Vol. 35, No. 8 ACTA AUTOMATICA SINICA August, 2009

A Granular Computing Approach to Knowledge Discovery in Relational Databases

QIU Tao-Rong[1, 2] LIU Qing[2] HUANG Hou-Kuan[1]

Abstract The main objective of this paper is to present granular computing algorithms of finding association rules with different levels of granularity from relational databases or information tables. Firstly, based on the partition model of granular computing, a framework for knowledge discovery in relational databases was introduced. Secondly, referring to granular computing, the algorithms for generating frequent k-itemsets were proposed. Finally, the proposed algorithms were illustrated with a real example and tested on two data sets under different supports. Experiment results show that the algorithms are effective and feasible. Moreover, the meanings of mining association rules based on granular computing are clearly understandable.

Key words Granular computing, information granule, knowledge discovery, association rule

Knowledge discovery in databases has become a research hot spot nowadays. The problem of mining association rules over basket data was introduced in [1]. Based on the concept of frequent itemsets, several mining algorithms to find association rules in transaction database, such as Apriori, AprioriTid, and so on, were proposed by Agrawal and his coworkers[2].

There have been many proposed algorithms that are variants of Apriori. These algorithms mainly differ in how they check "candidate" itemsets against the database. Apriori checks itemsets of length l for frequency during database pass l. AprioriTid algorithm does not use the database to count support. Instead, it uses a special encoding method for candidates from the previous pass and has better performance in later pass when the size of candidate itemsets becomes smaller compared to the size of the database. Apriori and AprioriTid can be combined into a hybrid algorithm, called AprioriHybrid that scales linearly with the number of transactions[2]. Partitioning algorithm[3] proposed by Savasere in 1995 might improve the performance of finding frequent itemsets. It identifies all frequent itemsets in memory-sized partitions of the database and then checks these against the entire database during the final pass. Sampling algorithm[4] was attributed to Toivonen in 1996. It can reduce the number of scanning databases to one in the best case and two in the worst case. In recent years, some effective new methods used for organizing, storing, analyzing, and processing data have been presented[5−12].

In general, association rules mined from relational databases are generalized. However, if some basic algorithms like Apriori, Apriori-like, and FP-tree are used to find association rules with different levels of granularity from relational databases, we have to extend relational databases. Thus, although we obtain available multilevel association rules, some disadvantages like redundant rules are generated.

Granular computing (GrC)[13−14] is a new concept and computational model, and may be regarded as a label of family of theories, methodologies, and techniques that make use of granules. A granule is a basic ingredient of GrC. It may be a subset, class, object, a group of concepts, or cluster of a universe[13−20].

Currently, many models and methods of GrC have been proposed and studied. Zadeh proposed a general framework of GrC based on fuzzy set theory[20]. In 1982, Pawlak put forward the theory of rough sets[14, 19, 21], namely a concrete example of GrC. Hobbs set up a theory of granularity[22], which is similar to the theory of rough sets in terms of formulation. Zhang developed a quotient space theory of problem solving based on hierarchical description and representation of a problem[23]. Lin proposed to use neighborhood systems for the formulation of GrC[14, 24−25]. Yao proposed a unified framework of GrC[14−18]. The new framework extends results obtained in the set-theoretic setting and extracts high-level common principles from a wide range of scientific disciplines; and many other researchers also proposed some available models and methods of GrC[14, 26−27].

Nowadays, the principles of GrC have been applied to many fields, such as medicine, economics, finance, business, environment, electrical engineering, and computer engineering. In the domain of data mining, GrC provides a conceptual framework for studying many issues. Yao applied the GrC model to the study of the consistent classification problems with respect to partitions of a universe[18]. In [28], he put forward an idea that one of the fundamental tasks of data mining is to search for the right level of granularity in data and knowledge representation. Lin presented a fast association rule algorithm (Bit-AssoRule) based on GrC[29−30]. But in his work, generating different levels of association rules were not considered. Furthermore, how to store bit maps was not very clear. Pedrycz and Park discussed development of the structure of the granular classifier[31]. Chen presented a novel model called the information granulation based data mining approach, which imitated the human ability to process information and acquired knowledge from information granules rather than from numerical data[32]. To get a true hybrid framework for taking operational decisions from data, [33] extended the algorithmic inference approach to the GrC paradigm.

In this paper, by applying the GrC technique, we define a granule as a set of entities that have the same properties in relational databases. So, a granule could be considered as an equivalent class of attribute values. By discussing the representation, storage, and operation of granules, we propose an approach (named G-Approach) to mining association rules from relational databases based on GrC and the taxonomy trees of attributes. G-Approach views granules as its basic processing elements to find association rules, and the meaning of granules is clear. Compared with Apriori, G-Approach can effectively reduce the number of can-

Received March 6, 2008; in revised form December 26, 2008
Supported by National Natural Science Foundation of China (60173054) and the Key Project of Jiangxi Province in China (20061B01002)
1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, P. R. China 2. Department of Computer, Nanchang University, Nanchang 330031, P. R. China
DOI: 10.3724/SP.J.1004.2009.01071

didate itemsets, save computing time, and find association rules of different levels by using granules with different levels of granularity according to a real example and experiment results.

The rest of this paper is organized as follows. A data model and related definitions are described in Section 1. Algorithms of generating frequent itemsets based on GrC are proposed in Section 2. A example in practice is illustrated in Section 3, and the performances of G-Approach and other algorithms are compared experimentally in Section 4, followed by some conclusions and avenues for future work in Section 5.

1 Data model and related definitions

For a given problem, if the solution is adopted by using GrC, then the following basic questions should be answered first: 1) How to define relevant information granules for a given problem? 2) How to define relevant operations on granules for a given problem? 3) How to construct from given granules the information granules satisfying given soft constrains[30]?

In this section, the first two questions will be discussed. Suppose in a relational database, the sum of data records is N, and that of attributes is M.

Definition 1 (Information table). Let 4-tuple $S = \langle U, A, V, f\rangle$ be an information table or a relational database table, where $U = \{u_1, \cdots, u_N\}$ is a non-empty finite set, and each element in U is called an individual, $A = \{a_1, \cdots, a_M\}$ is a finite attribute set, $V = \{V_{a_1}, \cdots, V_{a_M}\}$ is a set of attribute values, where $V_{a_i} = \{V_{a_i,1}, \cdots, V_{a_i,k}\}$ is the domain of attribute a_i, $V_{a_i,j}$ is a categorical value and f is a mapping, $f(u, a_i) : U \times A \to V_{a_i}$, such that $f(u, a_i) \in V_{a_i}$, for all $u \in U$ and $a_i \in A$.

Definition 2 (Concept hierarchy). Let $V_{a_i} = \{V_{a_i,1}, \cdots, V_{a_i,k}\}$ be the domain of attribute a_i, and each $V_{a_i,j}$ may be viewed as a concept. An attribute value hierarchy or concept hierarchy of attribute a_i is defined as a rooted tree T_{a_i} such that V_{a_i} is a set of the leaves of T_{a_i}.

The rooted tree T_{a_i}can be established by applying the generalization process to these values. In general, there are two ways of constructing the concept hierarchy: human predefines it and machine learns it.

