Dr.N.Chandra Sekhar Reddy, Dr.P.C.Rao.Vemuri, Ms.M.SaradaVaralakshmi, Mr B. Aswani Kumar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 1, January -February 2013, pp.2052-2064

Multifunctional Confidence Reliability Algorithm(MCRA) For Knowledge Discovery using Evaluation Of Learning Algorithms in Data Mining

Dr.N.Chandra Sekhar Reddy[1] Dr.P.C.Rao.Vemuri[2] Ms.M.SaradaVaralakshmi[3] Mr B. Aswani Kumar

1Professor and Head of the Department, Department of Computer Science and Engineering, St.Peter's Engineering College, Hyderabad, A.P, India
2 Professor and Principal, Department of Computer Science and Engineering, St. Peter's Engineering College, Hyderabad, A.P, India
3 Professor and Head of the Department, Department of Information Technology, St. Peter's Engineering College, Hyderabad, A.P, India
4 Associate Professor, Department of Computer Science and Engineering, St. Peter's Engineering College, Hyderabad, A.P, India

Abstract-

Association rule mining is the most popular technique in data mining. Mining association rules is a prototypical problem as the data are being generated and stored every day in corporate computer database systems. To manage this knowledge, rules have to be pruned and grouped, so that only reasonable numbers of rules have to be inspected and analyzed. In this paper, we present a detailed multifunctional itemset mining algorithm called MCRA. MCRA shows a number of additional features and performs the following, usually independent, tasks: identify frequent closed itemsets and associate generators to their closures. This makes MCRA a complete algorithm for computing classes of itemsets including generators and closed itemsets. These characteristics allow one to extract minimal non-redundant association rules, a useful and lossless representation of association rules. In addition, being based on the Pascal algorithm, MCRA has a rather efficient behavior on weakly and strongly correlated data. In particular, MCRA is able to perform the following, usually independent, tasks: identify frequent closed itemsets and associate generators to their closures. This allows one to find minimal non-redundant association rules.

Keywords -Confidence, Balanced tree, Association Rules, Data Mining, Multidimensional dataset, Pruning, Frequent itemset, Minimal Non-Redundant Association Rules.

I. IINTRODUCTION

Mining association rules is particularly useful for discovering relationships among items from large databases. A standard association rule is a rule of the form X→ Y which says that if X is true of an instance in a database, so is Y true of the same instance, with a certain level of significance as measured by two indicators, support and confidence. The goal of standard association rule mining is to output all rules whose support and confidence are respectively above some given support and coverage thresholds. These rules encapsulate the relational associations between selected attributes in the database, for instance, coke → potato chips: 0.02 support; 0.70 coverage denotes that in the database, 70% of the people who buy coke also buy potato chips, and these buyers constitute 2% of the database. This rule signifies a positive (directional) relationship between buyers of coke and potatochips. The mining process of association rules can be divided into two steps.1. Frequent Itemset Generation: generate all sets of items that have support greater than a certain threshold, called minsupport.2. Association Rule Generation: from the frequent itemsets, generate all association rules that have confidence greater than a certain threshold called minconfidence. Generating strong association rules from frequent itemsets often resultsin a huge number of rules, which limits their usefulness in real life applications.To solve this problem, different concise representations of association rules havebeen proposed, e.g. generic basis (GB), informative basis (IB), representativerules (RR), Duquennes-Guigues basis (DG), Luxenburger basis (LB),proper basis (PB), structural basis (SB), etc.

Kryszkiewicz showed that minimal non-redundant rules3 (MNR) with the cover operator, and the transitive reduction of minimal non-redundant rules3 (RMNR) with the cover operator and the confidence transitivity property are lossless, sound, and informative representations of all valid association rules. From the definitions of MNR and

RMNR it can be seen that we only need frequent closed itemsets and their generators to produce these rules. Frequent itemsets have several condensed representations, e.g. closed itemsets, generator representation, approximate free-sets, disjunction-free sets, disjunction-free generators, generalized disjunction-free generators, nonderivable itemsets, etc. From these representations, the one which consists of frequent closed itemsets and frequent generators gives rise to a concise set of association rules, which is lossless, sound, and informative. This set of rules, called the set of minimal non-redundant association rules (MNR), is not minimal in general case, but presents a good compromise between its size and time needed to generate it.Bastide et al. presented the Pascal algorithm and claimed thatMNR can be extracted with this algorithm. However, to obtainMNR from the output of Pascal, one has to do a lot of computing. First, frequent closed itemsets must also be known. Second, frequent generators must be associated to their closures. Here we propose an algorithm called MCRA, an extension of Pascal, which does this computing. Thus, MCRA allows one to easily construct MNR. Instead of Pascal, we might have selected another algorithm. The reason for choosing Pascal was as follows: among levelwise frequent itemset mining algorithms; it may be the most efficient. This is due to its pattern counting inference mechanism that can significantly reduce the number of expensive database passes. Furthermore, MCRA can be generalized, and thus it can be applied to any frequent itemset mining algorithm. The paper is organized as follows. In the next section, we overview the basic concepts and essential definitions. This is followed by the description of the three main features of the MCRA algorithm. We then present MCRA and give a running example. Then, the generation of minimal nonredundant association rules is presented. Next, we provide experimental results for comparing the efficiency of MCRA to Pascal and Apriori. Finally, we draw conclusions in the last section.

