Mining Generalized Knowledge from Transaction Databases

Show-Jane Yen

Department of Computer Science and Information Engineering

Fu Jen Catholic University,

Taipei, Taiwan, R.O.C.

Email: sjyen@csie.fju.edu.tw

Abstract

Mining association rules is an important task for knowledge discovery. We can analyze past transaction data to discover customer behaviors such that the quality of business decision can be improved. Various types of association rules may exist in a large database of customer transactions. The strategy of mining association rules focuses on discovering large itemsets which are groups of items which appear together in a sufficient number of transactions.

In this paper, we propose a graph-based approach to discover generalized association rules from a large database of customer transactions. This approach is to construct an association graph to indicate the associations between items, and then traverse the graph to generate large itemsets. Empirical evaluations show that our algorithm outperforms other algorithms which need to make multiple passes over the database.

Keywords: Data Mining, Knowledge Discovery, Association Rule, Association Pattern, Association Graph

1. Introduction

An *association rule* [1, 2, 3, 5] describes the associations among items in which when some items are purchased in a transaction, the others are purchased too. In order to find association rules, we need to discover all *large itemsets* from a large database of customer transactions. A large itemset is a set of items which appears often enough within the same transactions, that is, an itemset that is contained in a number of transactions above a certain minimum threshold.

The following definitions are adopted from [1, 2]. A transaction t *supports* an item x if x is in t. A transaction t supports an itemset X if t supports every item in X. The *support for an itemset* is defined as the ratio of the total number of transactions, which support this itemset to the total number of transactions in the database. To make the discussion easier, occasionally, we also let the total number of transactions which support the itemset denote the support for the itemset. Hence, a large itemset is an itemset whose support is no less than a certain *user-specified minimum support*. An itemset of length k is called a *k-itemset* and a large itemset of length k a *large k-itemset*.

After discovering all large itemsets, the association rules can be generated as follows: If the large itemset $Y=I_1I_2...I_k$, $k \geq 2$, all rules that reference items from the set $\{I_1, I_2, ..., I_k\}$ can be generated. The antecedent of each of these rules is a subset X of Y, and the consequent Y-X. The *confidence* of $X \Rightarrow Y-X$ in database D is the probability that when itemset X occurs in a transaction in D, itemset Y-X also occurs in the same transaction. That is, the ratio of the support for itemset Y to the support for itemset X. A generated rule is an association rule if its confidence achieves a certain user-specified *minimum confidence*. Assuming {coffee, sugar, milk} is a large itemset, an example of an association rule is "95% of the transactions in which coffee and sugar are purchased, milk is purchased too." This rule can be specified as "coffee, sugar $\Rightarrow$ milk

95%." The antecedent of this rule consists of coffee and sugar, and the consequent consists of milk alone. The percentage 95% is the confidence of the rule.