The concept hierarchy is a rooted tree with the height Q or Q levels ($1 \le Q \le\parallel V_{a_i} \parallel$), where $\parallel V_{a_i} \parallel$ denotes the cardinality of V_{a_i}. The level of a vertex v in the rooted tree is the length of the path from the root to v. Thus, the level of the root is 0. The attribute values in different levels support a partial order. If X and Y are two different levels in the given concept hierarchy and level X is lower than level Y, then we say that the concept at level X is more abstract than that at level Y.

According to Definitions 1 and 2, the domain of attribute a_i at level p in its concept hierarchy, denoted by $V^p_{a_i}$, is a set of both internal vertices where their levels are equal to p and some leaves where their levels are not greater than p.

Definition 3 (Information granule). An information granule is defined as the tuple $IG = (\varphi, m(\varphi))$, where φ refers to the intension of information granule IG, and $m(\varphi)$ represents the extension of information granule IG.

Let $S = \langle U, A, V, f\rangle$ be an information table or a relational database table. Let $B = \{a_1, a_2, \cdots, a_k\} \subseteq A$ be a subset of attributes and $\varphi = \{I_1, I_2, \cdots, I_k\}$ such that $I_i \in V_{a_i}$ be a set of attribute values corresponding to B. Then, the intension of an information granule can be defined as: $\varphi = \{I_1, I_2, \cdots, I_k\}$, and the extension can be defined as $m(\varphi) = \{u|f(u, a_1) = I_1 \wedge f(u, a_2) = I_2 \wedge \cdots \wedge f(u, a_k) = I_k,\ u \in U,\ a_i \in B,\ i = 1, 2, \cdots, k\}$. Here, $m(\varphi)$ describes the internal structure of the information granule. The collection of the extensions of all granules is denoted GK. The map $U \to GK$ is called the granulation of the information table or the relational database table.

Definition 4 (Size of information granule). Let $IG = (\varphi, m(\varphi))$ be an information granule, and its size can be defined as the cardinality of the extension of the information granule, namely, $\parallel m(\varphi) \parallel$. Intuitively, the size may be interpreted as the degree of abstraction or concreteness.

Definition 5 (Elementary granule). Let $IG = (\varphi, m(\varphi))$ be an information granule. If $\varphi = \{V_{a_i,j}\}$, then IG is called an elementary information granule of attribute a_i, or an elementary granule for short, where $V_{a_i,j}$ is the j-th attribute value of attribute a_i. Namely, $m(\varphi) = \{u|f(u, a_i) = V_{a_i,j},\ u \in U,\ a_i \in A\}$.

Definition 6 (Elementary granule at level *p*). Let T_{a_i} be the concept hierarchy of attribute a_i, and $V^p_{a_i,j}$ be the j-th attribute value of attribute a_i at level p in T_{a_i}. Then, a granule $(\{V^p_{a_i,j}\}, m(\{V^p_{a_i,j}\}))$ can be called an elementary information granule of attribute a_i at level p. There is $m(\{V^p_{a_i,j}\}) = \bigcup_{h=1}^{q} m(V_{a_i,h})$, where q is the number of the leaves of sub-tree with the root $V^p_{a_i,j}$, $V_{a_i,h}$ is one of leaves, i.e., an attribute value of attribute a_i, and $(\{V_{a_i,h}\}, m(\{V_{a_i,h}\}))$ is an elementary granule of attribute a_i.

Definition 7 (*k*-itemset). Let $I = \{I_1, \cdots, I_k\}$ be a k-itemset, where $I_i \in V_{a_i} (i = 1, 2, \cdots, k)$ is an attribute value of attribute a_i. The k-itemset I is listed in order according to the sequence of attributes, namely, $pri(a_1) > pri(a_2) > \cdots > pri(a_k)$, where $pri(a_i)$ refers to the order of attribute a_i.

Definition 8 (*k*-itemset granule). Let $I = \{I_1, \cdots, I_k\}$ be a k-itemset, and $B = \{a_1, a_2, \cdots, a_k\} \subseteq A$ be a subset of attributes. Then, information granule $IG = (I, m(I))$ is called a k-itemset granule, where $m(I) = \{u|f(u, a_1) = I_1 \wedge f(u, a_2) = I_2 \wedge f(u, a_k) = I_k, u \in U, a_i \in B, i = 1, \cdots, k\}$.

It should be ensured that a 1-itemset granule is an elementary granule satisfying the given conditions.

According to Definition 8, conclusions can be drawn as follows:

1) $m(I) = m(\{I_1\}) \cap m(\{I_2\}) \cap \cdots \cap m(\{I_k\})$.

2) If $I \subseteq V, J \subseteq V$, and $I \subseteq J$, then $m(I) \supseteq m(J)$. Namely, the size of information granular $(I, m(I))$ is greater or equal to that of information granule $(J, m(J))$, or $(I, m(I))$ is more abstract than $(J, m(J))$ as far as their concepts are concerned.

Definition 9 (Multi-dimension granular hierarchy). Let $(I, m(I))$ be a k_1-itemset granule and $(J, m(J))$ be a k_2-itemset granule. According to Definition 8, if $I \subseteq V, J \subseteq V$, and $I \subseteq J$, then $m(I) \supseteq m(J)$, i.e., the intension of $(J, m(J))$ is more concrete than that of $(I, m(I))$, denoted by $m(I) \prec m(J)$. Then, all these granules lead to a hierarchical structure by using the $\prec$ order, called a multi-dimensional granular hierarchy.

Definition 10 ($\otimes$ operation). Specialization of information granules, namely, the operation of information granules with respect to intersect $\otimes$: Let $I = \{I_1, \cdots, I_{k_1}\}, I \subseteq V$, and $J = \{J_1, \cdots, J_{k_2}\}, J \subseteq V$. Then, $IG1 = (I, m(I))$ be a k_1-itemset granule and $IG2 = (J, m(J))$ be a k_2-itemset granule. The operation of intersect between them is defined as: $IG = IG1 \otimes IG2 = (I \cup J, m(I) \cap m(J))$.

According to the definition, there comes a conclusion as follows:

$m(I) \cap m(J) = m(I \cup J) = m(\{I_1\}) \cap m(\{I_2\}) \cap \cdots \cap m(\{I_{k_1}\}) \cap m(\{J_1\}) \cap m(\{J_2\}) \cap \cdots \cap m(\{J_{k_2}\})$

Definition 11 ($\oplus$ operation). Generalization of information granules, namely the operation of information granules with respect to $\oplus$: Let $IG1 = (I, m(I))$ and $IG2 = (J, m(J))$ be two arbitrary granules. Then, the operation of $\oplus$ between them is defined as $IG = IG1 \oplus IG2 = (I \cap J, m(I \cap J))$.

Definition 12 (Association relationship). Let X be a k_1-itemset, where $X = \{x_1, \cdots, x_{k_1}\} \subseteq V$. Let Y be a k_2-itemset, where $Y = \{y_1, \cdots, y_{k_2}\} \subseteq V$, and satisfy $X \cap Y = \emptyset$. Let $(X, m(X))$ and $(Y, m(Y))$ be two arbitrary granules corresponding to X and Y, respectively. If there exists $m(X) \cap m(Y) \neq \emptyset$, then there is an association relationship between granule $(X, m(X))$ and granule $(Y, m(Y))$.