II.PROPOSED ALGORITHM

A.Derived algorithm (MCRA) -

MCRA has three main features, namely (1) pattern counting inference, (2) identifying frequent closed itemsets, and (3) identifying generators of frequent closed itemsets.

A.1 Pattern Counting Inference in Pascal and MCRA

The first part of MCRA is based on Pascal, which employs properties of the counting inference. In levelwise traversal of frequent itemsets, first the smallest elements of an equivalence class are discovered, and these itemsets are exactly the generators. Later, when finding a larger itemset, it is tested if it belongs to an already discovered equivalence class. If it does, the database does not have to be accessed to determine the support of the itemset. This way the expensive database passes and support counts can be constrained to the case of generators only. From some level on, all generators can be found, thus all remaining frequent itemsets and their supports can be inferred without any further databasepass.In Figure 1 (left) we show the output of Pascal when executed on dataset D (Fig 2): it finds frequent itemsets and marks frequent generators. Recalling the definitions ofMNR and RMNR, we see that this output is not enough. From our running example, the output of MCRA is shown in Figure 1 (right). Here one can see the equivalence classes of database D. Only the maximal (frequent closed itemset) and minimal elements (frequent generators) of each equivalence class are indicated. Support values are shown in the top right-hand corner of classes. As can be seen, the output of MCRA is necessary and sufficient for generating GB,IB, RIB, MNR, and RMNR.

A.2 Identifying Closed Itemsets among Frequent Itemsets in MCRA

The second part of MCRA consists in the identification of FCIs among FIs, adapting this idea from Apriori-Close [5]. By definition, a closed itemset has no proper superset with the same support. At the ith step all i-itemsets are marked as "closed". At the (i + 1)th iteration for each (i + 1)-itemset we test if it contains an i-itemset with the same support. If so, then the i-itemset is not a closed itemset since it has a proper superset with the same support, thus it is marked as "not closed". When the algorithm terminates with the enumeration of all FIs, itemsets still marked "closed" are the FCIs of the dataset.

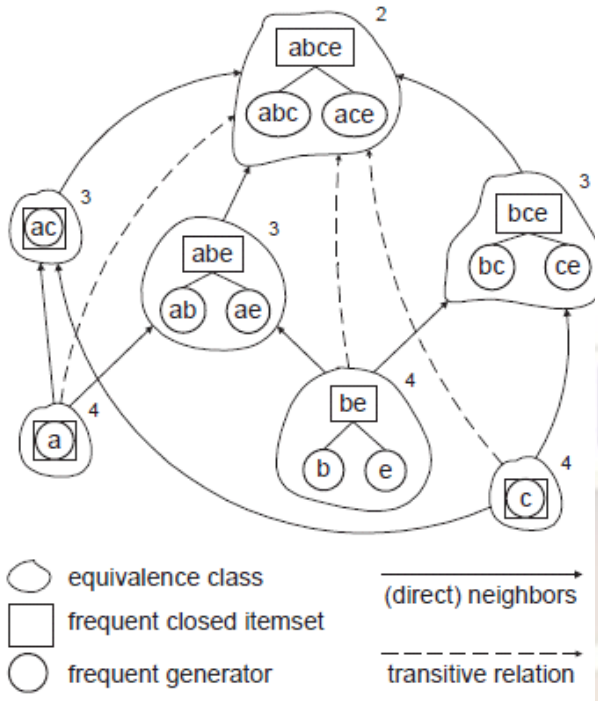


Fig.1 . Result of MCRA with min_supp = 0.2 (40%)

A.3 Associating the Generators to their Closures in MCRA

Because of the levelwise itemset search, when an FCI is found, all its frequent subsets are already known. This means that its generators are already computed,they only have to be identified. We show that the search space for generatorscan be narrowed to not closed ones. This is justified by the following properties:Property 4. A closed itemset cannot be a generator of a larger itemset.Property 5. The closure of a frequent not closed generator g is the smallest proper superset of g in the set of frequent closed itemsets.By using these two properties, the algorithm for efficiently finding generatorsis the following: generators are stored in a list l. At the ith iteration, frequent closed i-itemsets are filtered. For each frequent closed i-itemset z, the following steps are executed: find the subsets of z in list l, register them as generators of z, and delete them from l. Before passing to the (i+1)th iteration, add the i-itemsets that are not closed generators to list l. Properties 4 and 5 guarantee that whenever the subsets of a frequent closed itemset are looked for in list l,only its generators are returned. The returned subsets have the same support as the frequent closed itemset; it does not even have to be tested. Since only the generators are stored in the list, it means that we need to test far fewer elements than the whole set of FIs. Since at step i the size of the largest itemset in list l can be maximum (i − 1), we do not find the generators that are identical to their closures. If an FCI has no generator registered, it simply means that its generator is itself. As for the implementation, instead of using a “normal” list for storing generators, the trie data structure is suggested, since it allows a very quick lookup of subsets.

