OVERVIEW OF THE GUHA METHOD AS A DATA MINING TECHNIQUE

Viktor Todorovski
Faculty of Electrical Engineering
and Information Technology, UKIM
Skopje, R. of Macedonia

Ivan Chorbev
Faculty of Electrical Engineering
and Information Technology, UKIM
Skopje, R. of Macedonia

Suzana Loskovska
Faculty of Electrical Engineering
and Information Technology, UKIM
Skopje, R. of Macedonia

ABSTRACT

This paper is concerned with current applications and researches of GUHA, a method for hypothesis generation. The GUHA method is very promising in the field of association rules data mining. Some of the current software implementations of the GUHA procedures offer simple mechanism of integration with BI processes, especially where large data sets are being processed. To use data for processing, it must first be pre-processed or aggregated and after that, as a data set to be used by the implementations of the GUHA procedures. The most preferred processing data representation are bit string cards of categories, mainly because logical operations can be easily performed on such representation of data. Since GUHA has a significant statistical and logical background it is taken as very relevant in positive and negative association rules mining, and also in interesting hypothesis discovery.

I. INTRODUCTION

Unary Hypotheses Automaton or GUHA is a method consisted of procedures originated in Czechoslovak Academy of Sciences, in the mid sixties of the 20th century [1]. It has not been widely accepted until the 1980'es, but since then it emerges as a promising Data Mining method and strongly continues to develop. Its main principle is to let the computer generate and evaluate all hypotheses that may be interesting from the point of view of the given data and the studied problem. This principle has lead both to a specific theory and to several software implementations. Whereas the latter become quickly obsolete, the theory elaborated in the mean time has its standing value [4]. Typically, the hypotheses generated have the form "Many A's are B's" (B is highly frequented in A) of "A, S are mutually positively dependent" [2]. It can be said that GUHA is a method of automatic generation of hypotheses based on empirical data (thus a method of data mining). GUHA is a kind of automated exploratory data analysis: it systematically generates hypotheses supported by the data [3]. The nowadays popular term "association rules" in data mining, has occurred for the first time in the paper on GUHA [1] in 1966. A second feature, very important for GUHA, is its explicit logical and statistical foundations [2]. "The function of GUHA is to offer hypotheses, not to verify previously formulated hypotheses." This makes GUHA a method of exploratory data analysis (as opposed to confirmatory data analysis); since the 90-ties of the 20-th century the term "data mining" has been in use for such methods mainly if they deal with very large data sets.

Usually the data to be processed by GUHA procedures are represented as rectangular matrix with many columns (about ten and more) and many rows (hundreds or thousands). But in KDD (Knowledge Discovery in Databases) processes, which are usually data mining processes, we have relations between such matrixes (tables), therefore master-detail cardinality. In this type of cases the data is pre-processed and one large matrix is created. This is usual process in Business Intelligence and Data warehousing. As described in [6], relational data mining search for methods for KDD that can use more than one data table directly, without transforming the data into a single table first and then looking for patterns in such an engineered table. In many scenarios, leaving the data in relational form can save a lot of information that can be later expressed in the form of relational patterns. Single-table patterns are simply less expressive. However, in case of high pattern expressiveness there is higher computation time, which in some applications of data mining is very crucial.

Because GUHA is based on simple logical and statistical analysis as result we get easily measurable results of hypothesis support and confidence. This is used as an advantage in many new studies for simple decision support system (DSS) advice explanation or in background knowledge confirmation. The goal in [7] is "to contribute to the discussion concerning definition of valid novel, potentially useful, and ultimately understandable pattern". Data mining is a process used for identifying such patterns in data, where the GUHA method has significant theoretical and practical proof of effectiveness. Simplest forms of presenting patterns in an understandable context are aforementioned association rules. Usually we are not interested in the "classical" association rules of the form $X \neg\rightarrow Y$ where X and Y are sets of items. The intuitive meaning of $X \rightarrow Y$ is that transactions (e.g. supermarket baskets) containing set X of items tend to contain set Y of items. Widely accepted are the two measures of intensity of association rule, confidence and support. Paper [7] elaborate how these measures can be used to get interesting association rules.

