Using a Lattice Intension Structure to Facilitate User-Guided Association Rule Mining

Abdallah Alashqur
Applied Science University, Shafa Badran, Amman, Jordan
E-mail: alashqur@asu.edu.jo

Received: December 27, 2011 Accepted: January 9, 2012 Published: March 1, 2012
doi:10.5539/cis.v5n2p11 URL: http://dx.doi.org/10.5539/cis.v5n2p11

Abstract

Narrowing down the computational space is a key factor in improving the efficiency of an association rule mining system. One approach to achieve this is to let the user guide the association rule mining process by enabling the user to specify the types of association rules that he/she might be interested in. Instead of computing *all* that can be computed, the system limits its association rule mining process to the discovery of only the association rules that may be of interest to the user, therefore, reducing the computational space and complexity. In this paper, we introduce a new approach for achieving this by using a new structure called *lattice intension* structure.

Keywords: Association rules, Data mining, Frequent itemsets, Lattice traversal

1. Introduction

Mining association rules is a complex, time-consuming process. One of the reasons behind this complexity can be attributed to the fact that the mining process aims at finding *all* possible association rules that satisfy minimum support and confidence values in a given database. Since the number of association rules is potentially extremely large, this results in poor system performance. Numerous algorithms have been introduced in the literature for improving the performance of association rules mining (Wang, 2005; Xin, 2005; Rozenberg, 2006; Nadimi-Shahraki, 2008; Lucchese, 2006; Moonestinghe, 2006; Ma, 2008; Liu, 2003; Gouda, 2005). This paper presents a new approach for association rule mining, where we introduce a new structure called *lattice intension* structure that enables the user to guide the association rule mining process. The lattice intension structure gives a user the option to determine the *type* of association rules that are of interest. Instead of trying to discover *all* possible association rules, the system restricts its discovery process to only the association rules that are of interest to the user.

Briefly, our approach works as follows. Each node in the lattice intension structure represents a combination of attributes. The lattice intension structure helps the user in selecting a specific attribute combination that is of interest by selecting a node from the structure. The system uses the selection made by the user as a guiding directive. As a result, the system restricts its processing to the attribute combination selected by the user. Furthermore, when a user selects a node representing an attribute combination, the underlying system can present the user with the list of possible association rule intensions that involve these attributes, and the user may choose to further reduce the computational space by selecting one or more of the association rule intensions that appear on the list. The role of the data mining process becomes, simply, to find the association rules that are of interest to the user and that satisfy a minimum support and confidence values if the user provides such minimums.

This paper builds on our earlier work presented in (Alashqur, 2008) and (Alashqur, 2010). In (Alashqur, 2008), we introduced the concepts of *itemset intension* and *association rule intension* and distinguished them from *itemset extension* and *association rule extension.* In this paper, we introduce a new structure called *lattice intension* structure along the lines of itemset intension and association rule intension. Furthermore, we describe how the lattice intension structure is used to enable the user to guide the data mining process. In (Alashqur, 2010), we introduced an algorithm that is based on SQL and that makes use of the itemset intension and extension structures for mining relational databases and discovering *all* frequent itemsets. Based on our approach described in (Alashqur, 2010), in this paper we demonstrate how SQL can be used in association rule mining in the case when the user is given the ability to guide the mining process and restrict it to the discovery of some (instead of all) itemsets. In addition, we introduce an approach in which SQL can be used to compute association rules and their confidence in addition to computing itemsets.

The remainder of this paper is organized as follows. In section 2, we briefly summarize the definitions of itemset intension and extension and association rule intension and extension, to provide the necessary background for what is covered in the remaining sections of this paper. The detailed description of such structures can be found in (Alashqur, 2008). In Section 3, we introduce the concept of *lattice intension* and distinguish it from *lattice extension*. We show in Section 4 how a lattice intension structure can be used to facilitate user-guided association rule mining and how the system can narrow-down its SQL-based computations depending on the user's guidance. Conclusions are given in Section 5.

2. Background

In this section, we provide the background necessary for introducing the material covered in Section 3 and Section 4. We briefly describe the concepts of itemset extension, itemset intension, association rule extension, and association rule intension. A detailed description can be found in (Alashqur, 2008). The relation shown in Table 1, which is extracted from Alashqur (Alashqur, 2008), is used as a basis to describe these concepts.