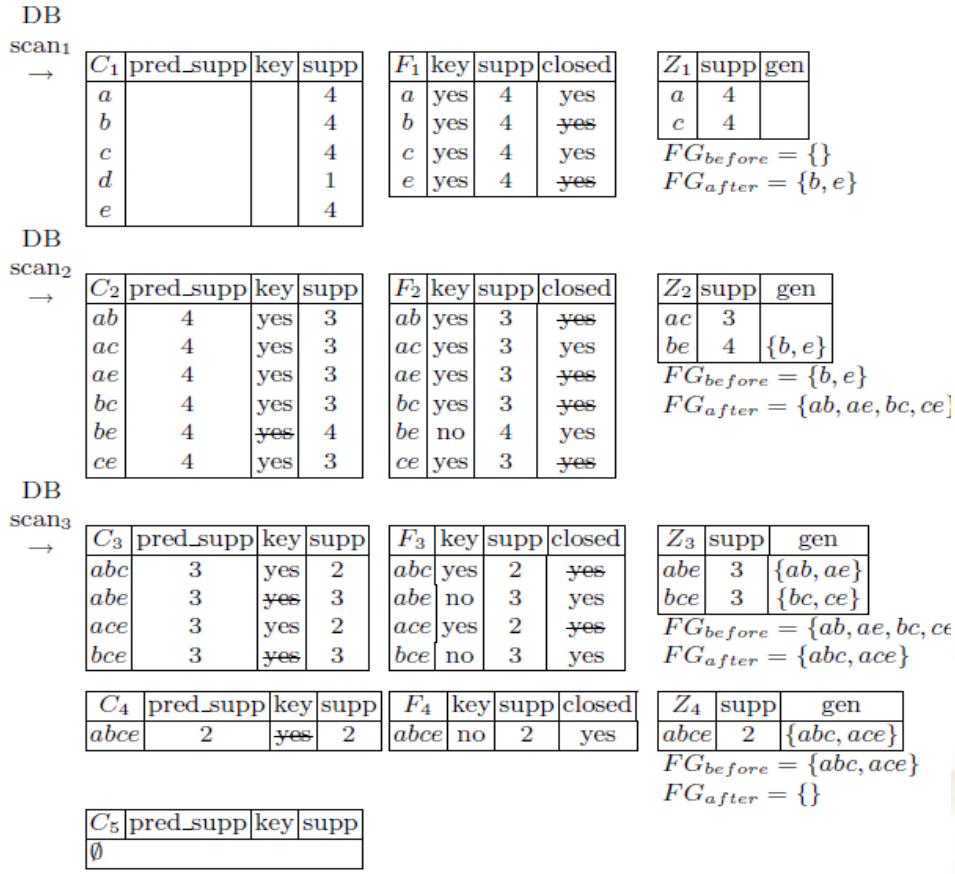


Fig.2 .Execution of MCRA on Dataset D with min_supp = 0.2 (40%)

BThe MCRA Algorithm
B.1 Pseudo Code

The main block of the algorithm is given in Algorithm 1. MCRA uses three differentkinds of tables, their description is provided in Tab(s). 1 and 2. We assume thatan itemset is an ordered list of attributes, since we will rely on this in the MCRAGen function (Algorithm 2).SupportCount Procedure: this method gets a Ci table with potentially frequent candidate itemsets, and it fills the support field of the table. This steprequires one database pass. For a detailed description consult [22].

Subsets function: this method gets a set of itemsets S, and an arbitraryitemset l. The function returns such elements of S that are subsets of l. Thisfunction can be implemented very efficiently with the trie data structure.Note that the empty set is only interesting, from the point of view of rulegeneration, if its closure is not itself. By definition, the empty set is always agenerator and its support is 100%, i.e. it is present in each object of a dataset(sup(∅) = |O|). As a consequence, it is the generator of an itemset whosesupport is 100%, i.e. of an itemset that is present in each object. In a booleantable it means a rectangle that fills one or more columns completely. In this case,the empty set is registered as a frequent generator (line 15 of Algorithm 1), andattributes that fill full columns are marked as “not keys” (line 10 of Algorithm 1).Since in our database D there is no full column, the empty set is not registeredas a frequent generator and not shown in Fig. 1 either.

B.2 Optimizing the Support Count of 2-itemsets

It is well known that many itemsets of length 2 turn out to be infrequent.Counting the support of 2-itemsets can be done more efficiently the followingway. Through a database pass, an upper-triangular 2D matrix can be built containing the support values of 2-itemsets. This technique is especially useful forvertical algorithms, e.g. Eclat [23] or Charm [10], where the number of intersection operations can thus be significantly reduced, but this optimization can alsobe applied to levelwise algorithms. Note that for a fair comparaison with otheralgorithms, we disabled this option in the experiments

.

Table 1. Tables used in MCRA.

