Mining negative generalized knowledge from relational databases

Yu-Ying Wu [a,*], Yen-Liang Chen [b], Ray-I Chang [c]

[a] Department of Information Management, Nanya Institute of Technology, Jhongli, Taiwan
[b] Department of Information Management, National Central University, Jhongli, Taiwan
[c] Department of Engineering Science and Ocean Engineering, National Taiwan University, Taipei, Taiwan

A R T I C L E   I N F O

A B S T R A C T

Attribute-oriented induction (AOI) is a useful data mining method for extracting generalized knowledge from relational data and users' background knowledge. Concept hierarchies can be integrated with the AOI method to induce multi-level generalized knowledge. However, the existing AOI approaches are only capable of mining positive knowledge from databases; thus, rare but important negative generalized knowledge that is unknown, unexpected, or contradictory to what the user believes, can be missed. In this study, we propose a global negative attribute-oriented induction (GNAOI) approach that can generate comprehensive and multiple-level negative generalized knowledge at the same time. Two pruning properties, the downward level closure property and the upward superset closure property, are employed to improve the efficiency of the algorithm, and a new interest measure, $nim(cl)$, is exploited to measure the degree of the negative relation. Experiment results from a real-life dataset show that the proposed method is effective in finding global negative generalized knowledge.

1. Introduction

The information explosion has been a serious challenge for current information gathering institutions. There is an urgent need for improved technology that can access, analyze, summarize, and interpret information intelligently and automatically [1]. Data mining has been proposed in response to this demand, and many varied approaches have been developed [2–5]. Data generalization, which generalizes a large set of task-relevant data from a relatively low conceptual level to higher conceptual levels, is one of the most important approaches for summarizing data characterization. Such a technique can generate concise and abstract knowledge, which helps in providing overall pictures of the data at different levels [6,7]. We call such data generalized knowledge.

Attribute-oriented induction (AOI), which was first introduced by Cai et al. [8], is a data mining technique used to extract generalized knowledge from relational data and users' background knowledge. This essential background knowledge takes the form of a concept hierarchy that is associated with each attribute in a relational database [9]. A concept hierarchy often refers to domain knowledge, and is a way to store relationships between specific concepts and generalized concepts [10]. The integration of the concept hierarchies and AOI methods can allow us to induce multi-level and cross-level generalized knowledge, which can provide good decision making support for different levels of management.

There is no doubt that the AOI technique is very useful for finding general characteristics from a relational database, and it has been applied in many areas, such as spatial patterns [11,12], medical science [13,14], intrusion detection [15,16], strategy making [17], and financial prediction [18]. However, as we can see from these applications, existing AOI approaches are only focused on mining positive generalized knowledge from databases. In the real world, there is a type of knowledge that is unknown, unexpected, or contradictory to what a user believes, called negative generalized knowledge in this study, which is more novel, meaningful, and interesting to the user than positive facts.

Thus, discovering negative generalized knowledge from a relational database can reveal more interesting phenomena. For example, a medical database may tell us that although many patients are Taiwanese and numerous patients have H1N1, only a few Taiwanese have this disease. This interesting negative fact suggests that, due to some unknown reasons, Taiwanese are more resistant to H1N1. However, this negative fact can never be found using traditional approaches, which only find positive relations or associations between data.

In this work, a generalized tuple is defined as an association of attribute values in concept hierarchies. For example, (Taiwanese, Pneumonia) and (Taiwanese, H1N1) are examples of generalized tuples. A generalized tuple is negative if its actual occurrence is significantly lower than its expected occurrence. The aim of this paper is to find all negative generalized tuples. To the best of our knowledge, the mining of negative tuples has been neglected in previous research on AOI; instead, the previous studies mined only positive generalized facts. As a result of the low frequency of negative

* Corresponding author. Tel.: +886 3 4361070; fax: +886 3 4360713.
E-mail address: yywu@alumni.ncu.edu.tw (Y.-Y. Wu).

generalized tuples, which might contradict our expectations, these tuples may never be found.

To resolve this problem, this study proposed a novel approach, called global negative AOI (GNAOI), which can generate comprehensive and multiple-level negative generalized knowledge at the same time. Only one pass over a database is necessary with the developed algorithm, and the efficiency of the algorithm is improved by the employment of two pruning properties, the downward level closure property and the upward superset closure property, and a new interest measure, $nim(cl)$, is exploited to measure the negative degree. Furthermore, it is not appropriate to generate all negative generalized tuples without information reduction, because too much information may hinder users from applying knowledge. Therefore, we must come up with some way of finding only the interesting negative generalized tuples. Accordingly, we use the following criteria to search for interesting negative generalized tuples:

(1) *Minimum support and expected support constraint:* Each value in a negative generalized tuple must satisfy the minimum support constraint. Meanwhile, a negative generalized tuple must satisfy the minimum expected support constraint. The rationale is that if a generalized tuple covers only a tiny fraction of data, it is more likely to represent noise rather than some other more interest characteristic. Therefore, we require that each generalized tuple must satisfy a minimum representability constraint.

(2) *Strong negative association:* A negative tuple represents a large deviation between the actual and the expected supports of a tuple. Therefore, we say two values have strong negative association if the probability of their co-occurrence is significantly smaller than the multiplication of their individual probabilities. The deviation of a negative generalized tuple is measured by the relative ratio of its actual support and its expected support occurrence. Only those generalized tuples with strong deviations are worth keeping.

(3) *Redundant measure constraint:* The multiple-level taxonomic structures means that a generalized tuple can induce many similar tuples simply by replacing some attribute values with their ancestors or descendants in the taxonomy. Accordingly, many generalized tuples become redundant because their information can be reasonably induced from other tuples; such tuples are unnecessary and should be removed.

The remainder of the paper is organized as follows. In Section 2, the AOI method and related work are reviewed. In Section 3, we define the problem of mining global negative generalized knowledge from a relational database. In Section 4, the mining algorithm, GNAOI, is presented. The results of our performance evaluation are given in Section 5. Finally, some conclusions are drawn in Section 6.

2. Related work

AOI is a set-oriented database mining method that generalizes the task-relevant subsets of the data, attribute by attribute [19]. The generalization of each attribute is associated with a concept hierarchy. The concept hierarchies represent necessary background knowledge, which controls the generalization process and can range from the single, most generalized root concept, to the most specific concepts corresponding to the specific values of attributes in the database [10,20].

There has been much research effort made using AOI approaches. Carter and Hamilton [21] proposed more efficient methods for AOI, while Cheung [22] used a rule-based conditional concept hierarchy to extend the traditional approach to a conditional AOI, thereby allowing different tuples to be generalized

through different paths, depending on other attributes of the tuples. Hsu [23] extended the basic AOI algorithm to the generalization of numeric values, and Chen and Shen [20] proposed a dynamic programming algorithm based on AOI techniques to find generalized knowledge from an ordered list of data. Several independent groups of researchers have combined fuzzy concept hierarchy procedures with AOI [2,24–26]. A fuzzy hierarchy of concepts reflects the degree to which a concept belongs to its direct parent concept. More than one direct parent of a single concept is allowed during fuzzy induction. AOI has also been integrated with other applications to induce domain dependent generalized knowledge [11–18,27,28].

The above methods, however, all concentrate on mining positive generalized knowledge. To the best of our knowledge, the mining of multi-level negative generalized tuples has not been addressed. The only studies similar to our work are those concerned with negative association rule mining methods. According to the classification scheme proposed by Tan et al. [29] there are three approaches for the mining of negative association rules or patterns. The first notes the absence of a certain item as a negative item [30,31]. After selecting a small set of negative items that we are interested in, we can augment the database by adding these negative items to corresponding transactions. In this way, existing association rule mining methods can be used to find negative association rules. However, as pointed out in Tan et al. [29], the major weakness of this approach lies in the fact that it can only handle a small number of negative items. When the number of negative items is large, the computation time becomes terribly long.