The relation of Table 1 contains data pertaining to ex-members of a gym club, which represents the data that is kept in the database for members who terminate their membership. This data includes AGE, GENDER, MEMBERSHIP_DURATION (how long a member maintained a valid membership in the club), HOME_DISTANCE (how far a member's residence is from the club location), and HOW_INTRODUCED (how a member was originally introduced to the club such as by referral or by seeing an advertisement in a newspaper). Table 1 shows this relation as populated with sample data. In real life situations, a large club, with many branches, may have millions of ex-members, thus millions of tuples may exist in such a relation.

2.1 Itemsets

In our approach, we define an itemset as a set of items such that no two items belong to the same attribute (i.e, no two items are drawn from the same attribute domain, where a domain represents the set of valid values defined for an attribute). For example, in Table 1, {m, short, far} is a valid itemset (IS) while {m, short, far, close} is not a valid IS since 'far' and 'close' are two items that belong to the same attribute, which is HOME_DISTANCE. Stated formally, the following is the definition of a valid itemset.

$$\{I_1, I_2, \ldots I_n\} \text{ is valid IS iff } (\neg\exists I_j)\,(\neg\exists I_k)\,(j \neq k \wedge (\mathrm{Attr}\,(I_j) = \mathrm{Attr}\,(I_k))$$

Where *I* is an item from the relation (i.e. an attribute value) and *Attr (I)* is a function that returns the attribute name of item *I*. Logical AND is represented by "$\wedge$".

In Table 1, the domains of attributes are assumed, for simplicity, to be mutually exclusive. If these domains are not mutually exclusive, then one must qualify attribute values by their attribute names. Therefore, in this case, an itemset like {short, news_paper} needs to be written as {MEMBERSHIP_DURATION.short, HOW_INTRODUCED.news_paper}. Note that, for clarity, throughout this paper, we use upper case letters for attribute names and lower case letters for attribute values. An itemset that contains k items is referred to as *k-itemset*.

The interestingness of an itemset is measured by the percentage of tuples in the relation that contain the itemset. This measure is referred to, in data mining literature, as *support*. In other words, the support is the probability *P* that the itemset exists in the relation.

$$Support(itemset) = P(itemset) = \frac{Num\ of\ tuples\ containing\ itemset}{Total\ Number\ of\ tuples} \times 100$$

The *support count,* on the other hand, is the absolute number of occurrences of an itemset in the relation.

As an example, based on the database state shown in Table 1, the *support count* of the 3-itemset {young, f, referral} is 1 since there is only one tuple that contains this itemset. Its support = (1/13) X 100 = 7.7%. The support can be zero in case if the itemset does not exist at all in the relation, such as {m, long}. Normally, the user of a data mining tool supplies a minimum support value *minsup*. The data mining tool then finds itemsets whose support is equal to or greater than *minsup*. Itemsets that satisfy the minimum support are referred to as *frequent* itemsets.

2.2 Itemset Intension

Following the terminology of the relational data model, the definition of *itemset intension* (*ISI*) was introduced in (Alashqur, 2008), and is summarized here. An *itemset intension* (*ISI*) is a subset of the attributes of a relation. For example, in Table 1, {HOME_DISTANCE, MEMBERSHIP_DURATION} is an *ISI*. The itemsets that consist of actual attribute values belonging to these two attributes are instantiations of this itemset intension and are referred to as *itemset extensions* or simply *itemsets* (itemsets are described in Section 2.1). In Table 1, the itemsets that are instantiations of the *ISI* {HOME_DISTANCE, MEMBERSHIP_DURATION} are as follows:

{close, long}, {far, long}, {close, short}, {far, short}.

An itemset *IS* is said to be an instantiation of an itemset intension *ISI* if the cardinality of *IS* is the same as the cardinality of *ISI* and each item in *IS* is drawn from a domain of an attribute in *ISI*. Let the symbol "$\sqsubset$" *denote* "instantiation of" and let *CAR (S)* be a function that returns the cardinality of set S. We formally define the relationship between an itemset and its itemset intension as follows.

$$\text{IS} \sqsubset \text{ ISI iff CAR (IS)} = \text{CAR (ISI) AND } (\forall I_j \in \text{IS}) (\text{Attr} (I_j) \in \text{ISI})$$

"I" is an item in the itemset *IS* and *Attr (I)* returns the attribute name of item I. Note that the formal definition of *itemset*, as described in Section 2.1, prevents any two values in an itemset from belonging to the same attribute.