C_i	potentially frequent candidate i-itemsets fields: 1) itemset, 2) pred_supp, 3) key, 4) support
F_i	frequent i-itemsets fields: 1) itemset, 2) key, 3) support, 4) closed
Z_i	frequent closed i-itemsets fields: 1) itemset, 2) support, 3) gen

Table 2.Fields of the tables of MCRA.

itemset	– an arbitrary itemset
pred_supp	– the minimum of the supports of all $(i-1)$-long frequent subsets of the itemset
key	– is the itemset a key generator?
closed	– is the itemset a closed itemset?
gen	– generators of a closed itemset

Algorithm 1 (MCRA):

```
1) fullColumn   false;
2) FG   {}; // global list of frequent generators
3) filling C1 with 1-itemsets; // copy attributes to C1
4) SupportCount(C1);
5) F1   {c 2 C1 | c.support _ min supp};
6) loop over the rows of F1 (l)
7) {
8) l.closed   true;
9) if (l.supp = |O|) {
10) l.key   false; // the empty set is its generator
11) fullColumn   true;
12) }
13) else l.key   true;
14) }
15) if (fullColumn = true) FG   {;};
16) for (i   1; true; ++i)
17) {
18) Ci+1   MCRA-Gen(Fi);
19) if (Ci+1 = ;) break; // exit from loop
20) if Ci+1 has a row whose “key” value is true, then
21) {
22) loop over the elements of the database (o) {
23) S   Subsets(Ci+1, o);
24) loop over the elements of S (s):
25) if (s.key = true) ++s.support;
26) }
27) }
28) loop over the rows of Ci+1 (c)
29) {
30) if (c.support _ min supp) {
31) if ((c.key = true) and (c.support = c.pred supp)):
32) c.key   false;
33) Fi+1   Fi+1 [ {c};
34) }
```

35) }
36) loop over the rows of Fi+1 (l) {
37) l.closed true;
38) S Subsets(Fi, l);
39) loop over the elements of S (s):
40) if (s.support = l.support) s.closed false;
41) }
42) Zi {l 2 Fi | l.closed = true};
43) Find-Generators(Zi);
44) }
45) Zi Fi;
46) Find-Generators(Zi);
47)
48) Result:
49) FIs: Si Fi
50) FCIs + their generators: Si Zi

Algorithm 2 (MCRA-Gen function):

Input: Fi – set of frequent itemsets
Output: table Ci+1 with potentially frequent candidate itemsets.
Plus: key and pred supp fields will be filled in Ci+1.
1) insert into Ci+1
select p[1], p[2], . . . , p[i], q[i]
from Fi p, Fi q
where p[1] = q[1], . . . , p[i − 1] = q[i − 1], p[i] < q[i]; // like in Apriori
2) loop over the rows of Ci+1 (c)
3) {
4) c.key true;
5) c.pred supp = |O| + 1; // number of objects in the database + 1 (imitating +1)
6) S (i − 1)-long subsets of c;
7) loop over the elements of S (s)
8) {
9) if (s /2 Fi) then Ci+1 Ci+1 \ {c}; // remove it if it is rare
10) else {
11) c.pred supp min(c.pred supp, s.support);
12) if (s.key = false) then c.key false; // by Prop. 2
13) }
14) }
15) if (c.key = false) then c.support c.pred supp; // by Th. 2
16) }
17) return Ci+1;

Algorithm 3 (Find-Generators procedure):
Method: fills the gen field of the table Zi with generators
Input: Zi – set of frequent closed itemsets
1) loop over the rows of Zi (z)
2) {
3) S Subsets(FG, z);
4) z.gen S;
5) FG FG \ S;
6) }
7) FG FG [{l 2 Fi | l.key = true ^ l.closed = false};

B.3 Running Example

Consider the following dataset D (Tab. 3) that we use for our examples throughout the paper.

	A	B	C	D	E
1	x	x		x	x
2	x		x		
3	x	x	x		x
4		x	x		x
5	x	x	x		x

Table 3. A toy dataset (D) for the examples

The execution of MCRA on dataset D with min supp = 2 (40%) is illustratedin Tab. 4. The algorithm first performs one database scan to count the supportsof 1-itemsets. The candidate itemset {D} is pruned because it is infrequent. Atthe next iteration, all candidate 2-itemsets are created and stored in C2. Thena database scan is performed to determine the supports of the six potentially frequent candidate itemsets. In C2 there is one itemset that has the same supportas one of its subsets, thus {BE} is not a key generator (see Th(s). 1 and 2). UsingF2 the itemsets {B} and {E} in F1 are not closed because they have a propersuperset in F2 with the same support. The remaining closed itemsets {A} and{C} are copied to Z1 and their generators are determined. In the global listof frequent generators (FG), which is still empty, they have no subsets, whichmeans that both {A} and {C} are generators themselves. The not closed keyitemsets of F1 ({B} and {E}) are added to FG.In C3 there are two itemsets, {ABE} and {BCE}, that have a non-key subset({BE}), thus by Prop. 2 they are not key generators either. Their support valuesare equal to the support of {BE} (Th. 2), i.e. their supports can be determinedwithout any database access. By F3 the itemsets {AB}, {AE}, {BC} and {CE}turn out to be “not closed”. The remaining closed itemsets {AC} and {BE} arecopied to Z2. The generator of {AC} is itself, and the generators of {BE} are{B} and {E}. These two generators are deleted from FG and {AB}, {AE}, {BC}and {CE} are added to FG.At the fourth iteration, it turns out in MCRA-Gen that the newly generatedcandidate itemset contains at least one non-key subset. By Prop. 2 the newcandidate itemset is not a candidate key generator, and its support is determineddirectly in MCRA-Gen by Th. 2. As there are no more candidate generators in C4,from this step on no more database scan is needed.In the fifth iteration no new candidate itemset is found and the algorithmbreaks out from the main loop. The largest frequent closed itemset is {ABCE},its generators are read from FG. When the algorithm stops, all frequent andall frequent closed itemsets with their generators are determined, as shown inTab. 5. In the table the “+” sign means that the frequent itemset is closed.