The second type is a neighborhood-based approach known as indirect association [32,33]. A pair of items a and b, is said to have indirect association if there is an item set Y such that $support(a, Y)$ and $support(b, Y)$ are both high, while $support(a, b)$ is low. In other words, a and b are replacements for each other in the sense that they are seldom purchased together, but at the same time, there are a number of other items commonly purchased together with them. Although this approach is interesting, it is incapable of finding negative associations between items at multiple-levels in hierarchies.

The third type relies on a concept hierarchy to estimate the expected support of an itemset. In this way, the methods can determine whether an itemset is negative or not [34,35]. Although the methods also use concept hierarchy to estimate the expected supports of the itemsets, there are some important differences between this method and our approach:

(1) Their databases are transactional, while ours is relational.
(2) They use a single hierarchy for the entire database, but we set a hierarchy for every attribute.
(3) The outputs are different. They generate negative association rules, but we generate negative generalized tuples.
(4) Their negative association rules are composed of original items, while our negative generalized tuples are composed of original items as well as generalized items from the hierarchies.
(5) They use a concept hierarchy to estimate the expected supports of itemsets rather than generating patterns with generalized items in the hierarchies. In this work, we do both.

In short, we take a fundamentally different approach to that which has been discussed thus far. Our goal is to develop an efficient mining method for discovering negative knowledge across different levels of taxonomies from relational databases.

3. Problem formulation

In this section, we define the problem of mining global negative generalized tuples from a relational database. Let $A = \{a_1, a_2, \ldots, a_n\}$

be a set of attributes. The value domain of each attribute a_i is associated with a corresponding concept hierarchy tree T_i. A concept hierarchy represents a taxonomy of values for an attribute domain that are partially ordered according to a specific-to-abstract relation. A value in a tree is also called a node. Let $v_{k,i}$ denote the k_{th} node of tree T_i, where the sequence of nodes is numbered according to some tree traversal order. Let $lvl(v_{k,i})$ denote the level of value $v_{k,i}$; let $leaf(T_i)$ denote the set of leaf nodes of tree T_i; and let $nonleaf(T_i)$ denote the set of non-leaf nodes of tree T_i.

Example 1. Let us consider a small research grant database with 20 tuples and 4 attributes, as shown in Table 1. To save space, all concept hierarchies are shown only in Appendix A. In Fig. A1 in Appendix A, there are four attributes with four concept hierarchies, T_1, T_2, T_3 and T_4, respectively. Suppose the node labels are numbered according to the level order traversal. Now, we can find the values $v_{10,2}$ = Spanish, $v_{3,2}$ = 'Liberal arts' in hierarchy T_2 as shown in Fig. 1. Moreover, we have $lvl(v_{1,2}) = 1$, $lvl(v_{9,2}) = 3$, $leaf(T_2)$ = {History, Music, Spanish, French, ..., Software_eng, Information_tech}.

Based on these notations, the problem can be stated as follows:

Definition 1 (*Valueset*). A valueset $l = \{v_{k1,i1},\ v_{k2,i2}, \ldots, v_{kr,ir}\}$ is a non-empty set of values, where no two values in the set belong to the same attribute, i.e., all $i_p \neq i_q$. A valueset with k values is called a k-valueset.

For example, l = {Delhi, History} is a 2-valueset, while {Delhi, New York} is not.

Definition 2 (*Tuple, Rid*). Let m be the number of attributes in the relation. A primitive tuple t is an m-valueset whose each value is in $leaf(T_i)$ for some i. Further, a generalized tuple g is an m-valueset whose each value is in $leaf(T_i)$ or $nonleaf(T_i)$ for some i. A relational database D is a set of primitive tuples. Each tuple in database D is associated with a unique identifier Rid.

Definition 3 (*Cover*). For each value $v_{k,i}$, let $cov(v_{k,i})$ denote the cover of value $v_{k,i}$, which is the set of $Rids$ of primitive tuples in D whose values of attribute i are $v_{k,i}$ or descendents of $v_{k,i}$ in the concept hierarchy T_i. For each valueset $l = \{v_{k1,i1}, v_{k2,i2}, \ldots, v_{kr,ir}\}$, $cov(l)$ = $cov(v_{k1,i1}) \cap \ldots \cap cov(v_{kr,ir})$ $(1 \leqslant r \leqslant n)$.

Example 2. In Table 1, each record is called a primitive tuple and has a unique number Rid. From Table 1, we find that cov(New York) = {2, 8, 12, 20}, cov(Physics) = {8, 11, 12} and cov{New York, Physics} = {8, 12}. After performing the generalized process, the cover of the abstract value is the union of the cover of all its descendents in the concept hierarchy tree. Therefore, we have cov(USA) = {2, 5, 8, 11, 12, 17, 19, 20}, cov(Science) = {2, 8, 11, 12, 14} and cov(USA, Science) = {2, 8, 11, 12}.

Definition 4 (*Support*). The support of a valueset l in a database D, $sup(l)$, is the number of $Rids$ in $cov(l)$ versus the total number of tuples in D, as shown in Eq. (1):

$$Sup(l) = \frac{|cov(l)|}{|D|}. \tag{1}$$

Example 3. The cardinality of $|cov$(USA, Science)$|$ = |{2, 8, 11, 12}| = 4 and $|D|$ = 20. So, we obtain sup(USA, Science) = 4/20 = 20%.

Real data tends to be dirty, noisy and error-prone and a lot of noise may be hidden in the data. Therefore if the frequency of some value among the values in the database is too small it is more likely to be a noise rather than information. If the support of a value is too small such values as noise and pruned to reduce the search space. Hence, a *minValueSup* threshold is necessary to ensure that the generalized tuple is statistically significant.

Definition 5 (*minValueSup*). A value $v_{k,i}$ is called a frequent value (fv) if $sup(v_{k,i}) \geqslant minValueSup$ as specified by the user. A candidate valueset (cl) is a non-empty valueset, where each value in the set is a frequent value.

Example 4. Consider the relation shown in Table 1. Assume that *minValueSup* = 15%. Table 2 shows partial results for the values at all levels, where the right hand column indicates whether these values are in set *FV* or frequent values. We keep the values with support $\geqslant$ *minValueSup* (15% in this example). The cover field records the $Rids$ of those primitive tuples in Table 1 whose values are descendents or the same as the table values in the concept hierarchy. Furthermore, all the values are sorted from abstract concepts to specific concepts.

In addition to the *minValueSup* constraint, the other pruning threshold is *minExpectedSup*. An expected support is the expected probability that a valueset would occur if all values in the valueset are independent. In probability theory, two values X and Y are independent if $P(X \cup Y) = P(X) \times P(Y)$. According to the theory, we can define the *expected support* of cl as follows:

$$exSup(cl) = sup(v_{k1}, i_1) \times sup(v_{k2}, i_2) \times \cdots \times sup(v_{kr}, i_r)(1 \leqslant r \leqslant n).$$

Since we are searching for negative generalized tuples whose actual frequencies are much lower than expected ones, we must set a *minExpectedSup* threshold to filter out the valuesets which cover only a tiny fraction of the data. By setting the threshold of *minExpectedSup* properly, we can remove many unimportant valuesets, and speed up the computation.

Definition 6 (*minExpectedSup*). A potential valueset (pl) is a cl whose expected support is not less than *minExpectedSup* threshold.

Table 1
Synthetic relation showing research grants with 20 tuples and 4 attributes.