Another very interesting area of future research is the data mining of negative and positive association rules in business intelligence. In [9] a theoretical and practical display of such methods is proposed. The negative association rules, in form of anti-patterns have high importance in banking or tax fraud detection software.

In the next sections, the GUHA method is going to be explained along with some of the association quantifiers. It is explained how data needs to be represented for processing. Current software implementations of GUHA procedures are presented.

II. GUHA METHOD AND ASSOCIATION RULES

A. Data representation

GUHA procedures natively work with matrix data, rows and columns representing the information about certain problem. The data should be discreet, non-stochastic in repetitive processing. Data to be processed can be represented as a rectangular matrix *M* where rows correspond to some objects and columns to their attributes. For example, objects may be patients and attributes are symptoms and diseases; or objects are bank transactions and attributes are various items describing them (for example as kind of loan etc.) – note that this widely accepted terminology – transactions and items – is going to be used. The value in the *i*-th row and *j*-th column is the value of *j*-th attribute for the *i*-th object. The value may be binary (yes-no, coded by 1 and 0), integers or real numbers, the "usual" case being the binary.

Logic is used to code hypotheses. Give each attributes (matrix column) a name (e.g. SEX; AGE; …). For any subset X of the domain of an attribute P; P:X is the property saying "the value of P is in X", e.g AMOUNT:(>10000$). If P is binary (e.g. IS-MARRIED) then P stands for P = YES and ¬P (negation) for P = NO. The formula P : X is called a *literal (atomic formula)*. One may form conjunctions of literals, e.g. P1 : X1 & P3 : X3 & P7 : X7 is a conjunction of three literals. The *i*-th object satisfies this conjunction if its value (in the *i*¬th row of the data matrix) of P1 is in X1; value of P3 is in X3 and value P7 is in X7. Such conjunctions describe composed properties of our objects [5].

For simplicity of data representation as is stated in [2], the data processed by GUHA can be a rectangular matrix of zeros and ones, the rows corresponding to objects and columns to attributes. (Needless to say, much more general data can be processed). The latter approach is used more with database data processing where rows and discreet column data is in bit string representation.

Another alternative approach on data representation in data mining with the GUHA method is the Bit String approach, as in [8]. The basic idea with the Bit String approach is to "index" and represent all data that is being analyzed in strings of bits. After that the procedure of evaluation of logical expressions is very straight forward. This approach of data representation makes it possible to use simple algorithm and data structures to efficiently compute necessary frequencies of the four fold table (see II.C).

The bit string represents each category of each attribute (i.e. each of its possible values) by one string of bits. This string is called card of category [10]. The attribute City is used as an example in Figure 1. The attribute City has 33 categories: Skopje, Kumanovo, Kratovo, Ohrid, …, Strumica,…,Krushevo.

In many data sets that we want to analyze, missing information in attributes is a usual problem. The GUHA method and its procedures have few techniques of handling such data.

		Cards of Categories				
Clicnt	City	Skopje	Ohrid	Strumica	Kratovo	...
A	Skopje	1	0	0	0	
B	Kumanovo	0	0	0	0	...
C	Kratovo	0	0	0	1	...
...	...	...	...	...	...	...
77Y	Skopje	1	0	0	0	
ZZZ	Ohrid	0	1	0	0	...

Figure 1. Cards of categories

Usually the software implementations of the GUHA procedures offer secured, deleting, optimistic and xxSec missing information handling. *Deleting*: Before evaluating a sentence, objects with missing information in antecedent or succedent are left out. *Optimistic*: No objects are left out. If a sentence is evaluated as true under optimistic processing of missing information, it means that the missing information could be replaced by valid values in such a way that the sentence would be true (i.e., that such completion exists). It is called optimistic because the best case scenario is assumed. *Secured (pessimistic)*: No objects are left out. If a sentence is evaluated as true under pessimistic processing of missing information, it means that it would be true under any completion of missing information. This method is called pessimistic because it is assumed the worst case scenario. xxSec: Before evaluating a sentence, objects with missing information in both antecedent and succedent are left out; the processing is then secured on the remaining set of objects. For more explanation on antecedent and succedent see II.B, and for more technical explanation of missing information processing by GUHA see [11] chapter 4.