Definition 13 (Association rules based on granules). If there is an association relationship between arbitrary granules $(X, m(X))$ and $(Y, m(Y))$, then an implication of form $(X, m(X)) \Rightarrow (Y, m(Y))$ is called an association rule based on granules.

Definition 14 (Support of k-itemset granules). Let $(I, m(I))$ be a k-itemset granule. The support of the k-itemset granule is defined as support$=\| m(I) \| / \| U \|$.

Definition 15 (Support of association rules). Let $(X, m(X))$ and $(Y, m(Y))$ be two arbitrary granules, and there is an association relationship between them. Then, the support of association rule $(X, m(X)) \Rightarrow (Y, m(Y))$ is defined as support $= \| m(X) \cap m(Y) \| / \| U \|$.

Definition 16 (Confidence of association rules). Let $(X, m(X))$ and $(Y, m(Y))$ be two arbitrary granules, and there is an association relationship between them. Then, the confidence of association rule $(X, m(X)) \Rightarrow (Y, m(Y))$ is defined as confidence $= \| m(X) \cap m(Y) \| / \| m(X) \|$.

2 Generating frequent k-itemsets based on GrC

2.1 Framework for finding association rules based on GrC

In this section, we investigate another related question of finding association rules from relational databases based on GrC. In other words, we use information granules that satisfy the given conditions to solve a given problem. A framework for finding association rules based on GrC is shown in Fig. 1.

Firstly, through data pre-processing, all quantitative attributes will be changed into categorical attributes. A discretization is required for continuous quantitative attributes. There are two possible approaches to the discretization problem. One can optimize coding by taking into account merely the similarities of objects in the attributes′ space or one can maximize the predictive properties of the information system in the stage of coding. In this paper, the first approach is adopted. Secondly, according to Definition 2, every concept hierarchy of attributes is created with respect to the domain of interest by the way of human predefining or machine learning. In this paper, the way of human predefining is used to construct the corresponding concept hierarchy for a given attribute. Thirdly, elementary granules are generated by scanning the given relational database table or information table once based on the given concept hierarchies of attributes. The number of elementary granules is equal to the number of the leaves in concept hierarchies. Fourthly, on the basis of the elementary granules and the given different levels of attributes and by following algorithms (G-Approach), frequent k-itemsets can be obtained. Finally, according to the minimum confidence provided, all association rules are mined.

In an actual relational database table or an information table, some attributes are useful and significant and others may be useless or insignificant for finding association rules, such as memo and name attributes. The former ones, called effective attributes, are determined by the user first and kept in a certain sequence.

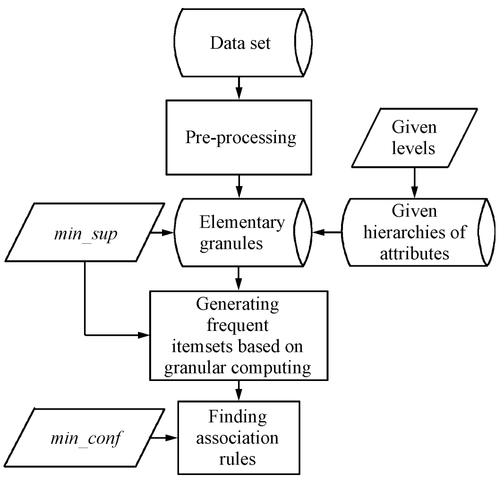

Data set
Pre-processing
Given levels
min_sup
Elementary granules
Given hierarchies of attributes
Generating frequent itemsets based on granular computing
min_conf
Finding association rules

Fig. 1 Framework for finding association rules based on GrC

2.2 Generating elementary granules and frequent 1-itemsets

For simplicity, an information granule (elementary granule or arbitrary k-itemset granule) is identified by using its forming conditions, namely, its intension. In other words, the intension of the information granule is viewed as its meaningful name. For example, a set of attribute values $\{V_{a_i,j}\}$ can stand for the elementary granule $(\{V_{a_i,j}\}, m(\{V_{a_i,j}\}))$ with regard to attribute value $V_{a_i,j}$. In this paper, a data structure for representing the elementary granule in memory space is defined as the 3-tuple $\langle$*count*, *item*, *pointer*$\rangle$, where the notation *count* denotes the number of individual objects included in the elementary granule. The notation *item* denotes the forming conditions of the elementary granule, and the notation *pointer* is used to link into a storing position of the individual objects. In the following algorithms, the notation $UIDSet$ denotes the storing position. An individual object is identified by its ID or a bit representation, and stored in an array or a linked list. The notation $Node_i$ denotes a set of elementary granules generated with respect to attribute a_i, called the granule table of attribute a_i, and the notation $Node_{i,j}$ stands for the j-th elementary granule or the j-th element in $Node_i$.

In the following, the algorithm of generating elementary granules is discussed first, then the algorithm generating elementary granules at level p is described.

Algorithm 1. Generating elementary granules from a relational database table or an information table.

Given data: The sequence of attributes and their corresponding concept hierarchies.

Input: A relational database table or an information table.

Output: Elementary granule table.

Algorithm description:

```
For $k = 1$ to $N$ //$N$ is the number of database records
For $i = 1$ to $m$ //$m$ is the number of effective attributes
{[to get the $j$-th attribute value $V_{a_i,j}$ of attribute $a_i$ in record $u_k$]
If $V_{a_i,j}$ exists in $Node_i$ then
    {$Node_{i,j}.count++$;
    $Node_{i,j}.pointer->UIDSet = Node_{i,j}.pointer->UIDSet \cup \{u_k\}$;}
else {
    new($p$); //generating a node
    $p->UIDSet = \{u_k\}$ ;
    $Node_{i,j}.count = 1$;
    $Node_{i,j}.item = \{V_{a_i,j}\}$;
    $Node_{i,j}.pointer = p$;}};
```

Every elementary granule $\{V_{a_i,j}\}$ is generated by running the algorithm, and all elementary granules of attribute a_i are organized into the granule table of attribute a_i. Thus, all granule tables of attributes form an elementary granule table.

Algorithm 2. Generating elementary granules with different levels.

Input: Corresponding levels $(p_1, p_2, \cdots, p_m)$, where p_i is a given level corresponding to the concept hierarchy of attribute a_i.

Output: Multi-level elementary granule table.

Algorithm description:

For $i = 1$ to m

Step 1. Get the domain of attribute a_i at the given level p_i from corresponding concept hierarchy. Let $V_{a_i}^{p_i} = \{V_{a_i,1}^{p_i}, V_{a_i,2}^{p_i}, \cdots, V_{a_i,q}^{p_i}\}$ be the domain of level p_i, where q is a cardinality of the set.

Step 2.

For $h = 1$ to q

Step 2.1. If $V_{a_i,h}^{p_i} \in V_{a_i}^{p_i}$ is not a leaf, then in order to construct an elementary granule at level p_i, we need to obtain all leaves of a subtree whose root is node $V_{a_i,h}^{p_i}$. Let $V_{a_i,h}^{p_i} = \{v_{a_i,1}, v_{a_i,2}, \cdots, v_{a_i,t}\}$ be a set of these leaves, where t is the number of the leaves.