Rid	Birthplace	Department	Age	Amount	Rid	Birthplace	Department	Age	Amount
1	Delhi	History	30	12250	11	LA	Physics	40	25000
2	New York	Math	55	26900	12	New York	Physics	42	25230
3	Taipei	History	30	14950	13	Bombay	Music	30	23980
4	Taichung	Music	60	24950	14	Calcutta	Math	30	20343
5	Orlando	Software_eng	52	28635	15	Delhi	History	30	12250
6	Tainan	Music	52	24000	16	Kaohsiung	Electrical_eng	52	15535
7	York	Spanish	65	42500	17	Orlando	Software_eng	52	28635
8	New York	Physics	42	25230	18	Delhi	History	23	9998
9	Cambridge	Spanish	60	59100	19	Chicago	Software_eng	60	27335
10	Paris	French	68	186925	20	New York	Information_tech	66	32905
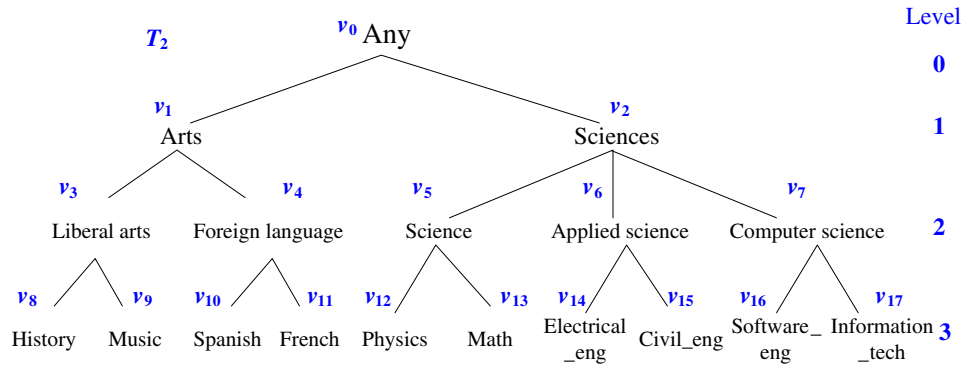
Fig. 1. Concept hierarchy T_2 with labeling.

In Section 1, we mentioned that an interesting negative generalized tuple must satisfy three additional constraints. Obviously, the potential valuesets in Definitions 5 and 6 satisfy the first constraint, including the minimal value support threshold and the minimal expected support threshold. In order to satisfy the second constraint, the strong negative association constraint, we must define the interest measure of a negative generalized tuple.

As mentioned above, a negative tuple represents a large deviation between the actual and the expected support of the tuple. The larger the deviation, the more interesting the tuple is considered to be [36]. Hence, the interest measure of a valueset is defined in terms of the *unexpectedness* of the valueset.

Definition 7 (*Interest, minInterest*). The negative interest measure of pl, $nim(pl)$, is obtained by

$$nim(pl) = \frac{exSup(pl)}{sup(pl)}. \qquad (2)$$

This measure is greater than one when the actual support value is lower than the expected support value, indicating a negative dependence. The larger the value is above 1, the greater is the negative dependence. A potential valueset pl is interesting if $nim(pl) \geqslant minInterest$, where $minInterest$ is a user-specified constant. An interesting negative generalized tuple ng is an interesting negative m-valueset.

Example 5. Following Example 4, Table 3 shows the partial results for the candidate valueset (cl). Assume that $minExpectedSup = 2\%$ and $minInterest = 2.5$. The last valueset cl_4 is not a potential valueset, because $exSup(cl_4)$ is less than $minExpectedSup$. Moreover, only the first valueset is interesting, because $nim(cl_1) \geqslant minInterest$.

In addition to the pruning and interest constraints, two redundant cases need to be considered.

Definition 8 (*Redundant valueset*). Case 1: A valueset $pl_a = \{v_{k1,i1}, v_{k2,i2}, \ldots, v_{kr,ir}, v_{kr+1,ir+1}\}$ is redundant if there is a valueset $pl_b = \{v_{k1,i1}, v_{k2,i2}, \ldots, v_{kr,ir}\}$, where $nim(pl_b) \geqslant nim(pl_a)$. Case 2: A valueset $pl_a = \{v_{k1,i1}, v_{k2,i2}, \ldots, v_{kr,ir}\}$ is redundant if there is a valueset $pl'_a = \{v'_{k1,i1}, v'_{k2,i2}, \ldots, v'_{kr,ir}\}$, where every value in pl_a is a descendant or the same as the corresponding value in pl'_a, and $nim(pl'_a) \geqslant nim(pl_a)$.

Example 6. Suppose there are four negative valuesets, as shown in Table 4. The valueset pl_1 is a subset of valueset pl_2, and $nim(pl_2) < nim(pl_1)$. We regard pl_2 as redundant and it should be removed. Another case is for the last two valuesets. Valueset pl_3 is more abstract than pl_4, and $nim(pl_3) \geqslant nim(pl_4)$. We keep the more general valueset in this case.

Table 2
Partial results for all level values.

Values	Support (%)	Cover	FV
Asia	45	1, 3, 4, 6, 13, 14, 15, 16, 18	Y
Sciences	50	2, 5, 8, 11, 12, 14, 16, 17, 19, 20	Y
Young	30	1, 3, 13, 14, 15, 18	Y
Old	55	2, 4, 5, 6, 7, 9, 10, 16, 17, 19, 20	Y
High	55	2, 5, 7, 8, 9, 10, 11, 12, 17, 19, 20	Y
…	…	…	…
France	**5**	**10**	**N**
Science	25	2, 8, 11, 12, 14	Y
21–30	30	1, 3, 13, 14, 15, 18	Y
25000–44999	45	2, 5, 7, 8, 11, 12, 17, 19, 20	Y
…	…	…	…
Delhi	15	1, 15, 18	Y
New York	20	2, 8, 12, 20	Y
Math	**10**	**2, 14**	**N**
20000–24999	20	4, 6, 13, 14	Y

3.1. Problem statement

Given a relational database D and a set of thresholds (*minValueSup*, *minExpectedSup*, *minInterest*), the aim is to find all non-redundant interesting negative generalized tuples.

4. Global negative attribute-oriented induction

In this section, we propose a novel GNAOI approach for mining multiple-level negative generalized tuples from a relational database. The algorithm combines attribute values with their concept hierarchies to induce the negative generalized knowledge. A hierarchy-information encoded table [37], which stores the generalized identifier (GID) code of the corresponding attribute value and cover lists [38] (i.e., Rid-lists) is adopted to save space and speed up the computation. Furthermore, both the downward level closure property and upward superset closure property are employed in the algorithm to efficiently generate global negative generalized knowledge. The input of the GNAOI algorithm includes three parts: (1) a task-related relation; (2) a set of concept hierarchies; and (3) a set of thresholds. The output consists of non-redundant interesting negative generalized tuples learned from the relational data. The major steps of the GNAOI are shown in Fig. 2. We first discuss how to collect and transform the input relational data into the hierarchical-information encoded table. Then we discuss in detail the algorithm designed for inducing all negative generalized tuples. Finally, the pruning and transformation processes are shown.

4.1. Collect and encode the task-relevant tuples

GNAOI first collects the dataset that is relevant to the learning task using relational database operations, e.g., projection and selec-

Table 3
Partial results for candidate valueset (CL).

CL	Sup (%)	exSup (%)	nim	PL	Interest
(cl_1) Sciences, Young	5	15	3	Y	**Y**
(cl_2) Old, Science, 25000–44999	5	6.187	1.237	Y	N
(cl_3) High, Science, New York	15	2.75	0.183	Y	N
(cl_4) Asia, Sciences, 21–30, 20000–24999	5	1.35	0.27	N	N

Table 4
Partial results for potential valueset (PL).

PL	nim	Pruning
(pl_1) Old, Science	2.75	
(pl_2) Old, Science, 25000–44999	1.237	Redundant
(pl_3) Asia, Sciences, Young	1.35	
(pl_4) Asia, Sciences, 21–30	1.35	Redundant

tion. The retrieved task-relevant tuples are stored in a table called the initial relation table. Each tuple in the initial relation has a unique identifier called *Rid*. Next, the initial relation can be transformed into a hierarchical-information encoded table, which stores the GID code of the corresponding attribute value. A GID is a number string containing the route information for an attribute value in a concept hierarchy.