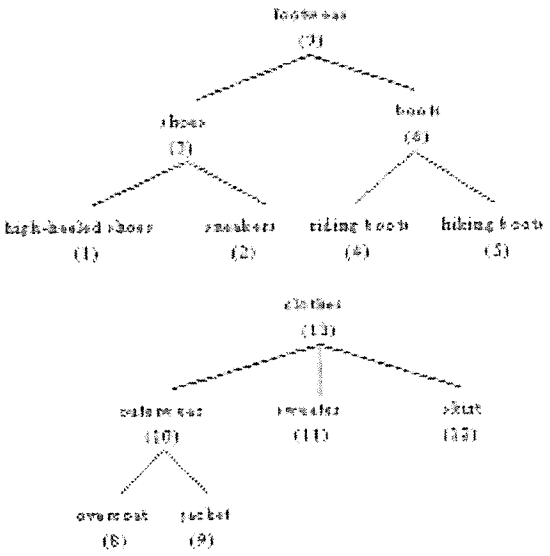

Figure 1: An example of concept hierarchies

In this paper, we propose a graph-based approach to discover generalized association rules. For the items which appear in the database, we call them the *database item*. A *concept hierarchy* of the items can usually be derived. An example of the concept hierarchy is shown in Figure 1, in which the terminal nodes are database items, and the nonterminal nodes are *generalized items*. If there is a path between nodes y and x, where y is a "higher concept" of x, then y is called an ancestor of x and x a descendant of y. In [4], the concept of "support" is extended such that a transaction supports an item x if x is in t or x is an ancestor of some items in t. Association rules may exist at higher concepts if the itemsets at the lower concepts cannot reach the minimum support. Hence, significant association rules may not be discovered if we only consider database items. A *generalized association rule* is introduced in [4], which describes the association among items which can be generalized or database items. A *generalized association pattern* is a large itemset in which each item is a generalized item or database item.

In our previous work [5], we proposed a graph-based approach to analyze a large amount of transaction data and to generate association rules. In this paper, we extend the graph-based approach to generate generalized association rules. We propose five phases to discover the association rules: 1. Numbering phase: in this phase, all itmes are assigned an integer number. 2. Large item generation phase: this phase generates large items and records related information. 3. Association graph construction phase: this phase constructs an association graph to indicate the associations between large items. 4. Association pattern generation phase: this phase generates all association patterns by traversing the constructed association graph. 5. Association rule generation phase: the association rules can be generated directly according to the corresponding association patterns.

This paper focuses on the association pattern generation, because after generating the association patterns, the association rules can be generated from the corresponding association patterns.

2. Mining Association Rules

An algorithm APG (Association Pattern Generation) is presented to generate association patterns, which is the same as the algorithm DLG [5]. In the following, we describe the four phases discussed in Section 1 for the algorithm APG.

2.1 Association graph construction

In the numbering phase, algorithm APG arbitrarily assigns each item a unique integer number. Suppose item i represents the item whose item number is i. In the large item generation phase, algorithm APG scans the database and builds a bit vector for each item. The length of each bit vector is the number of transactions in the database. If an item appears in the ith transaction, the ith bit of the bit vector associated with this item is set to 1. Otherwise, the ith bit of the bit vector is set to 0. The bit vector associated with item i is denoted as BV_i. The number of

1's in BV_i is equal to the number of transactions which support the item i, that is, the support for the item i.

Example 1: consider the database TDB1 in Table 1. Each record is a <TID, Itemset> pair, where TID is the identifier of the corresponding transaction, and Itemset records the items purchased in the transaction. Assume that the minimum support $\Im$ is 50% (i.e., 2 transactions).

TID	Itemset
100	C A D
200	E C B
300	A B C E
400	E B

Table 1: A database TDB1 of transactions

After the numbering phase, the numbers of the items A, B, C, D and E are 1, 2, 3, 4 and 5, respectively. In the large item generation phase, the large items found in the database TDB1 are items 1, 2, 3 and 5, and BV_1, BV_2, BV_3 and BV_5 are (1010), (0111), (1110) and (0111), respectively.

Property 2.1: The support for the itemset $i_1, i_2, ..., i_k$ is the number of 1's in $BV_{i1} \wedge BV_{i2} \wedge ... \wedge BV_{ik}$, where the notation "$\wedge$" is a logical AND operation.

In the association graph construction phase, APG constructs an association graph to indicate the associations between large items. For every two large items i and j (i < j) , if the number of 1's in $BV_i \wedge BV_j$ achieves the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset (i,j) is a large 2-itemset. Note that an ordered list notation is used to indicate the order of the items in an itemset for the following discussion.

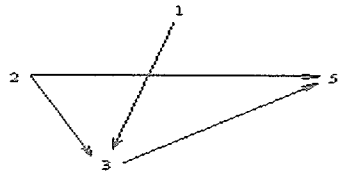

Figure 2. The association graph for Example 1

The association graph for the Example 1 is shown in Figure 2, and the large 2-itemsets are (1, 3), (2, 3), (2, 5) and (3, 5).

2.2 Association pattern generation

In the association pattern generation phase, the algorithm LGDE (Large itemset Generation by Direct Extension) is proposed to generate large k-itemsets (k>2). For each large k-itemset ($k \geq 2$), the last item of the k-itemset is used to extend the large itemset into k+1-itemsets.

