank Mr. Ren’ai Zhang for assistance in the experiment.

This work was partially supported by Chinese National High Technology Research and Development (863, No.2007.AA01Z197), the National Natural Science Foundation of P.R.China (No.60802066), Science and Technology Planning Project (No. 07-2-3-1-jch) of Qingdao China, and Deakin CRGS Grant 2008.

References