Example 7. In Table 1, the amount attribute is the fourth attribute in the tuple. Suppose the value of the amount is '19800'. This can be encoded as '4121', where the first digit '4' represents the fourth attribute in tuple, the second '1' represents the 'Low' at level-1, the third '2' represents '15000–24999' at level-2, and the fourth '1' indicates '15000–19999' at level-3 (Fig. 3).

It is beneficial to transform the original object values to GID codes. First, an encoded string contains complete route information for an attribute value which provides a self-explanatory advantage. Second, a code requires fewer bits than the corresponding object-identifier which reduces the size of the tuples. Encoding can also be performed during the collection of task-relevant data, so there is no extra encoding pass required. The taxonomic information for each value in Table 1 is encoded as a GID code, following the encoding rule mentioned above. The results are shown in Table 5 (T).

4.2. The NGT algorithm

The generalization of negative valuesets is a difficult task and the computational load is heavy. How to effectively identify poten-

tial negative tuples is the key problem. In this study, several useful constraints are used to generate and measure the potential valuesets. Both the downward level closure property and upward superset closure property are included in the algorithm to improve the generalization efficiency. The first step in this algorithm is to scan the encoded table T, and find the set of all values (CV) for all levels. Next, the set of frequent values (FV) for all levels is generated, as defined in Definition 5, and sorted by a specific rule.

Let VS_k be the set of negative k-valuesets. All k-valuesets ($k \geqslant 2$) are generated using VS_{k-1} and FV. The k-valueset generation is repeated until k is equal to the number of attributes in the tuple. It is important to note that 1-valuesets (VS_1) is the same as FV. Finally, we obtain all negative generalized tuples by uniting all k-valuesets. The details of the NGT algorithm are shown in Fig. 4.

First, a pass is made over T to produce the set of all values for all levels CV. Moreover, the support for each value at all levels is calculated. The cover (rids-list) is recorded at the same time by scanning T once. Using the cover technique, it is not necessary during the k-valuesets generalization process to scan the dataset any more. All frequent values of all levels (set FV) are generated in line 2. The steps can be formulated in Fig. 5.

The values in FV are then sorted according to their attribute and concept level, from abstract to specific. For example, if there is a FV = {11**, 1111, 12**, 111*, 112*, 122*, 1222}, then the sorted FV' = {11**, 111*, 1111, 112*, 12**, 122*, 1222}. The sorting is beneficial for the subsequent generalization. It is important to note that the sorting process can also be performed at the *init-pass* (line 1) step by adopting appropriate data structure; no extra sorting pass is required. Furthermore, the FV is the same as 1-valuesets (VS_1) which are used to generate the 2-valuesets.

After finding the 1-valuesets, lines 5–7 proceed with the derivation of the negative k-valuesets ($k \geqslant 2$). The negative valuesets in set VS_{k-1} (found in the $(k-1)$th pass) and the frequent values in set FV are used to generate the negative valuesets VS_k in each subsequent pass, say pass k, using the function *negative-gen*. In this function, the deviation between the actual and the expected support of each valueset is calculated. After this, two useful properties called *downward level closure property* and *upward superset closure property* are adopted to prune unnecessary candidates.

Algorithm GNAOI.

Input: (1) A task-related relation; (2) a set of concept hierarchies; (3) a set of thresholds.

Output. Non-redundant interesting negative generalized tuples mined from the task-related relation.

Method. Four steps:

    Step 1. Collect and encode task-relevant tuples.

    Step 2. Discover all negative generalized tuples.

    Step 3. Remove redundant negative generalized tuples.

    Step 4. Transform and Output final tuples.
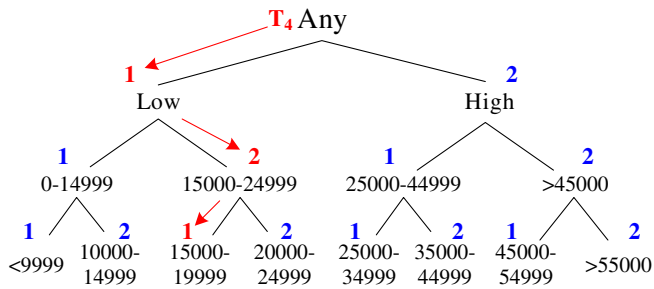
Fig. 2. GNAOI algorithm.

Fig. 3. Concept hierarchy for amounts showing encoded numbers.

Lemma 1 (Downward level closure property). *If a k-valueset $\{v_1, v_2, \ldots, v_k\}$ is not a potential valueset, then all of the valuesets $\{v'_1, v'_2, \ldots, v'_k\}$, where v'_i is either v_i or a descendant of v_i, are not potential valuesets.*

Proof. The support of a value is always larger on the abstract level than that on the specific level That is, the higher the concept level, the larger the expected support. Consequently, if a valueset is not a potential valueset at the higher level, then its valuesets at the lower level are also not. Therefore, the lemma follows.

For example, if $\{11^{**}, 32^{**}, 221^*\}$ is not a potential valueset, then $\{111^*, 321^*, 221^*\}$, $\{112^*, 32^{**}, 2212\}$, $\{11^{**}, 3213, 221^*\}$, $\{1123, 3211, 2212\}$ are also not potential valuesets. □

Lemma 2 (Upward superset closure property). *If a k-valueset $\{v_1, v_2, \ldots, v_k\}$ is not a potential valueset, then all of its $(k + 1)$-supersets are not potential valuesets.*

Proof. The expected support of a k-valueset is always larger than that of its $(k + 1)$-superset. That is, the more values in the valueset, the smaller the expected support. Consequently, if a valueset is not a potential valueset, then none of its supersets are either. Therefore, the lemma follows.

For example, if $\{11^{**}, 32^{**}\}$ is not a potential valueset, then $\{11^{**}, 32^{**}, 221^*\}$, $\{11^{**}, 32^{**}, 21^{**}, 42^{**}\}$ are also not potential valuesets.

Fig. 6 gives the *negative-gen* algorithm, which takes VS_{k-1} and FV as arguments, and returns VS_k, the set of all negative k-valuesets. The join step uses a double loop (line 1–3) to generate the k-valuesets. However, each valueset in the VS_{k-1} contains different attribute values. Before generating a new valueset, the algorithm must find the other attributes which do not exist in the valueset. It is a non-trivial task to find these attributes and join them with existent

Table 5
Results encoded from Table 1: *T*

Rid	Birthplace	Department	Age	Amount	Rid	Birthplace	Department	Age	Amount
1	1111	2111	3120	4112	11	1311	2211	3210	4211
2	1312	2212	3310	4211	12	1312	2211	3220	4211
3	1121	2111	3120	4112	13	1112	2112	3120	4122
4	1122	2112	3310	4122	14	1113	2212	3120	4122
5	1314	2231	3310	4211	15	1111	2111	3120	4112
6	1123	2112	3310	4122	16	1124	2221	3310	4121
7	1212	2121	3320	4212	17	1314	2231	3310	4211
8	1312	2211	3220	4211	18	1111	2111	3120	4111
9	1213	2121	3310	4222	19	1313	2231	3310	4211
10	1222	2122	3320	4222	20	1312	2232	3320	4211

Algorithm (NGT) For discovering all negative generalized tuples.

Input. (1) The hierarchical-information encoded table *T*; and (2) the set of thresholds.

Output. All interesting negative generalized tuples.

Procedure NGT(*T, minValueSup, minExpectedSup, minInterest*)

// |attr|: the number of attributes in the tuples

{

1 CV = init-pass(T);

2 $FV = \{v \mid v \in CV, sup(v) \geq minValueSup\}$;

3 sort(FV);

4 $VS_1 = FV$;

5 **for** ($k = 2$; $k \leq$ |attr|; k++) **do** {

6   VS_k = negative-gen(VS_{k-1}, FV);

7 }

8 $S = \cup_k VS_k$;

9 Convert S to negative generalized tuples, G;

}

Fig. 4. NGT algorithm.

for each value $v \in CV$ **do** {

    if $sup(v) \geq minValueSup$ **then**

        insert $<v>$ into FV;

}

Fig. 5. Steps for generating FV.

