Mining Positive and Negative Association Rules from Large Databases

Chris Cornelis
Dept. Appl. Math. and Comp. Sci.
Ghent University
Ghent, Belgium
Chris.Cornelis@UGent.be

Peng Yan, Xing Zhang, Guoqing Chen
School of Economics and Management
Tsinghua University
Beijing, China
{yanp,zhangx4,chengq}@em.tsinghua.edu.cn

***Abstract*— This paper is concerned with discovering positive and negative association rules, a problem which has been addressed by various authors from different angles, but for which no fully satisfactory solution has yet been proposed. We catalogue and critically examine the existing definitions and approaches, and we present an Apriori-based algorithm that is able to find all valid positive and negative association rules in a support-confidence framework. Efficiency is guaranteed by exploiting an upward closure property that holds for the support of negative association rules under our definition of validity.**

***Keywords*—association rules, data mining, Apriori**

I. Introduction

The idea of association rule (AR) mining already dates back to Hájek et al [7]. Its application to market basket analysis, i.e. to the discovery of co-occurence relationships among purchased items in supermarket transaction databases, gained enormous popularity after its re-introduction by Agrawal et al [2] in the mid-nineties. Apart from algorithmic efficiency and ease of implementation by algorithms like Apriori [12], its appeal is also due to its wide applicability; researchers have successfully exported the AR concept —originally specific to binary data tables— to a multitude of domains, involving quantitative, hierarchical, fuzzy, and many other kinds of databases. At the heart of all these methods is the ability to predict that, given the *presence* in a database record ("transaction") of one pattern ("frequent itemset") X, the record is likely to contain another pattern Y as well, concisely summed up by the (positive) rule $X \Rightarrow Y$. Recently, the problem of identifying negative associations (or "dissociations"), addressing also the *absence* of itemsets, has been explored and considered relevant (see e.g [1], [3], [4], [5], [11], [13], [14], [15], [16], [17]). For instance, a shop manager may like to be informed that "customers who buy an IBM notepad are unlikely to buy a D-link network card".

Mining negative association rules, however, raises a number of critical issues. First of all, in typical supermarket transaction databases, there are thousands of items (wares) and each record (customer transaction) only contains a few of them. For a database containing 10,000 items, such that each customer on average buys 10 of them, the density of the database is 0.1%. From the perspective of negative patterns (indicating absence of items), the density is a soaring 99.9%, leading to explosive amounts of, mostly uninteresting, rules. Secondly, the complexity of AR mining algorithms is exponential in the number of items; if for each item from the database a corresponding negated item (indicating absence of the original item) is considered, the computational costs will skyrocket. As a consequence, the way a valid negative association rule is defined plays an important role. Finally, as we will point out further on, the introduction of negative association rules invalidates some important pruning aids used to restrict the search space and guarantee efficiency in classical algorithms.

Existing approaches have tried to address these problems by incorporating attribute correlations or rule interestingness measures to filter out unimportant rules, or by relying on additional background information concerning the data. As opposed to this, the aim of this paper is to propose a computationally workable approach that stays within the strict bounds of the original support-confidence framework. Such an approach has the advantage of being intuitive to users (no additional parameters required), and may serve as a "skeleton" for the development of more specialized mining algorithms.

The paper is structured as follows: the next section recalls preliminaries about ARs, and compares two alternative ways to define negative ARs. In Section 3, existing strategies to mining negative ARs are reviewed. The core of the paper is Section 4, where first we propose a suitable definition of valid negative association rule, and then proceed to spell out, and evaluate, a new Apriori-based algorithm for finding all valid positive and negative association rules. Section 5 offers a brief conclusion and suggests future work in this direction.