Relational databases present the problem of 1:N cardinality of objects, or master-detail data relations. The term *virtual attributes* is proposed in [6], where a virtual attribute of the master is actually an *aggregate* or an *existential quantifier* of the detail attribute(s) values where discretization on the values must be performed.

B. Association rules

An association rule is usually understood as an expression of the form of X→Y, where X and Y are sets of items. The intuitive meaning is that transactions (e.g. supermarket baskets) containing set X of items tend to contain set Y of items. Two measures of intensity of association rule are used, confidence and support.

An association rule discovery task is a task to find all association rules of the form X→Y such that the support and confidence of X→Y are above the user-defined thresholds *minsup* and *minconf*. The conventional algorithm of association rules discovery proceeds in two steps. All frequent item sets are found in the first step. The frequent item set is the item set that is included in at least minsup transactions. The association rules with the confidence at least minconf are generated in the second step [8].

The relevant queries that the GUHA ASSOC procedure forms and tests in the data have the form A≈S, where A and S are interpreted as some (possibly compound) properties of objects, and ≈ stands for one of the statistical tests used in GUHA (represented by binary generalized quantifiers).

Property A is called ***antecedent***, S is called ***succedent*** (or with previous notation X and Y respectively). In each run, GUHA systematically forms and tests all relevant queries satisfying the defined task [11].

III. GUHA THEORY

A. GUHA Principle

To explain the GUHA principle of operation we can take an example where we assume that we have persons as objects, and the data matrix includes attributes called: 'age', 'sex', 'salary', 'does transactions', 'checks balance', 'uses reports', 'has kids', 'is married', 'good loaner' (note the different sets of values of these attributes). We can use all these attributes, where all but the last one will occur in antecedent and the last one will occur in succedent definition. Suppose we opt for implicational tests: we shall take ≈ for a 90% implication. This means that, if the statement should be valid, ninety percent of objects satisfying A must also satisfy S. We are looking for potential causes of being/not being *good loaner*. Examples of relevant queries:

- 1.A = 'male & over sixty & high salary & is married & does transactions & not have kids' , S = 'good loaner'
- 2.A= 'female & over sixty & average salary & is married & does transactions & has kids', S= 'bad loaner'

From the variable 'age', a property 'over sixty' will be created, for 'salary' high salary and average salary (and possibly also other properties) by the user. This is because GUHA only works with properties of objects, and the attribute 'age' or 'salary' is not a property. Similarly, from the variable 'sex' we can create two properties 'male' and 'female'.

The two sentences 1. and 2. are relevant, thus GUHA will form them and test them in the data, by-and-by. How is the test performed? Each of the objects falls into one of four distinct groups: either it satisfies both A and S, or it only satisfies A, or it only satisfies S, or it satisfies neither. The frequency of the four cases are counted (we denote them *a*,*b*,*c* and *d* respectively, see Table 1). All of the tests operate with these four numbers. In case of 90% implication, we find out whether the fraction *b*/*a* is greater or equal to 0.9. If so, the sentence is valid *in the data*.

B. GUHA Philosophy And Statistics

From a logician's point of view, relevant queries are sentences of a certain (first order) language, while the data matrix is a structure for this language. The structure is primary, and the language is determined by it.

The data matrix determines a (set-theoretical) structure determine a (set-theoretical) structure $\boldsymbol{M}=\langle M,\mathbf{v}_1,\ldots\mathbf{v}_n\rangle$, where M is a (non-empty, finite) set of objects (formal counterparts of the objects investigated), and $v_1,\ldots,v_n$ are unary functions from M to $R_1,\ldots,R_n$ respectively (suppose $R_1,\ldots,R_n$ are given). The language of $\boldsymbol{M}$ contains unary function symbols (unary functors) $V_1,\ldots,V_n$.

The data matrix may contain missing information. This will be represented in the most straightforward way, i.e., each of the sets of values $R_1,\ldots,R_n$ contains an element X, and if $v_i(o)$=X, then the variable *i*-th variable has missing information for object represented by *o*.