2.3 Association Rules

The association patterns among attribute values can be represented as association rules, where an association rule is an implication of the form:

$$\text{lhs} \rightarrow \text{rhs},$$

Each of the left had side (lhs) and right hand side (rhs) is a set of attribute values, provided that no attribute value exists in both lhs and rhs, i.e.,

$$lhs \cap rhs = \Phi \text{ .}$$

For instance, {referral} → {long} is an association rule relating the attribute value EMBERSHIP_DURATION.long to the attribute value HOW_INTRODUCED.referral. Each association rule has two metrics to measure its interestingness, *support* and *confidence*. The support of an association rule is the support of the itemset that contains all items in the rule, that is, the itemset containing the union of the items of the *lhs* and *rhs*. In other words,

$$\text{Support (lhs} \rightarrow \text{rhs)} = \text{support (lhs} \cup \text{rhs)} = \text{P (lhs} \cup \text{rhs)}$$

where P denotes *probability*. As an example, to find the support of the rule {referral} → {long}, we note that 5 out of 13 tuples in the relation of Table 1 contain both referral and long, therefore,

$$\textit{Support} \text{ (referral} \rightarrow \text{long)} = \textit{Support} \text{ \{referral, long\}} = (5/13) \text{ X } 100 = 38.5\%$$

We define the *confidence* of the rule (*lhs* → *rhs*) as the percentage of tuples that contain *rhs* from those that contain *lhs*. In other words, confidence is the conditional probability *P(rhs | lhs)*. *Confidence* can be expressed in terms of support as follows:

$$\textit{Confidence} \ (lhs \rightarrow rhs) = \frac{support(lh\ s \cup rhs)}{support(lh\ s)} \times 100$$

In addition to specifying a *minsup*, a *minconf* (minimum confidence) can also provided to the data mining process, which then discovers association rules that satisfy *minsup* and *minconf*.

2.4 Association Rule Intension

In addition to introducing the concept of Itemset Intension in (Alashqur, 2008), we also introduce the concept of Association Rule Intension. Association Rule intension is a rule template that is shared by multiple association rules. Similar to an itemset intension, an association rule intension is expressed in terms of attribute names instead of actual data values. For example,

AGE → MEMBERSHIP_DURATION

is an association rule intension. The following six association rules are possible instantiations of the above rule intension.

young → long	young → short	middle → long
middle → short	senior → long	senior → short

Generally, an association rule intension can be written as *LHS*$\rightarrow$ RHS where each of *LHS* and RHS represents a set of attribute names (hence, they are written in upper-case letters), provided that $LHS \cap RHS = \Phi$. An association rule of the form *lhs* $\rightarrow$ *rhs* (written in lower case letters) is said to be an *instantiation of* ($\sqsubset$) an association rule intension of the form $LHS \rightarrow RHS$ *if* $lhs \sqsubset LHS$ *AND* $rhs \sqsubset RHS$ (the symbol "$\sqsubset$" which stands for "instantiation of" is described in Section 2.2). In this case, we say that $(lhs \rightarrow rhs) \sqsubset (LHS \rightarrow RHS)$. In other words,

$$(\text{lhs}\rightarrow \text{rhs}) \sqsubset (\text{LHS} \rightarrow \text{RHS}) \text{ iff } (\text{lhs} \sqsubset \text{LHS}) \wedge (\text{rhs} \sqsubset \text{RHS})$$

3. Lattice Intension vs. Lattice Extension

In this section, we introduce two new lattice structures that we refer to as *lattice intension* and *lattice extension*, along the lines of itemset intension and itemset extension, which are described in Section 2.

A lattice intension is simply a lattice structure that represents all possible combinations of attributes in a given relation, where each node in the lattice structure represents an attribute combination.

Table 2 shows a sample relation R with three attributes X, Y, and Z along with an ID attribute that serves as a key of the relation. If we assume that the ID attribute is a sequential number, it makes sense to exclude it from any itemset and association rule computations. Therefore, the attributes to be considered for computing the support of itemsets are X, Y, and Z. We represent the different possible combinations of these attributes in the form of a lattice structure as shown in Figure 1. This lattice structure consists of combinations of attribute names, therefore we refer to it as *lattice intension*. The order of attributes in a lattice intesion structure is not important. On the other hand, we refer to a lattice structure that consists of attribute values as *lattice extension*.