Procedure negative-gen(VS_{k-1}, FV)

// *attr*(u): the attributes of valueset *u*

// |D|: the total number of tuples

// *nim*: the negative interest measure

{

1.   **for each** valueset $u \in VS_{k-1}$ **do** {

2.   find the attributes *n* does not exist in *attr*(*u*)

3.   **for each** value $v \in FV(n)$ **do**

4.   **if** <*u,v*> does not be marked **then** {

5.   calculate *exSup*(<u, *v*>);

6.   **if** *exSup*(<u, *v*>) > *minExpectedSup* **then** {

7.   *cov*(<*u,v*>) = intersection(*u,v*);

8.   *sup*(<*u,v*>) = |*cov*(<*u,v*>)| / |D|;

9.   calculate *nim*(<u, *v*>) ;

10.   **if** *nim*(<u, *v*>) > *minInterest* and *nim*(<u, *v*>) > *nim*(*u*) **then**

11.   insert <*u,v*> into VS_k;

12.   }

13.   **else**

14.   mark all the descendant valuesets of <*u,v*>;

15.   }

16.   }

}

Fig. 6. *Negative-gen* algorithm.

valuesets. For efficient joining, the *FV* is split into several *FV*(*n*), in which the argument *n* represents the attribute. For example, *FV*(1) = {11**, 111*, 1111, 112*, 12**, 122*, 1222 …}, and *FV*(2) = {22**, 221*, 2212, 222* …}. The data required for the *FV* can be structured as a dynamic array list or a tree structure. After the join step, the algorithm examines whether the new valueset is marked, which will be ignored directly. On the other hand, line 5 calculates the expected support of valueset $\langle u, v \rangle$. As mentioned in Definition 6, if *exSup*($\langle u, v \rangle$) is less than the *minExpectedSup* threshold, valueset $\langle u, v \rangle$ will be discarded (line 6) and no more the superset of valueset $\langle u, v \rangle$ will be generated. Further, according to the downward level closure property, all of the descendent *k*-valuesets of valueset $\langle u, v \rangle$ will be also discarded (line 14). On the other hand, if *exSup*($\langle u, v \rangle$) is larger than *minExpectedSup*, lines 7–9 are used to calculate the interest measure. Finally, the algorithm keeps the

k-valuesets whose *nim*($\langle u, v \rangle$) measures are larger than the *minInterest* threshold (lines 10 and 11). Here, we also check if $\langle u, v \rangle$ matches the condition of redundant case 1 in Definition 8, by examining if there exists any subset of $\langle u, v \rangle$ with a stronger negative interest value. If so, $\langle u, v \rangle$ is redundant. Otherwise, it is insert into VS_k.

After performing the *negative-gen* function, the NGT algorithm unites all *k*-valuesets, where *k* is from 2 to |*attr*| (line 8, Fig. 4). We thus obtain all negative generalized tuples. The final step is to convert all negative *k*-valuesets to the generalized tuple (line 9, Fig. 4). The step is given in Fig. 7. □

Example 8. The 2-valuesets {1231, 21**} can be converted to the generalized tuple {1231, 21**, any, any}, while the 3-valuesets {211*, 31**, 42**} can be converted to the generalized tuple {211*, 31**, 42**, any}.

Procedure convert(S)

// *S*: the set of all negative *k*-valuesets (*k* > 1)

// |*v*|: the number of values in the valueset

// |*attr*|: the number of attributes in the tuple

{

1   **for each** valueset $g \in S$ **do**

2   **if** |*v*| < |*attr*| **then**

3   **for** (*i*=1, *i* ≤ (|*attr*|-|*v*|), *i*++) **do**

4   insert <*any*> into *g*.

}

Fig. 7. *Convert* algorithm.

```
Procedure redundant-pruning(G)
// G: the set of all negative generalized tuples
// nim(g): the interest value of tuple g
{
1        interesting-sort(G);
2.       for each valueset g ∈ G do
3          for each valueset g' ∈ G that is after g do
4.           if nim(g') <= nim(g) and g' is a specific tuple of g then
5               remove g' from G;
}
```

Fig. 8. *Redundant-pruning* algorithm.

Table 6
Negative generalized tuples generated by the GNAOI algorithm.

Seq	Education	Age	Occupation	Constellation	Income	Card amount	Interest
1	Middle	–	Housewife (full-time)	–	Low	0–40000	4.175
2	–	Young-adult	Housewife (full-time)	–	Low	0–40000	3.981
3	–	–	Housewife (full-time)	–	Low	0–40000	3.934
4	–	Young-adult	Self-business	–	–	0–40000	5.613
5	–	Young-adult	Sales	–	Low	Low	5.235
6	–	–	Sales	Fire	Low	–	2.77
7	High	Advanced-age	Home worker	–	–	–	3.09
8	High	35–49	Home worker	–	–	–	3.268
9	Low	Young-adult	Worker	–	High	–	4.261
10	Middle	–	Farmer	–	Low	0–40000	4.132
11	–	35–49	Student	–	–	High	3.908
12	–	Advanced-age	Student	–	–	High	3.641
13	–	35–49	Student	–	High	–	4.119
14	–	Advanced-age	Student	–	High	–	3.838
15		Young-adult		Air	10001–30000	0–40000	6.652
16	–	15–24	Worker	Air	Low	0–40000	4.983
17	–	Young-adult	Worker	Air	1–10000	Low	3.831
18		Young-adult	–	Earth	<10001	80001–150000	5.600
19	–	Young-adult	–	Earth	<10001	High	3.607
20	Low	25–34	High-level		High		2.418

4.3. Pruning and transformation

The negative generalized tuples discovered by the NGT algorithm include multiple-level values. However, not every generalized tuple is worth being presented to the users. If a generalized tuple is more interesting than a tuple which is more specific (every value in the tuple is the same or at a lower level), the latter is considered a redundant tuple, as defined in Definition 8, Case 2. Using the algorithm shown in Fig. 8, all redundant generalized tuples can be removed. Since the interest value measures the negative dependence, the algorithm first sorts the generalized tuples by their interest value, from larger to smaller. The removal process is performed starting from the tuple which has the largest interest.

The final step is to transform and output all non-redundant negative generalized tuples. To do so, each GID code of g is converted to an original value. Next, the final tuples are output. For example, the encoded tuple ⟨111*, 211*, Any, 41**⟩ will be converted to ⟨India, Liberal arts, Any, Low⟩.

5. Experiments

In this section, we discuss several experiments conducted to evaluate the performance of the proposed algorithm. The GNAOI algorithm was implemented using the Java programming language and tested on a PC with an Intel Core Duo 3.0 GHz processor and
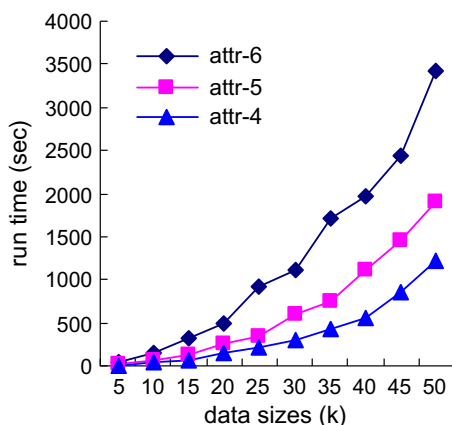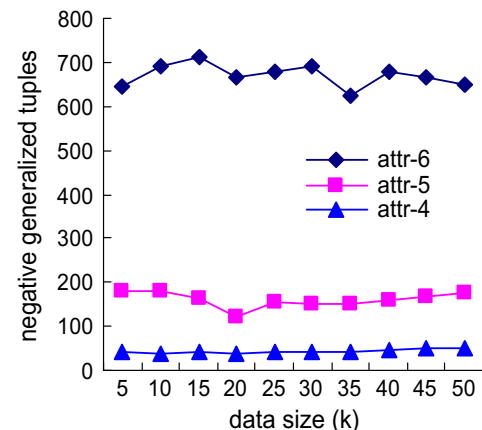

Fig. 9. Run time vs. data size.