Table 4. Execution of MCRA on dataset D with min supp = 2 (40%)

DB scan$_1$ →

C_1	pred_supp	key	supp
{A}			4
{B}			4
{C}			4
{D}			1
{E}			4

F_1	key	supp	closed
{A}	yes	4	yes
{B}	yes	4	~~yes~~
{C}	yes	4	yes
{E}	yes	4	~~yes~~

Z_1	supp	gen
{A}	4	
{C}	4	

$FG_{before} = \{\}$
$FG_{after} = \{B, E\}$

DB scan$_2$ →

C_2	pred_supp	key	supp
{AB}	4	yes	3
{AC}	4	yes	3
{AE}	4	yes	3
{BC}	4	yes	3
{BE}	4	~~yes~~	4
{CE}	4	yes	3

F_2	key	supp	closed
{AB}	yes	3	~~yes~~
{AC}	yes	3	yes
{AE}	yes	3	~~yes~~
{BC}	yes	3	~~yes~~
{BE}	no	4	yes
{CE}	yes	3	~~yes~~

Z_2	supp	gen
{AC}	3	
{BE}	4	{B, E}

$FG_{before} = \{B, E\}$
$FG_{after} = \{AB, AE, BC, CE\}$

DB scan$_3$ →

C_3	pred_supp	key	supp
{ABC}	3	yes	2
{ABE}	3	~~yes~~	3
{ACE}	3	yes	2
{BCE}	3	~~yes~~	3

F_3	key	supp	closed
{ABC}	yes	2	~~yes~~
{ABE}	no	3	yes
{ACE}	yes	2	~~yes~~
{BCE}	no	3	yes

Z_3	supp	gen
{ABE}	3	{AB, AE}
{BCE}	3	{BC, CE}

$FG_{before} = \{AB, AE, BC, CE\}$
$FG_{after} = \{ABC, ACE\}$

C_4	pred_supp	key	supp
{ABCE}	2	~~yes~~	2

F_4	key	supp	closed
{ABCE}	no	2	yes

Z_4	supp	gen
{ABCE}	2	{ABC, ACE}

$FG_{before} = \{ABC, ACE\}$
$FG_{after} = \{\}$

C_5	pred_supp	key	supp
∅			

The support values are indicated in parentheses. If MCRA leaves the generatorsof a closed itemset empty, it means that the generator is identical to the closeditemset (as this is the case for {A}, {C} and {AC} in the example). Due tothe property of equivalence classes, the support of a generator is equal to thesupport of its closure.

B.4 The Pascal+ Algorithm

Actually, MCRA can be specified to another algorithm that we call Pascal+. Previously we have seen that MCRA has three main features. Removing the thirdpart of MCRA (associating generators to their closures), we get Pascal+ thatcan filter FCIs among FIs, just like Apriori-Close. To obtain Pascal+ the FindGenerators() procedure calls must be deleted from Algorithm 1 in lines 43 and46.

Table 5. Output of MCRA

All frequent itemsets ($\bigcup_i F_i$)		All frequent closed itemsets with their generators ($\bigcup_i Z_i$)
{a} (4) +	{b, e} (4) +	{a} (4); [{a}]
{b} (4)	{c, e} (3)	{c} (4); [{c}]
{c} (4) +	{a, b, c} (2)	{a, c} (3); [{a, c}]
{e} (4)	{a, b, e} (3) +	{b, e} (4); [{b}, {e}]
{a, b} (3)	{a, c, e} (2)	{a, b, e} (3); [{a, b}, {a, e}]
{a, c} (3) +	{b, c, e} (3) +	{b, c, e} (3); [{b, c}, {c, e}]
{a, e} (3)	{a, b, c, e} (2) +	{a, b, c, e} (2); [{a, b, c}, {a, c, e}]
{b, c} (3)		

Table 6. Comparing sizes of different sets of association rules generated with MCRA

dataset (min_supp)	min_conf	$\mathcal{AR}$ (all strong rules)	$\mathcal{GB}$	$\mathcal{IB}$	$\mathcal{RIB}$	$\mathcal{MNR}$ ($\mathcal{GB} \cup \mathcal{IB}$)	$\mathcal{RMNR}$ ($\mathcal{GB} \cup \mathcal{RIB}$)
D (40%)	50%	50	8	17	13	25	21
T20I6D100K (0.5%)	90%	752,715	232	721,716	91,422	721,948	91,654
	70%	986,058		951,340	98,097	951,572	98,329
	50%	1,076,555		1,039,343	101,360	1,039,575	101,592
	30%	1,107,258		1,068,371	102,980	1,068,603	103,212
C20D10K (30%)	90%	140,651	967	8,254	2,784	9,221	3,751
	70%	248,105		18,899	3,682	19,866	4,649
	50%	297,741		24,558	3,789	25,525	4,756
	30%	386,252		30,808	4,073	31,775	5,040
C73D10K (90%)	95%	1,606,726	1,368	30,840	5,674	32,208	7,042
	90%	2,053,936		42,234	5,711	43,602	7,079
	85%	2,053,936		42,234	5,711	43,602	7,079
	80%	2,053,936		42,234	5,711	43,602	7,079
MUSHROOMS (30%)	90%	20,453	544	952	682	1,496	1,226
	70%	45,147		2,961	1,221	3,505	1,765
	50%	64,179		4,682	1,481	5,226	2,025
	30%	78,888		6,571	1,578	7,115	2,122