Lemma 2.1: If an itemset is not a large itemset, then any itemset which contains this itemset cannot be a large itemset.

Rationale: Because the itemset is not a large itemset, the support for the itemset is less than the minimum support. Hence, the support for an itemset which contains this itemset must be also less than the minimum support.

Lemma 2.2: For a large itemset $(i_1, i_2, ..., i_k)$, if there is no directed edge from item i_k to an item v, then itemset $(i_1, ..., i_k, v)$ cannot be a large itemset.

Rationale: Because there is no directed edge from item i_k to an item v, the itemset (i_k, v) is not a large 2-itemset. Hence, by Lemma 2.1, itemset $(i_1, ..., i_k, v)$ is not a large itemset.

Suppose $(i_1, i_2, ..., i_k)$ is a large k-itemset. If there is no directed edge from item i_k to an item v, then the itemset need not be extended into k+1-itemset, because $(i_1, ..., i_k, v)$ must not be a large itemset according to Lemma 2.2. If there is a directed edge from item i_k to an item u, then the itemset $(i_1, i_2, ..., i_k)$ is extended into k+1-itemset $(i_1, i_2, ..., i_k, u)$. The itemset $(i_1, i_2, ..., i_k, u)$ is a large k+1-itemset if the number of 1's in $BV_{i1} \wedge BV_{i2} \wedge ... \wedge BV_{ik} \wedge BV_u$ achieves the minimum support. If no large k-itemsets can be generated, the algorithm LGDE terminates.

For example, consider Example 1. For the large 2-itemset (2, 3), there is a directed edge from the last item 3 of the itemset (2, 3) to item 5 in the association graph shown in Figure 2. Hence, the 2-itemset (2, 3) is extended into 3-itemset (2, 3, 5). The number of 1's in $BV_2 \wedge BV_3$

$\wedge$ BV_5 (i.e., (0110)) is 2. Hence, the 3-itemset (2, 3, 5) is a large 3-itemset, since the number of 1's in its bit vector is no less than the minimum support threshold. The LGDE algorithm terminates because no large 4-itemsets can be further generated.

3. Mining Generalized Association Rules

We propose the algorithm GAPG (Generalized Association Pattern Generation) to discover all generalized association pattern. In the following, we also describe the four phases for algorithm GAPG.

3.1 A numbering method

To generate generalized association patterns, one can add all ancestors of each item in a transaction to the transaction and then apply the algorithm APG on the extended transactions. However, because if an item is a large item, then the 2-itemset which contains the item and its ancestor is also a large 2-itemset, the number of the edges in the association graph can be very large, and LGDE algorithm needs to take much more time to traverse the association graph to generate all large itemsets.

Lemma 3.1: [4] The support for an itemset X that contains both an item x_i and its ancestor χ_i will be the same as the support for the itemset X-χ_i.

Rationale: Suppose the itemset $X=(x_1, ..., x_i, \chi_i, x_{i+1}, ..., x_n)$. The support for itemset X is the number of 1's in $BV_{x1} \wedge ... \wedge BV_{xi} \wedge BV\chi_i \wedge BV_{xi+1} \wedge ... \wedge BV_{xn}$, and the support for itemset X-χ_i is the number of 1's in $BV_{x1} \wedge ... \wedge BV_{xi} \wedge BV_{xi+1} \wedge ... \wedge BV_{xn}$ according to Property 2.1. Because the set of the transactions which contain an item x_i is the subset of the set of the transactions which contain the ancestor x_i of item x_i, $BV_{xi} \wedge BV\chi_i = BV_{xi}$. Hence, the support for X is the same as the support for the itemset X-χ_i.

From Lemma 3.1, when an itemset X contains both an item x and its ancestor χ, if the itemset X-χ is a large itemset, then itemset X is also a large itemset. Lemma 3.1 can be employed to reduce the cost for large itemset generation. Hence, the problem of mining generalized association patterns becomes to find all generalized association patterns which do not contain both an item and its ancestor.