The lowest level in Figure 1 contains combinations of attributes consisting of only one attribute, the next level contains combinations consisting of two attributes, and so on. Each node in the lattice intension represents an *itemset intension* (see Section 2 for a description or *itemset intension*)

The number of levels in a lattice intension is equal to the number of attributes. The size of the lattice intension (i.e., the total number of nodes where each node represents an itemset intension) grows exponentially with the number of attributes that are considered. Let *N* be the number of itemset intensions (i.e., number of nodes) in a lattice intension, then

$$N = (2^A - 1)$$

where *A* represents the number of attributes. In the above lattice intension, there are three attributes X, Y, and Z. Therefore the number of itemset intensions is:

$$N = 2^3 - 1 = 7$$

Figure 2 shows the lattice intension representation for a relation that has five attributes A, B, C, D, and E. The number of nodes in this case is:

$$N = 2^5 - 1 = 31$$

We define a *lattice extension* as an instantiation of a lattice intension, just like an itemset extension is considered an instantiation of an itemset intension as described in Section 2. In other words, the nodes of a lattice extension consist of itemsets as opposed to itemset intensions. A set of attributes can be represented by only one lattice intension but may have multiple lattice extensions. A lattice extension can be obtained by substituting the data items (attribute values) for the attribute names in a lattice intension. The number of lattice extensions depends on the number of tuples in the relation.

Figure 3 shows a lattice extension based on the data of Table 2. This lattice extension is obtained by substituting the values x_5, y_3, and z_3 for the attributes X, Y, and Z in the lattice intension of Figure 1. The number shown next to the colon at each node represents the support count of the itemset based on the data of Table 2.

Similarly Figure 4 shows another lattice extension based on the data of Table 2. This lattice extension is obtained by substituting x_3, y_1, and z_1 for the attributes X, Y, and Z in the lattice intension of Figure 1. The zeroes next to the colon shown on some nodes of the lattice extension of Figure 4 indicate that the number of tuples in the relation of Table 2 that contain the itemset extension is zero. For example, there are no tuples in Table 2 that contain <x3, z1>.

The number of lattice extensions can be computed if we know the number of distinct values of each of the relation's attributes. Let R be a relation with n attributes *A1, A2, ... An*. Let the number of distinct values of attribute Ai be denoted by |*Ai*|. The number of lattice extensions (*NLE)* can be computed as follows.

$$NLE = |A1| * |A2| .... * |An| = \prod_{i=1}^{i=n} |Ai|$$

To compute the number of lattice extensions for the relation R of Table 2, we note that there are three distinct values of attribute Z, namely z1, z2, and z3. Similarly, there are four distinct values of attribute Y and six distinct values of attribute X. Substituting in the above equation, we can compute the total number of lattice extensions for the data of Table 2.

$$NLE = 6 * 4 * 3 = 72$$

Two of those lattice extensions are the ones shown in Figure 3 and Figure 4.

4. Guiding the Mining Process

In this section, we introduce our approach of using the lattice intension structure as a vehicle to aid the user in guiding the association rule mining process. The use of a lattice structure and its traversal techniques as a representation framework for association rule mining is known in the literature (Zaiane, 2005; Loo, 2002). However, existing approaches use the equivalent of a lattice extension. In our approach, we use a lattice intension structure since it is a more compact and efficient representation especially for very large relational databases. Furthermore, a lattice intension structure suites better our model of inter-attribute mining (Alashqur, 2010). Therefore, we use such a structure to facilitate user-guided association rule mining.

The main idea of our approach is to display the lattice intension and let the user select the node(s) that is considered important from his/her perspective. The node selected by the user represents a specific combination of attributes. It indicates that the user is interested in discovering association rules among attribute values of attributes that belong to the selected node. The underlying data mining system would then restrict the mining process to only those attributes, instead of mining the entire space of possible attribute combinations.

Consider the STUDENT relation shown in Table 3 that stores students' information in a college database. This relation stores the average grade in a letter format (a, b, etc.) that a student obtained in High School (HIGH_SCHOOL_AVERAGE), the family income range (FAMILY_INCOME) of the student's family, the college average grade of the student (COLLEGE_AVERAGE), and the range in which the SAT score of the student falls (SAT_SCORE_RANGE). The relation is populated with sample data as shown.

To keep the lattice intension structure simple, we represent each attribute with the first letter of its name. Thus 'H' is used for HIGH_SCHOOL_AVERAGE, 'F' for FAMILY_INCOME, 'C' for COLLEGE_AVERAGE, and 'S' for SAT_SCORE_AVERAGE. The ID attribute is ignored since it is not included in the mining process. Figure 5 shows the lattice intension structure for this relation.