Step 2.2. By scanning the given elementary granule table, an elementary granule $(\{v_{a_i,j}\}, m(\{v_{a_i,j}\}))$ can be taken, where $1 \leq j \leq t$. According to union operation of a set, an elementary granule at level p_i is generated as follows:

$$(\{V_{a_i,h}^{p_i}\}, m(\{V_{a_i,h}^{p_i}\})) = (\{V_{a_i,h}^{p_i}\}, \bigcup_{j=1}^{t} m(\{v_{a_i,j}\}))$$

If $V_{a_i,h}^{p_i}$ is a leaf, then an elementary granule $(\{V_{a_i,h}^{p_i}\}, m(\{V_{a_i,h}^{p_i}\}))$ at level p_i can be taken directly from the elementary granule table.

Step 2.3. Modify the granule table of attribute a_i.

In the following step, the elementary granule table at level p can also be viewed as an elementary granule table.

On the basis of the elementary granule table, frequent 1-itemsets can be generated easily. As noted, each frequent 1-itemset is the elementary granule satisfying minimum support. All frequent 1-itemsets also are organized into a table, called the granule table of frequent 1-itemsets. An algorithm for generating frequent 1-itemsets from the elementary granule table is described as follows.

Algorithm 3. Generating frequent 1-itemsets.

Input: Elementary granule table, minimum support: *min_sup*.

Output: Granule table of frequent 1-itemsets.

Algorithm description:

```
For $i = 1$ to $m$
For $j = 1$ to $\| Node_i \|$;
{ If $Node_{i,j}.count < min\_sup$ then
    { $q = Node_{i,j}$.pointer;
    delete($q$); delete($Node_{i,j}$); } }
```

2.3 Generating frequent *k*-itemsets

In order to facilitate the description of the algorithm of generating k-itemsets, several functions are first introduced as follows.

1) Function $getitem(s, i, n)$ is to get n elements from the i-th element in a sorted set s, including the i-th element.

2) Function $location(x, k)$ is to get a position of an elementary granule x in the elementary granule table of attribute a_k.

3) Function $getcount(p)$ is to get counts of nodes in a linked list p.

We use adjacency list to represent k-itemset granules like the representation of graphs. Vertex structure is the data structure of the elementary granule. But the pointer field is used as a link to a list. For each vertex, we keep a linked list of all k-itemset granules generated in order with regard to the vertex.

A node in the linked list is defined as the 5-tuple $\langle count, no, item, UIDpointer, next \rangle$, where the notation *count* denotes the number of individual elements of a k-itemset granule. The notation *no* is an index, which stands for the order of the attribute corresponding to the first element in the item field of the node. The notation *item* indicates all the elements except the first element in the k-itemset. The notation *UIDpointer* is used as a link to a list storing k-itemset granules. The *next* field links the next node together.

Attribute a_i corresponds to a k-itemset adjacency list, generally referred to as a k-itemset granule table of attribute a_i, denoted by notation $L_k\_Node_i$. The notation $L_k\_Node_{i,j}$ stands for the j-th elementary granule or element of the vertex table in $L_k\_Node_i$. In this paper, a k-itemset is formed by two $(k-1)$-itemsets, its forming regular is shown as Table 1.

Table 1 Forming regular of k-itemsets

First frequent $(k-1)$-itemset	Second frequent $(k-1)$-itemset
$L_{k-1}\_Node_1$	From $L_{k-1}\_Node_2$ to $L_{k-1}\_Node_m$
$L_{k-1}\_Node_2$	From $L_{k-1}\_Node_3$ to $L_{k-1}\_Node_m$
$\vdots$	$\vdots$
$L_{k-1}\_Node_{m-1}$	$L_{k-1}\_Node_m$

After generating frequent 1-itemsets, a frequent k-itemset $(k \geq 2)$ can be generated. It corresponds to a k-itemset granule that satisfies the minimum support. By using granule tables, some advantages can be used, and some heuristics can be provided to judge whether k-itemsets are frequent k-itemsets or not.

Suppose a 5-itemset $\{v_1, v_2, v_3, v_4, v_5\}$ is a candidate 5-itemset, and it consists of the combination of frequent 4-itemset $\{v_1, v_2, v_3, v_4\}$ in $L_4\_Node_1$ with frequent 4-itemset $\{v_2, v_3, v_4, v_5\}$ in $L_4\_Node_2$. Some details can be used to judge whether or not the 5-itemset is a frequent 5-itemset. First, if the number of the nodes in the linked list corresponding to $item = v_1$ is lower than 4, it is not a frequent 5-itemset. Second, if one of frequent 4-itemsets $\{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_3, v_5\}$, $\{v_1, v_2, v_4, v_5\}$, and $\{v_1, v_3, v_4, v_5\}$ does not exist in $L_4\_Node_1$, then it is not a frequent 5-itemset. Therefore, in order to reduce the number of candidate itemsets and save computing time in generating frequent k-itemsets, the above method can be used, especially when k is greater than 2.

Now we can describe algorithms of generating frequent k-itemsets based on GrC. First, the algorithm of generating frequent 2-itemsets is described as follows.

Algorithm 4. Generating frequent 2-itemsets.

Input: Granule table of frequent 1-itemsets, minimum support: *min_sup*.

Output: Granule table of frequent 2-itemsets.

Algorithm description:

```
For $i = 1$ to $m - 1$
Copy $(L_1\_Node_i)$; //copy frequent 1-itemsets
For $j = 1$ to $\| L_1\_Node_i \|$
For $r = i+1$ to $m$
For $k = 1$ to $\| L_1\_Node_r \|$ {
    $NewIG = L_1\_Node_{i,j} \otimes L_1\_Node_{r,k}$ //generating a 2-itemset granule
    if $\| NewIG \| \geq min\_sup$ then{
    new($p$); //generating a node
    $p->no = r$;
    $p->count = \| NewIG \|$;
    $p->item = L_1\_Node_{r,k}.item$;
    $p->UIDSet = L_1\_Node_{i,j}->UIDSet \cap L_1\_Node_{r,k}->UIDSet$;
    If $L_2\_Node_{i,j}.pointer <>$ Null then $rear->next = p$;
    else $Node_{i,j}.pointer = p$;
    $rear = p$;}}
```

Second, on the basis of frequent 2-itemsets, we can obtain frequent k-itemsets ($k \geq 3$) based on GrC.

Algorithm 5. Generating frequent k-itemsets ($k \geq 3$).

Input: Granule table of frequent 2-itemsets, minimum support: *min_sup*.

Output: Granule tables of frequent k-itemsets.