We suppose that the input variables are categorized, and that coefficients are defined on them. Categorization of a variable given by v_i is defined by a finite decomposition of its range R_i-X into nonempty sets (intervals). Suppose that for *i*th variable, represented by *vi*, the categories are $R_{i,1},\ldots,R_{i,ki}$. A coefficient of the *i*-th variable is an arbitrary nonempty proper subset of its categories. In the implementations of the GUHA methods, the creation of coefficients means creation of bit strings for the variable, and its negation (X [100010…010] and ¬X [011101…101]).

ASSOC uses two connectives, & and ¬, to build up formulas from atoms; the negation ¬ may, however, only be applied to atomic formulas. An **atomic formula** is an expression $(C)V_i(x)$, where V_i is the *i*-th functor of *L*, *C* is a coefficient of the *i*-th variable in $\boldsymbol{M}$, and x is a variable (in the sense of predicate logic, i.e., a attribute for objects).

The associated functions of quantifiers operate on four-fold tables. A **four-fold table** $T4_{\boldsymbol{M}}$ (A, S) = (a, b, c, d) isan ordered quadruple of natural numbers, defined in Table 1.

Table 1: four-fold table or 4FT

M	S	¬S	
A	a	b	r
¬A	c	d	s
	k	l	n

$a = a_{11} = Fr^{M}_{1}(A\&S)$; $b = a_{10} = Fr^{M}_{1}(A\&\neg S)$

$r = a + b = a_{1.} = Fr^{M}_{1}(A)$

$c = a_{01} = Fr^{M}_{1}(\neg A\&S)$; $d = a_{00} = Fr^{M}_{1}(\neg A\&\neg S)$

$s = c + d = a_{0.} = Fr^{M}_{1}(\neg A)$

$k = a + c = a_{.1} = Fr_1^{M}(S)$; $l = b + d = a_{.0} = Fr_1^{M}(\neg S)$

$n=a+b+c+d$

$Fr^{M}1(A)$ is the number of objects in M for which A is evaluated as true.

One of the most important notations with the GUHA method is ***quantifier,*** where a quantifier Q with associated function a_Q is:

- **associational**, if the following is true: if (a, b, c, d), (a', b', c', d') are two 4fts (where (a+b+c+d may differ from a'+b'+c'+d'), if aQ(a, b, c, d)=1, and if a' ≥ a, b' ≥ b, c' ≤ c, d' ≤ d, then a_Q(a', b', c', d')=1.
- **symmetric**, if the following is true: if (a,b,c,d) is a 4ft, a_Q(a, b, c, d)=1, then a_Q(a, c, b, d)=1.
- **implicational**, if the following is true: if (a, b, c, d), (a', b', c', d') are two 4fts (where (a+b+c+d may differ from a'+b'+c'+d'), if a_Q(a, b, c, d)=1, and if a' ≥ a, b' ≤ b, then a_Q(a', b', c', d')=1.
- a **quantifier of similarity**, if the following is true: if (a, b, c, d), (a', b', c', d') are two 4fts (where (a+b+c+d may differ from a'+b'+c'+d'), if a_Q(a, b, c, d)=1, and if a' ≥ a, b' ≤ b, c' ≤ c, then a_Q(a', b', c', d')=1.
- a **quantifier of equivalence**, if the following is true: if (a, b, c, d), (a', b', c', d') are two 4fts (where (a+b+c+d may

differ from a'+b'+c'+d'), if $a_Q(a, b, c, d)=1$, and if a'+d' ≥ a+d and b'+c' ≤ b+c, then $a_Q(a', b', c', d')=1$.

C. GUHA quantifiers

SIMPLE (simple deviation) quantifier has two parameters: BASE ≥ 1,δ≥ 0. This quantifier evaluates the association as following: A $\sim_{\delta,BASE}$ S is true iff a ≥ BASE and a·d > e^{δ}·b·c. The BASE parameter is the minimum user defined threshold for positive evaluation of the attribute and δ is the minimal confidence needed in percentage (minsup and minconf).

FISHER (Fisher's test) where we have parameters:

$$\sum_{i=a}^{\min(r,k)} \frac{\binom{k}{i}\cdot\binom{l}{r-i}}{\binom{n}{r}} = \sum_{i=0}^{\min(b,c)} \frac{r!\,s!\,k!\,l!}{n!(a+i)!(b-i)!(c-i)!(d+i)!} \le \alpha$$

Association rule A $\sim_{\alpha,BASE}$ S corresponds to a test (on the level α) of the null hypothesis of independence of A and S against the alternative one of the positive dependence.