II. Positive and Negative Association Rules

A. Support-Confidence Framework of Positive ARs

Let $\mathcal{D} = \{t_1, t_2, \ldots, t_n\}$ be a relational database of n records (or transactions) with a set of binary attributes (or items) $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$. For an itemset $X \subseteq \mathcal{I}$ and a transaction t in $\mathcal{D}$, 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 $supp(X) = \frac{|D_X|}{n}$, i.e the fraction of transactions containing X. A (positive)

association rule is of the form: $X \Rightarrow Y$, with $X, Y \subset \mathcal{I}$, $X \cap Y = \emptyset$. Support and confidence of $X \Rightarrow Y$ are defined as $supp(X \Rightarrow Y) = supp(XY)$ and $conf(X \Rightarrow Y) = \frac{supp(XY)}{supp(X)}$ respectively[1]. A valid positive association rule has support and confidence greater than given thresholds ms and mc. [2] Furthermore, an itemset X is called frequent if $supp(X) \geq ms$.

B. Negative Association Rules

What exactly constitutes a negative association rule? In literature, two formats prevail.

Definition I: in [1], [11], [15], [17], the traditional definition of itemset is maintained (so X, $Y \subset \mathcal{I}$), 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 ([15], [17]) that $supp(X \Rightarrow \neg Y) = supp(X\neg Y) = supp(X) - supp(XY)$ for $X, Y \subset \mathcal{I}$, 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 $Y \subset \mathcal{I}$ such that and $X \cap Y = \emptyset$, $\neg(XY)$ 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 will arise. In Section IV, we will show how to exploit the upward closure property to define an alternative definition of validity for negative ARs and to restrict the search space of our mining algorithm.

Definition II: in [3], [4], [16] authors view each item's opposite as a new item which can be inserted into $\mathcal{I}$ to derive $\mathcal{I}' = \mathcal{I} \cup \{\neg i_1, \neg i_2, \ldots, \neg i_m\}$. Since no transaction contains both an item and its opposite, we can restrict our attention to those itemsets that do not contain i_j and $\neg i_j$ simultaneously for $j = 1, \ldots, m$. An example of a possible negative rule under this definition is $\{i_1\} \Rightarrow \{\neg i_2, \neg i_3\}$; a transaction t supports it if $i_1 \in t$, $i_2 \notin t$ and $i_3 \notin t$. Hence the meaning of this rule is that "the appearance of i_1 in a transaction t induces that neither i_2 *nor* i_3 is likely to appear also in t". A record that contains i_1 and i_2, but not i_3, does *not* support this rule. It was shown in [16] that the support of itemsets in $\mathcal{I}'$ (and hence also support and confidence of negative ARs under Definition II) can be deduced from that of itemsets in $\mathcal{I}$.

Unlike Definition I, this definition allows pruning based on the downward closure property of $supp$, and hence is easier to interpret and to implement. Unfortunately, the total number of possible rules has now grown to $5^m - 2(3^m) + 1$, which is much more than under Definition I; it can also be calculated that the sum of support for all possible rules is $3^m - 2^{m+1} + 1$, which means that the average support equals 0 when m is large enough. Practically, these observations make the use of Definition II under the classical support-confidence framework reasonable only in relatively small databases.

[1]In this paper, we abbreviate $X \cup Y$ to XY for itemsets X, Y of $\mathcal{I}$.

III. Related Work

Based on the previous discussion, two important (and closely intertwined) problems should be solved in negative AR mining, that is: to keep the number of discovered rules in check and to find efficient algorithms. To reduce the search space, and to improve the quality of the mined rules, it is fruitful to introduce additional measures apart from support and confidence. Most papers consider some kind of correlation between attributes. For instance, Brin et al [4] used χ^2 tests to detect item dependencies within an itemset. An efficient algorithm for finding correlated itemsets ensued from the upward closure of the χ^2 statistic (if an itemset X is correlated, i.e. contains dependencies, then so is every superset of X), and this information may be used for finding negative ARs under Definition II. In a similar vein, Aggarwal and Yu [1] evaluated correlations in itemsets by a measure they called "collective strength", and provided an efficient algorithm to identify so-called "strongly collective itemsets" from databases (leading to negative ARs under Definition I). Some disadvantages of these approaches are that (a) they are not in line with (i.e., cannot be put on top of) the support-confidence framework and (b) afterwards, additional operations are still needed to determine the exact form of negative rules. On another count, *heuristic methods*, like the ones proposed by Savasere et al [11] and Yuan et al [17] incorporate domain knowledge, usually a taxonomy, into the algorithms, and only part of the rules are focused on to reduce the scale of the problem. In these methods, expected support or confidence of an itemset (based on itemsets' positions in the taxonomy) is compared to the actual value of these measures, and the larger the deviation, the more interesting such itemsets are considered to be. While they may reveal interesting negative ARs, domain knowledge may often not be readily available, limiting the applicability of such approaches.