Algorithm description:

```
For $k = 3$ to $m$
Copy $(L_{k-1}\_Node_i)$;
For $i = 1$ to $m - k+1$
For $j = 1$ to $\| L_{k-1}\_Node_i \|${
$p = L_{k-1}\_Node_{i,j}.pointer$;
$num = getcount(p)$; //getting the counts of nodes in a linked list $p$
If $num < k - 1$ then
continue;//does not generate candidate $k$-itemset
While $p <>$Null do{
$h = p.no$;
If $m - h + 2 \leq k$ then break;
$item1 = L_{k-1}\_Node_{i,j}.item \cup p.item$; //getting the first frequent $(k-1)$-itemset
$first-element = getitem(p.item, 1, 1)$;
$r = location(first-element, h)$;
$q = L_{k-1}\_Node_{h,r}.pointer$;
While $q <>$Null do{
$item2 = q.item \cup L_{k-1}\_Node_{h,r}.item$;//getting the second frequent ($k$-1)-itemset
If $getitem(item1, 2, k-2) == getitem(item2, 1, k-2)$ then{
//getting a candidate $k$-itemset
$NewIG\_item = item1 \cup item2$;
$NewIG\_UIDSet = p->UIDSet \cap q->UIDSet$;
If $\| NewIG\_UIDSet \| \geq min\_sup$ then {
new($NewIG$); //generating a node
$NewIG->no = h$;
$NewIG->count = \| NewIG\_UIDSet \|$;
$NewIG->item = getitem(NewIG\_item, 2, k-1)$;
$NewIG->UIDSet = NewIG\_UIDSet$;
If $L_k\_Node_{i,j}.pointer <> NULL$ then
$rear->next = p$;
else { $Node_{i,j}.pointer = NewIG$;}
$rear = NewIG$;}}
$q = q->next$; }
$p = p->next$; }}
```

2.4 Analysis of time complexity

The time complexity of algorithms mainly depends on three parameters: support, the number of attributes, and the number of records of a relational database table or an information table. For the Algorithms 1 and 3, the number of loops costs O(mN) clearly, where m is the number of effective attributes and N is the number of records of the database. For Algorithm 2, it also costs O(mN) at the worst case. For the algorithms of generating frequent k-itemsets (i.e., Algorithms 4 and 5), they cost variably with different k (the number of items) and supports. For the given support, when k value becomes greater, because of the number of candidate itemsets decreases, the running time of the algorithms becomes short. For the given k value, when the support becomes greater, the running time of the algorithms becomes shorter. So, for a given k value and the support, Algorithm 4 takes the most time. It costs O(m^2N^2) in the worst case. The overall running time of the algorithms is dominated by Algorithm 4.

3 Example

Table 2 is the information table of students′ circulation records that has been pre-processed. We simply take five attributes from the data structure of the information table in order to illustrate the proposed algorithms for finding association rules. In Table 2, $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ is a universe. Let *min_sup* be $s = 20\,\%$.

Table 2 Students′ circulation records

RID	Student-type	Gender	Degree	Book-type
1	A	M	BS	T1
2	C	F	PhD	T2
3	A	M	BS	T3
4	B	F	MS	T1
5	C	F	PhD	T3
6	A	M	BS	T5
7	B	M	BS	T4
8	D	F	PhD	T5
9	C	F	MS	T4
10	B	M	MS	T2
11	A	F	BS	T5
12	D	M	MS	T1
13	D	F	BS	T2
14	C	M	MS	T1
15	B	M	PhD	T5

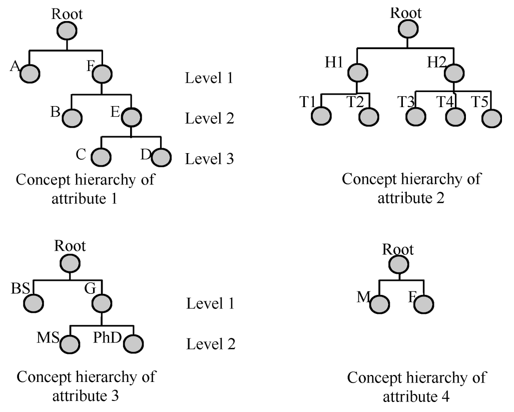

Root
A
F
B
E
C
D
Level 1
Level 2
Level 3
Concept hierarchy of attribute 1
Root
H1
H2
T1
T2
T3
T4
T5
Concept hierarchy of attribute 2
Root
BS
G
MS
PhD
Level 1
Level 2
Concept hierarchy of attribute 3
Root
M
F
Concept hierarchy of attribute 4

Fig. 2 Concept hierarchy

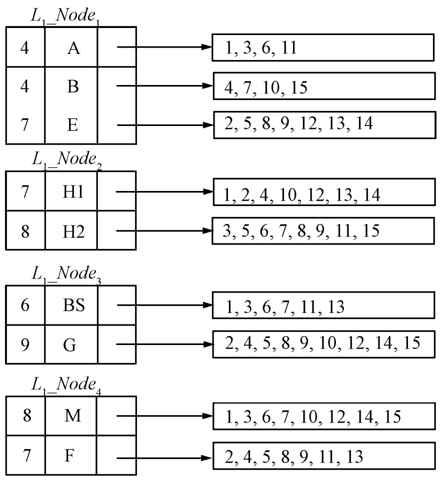
Suppose that the order of effective attributes is student-type, book-type, degree, and gender. Their hierarchies are given and shown in Fig. 2. In addition, suppose that the levels corresponding to each hierarchy are $L2, L1, L1$, and $L1$, respectively. Namely, $V^2_{\text{student-type}} = \{\text{A}, \text{B}, \text{E}\}$, $V^1_{\text{book-type}} = \{\text{H1}, \text{H2}\}$, $V^1_{\text{degree}} = \{\text{BS}, \text{G}\}$, $V^1_{\text{gender}} = \{\text{M}, \text{F}\}$.

3.1 Generating elementary granules

According to Algorithm 1, by scanning the data set once, the elementary granule table is generated. By the attribute of the student-type, four elementary granules are generated: ({A},{1,3,6,11}), ({B},{4,7,10,15}), ({C},{2,5,9,14}), and ({D},{8,12,13}), respectively. According to the attribute of book-type, elementary granules ({T1},{1,4,12,14}), ({T2},{2,10,13}), ({T3},{3,5}), ({T4},{7,9}), and ({T5},{6,8,11,15}) are generated. With regard to the attribute degree, we can obtain three elementary granules: ({BS},{1,3,6,7,11,13}), ({MS},{4,9,10,12,14}), and ({PhD},{2,5,8,15}). And up to the attribute gender, two elementary granules ({M},{1,3,6,7,10,12,14,15}) and ({F}, {2,4,5,8,9,11,13}) are constructed.

3.2 Generating frequent 1-itemsets

Frequent 1-itemsets can be generated by using Algorithm 3, and shown in Fig. 3, among them ({E}, m({E})) = ({E}, m({C}) ∪ m({D})), ({H1}, m({H1})) = ({H1}, m({T1}) ∪ m({T2})), ({H2}, m({H2})) = ({H2}, m({T3}) ∪ m({T4}) ∪ m({T5})), ({G}, m({G})) = ({G}, m({MS}) ∪ m({PhD})). In this example, all 1-itemsets are frequent 1-itemsets.