Implicational quantifiers are used in verification of sentences where only the values of *a* and *b* from the four-fold table are needed. The two-field table is $T2_M$ (A, S) =(a, b). *Lower critical implication* is categorized as an *Auxiliary implicational test* which is used in evaluating some of the quantifiers (e.g., LBOUND in LIMPL).

$$LBOUND(a,r) = \sum_{i=a}^{r}\binom{r}{i}\cdot CP^i\cdot(1-CP)^{r-i} \le \alpha$$

$$Parameters: 0 < CP < 1, 0 < \alpha \le 0.5$$

LIMPL (lower critical almost implication) A $\Rightarrow^{!}$ $C_{P,\alpha,BASE}$ S is true iff a ≥ BASE and LBOUND (a, r) ≤α with parameters: BASE ≥ 1, 0 < CP < 1, 0< α≤ 0.5. Association rule A $\Rightarrow^{!}_{CP,\alpha,BASE}$ S corresponds to a test (on the level α) of a null hypothesis H0: P(A|S) ≤ CP against the alternative one H1: P(A|S) > CP. If association rule A $\Rightarrow^{!}_{CP,\alpha,BASE}$ S is true in data matrix *M* then the alternative hypothesis is accepted.

Another widely used implication test with GUHA is *FIMPLE (founded almost implication)*. With this test we have defined parameters: BASE ≥ 1, 0 < CP ≤ 1. A =>CP,BASE S is true iff a ≥ BASE and (a / r) ≥ CP. The numerical characteristics, the relative frequency PROB (a, r)= a / r,

$$A.S.E = \sqrt{(a/r)\cdot(1-a/r)\cdot(1/r)}$$

The association rule A $\Rightarrow_{CP,BASE}$ S can be interpreted as "100·CP per cent of objects satisfying A satisfy also S" or "A implies S on the level 100·p per cent".

Double almost-implications or *quantifiers of similarity* like FIMPL2, LIMPL2, UIMPL2 are symmetric (and similarity quantifiers in the sense of the above definition). Of these we will have only *FIMPL2* or *founded double almost-implication.* This quantifier is restricted with parameters: BASE ≥ 1, 0<CP≤1, where A<=>$_{CP,BASE}$S is true iff A=>$_{CP,BASE}$S and S=>$_{CP,BASE}$A. The association rule A<=>$_{CP,BASE}$S can be interpreted as "100·CP per cent of objects satisfying A or S satisfy both A and S" or "A∧ S implies A∨ S on the level 100·CP per cent".

Quantifiers of equivalence like FEQ, LEQ, UEQ are symmetric and equivalence quantifiers in the sense of the above definition. The *FEQ or CP-almost-equivalence* is defined with parameters: BASE ≥ 1, 0 < CP ≤ 1. The association rule A ≡CP,BASE S is true iff PROB (a + d, a+b+c+d) = (a + d) / (a + b + c + d) ≥ CP. The association rule A ≡$_{CP,BASE}$ S can be interpreted as "100·CP per cent of objects have the same value for A and S".

GUHA method also has testing of the hopefulness of a sentence A $\approx_M$ S. In other words we set the limit aside BASE, and we have BASANT that a hypothesis must confront in sense that it's computed association is hopeful. This can be used only when the user is highly involved with the data being processed. For each quantifier, there is a number BASANT ≥ BASE, such that if a sentence A ≈M S is true, then a ≥ BASANT. In a model *M* without missing information, a sentence A≈M S is hopeless if a = Fr^M_1(A&S) < BASANT. An antecedent A is hopeless if r = Fr^M_1(A) < BASANT, a succedent S is hopeless if l = Fr^M_1(S) < BASANT. We can extend the definition of hopefulness to literals as one-member conjunctions (disregarding the relevance criteria for the moment).