On the other hand, *Apriori-based* implementations are very attractive since they can guarantee efficiency for large scale databases, do not depend on other algorithms and do not require additional information on the data. For example, Wu et al [15] presented an Apriori-based framework for mining both positive and negative ARs (in the sense of Definition I) that focuses on rule dependency measures (rather than on itemset dependency measures), and in particular uses Piatetsky-

Shapiro's argument [10] that *a rule $X \Rightarrow Y$ is not interesting if $supp(X \Rightarrow Y) \approx supp(X)supp(Y)$*. For instance, for the rule $X \Rightarrow \neg Y$ to be valid, the authors in [15] required $interest(X, \neg Y) = |supp(X\neg Y) - supp(X)supp(\neg Y)| \geq mi$, with mi an additional threshold. Furthermore, $X \Rightarrow \neg Y$ is only considered a valid negative AR if both X and Y are frequent. Apart from being intuitive (to be relevant, mined patterns should involve frequent itemsets only), this criterion also warrants a significant efficiency boost: if X is infrequent, then no negative ARs with X or any of its supersets in the antecedent or consequent have to be considered. There are however substantial problems with this approach, since the algorithm proposed in [15] cannot generate all valid negative ARs. Without going into the details here, the problem is due to the fact that $interest$, which is used during algorithm execution for pruning itemsets, does not have a downward closure property like $supp$.

Another Apriori-based algorithm was given by Antonie and Zaïane [3] for the purpose of simultaneously generating positive ARs and (a subclass of) negative ARs under Definition II, using Pearson correlation coefficient for measuring interestingness. Their approach suffers from a similar problem like the one proposed in [15].

IV. Efficient Discovery of Both Positive and Negative Association Rules

The above-described Apriori-based implementations are efficient but cannot generate all valid positive and negative ARs. In this section, we try to solve that problem without paying too high a price in terms of computational costs. In accordance with the observations made in Section II-B, we work with negative ARs under Definition I. For simplicity, we also limit ourselves to support and confidence to determine the validity of ARs.

A. Valid Positive and Negative Association Rules

By a valid AR, we mean any expression $C_1 \Rightarrow C_2$, $C_1 \in \{X, \neg X\}$, $C_2 \in \{Y, \neg Y\}$, $X, Y \subset \mathcal{I}$, $X \cap Y = \emptyset$, s.t.

1) $supp(C_1 \Rightarrow C_2) \geq ms$
2) $supp(X) \geq ms, supp(Y) \geq ms$
3) $conf(C_1 \Rightarrow C_2) \geq mc$
4) $C_1 \Rightarrow C_2$ is minimal w.r.t. support of its negative itemsets

In 1) it is possible to replace ms by another value (support threshold for negative ARs, e.g. based on the density of $\mathcal{D}$), but this is not necessary; in general, a larger threshold yields more significant, and hence more interesting negative rules. Condition 2) is based on the work of Wu et al [15] (see Section III). The distinguishing part of this definition is condition 4); it means that if $C_1 = \neg X$, then there should not exist $X' \subset X$ such that $supp(\neg X' \Rightarrow C_2) \geq ms$ (analogously for C_2). This condition is based on the upward closure property of $supp$ for negative itemsets from Section II-B; it is reasonable to consider only the boundary of negative ARs with minimal negative parts, since clearly the non-negative ARs are redundant. Like the downward closure property for positive itemsets, it will be used as an efficient pruning device in our algorithm.

Our algorithm for mining both positive and negative valid ARs (PNAR) is built up conceptually around a partition of the itemset space into 4 parts, which is shown in Fig. 1. Each part is then dealt with separately. Before spelling out the steps of the algorithm, we introduce the relevant notations.