C. Finding Minimal Non-Redundant Association Ruleswith MCRA

Generating all strong association rules from frequent itemsets produces too manyrules, many of which are redundant. For instance in dataset D with min supp =2 (40%) and min conf = 50% no less than 50 rules can be extracted. Consideringthe small size of the dataset, 5×5, this quantity is huge. How could we find themost interesting rules? How could we avoid redundancy and reduce the number of rules? Minimal non-redundant association rules (MNR) can help us.By Definitions 1 – 5, an MNR has the following form: the antecedent isa frequent generator, the union of the antecedent and consequent is a frequentclosed itemset, and the antecedent is a proper subset of this frequent closeditemset. MNR also has a reduced subset called RMNR. Since a generator is aminimal subset of its closure with the same support, non-redundant associationrules allow to deduce maximum information with a minimal hypothesis. Theserules form a set of minimal non-redundant association rules, where "minimal"means "minimal antecedents and maximal consequents". Among rules with thesame support and same confidence, these rules contain the most information andthese rules can be the most useful in practice [19]. For the generation of suchrules the frequent closed itemsets and their associated generators are needed.Since MCRA can find both, the output of MCRA can be used directly to generatethese rules.

The algorithm for finding MNR is the following: for each frequent generator P1 find its proper supersets P2 in the set of FCIs. Then add the ruler : P1 → P2 \ P1 to the set of MNR. For instance, using the generator {E}in Fig. 1, three rules can be determined. Rules within an equivalence class formthe generic basis (GB), which are exact association rules (E ⇒B), while rulesbetween equivalence classes are approximate association rules (E → BC andE → ABC). For extracting RMNR the search space for finding frequent closedproper supersets of generators is reduced to equivalence classes that are directneighbors (see Def. 10), i.e. transitive relations are eliminated. Thus, for instance, in the previous example only the first two rules are generated: E ⇒Band E → BC. A comparative table of the different sets of association rules extracted with MCRA are shown in Tab. 6.11 In sparse datasets, like T20I6D100K,the number ofMNR is not much less than the number of AR, however in dense,highly correlated datasets the difference is significant. RMNR always representmuch less rules than AR, in sparse and dense datasets too.As shown in Tab. 5, MCRA finds everything needed for the extraction of minimal nonredundant association rules. For a very quick lookup of frequent closedproper supersets of frequent generators we suggest storing the frequent closeditemsets in the trie data structure.

III. EXPERIMENT AND RESULT

We evaluated MCRA against Apriori and Pascal. We have implemented thesealgorithms in Java using the same data structures, and they are all part of theplatform Coron [24]. The experiments were carried out on an Intel Pentium IV2.4 GHz machine running GNU/Linux operating system, with 512 MB of RAM.All times reported are real, wall clock times as obtained from the Unix timecommand between input and output. Table 7 shows the characteristics of thedatabases used in our evaluation. It shows the number of objects, the numberof different attributes, the average transaction length, and the largest attributein each database.

Table 7. Characteristics of databases

	# Objects	# Attributes	Avg. length	Largest attr.
T20I6D100K	100,000	893	20	1,000
C20D10K	10,000	192	20	385
MUSHROOMS	8,416	119	23	128

The T20I6D100K12 is a sparse dataset, constructed according to the properties of market basket data that are typical weakly correlated data. The numberof frequent itemsets is small, and nearly all FIs are closed. The C20D10K is acensus dataset from the PUMS sample file, while the Mushrooms describemushrooms characteristics. The last two are highly correlated datasets. It hasbeen shown that weakly correlated data, such as synthetic data, constitute easycases for the algorithms that extract frequent itemsets, since few itemsets arefrequent. For such data, all algorithms give similar response times. On the contrary, dense and highly-correlated data constitute far more difficult cases for theextraction due to large differences between the number of frequent and frequentclosed itemsets. Such data represent a huge part of real-life datasets.

3.1 Weakly Correlated Data

The T20I6D100K synthetic dataset mimics market basket data that are typicalsparse, weakly correlated data. In this dataset, the number of frequent itemsets issmall and nearly all frequent itemsets are generators. Apriori, Pascal and MCRAbehave identically. Response times for the T20I6D100K dataset are presentednumerically in Tab. 8.Table 8 also contains some statistics provided by MCRA about the datasets. Itshows the number of FIs, the number of FCIs, the number of frequent generators,the proportion of the number of FCIs to the number of FIs, and the proportionof the number of frequent generators to the number of FIs, respectively. As wecan see in T20I6D100K, above 0.75% minimum support all frequent itemsets areclosed and generators at the same time. It means that each equivalence class hasonly one element. Because of this, MCRA and Pascal cannot use the advantage ofpattern counting inference and they work exactly like Apriori.