In a system built based on our approach, the lattice intension structure would be presented on the screen for the user to explore. The user then selects a node of interest. Assume that the user clicks on the node HCS as the node of interest for association rule mining. This implies that association rules among the *three* attributes HIGH_SCHOOL_AVERAGE, COLLEGE_AVERAGE, and SAT_SCORE_RANGE are to be discovered by the system along with their support and confidence values. The nodes that come under HCS are highlighted as shown in Figure 6, thus forming a sub-lattice structure.

The number of possible association rule intensions that can be constructed from *N* attributes, can be computed by the following formula:

$$\textit{Number of Association Rule Intensions} = 2^N - 2$$

There are three attributes represented by the selected node HCS. Therefore, the number of association rule intensions that can be constructed from these three attributes is:

$$\textit{Number of Association Rules} = 2^3 - 2 = 6$$

These six association rule intensions are as shown below. (In theory, association rules consisting of only one attribute in the *lhs* and one attribute in the *rhs* can also be presented to the user for selection, provided that these attributes exist in the selected lattice node.)

1) HIGH_SCHOOL_AVERAGE, COLLEGE_AVERAGE → SAT_SCORE_RANGE
2) HIGH_SCHOOL_AVERAGE → COLLEGE_AVERAGE, SAT_SCORE_RANGE
3) COLLEGE_AVERAGE , SAT_SCORE_RANGE → HIGH_SCHOOL_AVERAGE

4) COLLEGE_AVERAGE → SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE

5) SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE → COLLEGE_AVERAGE

6) SAT_SCORE_RANGE → HIGH_SCHOOL_AVERAGE, COLLEGE_AVERAGE

In a system built based on our approach, after a user selects a node in the lattice, he/she would be presented by the list of possible association rule intensions similar to the list shown above. The user can then further narrow down the search space by selecting one or more of the association rule intensions and supply *minsup* and *minconf* values. The system then generates all association rule extensions whose intensions have been selected by the user and that satisfy the *minsup* and *minconf* constraints.

To demonstrate how the computation process is performed, assume that the user selects the association rule intension number 5 from the above list. This means that the user is interested in discovering the impact of the combination of SAT score and high school average on the college average grade for students. To compute the confidence of association rules of this type, we need to compute the support of 3-itemsets that are instantiations of the itemset intension {SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE, COLLEGE_AVERAGE}, then we need to compute the support of 2-itemsets whose itemset intension is {SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE}, which represents the *lhs* of the selected association rule intension. By dividing the support of a 3-itemset over the support of the corresponding 2-itemset representing the *lhs*, we obtain the confidence.

To perform the support computation, we follow the approach described in (Alashqur, 2010), where we introduced a SQL-based algorithm named RDB-Miner for computing the support of itemsets. The algorithm computes the support of all possible itemsets. In this paper, however, we use a similar approach, with the difference that SQL is used to compute the support of only the itemsets needed based on the user guidance.

The SQL statement that computes the support count of the 3-itemsets for association rule number 5 (i.e., the association rule intension selected by the user) is shown below.

```
SELECT      SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE,
            COLLEGE_AVERAGE, COUNT (*) AS SUP_COUNT
FROM        STUDENT
GROUP BY    SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE,
            COLLEGE_AVERAGE
```

The result of the above query is shown in Table 4. On the other hand, the SQL statement to compute the support of 2-itemsets whose intension is {SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE}, which represent the *lhs* of association rule intension number 5, is shown below.

```
SELECT      SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE,
            COUNT (*) AS SUP_COUNT
FROM        STUDENT
GROUP BY    SAT_SCORE_RANGE, HIGH_SCHOOL_AVERAGE
```

The result is shown in Table 5. As described in Section 2, the confidence of extensional association rules that are instantiations of association rule intension number 5 can be computed from the data in Table 4 and Table 5, by dividing the *support count* from Table 4 over the corresponding *support count* from table 5. The following is an example.

Confidence ('500-599', c → b) = (support ('500-599',c,b)) / (support ('500-599', c)) = 0.5 = 50%

Where the nominator is taken from the first row of Table 4 (i.e. the row that has the three items "c, b, 500-599"). The denominator is taken from first row of Table 5, which represents the *lhs* component (i.e., "c, 500-599").

Below, we show all association rule extensions based on the data of Table 2, that are instansiations of association rule intension number 5. The confidence, which is shown next to each association rule, is computed in the way described above from the values in Table 4 and Table 5.

'500-599', c → b	conf. = (1/2) = 50%
'500-599', c → c	conf. = (1/2) = 50%
'600-699', b → b	conf. = (2/2) = 100%

‘700-799’, a → a	conf. = (2/3) = 67%
‘700-799’, a → b	conf. = (1/3) = 33%

If the user supplies a *minsup* and *minconf* values, then only the association rules from the above list that satisfy these conditions are actually returned to the user.

