Prince: An Algorithm for Generating Rule Bases Without Closure Computations

Conference Paper *in* Lecture Notes in Computer Science · September 2005

DOI: 10.1007/11546849_34 · Source: dx.doi.org

3 authors:

Tarek Hamrouni
Tunis El Manar University
74 PUBLICATIONS **871** CITATIONS

SEE PROFILE

Sadok Ben Ben Yahia
University of Southern Denmark
517 PUBLICATIONS **4,444** CITATIONS

SEE PROFILE

Yahya Slimani
Manouba University
190 PUBLICATIONS **1,677** CITATIONS

SEE PROFILE

Prince: An Algorithm for Generating Rule Bases Without Closure Computations*

T. Hamrouni, S. Ben Yahia, and Y. Slimani

Département des Sciences de l'Informatique,
Campus Universitaire, 1060 Tunis, Tunisie
{sadok.benyahia, yahya.slimani}@fst.rnu.tn

Abstract. The problem of the relevance and the usefulness of extracted association rules is becoming of primary importance, since an overwhelming number of association rules may be derived, even from reasonably sized databases. To overcome such drawback, the extraction of reduced size generic bases of association rules seems to be promising. Using the concept of minimal generator, we propose an algorithm, called PRINCE, allowing a shrewd extraction of generic bases of rules. To this end, PRINCE builds the partial order. Its originality is that this partial order is maintained between minimal generators and no more between closed itemsets. A structure called *minimal generator lattice* is then built, from which the derivation of the generic association rules becomes straightforward. An intensive experimental evaluation, carried out on benchmarking sparse and dense datasets, showed that PRINCE largely outperforms the pioneer level-wise algorithms, *i.e.*, CLOSE, A-CLOSE and TITANIC.

Keywords: Data mining, Formal Concept Analysis, generic rule bases, minimal generator lattice.

1 Introduction

The last decade was marked by an "obsessional" algorithmic effort to reduce the computation time of the interesting pattern extraction step. The obtained success is primarily due to prowesses of programming conjuncted with intensive handling of compact data structures in the main memory. However, it seems obvious that this frenzy made loses sight of the essential objective of this step, *i.e.*, to extract a reliable knowledge, of exploitable size for end-users. Thus, almost all these algorithms focused on enumerating maximal or closed patterns – presenting a frequency of appearance considered to be satisfactory [1]. As a drawback of this success, this enumeration will generate an impressive and not exploitable number of association rules, even for a reasonable context size. In this situation, the approach based on the extraction of closed patterns (or itemsets) [2] presented a clear promise to reduce considerably the association rule list size. This approach, relying on the Formal Concept Analysis (FCA) mathematical background [3],

* A French version was previously published in French at INFORSID'2005: http://inforsid2005.imag.fr/index.htm.

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2005, LNCS 3589, pp. 346–355, 2005.

proposes to reduce the search space by detecting intrinsic structural properties. Thus, the use of monotonous and non monotonous constraints, during the browsing of the search space, is a typical behavior of this approach [4]. Therefore, the problem of mining association rules might be reformulated, under the (frequent) closed itemsets discovery point of view, as the following two steps [4]:

1. Discover two distinct "closure systems", *i.e.*, sets of sets which are closed under the intersection operator, namely the set of closed itemsets and the set of minimal generators. Also, the *upper cover* (Cov^u) of each closed itemset should be available.
2. From all the discovered information during the first step, *i.e.*, both closure systems and the upper cover sets, derive generic bases of association rules (from which all the remaining rules can be derived).

The essential report after an overview of the state of the art of algorithms [2,5,6,7,8] aiming to discover closed itemsets can be summarized in what follows:

1. Modest performances on sparse contexts. Indeed, computing itemset closures in this type of contexts is heavily handicapping these algorithm performances.
2. Negligence of maintaining the order covering the relationship between the closed itemsets.

In this paper, we propose a new algorithm, called Prince, aiming to extract generic bases of association rules. Prince performs a level-wise browsing of the search space. Its main originality is that it is the only one who accepted to bear the cost of building the partial order. Interestingly enough, to amortize this prohibitive cost, the partial order is maintained between minimal generators and not between closed itemsets. Hence, itemset closures are not computed but derived when Prince performs a simple sweeping of the partially ordered structure to derive generic bases of association rules. Experimental results, carried out on typical benchmark datasets, are very encouraging and showed that this astute partial order construction is not handicapping. Indeed, Prince performances largely outperformed those of well known level-wise browsing algorithms, *i.e.*, Close [2], A-Close [5], and Titanic [7]. These last determine at least the "key" information provided by the minimal generator set. Due to space limit, we report our results only on two datasets.

The remainder of the paper is organized as follows. Section 2 presents the basic mathematical foundations for the derivation of generic bases of association rules. We describe related work and motivations in section 3. Section 4 is dedicated to the presentation of the algorithm Prince. Experimental results showing the utility of the proposed approach are presented in section 5. The conclusion and future work are presented in section 6.

2 Mathematical Background

Due to lack of available space, interested reader for key results from the Galois lattice-based paradigm in FCA is referred to [9].

Frequent Closed Itemset: An itemset $I \subseteq \mathcal{I}$ is said to be *closed* if $\omega(I) = I$ [1] [2]. I is said to be *frequent* if its *relative support*, $\text{Supp}(I) = \frac{|\psi(I)|}{|\mathcal{O}|}$, exceeds a user-defined minimum threshold, denoted *minsup*. In the remainder, we will use the absolute support.