3.2 Strongly Correlated Data

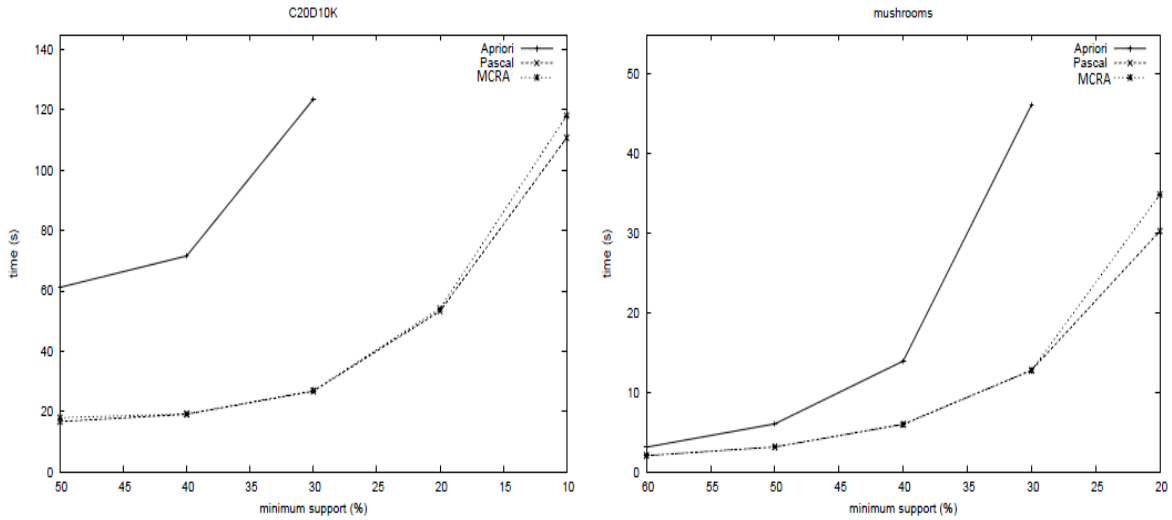
Response times obtained for the C20D10K and Mushrooms datasets are givennumerically in Tab. 8, and graphically in Fig. 2, respectively. In these twodatasets, the number of frequent generators is much less than the total number of frequent itemsets. Hence, using pattern counting inference, MCRA has toperform much fewer support counts than Apriori. We can observe that in allcases the execution times of MCRA and Pascal are almost identical: adding thefrequent closed itemset derivation and the identification of their generators tothe frequent itemset discovery does not induce serious additional computationtime. Apriori is very efficient on sparse datasets, but on strongly correlated datathe other two algorithms perform much better.

Table 8. Response times of MCRA and other statistics

min_supp (%)	Apriori	Pascal	MCRA	# FIs	# FCIs	# FGs	$\frac{\#FCIs}{\#FIs}$	$\frac{\#FGs}{\#FIs}$
T20I6D100K								
2	72.67	71.15	71.13	378	378	378	100.00%	100.00%
1	107.63	106.24	107.69	1,534	1,534	1,534	100.00%	100.00%
0.75	134.49	132.00	133.00	4,710	4,710	4,710	100.00%	100.00%
0.5	236.10	228.37	230.17	26,836	26,208	26,305	97.66%	98.02%
0.25	581.11	562.47	577.69	155,163	149,217	149,447	96.17%	96.32%
C20D10K								
50	61.18	16.68	17.94	1,823	456	456	25.01%	25.01%
40	71.60	19.10	19.22	2,175	544	544	25.01%	25.01%
30	123.57	26.74	26.88	5,319	951	967	17.88%	18.18%
20	334.87	53.28	54.13	20,239	2,519	2,671	12.45%	13.20%
10	844.44	110.78	118.09	89,883	8,777	9,331	9.76%	10.38%
MUSHROOMS								
60	3.10	2.04	2.05	51	19	21	37.25%	41.18%
50	6.03	3.13	3.13	163	45	53	27.61%	32.52%
40	13.93	6.00	5.94	505	124	153	24.55%	30.30%
30	46.18	12.79	12.75	2,587	425	544	16.43%	21.03%
20	554.95	30.30	34.88	53,337	1,169	1,704	2.19%	3.19%

3.3 Comparing Pascal+ and Pascal

We also compared the efficiency of Pascal+ with Pascal. Pascal+ gives almostequivalent response times to Pascal on both weakly and strongly correlated data,i.e. the filtering of closed itemsets among frequent itemsets is not an expensivestep. As Pascal is more efficient than Apriori on strongly correlated data (seeTab. 8), Pascal+ is necessarily more efficient than Apriori-Close. If we need bothfrequent and frequent closed itemsets then Pascal+ is recommended instead ofApriori-Close.