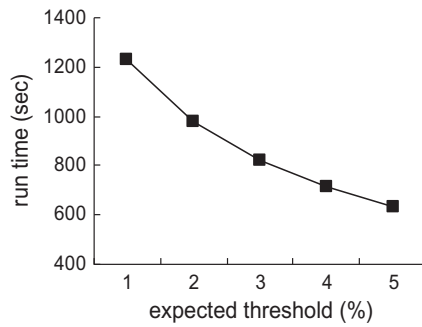**Fig. 10.** Number of tuples vs. data size.

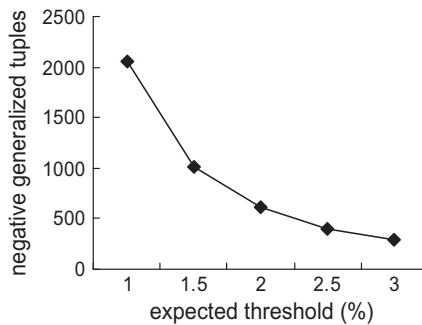**Fig. 11.** Run time vs. *minExpectedSup* threshold.

Fig. 12. Number of tuples vs. *minExpectedSup* threshold.

2 GB main memory under the Windows XP operating system. A real dataset of credit card data was used in the experiments. To make the time measurements more reliable, no other application was running on the machine while the experiments were running.

Since the traditional AOI method does not focus on negative tuples, the objective of this experiment was not to compare the two algorithms. Instead, we focused on evaluating the effectiveness and performance of the GNAOI algorithm under the various parameter settings. Specifically, its performance was studied with respect to four factors, including: (1) the number of tuples; (2) the number of attributes; (3) the minimal expected support thresholds; and (4) the interest thresholds. The run time and the variation of the number of negative generalized tuples with the change in value of the factors were noted.

5.1. Real dataset

The real dataset used in this study consists of credit card data collected from the survey research center at a university in Taiwan.
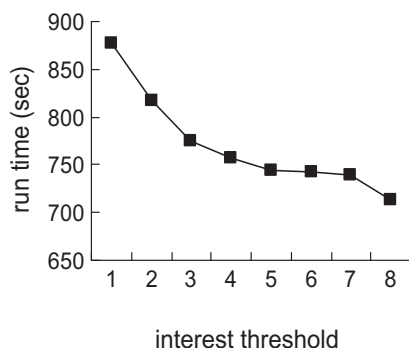

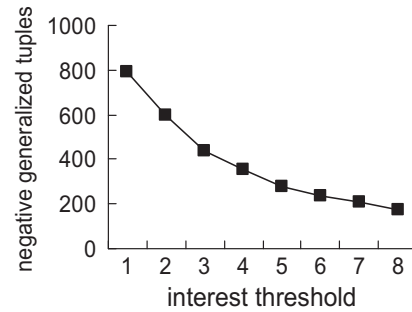**Fig. 13.** Run time vs. *minInterest* threshold.

Fig. 14. Number of tuples vs. *minInterest* threshold.

The dataset consists of 2 million tuples, each of which has 27 attributes including card type, frequency, sex, age, occupation, marital status, education, blood type, income, constellation, consumption pattern, and so on. We invited two domain experts who select six attributes whose values are well suited to repeated generalization and create concept hierarchies from them. The selected attributes are education, age, occupation, constellation, income and credit card consumption. The concept hierarchy for each attribute is shown in Fig. A2 in Appendix A. Since the attribute values of original dataset are stored as specific codes defined by the research center, we must first transform these attribute values to the leaf concepts of concept hierarchies, i.e., the GID code, before the generalization tasks can be started.

5.2. Negative generalized tuples

In this section, we discuss the negative generalized knowledge produced by the GNAOI algorithm. In the test, we randomly extracted 50000 tuples, each with 6 attributes, from the credit card dataset to mine the negative generalized tuples. The parameters are as follows: minValueSup = 10%, minExpectedSup = 2% and the interest measure threshold is 1.2. Applying the GNAOI algorithm and repeating the extraction and mining process 50 times, we get the average amount of interesting negative generalized tuples is 637. Some interesting negative tuples are shown in Table 6. The values 'Any' are represented by '–'.

Each of the tuples in Table 6 can be interpreted as negative generalized knowledge. The table shows some interesting observations. From tuples 1, 2, and 3, we see that the card consumption of full-time housewives with low incomes may be higher than expected. One possible reason could be that since housewives usually manage a whole household they spend more money with credit cards. Tuple 5 shows young-adult sales may earn low income but spending more money. Tuples 7 and 8 show advanced-age home workers are seldom high education, especially between 35 and 49 years old. Tuples 11 and 12 indicate that older students are conservative about spending money with credit cards. Tuples 13 and 14 show that older students seldom have high income, especially in the 35–49 age bracket. From tuples 15, 16 and 17, we see that young-adult workers with the Air element constellation and low income may spend more money with credit cards. In contrast, tuples 18 and 19 show that the Earth element constellation group is conservative at spending money with credit cards. Tuple 20 points out those younger high-level workers with high income are seldom low education.

5.3. Performance evaluations

In this section, we describe a series of experiments evaluating the performance of the process. In the first, we evaluate the run time for the GNAOI algorithm using varied data size (number of

tuples) from 5,000 to 50,000 but fixed all thresholds. The thresholds are as follows: minValueSup = 10%, minExpectedSup = 2%, and the interest measure threshold is 1.2. In addition, different numbers of attributes are evaluated: 4 attributes (attr-4); 5 attributes (attr-5); and 6 attributes (attr-6). We repeat the evaluations 50 times. Each time, the evaluated datasets are randomly extracted from source database. The average results of computation are summarized in Fig. 9. From this figure, we can see that the time required increases as the number of tuples increases. Moreover, when more attributes are given, more run time is required. This is because the GNAOI algorithm generates the candidate valuesets by recursively combining distinct attribute values. Hence, the number of attributes has a significant effect on the run time.

We also study how the number of generalized tuples changes as the data size changes. Fig. 10 shows the results of filtering for different data sizes and different numbers of attributes. The results show that more negative generalized tuples are generated as more attributes are considered. The reason is that more attributes result in more negative generalized tuples, which in turn generates more interesting tuples. However, the number of generalized tuples remains almost the same for the same number of attributes, even though the data size changes. The reason for this may be that, since the data are all drawn from the same dataset, the data sets have similar properties, even though their sizes are different.

Next, we study how the minExpectedSup threshold influences the run time of the GNAOI algorithm. To this end, we set five thresholds and fix the data size at 25000 and the number of attributes at 6. The other thresholds are set as follows: minValueSup = 10% and the interest measure threshold is 1.2. We also repeat this evaluation 50 times. Each time, we randomly extracted 25000 tuples from source database. Fig. 11 shows the average performance curve of computation. Time required decreases as the expected support threshold values increase. The reason is that when the thresholds increase, the number of candidate valuesets decrease, which in turn decreases the run time.