IV. POSITIVE AND NEGATIVE ASSOCIATION RULES

A. Positive Association Rules

In [9] the positive and negative association rules are subject of elaboration. There is a *Support-Confidence Framework of Positive ARs (Association Rules)*. This framework is defined as we have $D = \{t_1, t_2, \ldots, t_n\}$ which can be a relational database of *n* records (or transactions) with a set of binary attributes (or items) $I = \{i_1, i_2, \ldots, i_m\}$. For an itemset $X \subseteq I$ and a transaction *t* in *D*, we say that *t* supports *X* if *t* has value 1 for all the attributes in *X*; for conciseness, we also write $X \subseteq t$. By D_X we denote the records that contain all attributes in *X*. The support of *X* is computed as $supp(X)= |D_X| \; n$, i.e the fraction of transactions containing *X*. A (positive) association rule is of the form: $X \Rightarrow Y$, with $X, Y \subset I, X \cap Y = \varnothing$. Support and confidence of $X \Rightarrow Y$ are defined as $supp(X \Rightarrow Y)= supp(XY)$ and $conf(X \Rightarrow Y)= supp(XY)/ supp(X)$ respectively. A valid positive association rule has support and confidence greater than given thresholds *ms* and *mc*. Furthermore, an itemset *X* is called frequent if $supp(X) \ge ms$.

B. Negative Association Rules

In [9] to define negative Association Rules for simplicity and to have more efficient algorithm, they have chosen the definition where the traditional itemset is maintained (so $X,Y \subset I$), and to each *positive* rule $X \Rightarrow Y$ correspond three negative ones, $X \Rightarrow \neg Y$, $\neg X \Rightarrow Y$ and $\neg X \Rightarrow \neg Y$ .A transaction *t* supports $X \Rightarrow \neg Y$ if $X \subseteq t$ and $Y \not\subseteq t$. Hence, the meaning of a rule like $\{i_1\} \Rightarrow \neg \{i_2, i_3\}$ is that "the appearance of i_1 in a transaction *t* induces that i_2 and i_3 are unlikely to appear simultaneously in *t*"; hence a record containing i_1 and i_2, but not i_3, supports this rule. It can be

verified that $supp(X \Rightarrow \neg Y) = supp(X \neg Y) = supp(X) - supp(XY)$ for $X, Y \subset I$, and similarly support and confidence of the other kinds of negative ARs can be be straightforwardly deduced from the corresponding positive itemset supports. The total number of positive and negative ARs that can be generated is $4(3^m - 2^m + 1 + 1)$, of which $3^m - 2^m + 1 + 1$, i.e., one fourth, are positive. Although theoretically the complexity of mining both positive and negative ARs is in the same level as that of mining only positive ARs, naive implementations will run into trouble; this is because one of the most important pruning aids in Apriori, namely that supersets of infrequent (positive) itemsets need not be considered as they cannot be frequent (downward closure property of *supp*), is no longer valid for negative itemsets. For the latter, a dual upward closure property of *supp* holds: if $supp(\neg X) \geq ms$, then, for every $Y \subset I$ such that and $X \cap Y = \varnothing$, $\neg(XY)$ also meets the support threshold. As a consequence, all these frequent negative itemsets must be further considered, and many meaningless negative rules with large consequents can arise. In [9] section IV, is shown how to exploit the upward closure property to define an alternative definition of validity for negative ARs and to restrict the search space of their mining algorithm. For simplicity, they also limit themselves to support and confidence to determine the validity of ARs.

C. GUHA And Negative Association Rules

Very important area of future research is the negative and positive association rules in means of data mining and business intelligence. When we are searching for hypotheses and association rules between attributes in the data, the hypotheses with negative association are also important as the positive related ones, as they are more useful as anti-patterns and therefore more interesting in many domains.

The GUHA method procedures are very feasible of their using in negative association rules data mining framework, especially of performance aspect. Computation of statistics for negative support can be clearly seen in Table 1, where *b, c* and *d* are subject of examination and computation. With bit strings used, the time of computation is predicted to be linear related with positive evaluation of expressions, so with simple modifications on the implementation of the GUHA procedures, statistics for negative association rules can be easily obtained, including hypothesis propositions.

D. GUHA Implementations

1) GUHA +-

GUHA+- is a student implementation of the GUHA method. The very first version of GUHA+-was produced as a teamwork project, a compulsory part of student curriculum at the Faculty of Mathematics and Physics, Charles University, Prague, in September 1998. The teamwork was supervised by Institute of Computer Science (ICS), Academy of Sciences of the Czech Republic.