B. Notations

X, Y: positive itemsets
I, I': itemsets, which can be positive or negative;
$L(P_i)$: frequent itemsets in Part i;
$L(P_i)_k$: k-length frequent itemsets in Part i;
$L(P_4)_{k,p}$: in Part 4, frequent itemsets with k-length negative part and p-length positive part;
$C(P_i)$: candidate itemsets in Part i (analogously: $C(P_i)_k, C(P_4)_{k,p}$);
$S(P_i)$: positive itemsets whose support needs to be calculated via DB scan in Part i; if $i = 1$, $S(P_i) = C(P_i)$ (analogously: $S(P_i)_k, S(P_4)_{k,p}$);
S: the set of positive itemsets whose supports are known; this includes $L(P_1)$, $C(P_1)$ and $S(P_i)$ after DB scan;
for itemset $I = \neg XY$, I.negative$= X$ and I.positive$= Y$

C. Main Procedure of PNAR

1: Generate all positive frequent itemsets $L(P_1)$, and put $S = C(P_1)$
2: For all itemsets I in $L(P_1)$, insert I into $L(P_2)$ if $1 - supp(I) \geq ms$
3: Generate all negative frequent itemsets $L(P_3)$
4: Generate all negative frequent itemsets $L(P_4)$
5: Generate all valid positive and negative ARs

Line 1 is the same as in classical AR mining, and can be implemented by Apriori, but also by other algorithms like Frequent-Pattern Tree [6], Close [9], Charm [18], ... The supports in Line 2 can be immediately derived after we get all frequent positive itemsets, since $supp(\neg I) = 1 - supp(I)$, for I in $\mathcal{I}$. For line 3 and line 4, we can easily calculate the support of an itemset $\neg XY$ (or $\neg X\neg Y, X\neg Y$) in Part 3 and Part 4 if we know the support of X, Y and XY. Since antecedent and consequent of a negative rule must be frequent by our definition, for all such potential valid negative ARs, $supp(X)$ and $supp(Y)$ are known. The computational efficiency of the algorithm depends highly on the number of itemsets $I = XY$ that need further checking via database scan. In the following two subroutines, the upward closure property of $supp$ is used to reduce that number.

D. Generating Frequent Itemsets in P_3

1: $L(P_3) = \emptyset, C(P_3) = \emptyset, S(P_3) = \emptyset$
2: $C(P_3)_2 = \{\neg\{i_1\}\neg\{i_2\} | i_1, i_2 \in L(P_1)_1, i_1 \neq i_2\}$
3: **for** $\{k = 2; C(P_3)_k \neq \emptyset; k++\}$ **do**
4: **for all** $I = \neg X\neg Y \in C(P_3)_k$ **do**
5: **if** $supp(I) \geq ms$ **then**
6: insert I into $L(P_3)_k$
7: **else**