We also demonstrate how the number of negative generalized tuples changes as the minExpectedSup values change. Fig. 12 shows the results for different thresholds. One can see that the
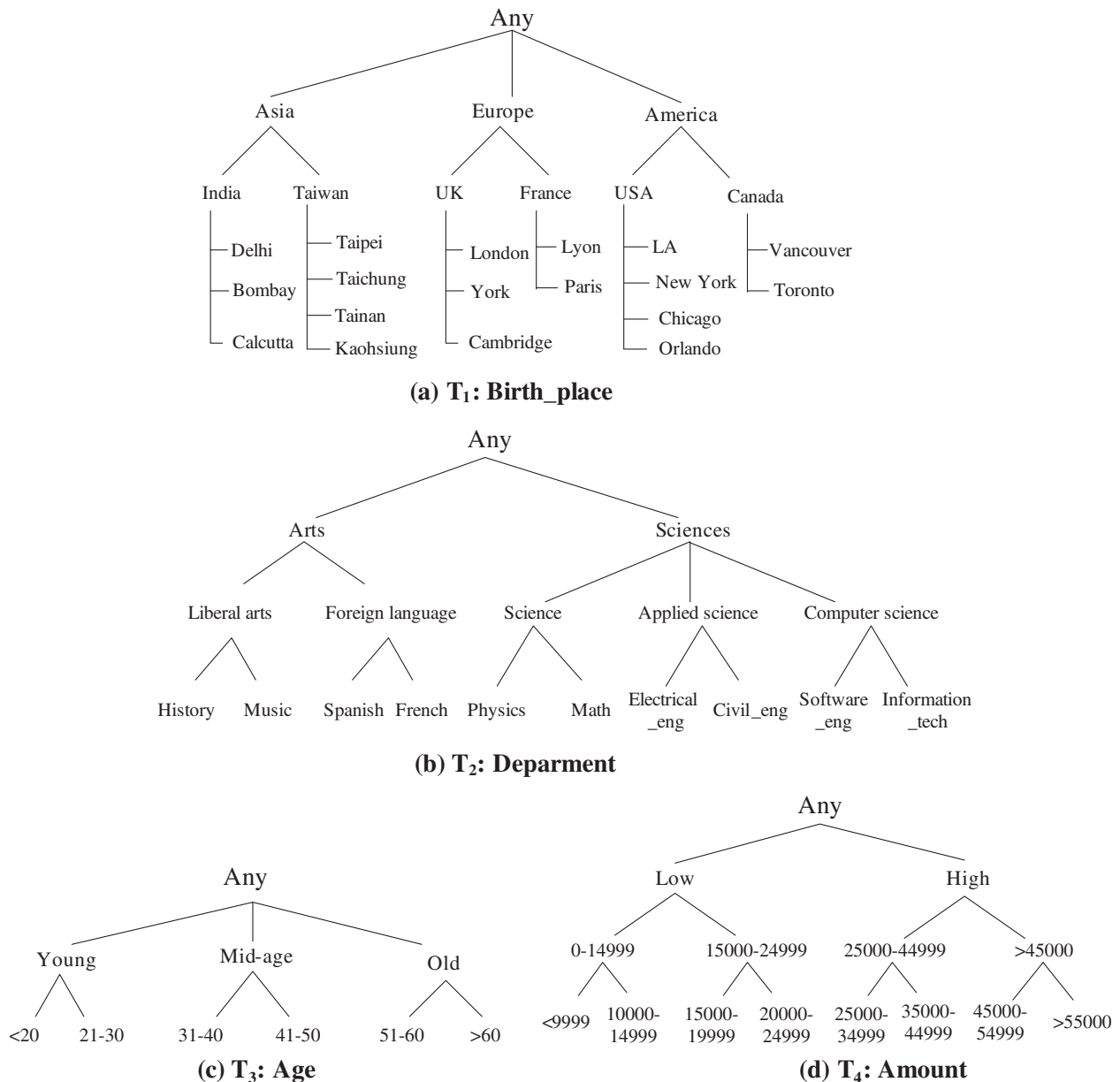
(a) T₁: Birth_place

(b) T₂: Deparment

(c) T₃: Age

(d) T₄: Amount

Fig. A1. Concept hierarchies for research grant data.

(a) T₁: Education

(b) T₂: Age

(c) T₃: Occupation

(d) T₅: Constellation

(e) T₄: Income (NT$)

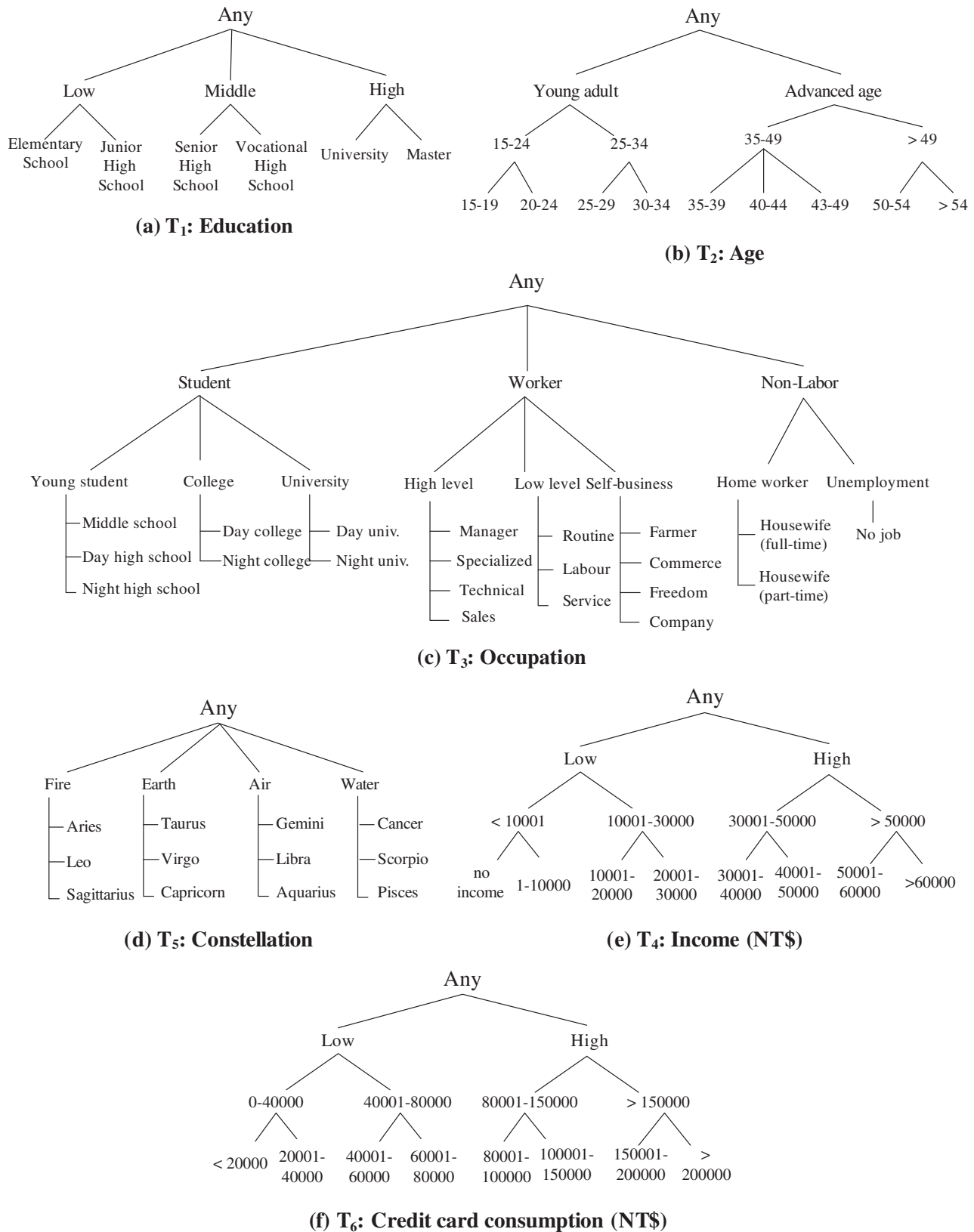**(f) T₆: Credit card consumption (NT$)**

Fig. A2. Concept hierarchies for credit card dataset.

number of negative generalized tuples decreases as the threshold values get larger. This result is quite reasonable, because the thresholds are used to filter out the valuesets; smaller thresholds naturally result in more valuesets.

As mentioned earlier, an interest measure is proposed to filter out uninteresting tuples. In the final experiment, we repeat 50 times to evaluate the influence of the interest measures. The experimental parameters are as follows: data size is fixed at 25000 tuples; number of attributes is 6; minValueSup is 10%; and minExpectedSup is 2%. The run time curve with different interest measures, from 1.1 to 1.8 is illustrated in Fig. 13. Fig. 14 shows the variation of the number of negative generalized tuples. From these figures, we can see that by setting an appropriate interest measure, one can easily filter out many uninteresting negative generalized tuples and reduce the computation time.