IV.CONCLUSION

In this paper we presented a multifunctional itemset miner algorithm called MCRA,which is a refinement of Pascal. With pattern counting inference, using the generators of equivalence classes, it can reduce the number of itemsets counted andthe number of database passes. In addition, it can identify frequent closed itemsets among frequent itemsets, and it can associate generators to their closure.Weshowed that these extra features are required for the generation of minimal nonredundant association rules. MCRA can also be specified to another algorithm thatwe call Pascal+. Pascal+ finds both frequent and frequent closed itemsets, likeApriori-Close. We compared the performance of MCRA with Apriori and Pascal.The results showed that MCRA gives almost equivalent response times to Pascalon both weakly and strongly correlated data, though MCRA also identifies closeditemsets and their generators.An interesting question is the following: can the idea of MCRA be generalizedand used for any arbitrary frequent itemset miner algorithm, be it either breadthfirst or depth-first? Could we somehow extend these algorithms in a universalway to produce such results that can be used directly to generate not only allstrong association rules, but minimal non-redundant association rules too? Wethink that the answer is positive, but detailed study of this will be subject offurther research.

V. REFERENCE

1. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining minimalnon-redundant association rules using frequent closed itemsets. In Lloyd, J.et al..,ed.: Proc. of the Computational Logic (CL'00). Volume 1861 of Lecture Notes inArtificial Intelligence – LNAI., Springer (2000) 972–986.
2. Kryszkiewicz, M.: Representative association rules. In: PAKDD '98: Proceedings of the Second Pacific-Asia Conference on Research and Development in KnowledgeDiscovery and Data Mining, London, UK, Springer-Verlag (1998) 198–209.
3. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informativesr´esultant d'un tableau de donn´ees binaires. Math´ematiqueset Sciences Humaines95 (1986) 5–18.
4. Luxenburger, M.: Implications partielles dans un contexte. Math´ematiques, Informatiqueet Sciences Humaines 113 (1991) 35–55.
5. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Closed set based discovery ofsmall covers for association rules. In: Proc. 15emes Journees Bases de DonneesAvancees, BDA. (1999) 361–381.
6. Kryszkiewicz, M.: Concise representations of association rules. In: Pattern Detection and Discovery. (2002) 92–109.
7. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of associationrules using closed itemset lattices. Inf. Syst. 24(1) (1999) 25–46.
8. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. Lecture Notes in Computer Science 1540 (1999) 398–416.
9. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing IcebergConcept Lattices with TITANIC. Data and Knowledge Engineering 42(2) (2002)189–222.
10. Zaki, M.J., Hsiao, C.J.: CHARM: An Efficient Algorithm for Closed Itemset Mining. In: SIAM International Conference on Data Mining SDM'02. (2002) 33–43.
11. Kryszkiewicz, M.: Concise representation of frequent patterns based on disjunctionfree generators. In: ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society (2001)305–312.
12. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequentpatterns with counting inference. SIGKDD Explor. Newsl. 2(2) (2000) 66–75.
13. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Approximation of frequency queries bymeans of free-sets. In: Proceedings of PKDD 2000, Lyon, France, Springer Berlin/ Heidelberg (2000) 75–85.
14. Bykowski, A., Rigotti, C.: A condensed representation to find frequent patterns.In: PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGARTsymposium on Principles of database systems, ACM Press (2001) 267–273.
15. Kryszkiewicz, M., Gajek, M.: Why to apply generalized disjunction-free generatorsrepresentation of frequent patterns? In Hacid, M.S., Ra, Z., Zighed, D., Kodratoff,Y., eds.: Proceedings of Foundations of Intelligent Systems: 13th

InternationalSymposium, ISMIS 2002, Lyon, France, Springer-Verlag Berlin / Heidelberg (2002)383–392.

16. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In: PKDD'02: Proceedings of the 6thEuropean Conference on Principles of Data Mining andKnowledge Discovery, London, UK, Springer-Verlag (2002) 74–85.
17. Calders, T., Goethals, B.: Depth-first non-derivable itemset mining. In: Proc.SIAM Int. Conf. on Data Mining SDM '05, Newport Beach (USA). (2005).
18. Harms, S., Deogun, J., Saquer, J., Tadesse, T.: Discovering representative episodalassociation rules from event sequences using frequent closed episode sets and eventconstraints. In: ICDM '01: Proceedings of the 2001 IEEE International Conferenceon Data Mining, Washington, DC, USA, IEEE Computer Society (2001) 603–606.
19. Pasquier, N.: Mining association rules using formal concept analysis. In: Proc. Ofthe 8th International Conf. on Conceptual Structures (ICCS '00), Shaker-Verlag(2000) 259–264.
20. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Pascal : un algorithme d'extraction des motifs frquents. Technique et science informatiques 21(1)(2002) 65–95.
21. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations.Springer, Berlin/Heidelberg (1999).
22. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in knowledge discovery and data mining.American Association for Artificial Intelligence (1996) 307–328.
23. Zaki, M.J.: Scalable Algorithms for Association Mining. IEEE Transactions onKnowledge and Data Engineering 12(3) (2000) 372–390.
24. Szathmary, L., Napoli, A.: Coron : A framework for levelwise itemset mining algorithms. In Ganter, B., Godin, R., Mephu Nguifo, E., eds.: Suppl. Proc. of ICFCA'05, Lens, France. (2005) 110–113.