L_1_Node$_1$
4 A → 1, 3, 6, 11
4 B → 4, 7, 10, 15
7 E → 2, 5, 8, 9, 12, 13, 14
L_1_Node$_2$
7 H1 → 1, 2, 4, 10, 12, 13, 14
8 H2 → 3, 5, 6, 7, 8, 9, 11, 15
L_1_Node$_3$
6 BS → 1, 3, 6, 7, 11, 13
9 G → 2, 4, 5, 8, 9, 10, 12, 14, 15
L_1_Node$_4$
8 M → 1, 3, 6, 7, 10, 12, 14, 15
7 F → 2, 4, 5, 8, 9, 11, 13

Fig. 3 Granule table of frequent 1-itemsets

3.3 Generating frequent 2-itemsets

Based on the granule table of frequent 1-itemsets, we can obtain frequent 2-itemsets by using Algorithm 4. The results are shown in Fig. 4. In order to generate a frequent 2-itemset, a candidate 2-itemset is formed in order, thereafter generating a 2-itemset granule by intersection operation between two corresponding elementary granules. If the support of the 2-itemset granule satisfies the minimum support, the candidate 2-itemset is a frequent 2-itemset. In the example, candidate 2-itemset {A, H1} is generated by intersection operation between the two 1-itemsets granules {A} and {H1}. Because of the support of the granule being less than the minimum support, the candidate 2-itemset {A, H1} is not a frequent 2-itemset. Other candidate 2-itemsets, such as {A, H2}, {B, H1}, {B, H2}, $\cdots$, {G, F}, are formed one by one in order. Among them, candidate, 2-itemsets, such as {A, H2}, {A, BS}, $\cdots$, {G, M}, and {G, F}, are frequent 2-itemsets.

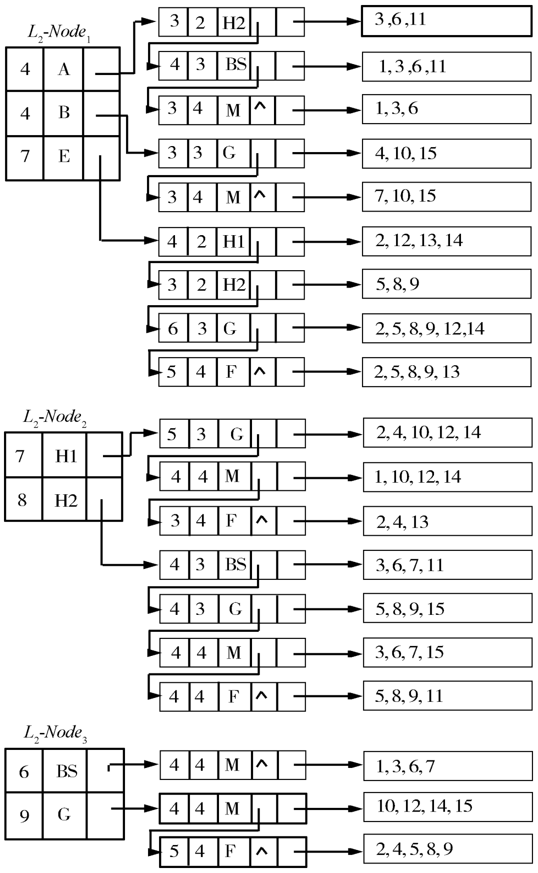


Fig. 4 Granule table of frequent 2-itemsets

3.4 Generating frequent *k*-itemsets

Based on the granule table of frequent 2-itemsets, we can generate frequent k-itemsets ($k \geq 3$) by using Algorithm 5. The results are shown in Fig. 5. According to Algorithm 5, the first frequent 2-itemset {A, H2} can be obtained easily in L_2_$Node_1$. With the help of the field *no*, we can get another frequent 2-itemset {H2, BS} in L_2_$Node_2$. Since they can form candidate 3-itemset {A, H2, BS}, a corresponding 3-itemset granule is generated by using intersection operation between two 2-itemsets granules. By computing, the support of the 3-itemset granule satisfies the minimum support, so the candidate 3-itemset {A, H2, BS} is a frequent 3-itemset. But, for frequent 2-itemset {A, M} in L_2_$Node_1$, since the value of the field *no* (equal to 4) is the last attribute in order, a candidate 3-itemset can not be formed by combining frequent 2-itemset {A, M} with any other frequent 2-itemset from L_2_$Node_2$ and L_2_$Node_3$. By using the heuristics provided, some candidate 3-itemsets, such as {A, H2, G}, {A, H2, M}, {A, H2, F}, and so on, can not be formed. And other candidate 3-itemsets like {H1, G, M} can be formed. Finally, all frequent 3-itemsets are obtained.

After all frequent k-itemsets ($k \geq 3$) are generated, frequent (k+1)-itemsets can be generated. In this example, since the supports of all candidate 4-itemsets do not satisfy the minimum support, none of the frequent 4-itemsets can be generated.

So far, all frequent itemsets have been generated. Suppose that the minimum confidence is 80 %, and an implication of association rules is $X \Rightarrow Y$, where only one item in Y set exists, we can find association rules as follows:

A ⇒ BS, E ⇒ G

A ∧ H2 ⇒ BS, A ∧ M ⇒ BS, E ∧ H2 ⇒ G, E ∧ F ⇒ G, G ∧ F ⇒ E

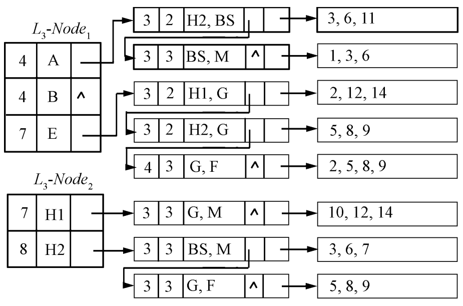


Fig. 5 Granule table of frequent 3-itemsets

4 Experiment

The aim of this experiment mainly includes two aspects. One is to evaluate feasibility and effectiveness of G-Approach. The other is to examine the ability of generating multi-dimensional generalized association rules by using the taxonomy information on attributes.

Two kinds of datasets are considered in our experiment. One is dense data, including mushroom and chess, which are taken from the Irvine Machine Learning Database Repository. The other is sparse data taken from a real-life data set, called circulation data set. These data sets and their features are described in Table 3.

Table 3 Data sets and their characteristics

Data sets	Records	Items	Attributes
Mushroom	8 124	120	23
Chess	3 196	76	37
Circulation	57 600	36	4

The circulation data set containing the information of students′ circulation records is produced from one medium sized library. The library has accumulated 57 600 entries of records with 4 kinds of attributes and 36 different values. And it averagely has 9 attribute values at every attribute in this database after pretreatment

In order to compare with G-Approach, we select another two popular algorithms, which are Apriori and FP-tree, respectively. The two algorithms are taken from the web site http://fimi.cs.helsink.fi/.

All experiments are performed on a lightly loaded P4 2.4 GHz CPU, 256 M of RAM, Windows 2000 (professional), and VC++6.0. The software developed is based on C++.

4.1 Testing 1: mining association rules without considering concept hierarchies of attributes

In order to validate the effectiveness and feasibility of this G-approach, we compare the performance of G-Approach and the other two algorithms on two kinds of data sets under different supports without considering taxonomy trees of attributes in each data set. First, we mainly examine how G-Approach scales up with different kinds of data sets and the number of attributes. Next, we investigate the scale-up as we increase the number of records (namely, transaction volume) from 8 000 to 16 000 on mushroom.

Experiment results are shown in Tables 4 ∼ 7.