Below, we go beyond what is presented in (Alashqur, 2010), and show that SQL can be used not only to compute the support of itemsets, but also to compute the confidence of association rules. For example, to return the above association rule extensions whose association rule intension is:

SAT_SCORE_RANGE , HIGH_SCHOOL_AVERAGE → COLLEGE_AVERAGE

the following SQL statement can be executed against Table 4 and Table 5.

```
SELECT    T4.HIGH_SCHOOL_AVERAGE              AS H_LHS1,
          T4.COLLEGE_AVERAGE                  AS C_LHS2,
          T4.SAT_SCORE_RANGE                  AS S_RHS,
          (T4.SUPP_COUNT/T5.SUPP_COUNT)       AS CONFIDENCE
FROM      Table_4 T4, Table_5 T5
WHERE     T4. HIGH_SCHOOL_AVERAGE = T5.HIGH_SCHOOL_AVERAGE
           AND T4.SAT_SCORE_RANGE = T5.SAT_SCORE_RANGE
```

The above statement returns four columns. The first two represent the *lhs* of each rule extension, namely H_LHS1 and C_LHS2, that represent the two attributes HIGH_SCHOOL_AVERAGE and COLLEGE_AVERAGE, respectively. The third column, S_RHS, represents the right hand side (*rhs*) of rule extensions, namely SAT_SCORE_RANGE. The forth column represents the confidence value of the rule. The returned result is shown in Table 6.

5. Conclusion

In this paper, we introduced a new approach for user-guided association rule mining. In our approach, a user is presented with a lattice intension structure that encodes the different possible attribute combinations. By selecting a node from the structure, the user identifies the attribute combination of interest. This selection directs the system to discover any association rules among data values that belong to the attributes represented by the selected node. Furthermore, the user may be presented with a list of possible association rule intensions corresponding to the selected node. He/she would then be prompted to select one or more of these association rule intensions. The system attempts to discover the association rules that have been identified as important from the user’s perspective. This approach simplifies the data mining process specially in very large databases and makes it more user-oriented and interactive. This is, of course, in addition to improving the performance.

Acknowledgement

The author is grateful to the Applied Science Private University, Amman, Jordan, for the partial financial support granted to cover the publication fee of this research article.

References

Alashqur, A. (2008). Mining Association Rules: A Database Perspective. *International Journal of Computer Science and Network Security IJCSNS*, 8(12).

Alashqur, A. (2010). RDB-MINER: A SQL-Based Algorithm for Mining True Relational Databases. *Journal of Software*, 5(9), 998-1005.

Gouda, K., & Zaki, M. (2005). GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets. *Data Mining and Knowledge Discovery: An International Journal*, 11(3), 223-242. http://dx.doi.org/10.1007/s10618-005-0002-x

Liu, G., H. Lu, Y. Xu, & J. X. Yu. (2003). Ascending Frequency Ordered Prefix-tree: Efficient Mining of Frequent Patterns. Proc. 2003 Int. Conf. on Database Systems for Advanced Applications (DASFAA03), Kyoto, Japan.

Loo, K., Chi Lap Yip, Ben Kao, & David Cheung. (2002). A lattice-based approach for I/O efficient association rule mining. *Information Systems*, 27(1), 41-74. http://dx.doi.org/10.1016/S0306-4379(01)00046-1

Lucchese, B., S. Orlando, & R. Perego. (2006). Fast and memory efficient mining of frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), 21-36. http://dx.doi.org/10.1109/TKDE.2006.10

Ma, L., & Yi Qi. (2008). An Efficient Algorithm for Frequent Closed Itemsets Mining. *International Conference on Computer Science and Software Engineering (CSSE)*, 259-262. http://dx.doi.org/10.1109/CSSE.2008.1042

Moonestinghe, H., S. Fodeh, & P. N. Tan. (2006). Frequent closed itemset mining using prefix graphs with an efficient flow-based pruning strategy. *Proceedings of the 6th International Conference on Data Mining, Hong Kong*, 426-435.

Nadimi-Shahraki, M., Norwati Mustapha, Md Nasir Sulaiman, & Ali B. Mamat. (2008). A New Algorithm for Discovery Maximal Frequent Itemsets. *International Conference on Data Mining (DMIN)*, 2008, 309-312.