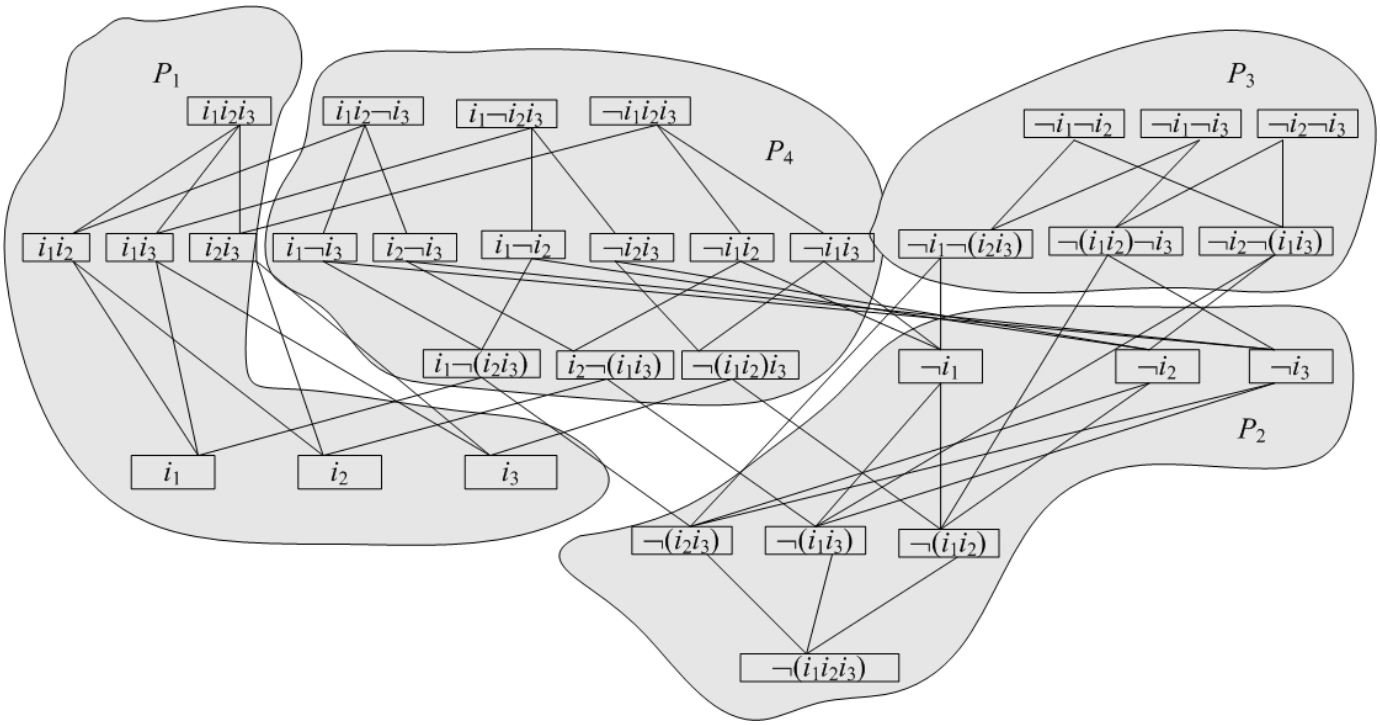

Fig. 1. Partition of itemset space into four parts

```
8:        for all i ∉ XY do
9:          Cand =check_candidates(I, i)
10:         // Cand ⊆ {¬(X{i})¬Y, ¬X¬(Y{i})}
11:         C(P3)k+1 = C(P3)k+1 ∪ Cand
12:         if Cand ≠ ∅, XY{i} ∉ S and
            (∄I' ⊂ XY{i})(supp(I') = 0) then
13:           insert XY{i} into S(P3)k+1
14:         end if
15:       end for
16:     end if
17:   end for
18:   compute support of itemsets in S(P3)k+1
19:   S = S ∪ S(P3)k+1
20: end for
```

We start the algorithm from negative itemsets with 2 items, and then add items to each of the negative parts. We discuss some of the steps in detail:

- Line 3, the algorithm will terminate when no candidate itemsets are generated.
- Line 5, if $supp(I) = supp(\neg X \neg Y) \geq ms$, because of the minimality constraint in the definition of negative ARs, no item will be added to the $\neg X$ or $\neg Y$ part because obviously $I' = \neg(X \cup \{i\})\neg Y$ and $I' = \neg X \neg(Y \cup \{i\})$ will be frequent (upward closure property of $supp$). It should be noted that it is not necessary to scan $\mathcal{D}$ to calculate support of 2-length itemsets in $C(P_3)_2$ because all related positive itemsets must be in $C(P_1)_2$.
- Line 8-9, for every item i not yet in X or Y, function check_candidates generates the candidate itemsets $\neg(X\{i\})\neg Y$ and $\neg X\neg(Y\{i\})$ and then checks whether they can be pruned. For instance, for $I' = \neg(X\{i\})\neg Y$, it should hold that $X\{i\}$ is frequent, i.e. $X\{i\} \in L(P_1)$; moreover if there exists $\neg X'\neg Y'$ in $L(P_3)$ such that $X' \subseteq X\{i\}$ and $Y' \subseteq Y$, then I' needs not be considered according to the upward closure property.
- Line 12-13, $supp(XY\{i\})$ has to be checked if it has not yet been computed and if none of its subsets is known to have support 0 (in which case $supp(XY\{i\}) = 0$ by the downward closure property of $supp$).
- Line 18, this step needs DB scan.