Tables 4 CPU time for generating frequent itemsets on mushroom (records = 8 000) (s)

Support (%)	G-Approach	FP-tree	Apriori
70	0.197	0.13	2.343
60	0.205	0.151	2.39
50	0.296	0.168	2.578
40	0.86	0.57	3.98
30	2.41	1.603	9.04

Table 5 CPU time for generating frequent itemsets on mushroom (records = 16 000) (s)

Support (%)	G-Approach	FP-tree	Apriori
70	0.351	0.24	4.617
60	0.43	0.27	4.734
50	0.583	0.301	5.156
40	1.69	0.81	7.89
30	5.953	2.03	18.01

Tables 6 CPU time for generating frequent itemsets on circulation (records = 57 600) (s)

Support (%)	G-Approach	FP-tree	Apriori
10	0.51	0.19	2.14
9	0.51	0.20	2.14
8	0.68	0.201	3.82
7	0.71	0.212	3.826
6	2.45	0.212	80.01
5	3.57	0.28	155.04

Tables 7 CPU time for generating frequent itemsets on chess (records = 3 196) (s)

Support (%)	G-Approach	FP-tree	Apriori
90	0.39	0.64	4.291
85	1.375	1.75	10.218
80	4.062	4.82	28.343
75	25.687	14.798	74.296
70	120.468	49.937	203.09

4.2 Testing 2: finding association rules at different levels of granularity

Since the basic algorithms such as FP-tree and Apriori for finding association rules do not take the presence of taxonomies into consideration, the association rules are restricted to the leaf-level values in the taxonomy. In order to get different levels of granularity from relational databases, an obvious solution to the problem is to replace each record with an “extended record”. The extended record contains all the values of attributes in the original record as well as all the ancestors of each value of the attribute in the original record. Thus, two algorithms FP-tree and Apriori can be run on these extended records to get association rules at any level of the taxonomy (or different levels of granularity).

The way to extend records can be illustrated as follows.

Let a record in an original data set be n-tuple $(v_1, v_2, \cdots, v_n)$, and suppose there are h internal nodes in the taxonomy tree of attribute a_i. If the attribute a_i is considered, then n-tuple is extended into $(n + h)$-tuple $(v_1, v_2, \cdots, v_{i-1}, v_{i1}, v_{i2}, \cdots, v_{ih}, v_i, \cdots, v_n)$, where $(v_{i1}, v_{i2}, \cdots, v_{ih})$ is the list of internal nodes of the taxonomy tree of attribute a_i, and is organized in order. Value v_{ij} is the root of the taxonomy tree. If value v_{ij} is not the ancestor, then it is denoted by symbol "$*$".

For example, in Fig. 2, the taxonomy tree of attribute 1 has 3 internal nodes. Let a record of the circulation data set be 4-tuple (B, T3, MS, F). If we want to extend attribute 1, then the tuple (B, T3, MS, F) is replaced by (Root-1, F, $*$, B, T3, MS, F), where only two internal nodes Root-1 and F are the ancestors of the node B, and the symbol $*$ stands for a special sign in the record. If we want to extend all attributes, then the mentioned tuple (B, T3, MS, F) is substituted with (Root-1, F, $*$, B, Root-2, $*$, H2, T3, Root-3, G, MS, Root-4, F) with 13 attributes values.

Obviously, we can generate different levels of association rules on these extended records. But many redundant rules are generated along with the available rules. We call the frequent itemsets including special sign "$*$" as redundant frequent itemsets. In order to measure the size of redundant frequent itemsets, we calculate the ratio of redundant frequent itemsets to total frequent itemsets under the given different minimum support.

In our testing, the taxonomy trees of attributes in the circulation data set are similar to Fig. 2. We only extend the circulation data set in attributes 1 and 2. Thus, we obtain the extended circulation data set with 9 attributes. FP-tree and Apriori run on the extended circulation data set, and G-Approach runs on the original circulation data set. Table 8 shows the CPU time taken by the three algorithms as the minimum support decreases from 60 % to 10 %. Table 9 shows the ratio of redundant frequent itemsets to total frequent itemsets under the given different minimum support in mining association rules with the given levels of granularity by using FP-tree and Apriori on the circulation data set.

Table 8 CPU time for generating frequent itemsets at the given levels of granularity on circulation (s)

Support (%)	G-Approach	FP-tree	Apriori
60	0.156	0.153	2.486
50	0.171	0.168	2.562
40	0.187	0.191	2.765
30	0.203	0.218	2.921
20	0.343	0.31	7.968
10	0.515	0.678	16.046

Table 9 The size of redundant frequent itemsets

Support (%)	Redundant frequent itemsets rate (%)
60	26.3
50	21.05
40	25.53
30	25.31
20	63.01
10	62.8

4.3 Discussion of experiment results

In Subsection 4.1, empirical evaluation shows that G-Approach is effective and flexible on two different kinds of data sets. The strategy to generate association rules for G-Approach and Apriori is to find frequent itemsets. But because of potentially large number of database scans and candidate itemsets, Apriori does not perform very well for each data set. From Tables 4 ∼ 7, G-Approach functions better than Apriori in all cases. Compared with Apriori, the superior performance of G-Approach on these data sets arises from generating fewer candidate itemsets by using optimized search space and some heuristics in generating frequent itemsets based on GrC.

FP-tree is an optimal approach to mining frequent patterns without candidate generation. It mainly takes time in generating frequent patterns tree and discovering the frequent patterns by using the FP-growth approach. Compared with G-Approach, FP-tree performs somewhat better than G-Approach on two different kinds of data sets. However, experiment results show that the proposed G-Approach is basically equal to FP-tree in the performance with respect to run-time when support is above some threshold. From Tables 4 ∼ 7, it is shown that the performance of G-Approach relative to FP-tree has little difference for three data sets of mushroom with support being greater than 40 %, students′ circulation with support being greater than 7 %, and chess with support being greater than 75 %. Both G-Approach and FP-tree exhibit similar scale-up with the number of records.

In Subsection 4.2, as far as mining association rules with different levels of granularity are concerned, G-Approach is more convenient and effective than both FP-tree and Apriori. Table 8 shows that the performance of G-Approach relative to FP-tree and Apriori for the students′ circulation data set. G-Approach performs better than Apriori in all different support cases. And G-Approach does better than FP-tree in three cases with 10 %, 30 %, and 40 % supports, respectively.

Based on multiple taxonomies over attributes and operations between elementary information granules of attributes, multi-dimensional granular hierarchies are generated. So G-Approach can mine association rules with different levels of granularity from original relational databases without redundant rules.

FP-tree and Apriori can find associations between attributes at any level of the taxonomy, but they have to run on extended data sets. For each extended data set, many redundant frequent itemsets and corresponding redundant association rules are generated as the number of attributes increases. Table 9 shows that the number of frequent itemsets generally increases when the given minimum support decreases. Otherwise, applying basic algorithms for mining association rules with different levels of granularity, we must transform original data sets into extended data sets, which results in some disadvantages such as sparse data turning into dense data, the number of attributes increasing, larger itemsets, and so on. Thus, it probably takes more CPU time for FP-tree and Apriori running on extended data sets.