Rozenberg, B., & E. Gudes. (2006). Association rules mining in vertically partitioned databases. *Data and Knowledge Engineering*, 59(2), 378-396. http://dx.doi.org/10.1016/j.datak.2005.09.001

Wang, J., J. Han, Y. Lu, & P. Tzvetkov. (2005). TFP: An effcient algorithm for mining top-k frequent closed itemsets. *IEEE Trans. Knowledge and Data Engineering*, 17, 652-664. http://dx.doi.org/10.1109/TKDE.2005.81

Xin, D., J. Han, X. Yan, & H. Cheng. (2005). Mining compressed frequent-pattern sets. In *Proc. 2005 Int.Conf. Very Large Data Bases (VLDB'05)*, pages 709-720, Trondheim, Norway, Aug. 2005.

Zaïane, O. Mohammad El-Hajj. (2005). Pattern lattice traversal by selective jumps. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005, 729-735.

Table 1. Relation GYM_EX_MEMBERS

ID	AGE	GENDER	MEMBERSHIP DURATION	HOME_ DISTANCE	HOW_ INTRODUCED
1	young	f	Long	close	News_paper
2	middle	m	Short	far	News_paper
3	senior	f	Long	close	referral
4	senior	f	Long	close	referral
5	young	f	Long	far	News_paper
6	middle	m	Short	close	News_paper
7	senior	m	Short	far	News-paper
8	senior	f	Long	close	referral
9	young	f	Long	close	referral
10	middle	f	Long	far	News_paper
11	middle	m	Short	far	News_paper
12	senior	f	Long	close	referral
13	senior	m	Short	far	referral

Table 2. Relation R

ID	X	Y	Z
1	x1	y1	z1
2	x2	y2	z1
3	x3	y1	z2
4	x4	y2	z2
5	x1	y1	z1
6	x2	y2	z1
7	x3	y1	z2
8	x4	y2	z2
9	x5	y1	z1
10	x5	y3	z3
11	x5		z3
12	x6	y4	

Table 3. Relation STUDENT

ID	HIGH SCHOOL AVERAGE	FAMILY INCOME	COLLEGE AVERAGE	SAT SCORE RANGE
100	c	20k – 30k	b	500-599
101	b	20k – 30k	b	600-699
102	a	30k – 40k	a	700-799
103	b	40k – 50k	b	600-699
104	a	20k – 30k	a	700-799
105	a	50k – 60k	b	700-799
106	c	30k - 40k	c	500-599

Table 4. Support Count of 3-itemsets

HIGH_SCHOOL_AVERAGE	COLLEGE_AVERAGE	SAT_SCORE_RANGE	SUP_COUNT
c	B	500-599	1
b	B	600-699	2
a	A	700-799	2
a	B	700-799	1
c	C	500-599	1

Table 5. Support Count of 2-itemsets representing the LHS

HIGH_SCHOOL_AVERAGE	SAT_SCORE_RANGE	SUP_COUNT
c	500-599	2
b	600-699	2
a	700-799	3

Table 6. Association Rules and their Confidence

H_LHS1	C_LHS2	S_RHS	CONFIDENCE
500-599	c	b	0.50
500-599	c	c	0.50
600-699	b	b	1.00
700-799	a	a	0.67
700-799	a	b	0.33