This implementation only includes the 4ft based procedures, its fast and is provided with good user operation manual. Data specification is moderately easy, depending on initial data set format, and connection with formatted data set is very straight forward, offering various connection technologies. Hypothesis selection however is very difficult process, and the selector is restricted to some predefined categories. Selection can be modified, but it requires more specific knowledge of the software, and is not much user friendly.

2) LISp Miner

LISp-Miner is an academic project for support research and teaching of knowledge discovery in databases. It is suitable for students, pilot and mid-size KDD projects. The core of the system is consisted of several KDD procedures capable to give answers to various standard and non-standard analytical questions. Modules to solve some additional tasks are also included, especially in the area of rules formulation and translation in natural language. The LISp-Miner software can be freely downloaded, and has documented demonstration (see [12]) of its usage. Detailed instructions how to start and various types of user support are also available. There are various research activities and applications both in data analysis and in teaching related to the LISp-Miner project.

The LISp-Miner project is managed by teachers and students of University of Economics, Prague. Other researchers take part in the project realization too. There are several tens of publications related to the LISp-Miner project. This implementation covers more GUHA procedures in terms of KDD procedures as: 4ft-Miner, SD4ft-Miner, KL-Miner, CF-Miner, SDKL-Miner, SDCF-Miner, Action4ftMiner, and KEX. LISp Miner includes additional modules as data Preprocessing – TimeTransf, LM Knowledge and Sewebar as rules translation modules.

V. CONCLUSION

The GUHA method is very promising technique for attribute association rules data mining. It is proven very fast on large data processing mainly because of its logical and statistical foundations. GUHA implementations are standard software implementations, with which GUHA has been proven as efficient data mining tool. GUHA offers very simple logic of functioning and easily and user friendly results, in means of hypothesis (attribute association) discovery.

GUHA can be involved in positive and negative association rules mining, especially with de-normalized or aggregated relational data, where with number of financial, medical and other types of application can be used.

Further work may include some practical experiments in domains of Medical informatics, financial analysis, user behavior, and negative association rules. Most appropriate is the LISp Miner implementation cause of the extensible nature and still developing and active team.

REFERENCES

[1] Hàjek P., Havel I., Chytil M.: The GUHA method of automatic hypotheses determination, Computing 1 (1966) 293-308.

[2] Petr Hàjek, Tomàsh Feglar, Jan Rauch, David Coufal, The GUHA method, data preprocessing and mining. (Position paper.), Technical report No. 867.
[3] Esko Turunen, GUHA method for data mining, Tampere University of Technology (TUT), Department of Mathematics, PO Box 553, FIN¬33101, Tampere, Finland
[4] Petr Hàjek, Tomàsh Havrànek, Mechanizing Hypothesis Formation ¬Mathematical Foundations for a General Theory, Springer-Verlag Berlin Heidelberg New York in 1978, ISBN 3-540-08738-9, Copyright reverted to the authors by a letter of 7/03/01
[5] Petr Hàjek, Briefly on the GUHA method of data mining, Journal of Telecommunications and Information Technology 3/2003 112 – 114
[6] Tomàsh Karbant, Relational Data Mining and GUHA, Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Praha
[7] Rauch, J., Interesting Association Rules and Multi-relational Association Rules, Communications of Institute of Information and Computing Machinery, Taiwan. Vol. 5, No. 2, May 2002. pp. 77–82
[8] Rauch, J. -Simunek, M., Alternative Approach to Mining Association Rules, in FDM 2002, The Foundation of Data Mining and Knowledge Discovery, The Proceedings of the Workshop of ICDM02, pp 157-162.
[9] Chris Cornelis, Peng Yan, Xing Zhang, Guoqing Chen, Mining Positive and Negative Association Rules from Large Databases
[10] Rauch, J.: Some Remarks on Computer Realisations of GUHA Procedures, International Journal of Man-Machine Studies 10, 1978, pp. 23-28.
[11] Zuzana Honzikova, Institute of Computer Science at Academy of Sciences, Czech Republic, Guha +-User's Guide, 1999
[12] http://lispminer.vse.cz/index.html