5 Conclusion and future work

This paper discusses the basic concepts of information granules and application of GrC in mining association rules from relational databases. Based on the GrC, algorithms of generating frequent itemsets are proposed for finding association rules with different levels of granularity from a relational database table or an information table. The proposed algorithms use granules as basic processing elements to find association rules, and the meaning of granules is clear. Compared with FP-tree and Apriori, the proposed G-Approach is effective, flexible and more convenient for finding association rules with different levels of granularity and makes problem solving feasible. Experiment results show that the theory of GrC is effective and feasible and with high applicability. In addition, with slight changes of the algorithms, it is also convenient to find the maximal frequent itemsets.

Further work involves such aspects as optimizing the algorithm, testing further in a large relational database table or an information table, making comparison with other methods in all aspects, and modifying the algorithm so as to find association rules for a dynamic database.

References

1 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD International Conference on Management of Data. Washington D. C., USA: ACM, 1993. 207−216

2 Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proceedings of the 20th Very Large Database Conference. Santiago, Chile: ACM, 1994. 487−499

3 Savasere A, Omiecinski E, Navathe S B. An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st International Conference on Very Large Database. San Francisco, USA: Morgan Kaufmann Publishers, 1995. 432−444

4 Toivonen H. Sampling large databases for association rules. In: Proceedings of the 22nd International Conference on Very Large Database. San Francisco, USA: Morgan Kaufmann Publishers, 1996. 134−145

5 Han J W, Kamber M [Author], Fan Ming, Meng Xiao-Feng [Translator]. *Data Mining: Concepts and Techniques*. Beijing: China Machine Press, 2001. 152−166 (in Chinese)

6 Dunham M H. *Data Mining: Introductory and Advanced Topics*. Beijing: Tsinghua University Press, 2003. 164−191

7 Liu Jun-Qiang, Sun Xiao-Ying, Pan Yun-He. Survey on association rules mining technology. *Computer Science*, 2004, **31**(1): 110−113 (in Chinese)

8 Au W H, Chan K C C. Mining changes in association rules: a fuzzy approach. *Fuzzy Sets and Systems*, 2005, **149**(1): 87−104

9 Yun U. An efficient mining of weighted frequent patterns with length decreasing support constraints. *Knowledge-Based Systems*, 2008, **21**(8): 741−752

10 Hong T P, Horng C Y, Wu C H, Wang S L. An improved data mining approach using predictive itemsets. *Expert Systems with Applications*, 2009, **36**(1): 72−80

11 Ma Hai-Bin. The Techniques Research on Frequent Pattern Mining [Ph. D. dissertation], Fudan University, China, 2005

12 Hu T M, Sung S Y, Xiong H, Fu Q. Discovery of maximum length frequent itemsets. *Information Sciences: an International Journal*, 2008, **178**(1): 69−87

13 Bargiela A, Pedrycz W. *Granular Computing: An Introduction*. Boston: Kluwer Academic Publishers, 2003. 1−17

14 Miao Duo-Qian, Wang Guo-Yin, Liu Qing, Lin Tsan-Young, Yao Yi-Yu. *Granular Computing: Past, Present, and the Future Perspectives*. Beijing: Academic Press, 2007. 1−178 (in Chinese)

15 Yao Y Y. Perspectives of granular computing. In: Proceedings of the International Conference on Granular Computing. Beijing, China: IEEE, 2005. 85−90

16 Yao Y Y. The art of granular computing. In: Proceedings of the International Conference on Rough Sets and Intelligent Systems Paradigms. Warsaw, Poland: Springer, 2007. 101−112

17 Yao Yi-Yu. The rise of granular computing. *Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)*, 2008, **20**(3): 299−308 (in Chinese)

18 Yao Y Y, Yao J T. Granular computing as a basis for consistent classification problems. In: Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Seoul, Korea: ACM, 2003. 101−106

19 Liu Qing. *Rough Sets and Rough Reasoning*. Beijing: Academic Press, 2005. 100−115 (in Chinese)

20 Zadeh L A. Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*, 1997, **90**(2): 111−127

21 Nguyen S H, Skowron A, Stepaniuk J. Granular computing: a rough set approach. *Computational Intelligence*, 2001, **17**(3): 514−544

22 Hobbs J R. Granularity. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, USA: Morgan Kaufmann, 1985. 432−435

23 Zhang L, Zhang B. The quotient space theory of problem solving. *Fundamenta Informaticae*, 2004, **59**(2-3): 287−298

24 Lin T Y. Granular computings II: infrastructures for AI-engineering. In: Proceedings of the International Conference on Granular Computing. Atlanta, USA: IEEE, 2006. 2−7

25 Lin T Y. Formal models for granular computings highlights and fallacies. In: Proceedings of the International Conference on Granular Computing. Hanzhong, China: IEEE, 2008. 5−10

26 Pei D W. Some models of granular computing. In: Proceedings of the International Conference on Granular Computing. Silicon Valley, USA: IEEE, 2007. 11−16

27 Liu Q, Sun H, Wang Y. Granulations based on semantics of rough logical formulas and its reasoning. In: Proceedings of the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. Toronoto, Canada: Springer, 2007. 419−426

28 Yao Y Y. Granular computing for data mining. In: Proceedings of the International Society for Optical Engineering Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security. Orlando, USA: IEEE, 2006. 1−12

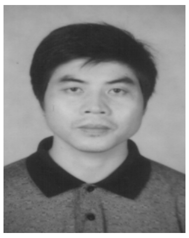
29 Lin T Y, Louie E. Finding association rules by granular computing: fast algorithms for finding association rules. In: Proceedings of the 12th International Conference on Data Mining, Rough Sets and Granular Computing. Berlin, German: Springer, 2002. 23−42

30 Lin T Y, Louie E. Modeling the real world for data mining: granular computing approach. In: Proceedings of the 9th International Fuzzy System Association World Congress. Vancouver, Canada: IEEE, 2001. 3044−3049

31 Pedrycz W, Park B J, Oh S K. The design of granular classifiers: a study in the synergy of interval calculus and fuzzy sets in pattern recognition. *Pattern Recognition*, 2008, **41**(12): 3720−3735

32 Chen M C, Chen L S, Hsu C C, Zeng W R. An information granulation based data mining approach for classifying imbalanced data. *Information Sciences*, 2008, **178**(16): 3214−3227

33 Apolloni B, Bassis S. Algorithmic inference: from information granules to subtending functions. *Nonlinear Analysis: Hybrid Systems*, 2008, **2**(2): 665−683


QIU Tao-Rong Ph. D. candidate at Beijing Jiaotong University and professor at Nanchang University. He received his master degree from Nanjing University of Technology in 1991. His research interest covers rough sets, GrC, and knowledge discovery. Corresponding author of this paper. E-mail: taorongqiu@163.com

LIU Qing Professor in the Department of Computer, Nanchang University. His research interest covers rough set, rough reasoning, GrC granular reasoning, and artificial intelligence.
E-mail: qliu_ncu@yahoo.com.cn

HUANG Hou-Kuan Professor at the School of Computer and Information Technology, Beijing Jiaotong University. His research interest covers agent, knowledge discovery, data mining, and artificial intelligence.
E-mail: hkhuang@center.njtu.edu.cn