E. Generating Frequent Itemsets in P_4

For the itemsets with both a positive part and a negative part, we first generate frequent itemsets with one negative item, then two negative items, etc. Both the downward and upward closure properties are using as pruning devices.

```
1:  L(P4) = ∅, C(P4) = ∅, S(P4) = ∅
2:  C(P4)1,1 = {¬{i1}{i2} | i1, i2 ∈ L(P1)1, i1 ≠ i2}
3:  for {k = 1; C(P4)k,1 ≠ ∅; k++} do
4:    for {p = 1; C(P4)k,p ≠ ∅; p++} do
5:      for all I ∈ C(P4)k,p do
6:        if supp(I) ≥ ms then
7:          insert I into L(P4)k,p
8:        end if
9:      end for
10:     for all joinable I1, I2 ∈ L(P4)k,p do
11:       X = I1.negative, Y = I1.positive∪I2.positive
12:       I = ¬XY
13:       if (∄X' ⊂ X)(supp(¬X'Y) ≥ ms) and
```

```
            (∄Y′ ⊂ Y)(supp(¬XY′) < ms) then
14:           insert I into C(P4)k,p+1
15:           if XY ∉ S and ∄I′ ⊂ XY, supp(I′) = 0 then
16:             insert XY into S(P4)k,p+1
17:           end if
18:         end if
19:       end for
20:       compute support of itemsets in S(P4)k,p+1
21:       S = S ∪ S(P4)k,p+1
22:     end for
23:     for all X ∈ L(P1)k+1, i ∈ L(P1)1 do
24:       if (∄X′ ⊂ X)(¬X′{i} ∈ L(P4) then
25:         C(P4)k+1,1 = C(P4)k+1,1 ∪ ¬X{i}
26:       end if
27:     end for
28: end for
```

Below are some comments pertaining to the above pseudocode:

- Line 10–12 correspond to the join operation in classical AR mining. I_1 and I_2 are joinable if $I_1 \neq I_2$, I_1.negative = I_2.negative, I_1.positive and I_2.positive share the same $k-1$ items, and I_1.positive$\cup I_2$.positive $\in L(P_1)_{p+1}$. For example, $(\neg X)(Y'\{i\})$ and $(\neg X)(Y'\{j\})$ generate $(\neg X)(Y'\{i,j\})$.
- Line 13–14 exploit the upward and downward closure property of negative itemsets. For example, for a candidate itemset $\neg(\{i_1,i_2\})\{i_3,i_4\}$, it requires that $\neg(\{i_1\})\{i_3,i_4\}$ and $\neg(\{i_2\})\{i_3,i_4\}$ are not frequent, and that $\neg(\{i_1,i_2\})\{i_3\}$ and $\neg(\{i_1,i_2\})\{i_4\}$ are frequent. Note that due to the structure of the algorithm, the supports of $X'Y$ and $\neg XY$ are known.
- Line 20 is the only place where a DB scan occurs.
- Lines 23–27 generate candidate itemsets with k–length negative and 1-length positive parts. Candidates are filtered by the condition on Line 24.

F. Generating Rules

Rule generation, then, is straightforward. According to their particular form, we distinguish four classes: R_1 ($X \Rightarrow Y$), R_2 ($\neg X \Rightarrow \neg Y$), R_3 ($X \Rightarrow \neg Y$) and R_4 ($\neg X \Rightarrow Y$).

```
1: generate all positive association rules in R1
2: for all I = ¬X¬Y ∈ L(P3) do
3:   if conf(¬X ⇒ ¬Y) ≥ mc then
4:     insert ¬X ⇒ ¬Y into R2
5:   end if
6: end for
7: for all I = ¬XY ∈ L(P4) do
8:   if conf(X ⇒ ¬Y) ≥ mc then
9:     insert X ⇒ ¬Y into R3
10:  end if
11:  if conf(¬X ⇒ Y) ≥ mc then
12:    insert ¬X ⇒ Y into R4
13:  end if
14: end for
```

It can be verified (detailed proof omitted) that PNAR is complete, i.e. finds all valid positive and negative ARs.