6. Conclusion

We have proposed an effective method, the GNAOI method, for generating all negative interesting generalized knowledge at one time. The method employs the expected support concept to filter out uninteresting knowledge. In addition, three constraints and two useful closure properties are designed to reduce the search space. The method provides a simple but efficient way to carry out negative knowledge generalization from a relational database. A real-life dataset is used for experiment and the results show that the proposed method can induce more comprehensive negative generalized knowledge from relational database. There are two possible directions for future research. First, in complicated business or distributed environment, there is an urgent need for mining negative generalized knowledge from multi-databases. The proposed GNAOI method can be integrated with existing multi-database mining method [39,40] to expand its application scope. Second, it would be interesting to extend the usage of background knowledge by using domain generalization graphs rather than concept trees or using fuzzy concept trees rather than crisp concept trees.

Appendix A. Concept hierarchies

Figs. A1 and A2.

References

[1] S.Y. Chen, X. Liu, The contribution of data mining to information science, Journal of Information Science 30 (6) (2004) 550–558.
[2] R.R. Yager, F.E. Petry, A multicriteria approach to data summarization using concept ontologies, IEEE Transactions on Fuzzy Systems 14 (6) (2006) 767–780.
[3] C. Chen, Z. Rao, Mrm: a matrix representation and mapping approach for knowledge acquisition, Knowledge-Based Systems 21 (4) (2008) 284–293.
[4] X. Wang, W. Zhang, Relations of attribute reduction between object and property oriented concept lattices, Knowledge-Based Systems 21 (5) (2008) 398–403.
[5] R.A. Angryk, Attribute-oriented defuzzification of fuzzy database tuples with categorical entries, Control and Cybernetics 38 (2) (2009) 419–453.
[6] W.J. Frawley, G. Piatetsky, C.J. Matheus, Knowledge Discovery in Database: An Overview, AAAI/MIT Press, Cambridge, MA, 1991.
[7] J. Han, M. Kamber, Data Mining: Concepts and Techniques, second ed., Morgan Kaufman, New York, 2006.
[8] Y. Cai, N. Cercone, J. Han, An attribute-oriented approach for learning classification rules from relational databases, in: Proceedings of the Sixth International Conference on Data Engineering, Washington, DC, 1990, pp. 281–288.
[9] M.S. Chen, J. Han, P.S. Yu, Data mining: an overview from a database perspective, IEEE Transactions on Knowledge and Data Engineering 8 (6) (1996) 866–883.
[10] J. Han, Y. Cai, N. Cercone, Data-driven discovery of quantitative rules in relational databases, IEEE Transactions on Knowledge and Data Engineering 5 (1) (1993) 29–40.
[11] E.M. Knorr, R.T. Ng, Extraction of spatial proximity patterns by concept generalization, in: Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996, pp. 347–350.
[12] W. Lu, J. Han, B.C. Ooi, Discovery of general knowledge in large spatial databases, in: Far East Workshop on Geographic Information Systems, Singapore, 1993, pp. 275–289.
[13] Q.-H. Liu, C.-J. Tang, C. Li, Q.-W. Liu, T. Zeng, Y.-G. Jiang, Traditional Chinese medicine prescription mining based on attribute-oriented relevancy induction, Journal of Computer Applications 27 (2) (2007) 449–452.
[14] S. Tsumoto, Knowledge discovery in clinical databases and evaluation of discovered knowledge in outpatient clinic, Information Sciences 124 (1–4) (2000) 125–137.
[15] S.O. Al-Mamory, H. Zhang, Intrusion detection alarms reduction using root cause analysis and clustering, Computer Communications 32 (2) (2009) 419–430.
[16] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Transactions on Information and System Security (TISSEC) 6 (4) (2003) 443–471.
[17] S.T. Li, L.Y. Shue, S.F. Lee, Business intelligence approach to supporting strategy-making of ISP service management, Expert Systems with Applications 35 (3) (2008) 739–754.
[18] J. Sun, H. Li, Data mining method for listed companies' financial distress prediction, Knowledge-Based Systems 21 (1) (2008) 1–5.
[19] J. Han, Y. Fu, Exploration of the power of attribute-oriented induction in data mining, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1996, pp. 399–421.
[20] Y.-L. Chen, C.-C. Shen, Mining generalized knowledge from ordered data through attribute-oriented induction techniques, European Journal of Operational Research 166 (1) (2005) 221–245.
[21] C.L. Carter, H.J. Hamilton, Efficient attribute-oriented generalization for knowledge discovery from large databases, IEEE Transactions on Knowledge and Data Engineering 10 (2) (1998) 193–208.
[22] D.W. Cheung, H.Y. Hwang, A.W. Fu, J. Han, Efficient rule-based attribute-oriented induction for data mining, Journal of Intelligent Information Systems 15 (2) (2000) 175–200.
[23] C.-C. Hsu, Extending attribute-oriented induction algorithm for major values and numeric values, Expert Systems with Applications 27 (2) (2004) 187–202.
[24] R.A. Angryk, F.E. Petry, Knowledge discovery in fuzzy databases using attribute-oriented induction, in: Foundations and Novel Approaches in Data Mining, 2006, pp. 169–196.
[25] K.M. Lee, Mining generalized fuzzy quantitative association rules with fuzzy generalization hierarchies, in: Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, 2001, pp. 2977–2982.
[26] G. Raschia, N. Mouaddib, Saintetiq: a fuzzy set-based approach to database summarization, Fuzzy Sets and Systems 129 (2) (2002) 137–162.
[27] X. Zhu, Z. Huang, Conceptual modeling rules extracting for data streams, Knowledge-Based Systems 21 (8) (2008) 934–940.
[28] Z. Xu, On multi-period multi-attribute decision making, Knowledge-Based Systems 21 (2) (2008) 164–171.
[29] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison Wesley, Boston, 2006.
[30] M.L. Antonie, O.R. Zaiane, Mining positive and negative association rules: an approach for confined rules, in: The 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, 2004, pp. 27–38.
[31] X. Wu, C. Zhang, S. Zhang, Efficient mining of both positive and negative association rules, ACM Transactions on Information Systems (TOIS) 22 (3) (2004) 381–405.
[32] P.-N. Tan, V. Kumar, J. Srivastava, Indirect association: mining higher order dependencies in data, in: Proceedings of the 4th European Conference of Principles and Practice of Knowledge Discovery in Databases, Lyon, France, 2000, pp. 632–637.
[33] W.G. Teng, M.J. Hsieh, M.S. Chen, On the mining of substitution rules for statistically dependent items, in: Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 2002, pp. 442–449.
[34] A. Savasere, E. Omiecinski, S. Navathe, Mining for strong negative associations in a large database of customer transactions, in: Proceedings of The Fourteenth International Conference on Data Engineering, Orlando, Florida, 1998, pp. 494–502.
[35] X. Yuan, B.P. Buckles, Z. Yuan, J. Zhang, Mining negative association rules, in: Seventh International Symposium on Computers and Communications, ISCC 2002, 2002, pp. 623–628.
[36] L. Geng, H.J. Hamilton, Interestingness measures for data mining: a survey, ACM Computing Surveys 38 (3) (2006) 1–32.
[37] J. Han, Y. Fu, Mining multiple-level association rules in large databases, IEEE Transactions on Knowledge and Data Engineering 11 (5) (1999) 798–805.
[38] M.J. Zaki, Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering 12 (3) (2000) 372–390.
[39] S. Zhang, X. Wu, C. Zhang, Multi-database mining, IEEE Computational Intelligence Bulletin 2 (1) (2003) 5–13.
[40] S. Zhang, C. Zhang, X. Wu, Knowledge Discovery in Multiple Databases, Springer, New York, 2004.