In the numbering phase, GAPG algorithm applies numbering method PON (POstorder Numbering method) to number items at the concept hierarchies. For each concept hierarchy, PON numbers each item at the concept hierarchy according to the following order: for each item at the concept hierarchy, after all descendants of the item are numbered, PON numbers this item immediately, and all items are numbered increasingly. After all items at a concept hierarchy are numbered, PON numbers items at another concept hierarchy.

Lemma 3.2: If the numbering method PON is adopted to number items, and for every two items i and j (i<j), item ϑ is an ancestor of item i but not an ancestor of item j, then ϑ < j.

Rationale: According to PON numbering method, after all descendants of an item are numbered, this item is numbered immediately, and these items are numbered increasingly. Hence, for an item i, if it is numbered, then its ancestor ϑ must be numbered before the other item j which is not a descendant of item ϑ is numbered. So, ϑ < j.

Example 2: Consider the database TDB2 in Table 2 and the concept hierarchies in Figure 1. Assume that the minimum support $\Im$ is 40% (i.e., 2 transactions).

TID	Itemset
100	{high-heeled shoes, riding boots, overcoat, sweater, skirts}
200	{high-heeled shoes, sneakers, skirts}
300	{high-heeled shoes, skirts}
400	{hiking boots, jackets, sweater}
500	{overcoat, sweater}

Table2: A database TDB2 of transactions

After applying PON method on the concept hierarchies in Figure 1, all items at the concept hierarchies

are numbered, where the number within the parentheses below each item in Figure 1 is the number of the item.

3.2 Large item generation

In the large item generation phase, GAPG algorithm builds a bit vector for each database item, and finds all large items (include database items and generalized items).

Lemma 3.3: Suppose database items i_1, i_2, ..., and i_m are all descendants of the generalized item i_n. The bit vector BV_{in} associated with item i_n is $BV_{i1} \vee BV_{i2} \vee ... \vee BV_{im}$, and the number of 1's in $BV_{i1} \vee BV_{i2} \vee ... \vee BV_{im}$ is the support for item i_n, where the notation "$\vee$" is a logical OR operation.

Rationale: If item x is in a transaction, then its ancestor χ is also in the transaction. Because item i_n is the common ancestor of databaseitems i_1, i_2, ..., and i_m which are all descendants of item i_n , the transactions which contain items i_1, i_2, ..., or i_m also contain item i_n. Hence, the bit vector BV_{in} associated with item i_n is $BV_{i1} \vee BV_{i2} \vee ... \vee BV_{im}$.

From Lemma 3.3, the bit vector associated with a generalized item is obtained by performing logical OR operations on the bit vectors associated with the database items which are all descendants of the generalized item. For Example 2, in the large item generation phase, the bit vectors associated with database items 1, 2, 4, 5, 8, 9, 11 and 12 are built. The items 8, 9, 11 and 12 are all descendants of the generalized item 13. Hence, $BV_{13}=BV_8 \vee BV_9 \vee BV_{11} \vee BV_{12}$ according to Lemma 3.3.

3.3 Generalized association graph construction

In the association graph construction phase, GAPG constructs a *generalized association graph* to be traversed. The method to construct a generalized association graph is similar to the method to construct an association graph. For every two large items i and j (i<j), if item j is not an ancestor of item i and the number of 1's in $BV_i \wedge BV_j$ achieves the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset (i, j) is a large 2-itemset.

Lemma 3.4: If an itemset X is a large itemset, then any itemset generated by replacing an item in itemset X with its ancestor is also a large itemset.

Rationale: Because itemset X is a large itemset, the support for itemset X is no less than the minimum support. Suppose X' is an itemset generated by replacing an item in itemset X with its ancestor. Since the set of the transactions which contain an item is the subset of the set of the transactions which contain an ancestor of the item, the support for itemset X' is no less than the support for itemset X. Hence, the itemset X' is also a large itemset.