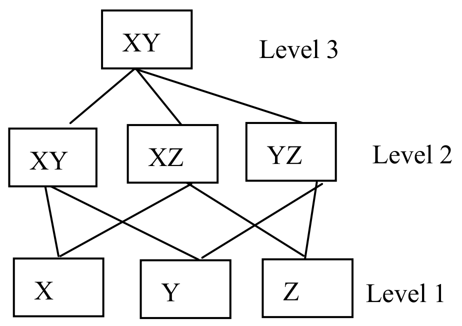

Figure 1. Lattice Intension

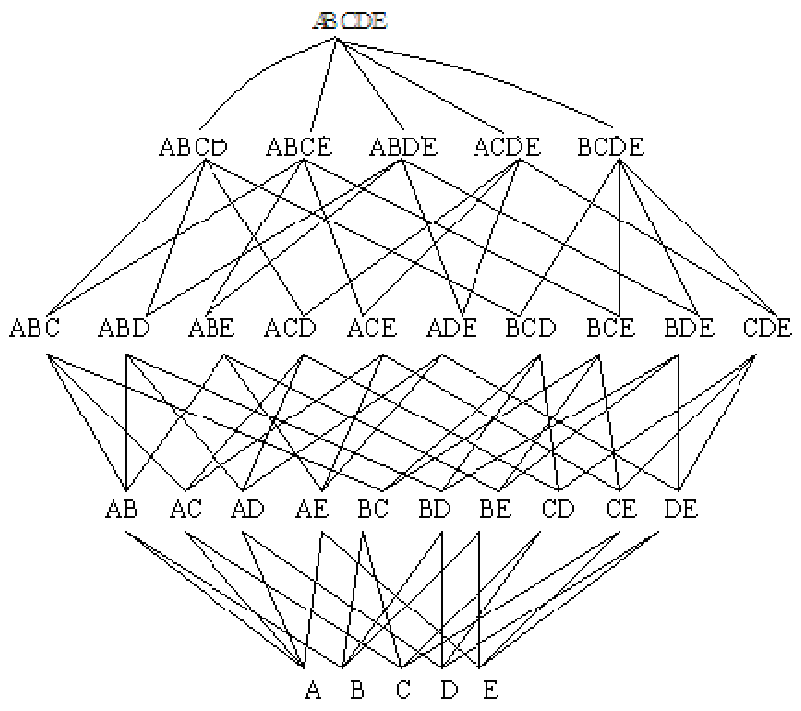
ABCDE
ABCD ABCE ABDE ACDE BCDE
ABC ABD ABE ACD ACE ADE BCD BCE BDE CDE
AB AC AD AE BC BD BE CD CE DE
A B C D E

Figure 2. Lattice Intension for Five Attributes

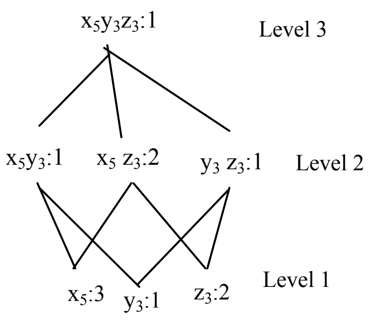

Figure 3. A Lattice Extension

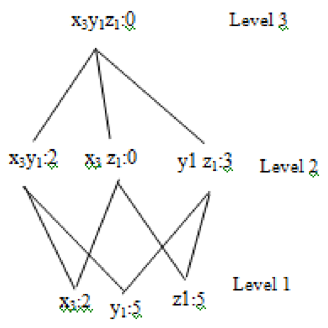

Figure 4. Another Lattice Extension

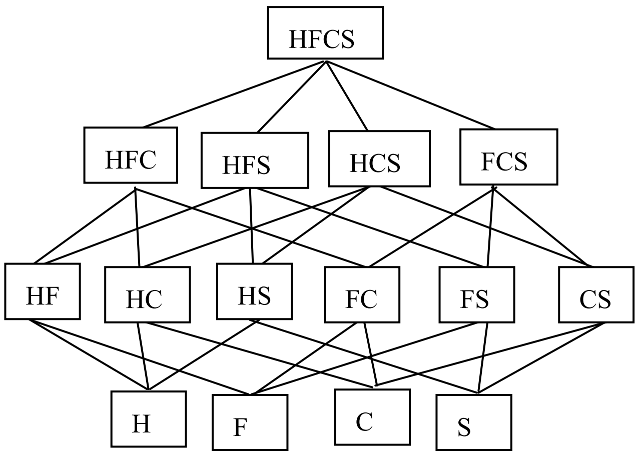

Figure 5. Lattice intension for the attributes of Table 3

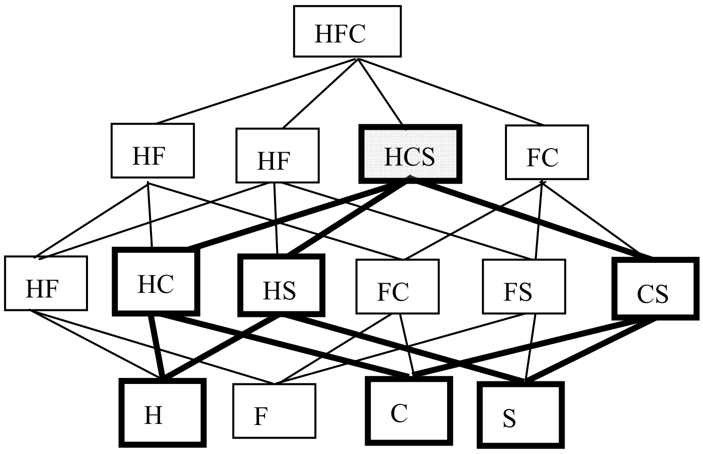

Figure 6. Highlighted Sub-Lattice Structure