G. Experimental Results

To test the efficiency of PNAR, we have applied it to synthetic data generated according to the supermarket basket data generation algorithm described in [12] with changing values for ms, n (number of transactions), m (number of items) and $|T|$ (average transaction size). The default values for these parameters are $ms = 0.01$, $n = 1000K$, $m = 1000$, $|T| = 10$. Moreover, $mc = 0.7$ and the average size of maximal frequent itemsets (i.e., without frequent supersets) is taken to be 4.

We have compared PNAR with a naive, straightforward algorithm called S-PNAR (simple PNAR). S-PNAR mines all valid positive and negative ARs by the following steps: (1) find all frequent itemsets, i.e. $L(P_1)$, (2) for X and Y in $L(P_1)$, compute $supp(XY)$ if it is not yet known, (3) check the validity of all rules $X \Rightarrow Y$, $\neg X \Rightarrow \neg Y$, $X \Rightarrow \neg Y$ and $\neg X \Rightarrow Y$. The results are shown in Figure 2. It is clear from these graphs that PNAR has good scalability for the different parameters, and that the efficiency gain thanks to the proposed optimizations is significant.

Furthermore, note that since PNAR needs additional passes over the database to derive all required supports, obviously it is more time-consuming than Apriori. However, Apriori only finds positive ARs; moreover, experiments have revealed that the execution time of PNAR is never greater than 3.5 times that of Apriori. This is especially remarkable since we noted in Section II-B we have seen that the number of potentially valid positive and negative ARs is four times larger than that of positive ARs only. Finally, we note that a comparison with the algorithms in [3] and [15] is not possible since the latter do not find all valid positive and negative ARs.

V. Concluding Remarks and Future Work

Negative AR mining is a highly relevant, yet difficult problem in data mining which recently has caught many researchers' interest. This paper has contributed to this emerging topic by cataloguing, and identifying problems with, existing negative AR definitions and mining approaches, and by proposing a new Apriori-based algorithm (PNAR) that exploits the upward closure property of negative association rules under Definition I. Compared to a naive implementation, PNAR is very efficient in finding all positive and negative ARs within a reasonable time framework. The present algorithm does not make use of interestingness measures; for future work, we plan to incorporate them into our implementation, in order to improve the quality and usefulness of the mined negative association rules, and to further reduce the computational workload. Among some of the relevant applications of this work, we envisage PNAR-based classifiers that generalize the popular CBA (Classification Based on Association rules [8]) tools, as well as mining for negative sequential patterns such as "customers who bought shampoo are unlikely to buy it again within the next month".

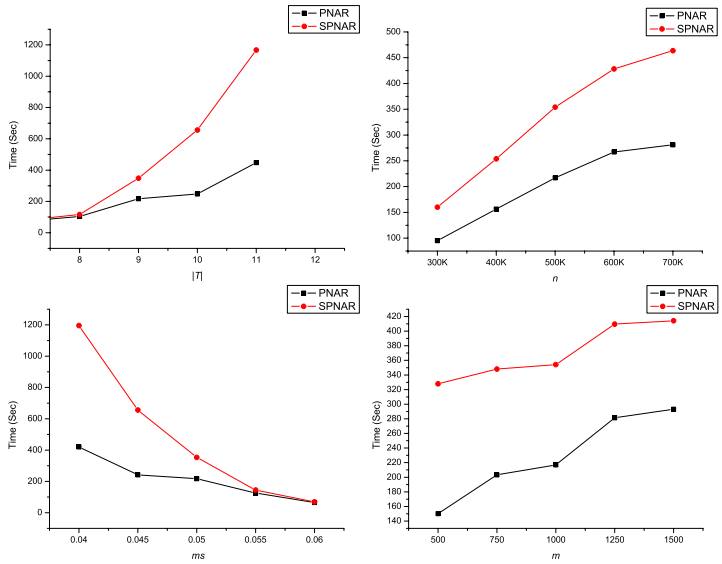

Fig. 2. Comparison between PNAR and S-PNAR for changing values of $|T|$,n, ms and m.