Lemma 3.5: If (the number of 1's in $BV_i \wedge BV_j$) $\geq$ minimum-support, then for each ancestor u of item i and for each ancestor v of item j, (the number of 1's in $BV_u \wedge BV_j$) $\geq$ minimum-support and (the number of 1's in $BV_i \wedge BV_v$) $\geq$ minimum-support.

Rationale: Because the number of 1's in $BV_i \wedge BV_j$ is no less than the minimum support, the itemset (i, j) is a large itemset. Because item u is an ancestor of item i and item v is an ancestor of item j, according to Lemma 3.4, itemsets (u, j) and (i, v) are also large itemsets.

From Lemma 3.5, if an edge from item i to item j is created, the edges from item i to the ancestors of item j, which are not ancestors of item i, are also created. According to Lemma 3.2, the numbers of the ancestors of item i, which are not the ancestors of item j, are all less than j. Hence, if an edge from item i to item j is created, the edges from the ancestors of item i, which are not ancestors of item j, to item j are also created.

For example, consider Example 2. In the association graph construction phase, because the number of 1's in $BV_1 \wedge BV_{12}$ is 3 ($\geq$ 2), the 2-itemset (1, 12) is a large 2-itemset, and a directed edge from item 1 to item 12 is created. Items 3 and 7 are ancestors of item 1 but not ancestors of item 12, and item 13 is an ancestor of item 12 but not ancestor of item 1. Hence, the directed edges from items

3 and 7 to item 12 and from item 1 to item 13 are also created.

The algorithm GAGC (Generalized Association Graph Construction) is proposed to construct a generalized association graph, which is described as follows:

```
for every two large items i and j (i < j) do
  if item j is not an ancestor of item i and
    there is no directed edge from item i to item j then
      if (the number of 1's in BV_i ∧ BV_j) ≥
          minimum-support then
      begin
        create a directed edge from item i to item j
        generate a large 2-itemset (i, j)
        foreach ancestor ς of item j do
          if ς is not an ancestor of item i then
          begin
            create a directed edge from item i to item ς
            (Lemma 3.5)
              generate a large 2-itemset (i, ς)
          end
          foreach ancestor ϑ of item i do
            if ϑ is not an ancestor of item j then
            begin
              create a directed edge from item ϑ to item j
                (Lemmas 3.1 and 3.5)
              generate a large 2-itemset (ϑ, j)
            end
      end
```

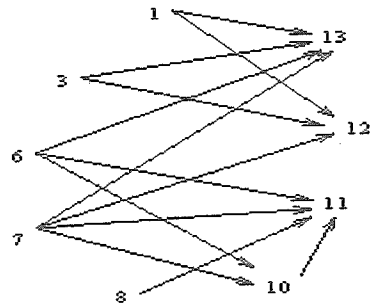

Figure 3. The generalized association graph for Example 2

After applying GAGC algorithm in the association graph construction phase, the generalized association graph for Example 2 is constructed in Figure 3 where there are no edges between an item and its ancestors.

3.4 Generalized association pattern generation

In the association pattern generation phase, GAPG applies LGDE algorithm to generate all generalized association patterns by traversing the generalized association graph.

Theorem 3.1: If the numbering method PON is adopted to number items and the algorithm GAGC is applied to construct a generalized association graph, then any itemset generated by traversing the generalized association graph (i.e., performing LGDE algorithm) will not contain both an item and its ancestor.

Proof: We use mathematical induction to prove this theorem.

Basis of induction: By GAGC algorithm, because there is no edge between an item and its ancestor, any large 2-itemset does not contain an item and its ancestor.

Inductive hypothesis: We assume that any large k-itemset $(i_1, i_2, ..., i_k)$ does not contain both an item and its ancestor.

Inductive step: Suppose there is a directed edge from item i_k to item w in the generalized association graph constructed by applying GAGC algorithm. By LGDE algorithm, the large k-itemset $(i_1, i_2, ..., i_k)$ is extended into k+1-itemset $(i_1, i_2, ..., i_k, w)$. Suppose items $\vartheta_1, \vartheta_2, ...$ and ϑ_{k-1} are the ancestors of items $i_1, i_2, ...$ and i_{k-1}, respectively, but none are ancestors of item i_k. Because items are numbered by PON method, $i_k > \vartheta_j$ $(1 \le j \le k-1)$ (Lemma 3.2). Hence, there are no edges from item i_k to the ancestors of items $i_1, i_2, ...$ and i_k. So, item w cannot be an ancestor of item $i_1, i_2, ...$ or i_k.

4. Performance Evaluation

In this section, we evaluate the performance of the two algorithms APG and GAPG which are proposed to discover the two types of association patterns: primitive association patterns, generalized association patterns, respectively. In [5], we have evaluated the performance of APG (called DLG in [5]) and demonstrated that APG has a better performance than other approaches [1, 2, 3]. In the following we analyze the performance of algorithm GAPG. We evaluate the performance of GAPG algorithm by comparing this algorithm with the algorithm Cumulate [4].

We first generate the synthetic data for the experiment by applying the method described in [4]. The parameters for the synthetic data are set as follows: the number of transactions is 100,000, the number of items is 100,00, the number of concept hierarchies is 100, the fanout is 5, the

number of the potentially large itemsets is 5000, the average size of the transactions is 10 and the average size of the potentially large itemsets is 5. Figure 4 shows the relative execution time for Cumulate algorithm and GAPG algorithm over various minimum supports, ranging from 0.5% to 3.5%.

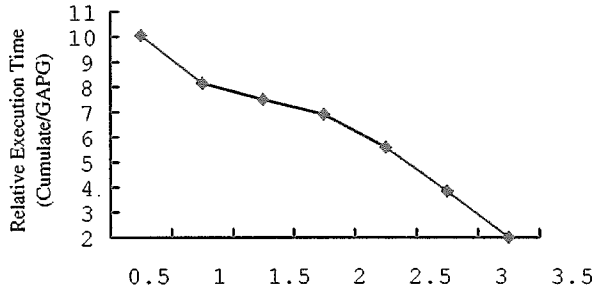

Figure 4: Relative Execution Time

Suppose in the kth iteration, the set GL_k of the large k-itemsets is generated. In the first iteration, Cumulate algorithm scans the database to add the ancestors of each item in a transaction to the transaction, and count the support for each item in the extended database. Suppose the average number of the ancestors of each item is m. The number of the items in an extended transaction will be (m-1) times larger than the number of the items in the original transaction. Hence, the size of the extended database will also be (m-1) times larger than the size of the original database. It is very costly to scan so large database to generate large items.

For GAPG algorithm, it first applies PON method to number items at the concept hierarchies. Suppose there are n items in the concept hierarchies. The time complexity to number all items is O(n). After numbering all items in the concept hierarchies, GAPG scans the database to count the support and build a bit vector for each item in the database. The support for the generalized items can be obtained by performing logical OR operations on the bit vectors associated with some specific items. Hence, in the first iteration, the two algorithms, Cumulate and GAPG, takes a similar time to generate large items.

In the second iteration, Cumulate algorithm generates candidate 2-itemsets by combining every two large items and deletes any candidate 2-itemset that consists of an item and its ancestor. For these remaining candidate 2-itemsets, Cumulate adds the ancestors of each item in a transaction, which are present in any of the candidates, to the transaction and count the support for each candidate 2-itemset by scanning each extended transaction. However, the number of the candidate 2-itemsets to be counted and the size of the extended database both are still very large. It is very time consumming to search such a large number of candidates and scan the large extended database.

For GAPG algorithm, it applies GAGC algorithm to construct a generalized association graph. Suppose the average number of the ancestors of each large item is p. GAGC algorithm at most needs to perform $|GL_1|(|GL_1|-1)/2-|GL_1| \times p$ logical AND operations on the bit vectors to construct a generalized association graph and generate large 2-itemsets.