ACKNOWLEDGMENT

This work was partly supported by the National Natural Science Foundation of China (70231010/70321001) and the Bilateral Scientific and Technological Cooperation Between China and Flanders (174B0201). Chris Cornelis would like to thank the Research Foundation–Flanders for funding his research.

REFERENCES

[1] C. C. Aggarwal and P.S. Yu, "A New Framework for Itemset Generation", *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998, pp. 18–24.

[2] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", *Proc. ACM-SIGMOD Intl. Conf. on Management of Data*, 1993, 207–216.

[3] M.L. Antonie and O.R. Zaïane, "Mining Positive and Negative Association Rules: an Approach for Confined Rules", *Proc. Intl. Conf. on Principles and Practice of Knowledge Discovery in Databases*, 2004, pp 27–38.

[4] S. Brin, R. Motwani and C. Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations", *Proc. ACM SIGMOD on Management of Data*, 1997, pp 265–276.

[5] O. Daly and D. Taniar, "Exception Rules Mining Based On Negative Association Rules", *Lecture Notes in Computer Science*, Vol. 3046, 2004, pp 543–552.

[6] J. Han, J. Pei, Y. Yin and R. Mao, "Mining Frequent Patterns without Candidate Generation: a Frequent-Pattern Tree Approach", *Data Mining and Knowledge Discovery*, Vol. 8, 2004, pp 53–87.

[7] P. Hájek, I. Havel and M. Chytil, "The GUHA Method of Automatic Hypotheses Determination", *Computing*, Vol. 1, 1966, pp 293–308.

[8] B. Liu, W. Hsu and Y. Ma, "Integrity Classification and Association Rule Mining", *Proc. Fourth Intl. Conference on Knowledge Discovery and Data Mining*, New York, NY, 1998, pp 80–86.

[9] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, "Efficient Mining of Association Rules Using CLOSED Itemset Lattices", *Information Systems*, Vol. 24(1), 1999, pp 25–46.

[10] G. Piatetsky-Shapiro, "Discovery, Analysis and Presentation of Strong Rules", *Knowledge Discovery in Databases*, AAAI/MIT, Menlo Park, CA, 1991, pp 229–248.

[11] A. Savasere, E. Omiecinski and S. Navathe, "Mining for Strong Negative Associations in a Large Database of Customer Transactions", *Proc. Intl. Conf. on Data Engineering*, 1998, pp 494–502.

[12] R. Srikant and R. Agrawal, "Fast Algorithms for Mining Association Rules", *Proc. VLDB Conference*, 1994, pp 487–499.

[13] D.R. Thiruvady and G.I. Webb, "Mining Negative Association Rules Using GRD", *Proc. Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining*, 2004, pp 161–165.

[14] Q. Wei and G. Chen, "Association Rules with Opposite Items in Large Categorical Database", *Proc. Intl. Conf. on Flexible Query Answering Systems*, 2000, pp 507–514.

[15] X. Wu, C. Zhang and S. Zhang, "Efficient Mining of Both Positive and Negative Association Rules", *ACM Trans. on Information Systems*, vol. 22(3), 2004, pp 381–405.

[16] P. Yan, G. Chen, C. Cornelis, M. De Cock and E.E. Kerre, "Mining Positive and Negative Fuzzy Association Rules", LNCS 3213, 2004, pp 270–276.

[17] X. Yuan, B.P. Buckles, Z. Yuan and J. Zhang, "Mining Negative Association Rules", *Proc. Seventh Intl. Symposium on Computers and Communication*, Italy, 2002, pp 623–629.

[18] M.J. Zaki and C.J. Hsiao, "CHARM: an Efficient Algorithm for Closed Itemset Mining", *Proc. Second SIAM Intl. Conf. on Data Mining*, Arlington, VA, 2002, pp 12–28.