In the kth (k>2) iteration, Cumulate generates candidate k-itemsets by applying join-based algorithm [2]. After generating candidate k-itemsets, Cumulate adds the ancestors of each item in a transaction, which are present in any of the candidate k-itemsets, to the transaction and count the support for each candidate k-itemset by scanning each extended transaction. Hence, the execution time of Cumulate depends on the number of generated candidate itemsets and the amount of data that has to be scanned.

For GAPG algorithm, it applies LGDE algorithm to generate large k-itemsets. LGDE extends each large k-1-itemset into k-itemsets according to the generalized association graph and performs logical AND operations. Suppose the average out-degree of each item in the generalized association graph is q. LGDE performs $(k-1) \times |GL_{k-1}| \times q$ logical AND operations to find all large k-itemsets. Hence, as the minimum support decreases, the number of logical AND operations performed increases

because the two values $|GL_{k-1}|$ and q increase.

Since the number of the candidate itemsets to be counted and the size of the extended database to be scanned by Cumulate algorithm are much larger than the logical AND operations performed by GAPG algorithm, and GAPG needs only one database scan but Cumulate needs to scan the extended database in each iteration, GAPG always outperforms Cumulate for various minimum supports which is shown in Figure 4. Figure 5 shows the relative execution time for Cumulate and GAPG for various database sizes, in which the minimum support is set to 1%.

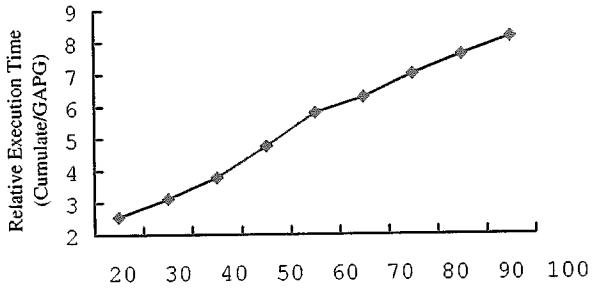

Figure 5. Relative Execution Time

The GAPG algorithm outperforms the Cumulate algorithm significantly, and the performance gap increases as the minimum support decreases or the database size increases because the number of candidate itemsets and the number of database scans increases for Cumulate.

5. Conclusion and Future Work

We propose a graph-based approach to discover association rules and generalized association rules. The approach includes the five phases: numbering phase, large item generation phase, association graph construction phase, association pattern generation phase and association rule generation phase.

We present the two algorithms: APG and GAPG to generate association patterns and generalized association patterns, respectively. In [11], the algorithm PAPG has been demonstrated to have a better performance than other approaches [2, 3, 5]. In this paper, we compare GAPG algorithm to the previously known algorithm Cumulate [7], respectively. The experimental results show that GAPG outperform Cumulate. When the minimum support decreases or the size of the database increases, the performance gap increases because the number of candidate itemsets generated by GAPG increases and the number of database scans also increases.

For our approach, the related information may not fit in the main memory when the size of the database is very large. In the future, we shall consider this problem by reducing the memory space requirement. Also, we shall apply our approach on different applications, such as document retrieval and resource discovery in the world-wide web environment.

Acknowledgement

This work was partially supported by the Republic of China National Science Council under Contract No. NSC 89-2213-E-030-003.

Reference

[1] Agrawal R. and et al., "Mining Association Rules Between Sets of Items in Large Databases", *Proceedings of ACM SIGMOD*, pp.207—216, 1993.

[2] Agrawal R. and Srikant R., "Fast Algorithm for Mining Association Rules", *Proceedings of Very Large Data Bases*, pp.487-499, 1994.

[3] Park J.S., Chen M.S. and Yu P.S., "An Effective Hash-Based Algorithm for Mining Association Rules", *Proceedings of ACM SIGMOD*, Vol.24, No.2, pp.175--186, 1995.

[4] Srikant R. and Agrawal R., "Mining Generalized Association Rules", *Proceedings of Very Large Data Bases*, pp.407--419, 1995.

[5] Yen S.J. and Chen .L.P., "An Efficient Approach to Discovery Knowledge from Large Databases", *Proceedings of Parallel and Distributed Information Systems*, pp.8--18, 1996.