Minimal Generator [10]: An itemset $g \subseteq \mathcal{I}$ is said to be *minimal generator* of a closed itemset f, if and only if $g'' = f$ and does not exist $g_1 \subseteq g$ such that $g_1'' = f$. The set MG_f of the minimal generators of f is: $MG_f = \{g \subseteq \mathcal{I} \mid g'' = f \wedge \nexists\, g_1 \subset g \text{ such as } g_1'' = f\}$.

Iceberg Galois Lattice: When only frequent closed itemsets are considered with set inclusion, the resulting structure $(\hat{\mathcal{L}}, \subseteq)$ only preserves the *Join* operator [9]. This is called a *join semi-lattice* or *upper semi-lattice* and referred to as "*Iceberg Galois Lattice*" [7]

Minimal Generator Lattice: A *minimal generator lattice* is equivalent to an Iceberg Galois lattice such as each equivalence class contains only the associated minimal generators [11].

3 Related Work and Motivations

Since the apparition of the approach based on the extraction of the frequent closed itemsets [2], several generic bases were introduced among which those of Bastide *et al.* [10] and which are defined as follows:

1. The *generic basis for exact association rules* is defined as follows:

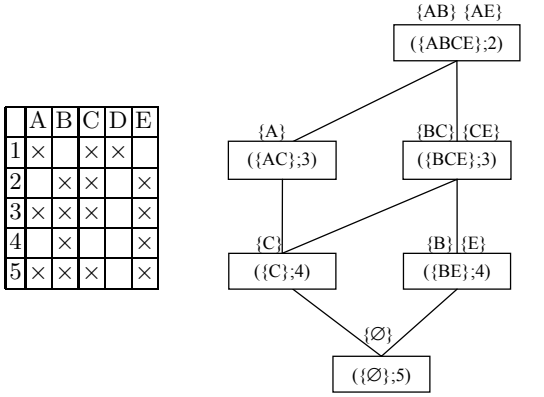
Definition 1. *Let $\mathcal{FCI}_\mathcal{K}$ be the set of frequent closed itemsets extracted from the extraction context $\mathcal{K}$. For each frequent closed itemset $f \in \mathcal{FCI}_\mathcal{K}$, let MG_f be the set of its minimal generators. The generic basis of exact association rules GB is given by: $GB = \{R\text{: } g \Rightarrow (f - g) \mid f \in \mathcal{FCI}_\mathcal{K}$ and $g \in MG_f$ and $g \neq f$ [2]$\}$.*

2. The *transitive reduction of the informative base* [10] is defined as follows:

Definition 2. *Let $\mathcal{FMG}_\mathcal{K}$ be the set of frequent minimal generators extracted from the extraction context $\mathcal{K}$. The transitive reduction RI is given by: $RI = \{R \mid R\text{: } g \Rightarrow (f - g) \mid f \in \mathcal{FCI}_\mathcal{K}$ and $g \in \mathcal{FMG}_\mathcal{K}$ and $g'' \prec f$ [3] and Conf(R) $\geq$ minconf $\}$.*

In the remainder of the paper, we will refer to the generic association rules formed by the couple (GB,RI). This couple is informative, sound and lossless [10,12] and the rules forming it are referred as *informative association rules*. Thus, given an Iceberg Galois lattice – in which each frequent closed itemset is decorated by its list of the minimal generators – the derivation of these rules can

[1] In the remainder, the closure operator ω is indicated by $''$.
[2] The condition $g \neq f$ ensures discarding non-informative rules of the form $g \Rightarrow \emptyset$.
[3] The notation $\prec$ indicates that f covers g'' in $(\hat{\mathcal{L}}, \subseteq)$.



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

Exact generic rules	
R_1: $E \Rightarrow B$	R_2: $B \Rightarrow E$
R_3: $A \Rightarrow C$	R_4: $BC \Rightarrow E$
R_5: $CE \Rightarrow B$	R_6: $AB \Rightarrow CE$
R_7: $AE \Rightarrow BC$	
Approximative generic rules	
R_8: $\emptyset \overset{0.8}{\Rightarrow} BE$	R_{13}: $E \overset{0.75}{\Rightarrow} BC$
R_9: $\emptyset \overset{0.8}{\Rightarrow} C$	R_{14}: $A \overset{0.66}{\Rightarrow} BCE$
R_{10}: $C \overset{0.75}{\Rightarrow} A$	R_{15}: $BC \overset{0.66}{\Rightarrow} AE$
R_{11}: $C \overset{0.75}{\Rightarrow} BE$	R_{16}: $CE \overset{0.66}{\Rightarrow} AB$
R_{12}: $B \overset{0.75}{\Rightarrow} CE$	

Fig. 1. Left: Extraction context $\mathcal{K}$ **Center**: The associated *Iceberg Galois lattice* for *minsup*=2. **Right**: *Informative association rules* for *minsup*=2 and *minconf*=0.5.

be performed straightforwardly. Indeed, generic approximative rules represent "inter-node" implications, assorted with the confidence measure, between two adjacent comparable equivalence classes, *i.e.*, from a frequent closed itemset to another frequent closed itemset immediately covering it.

According to the definition of the informative association rules, we note that they require the extraction of the frequent closed itemsets and their associated minimal generators as well as the upper cover of each frequent closed itemset. Then, constructing the partially ordered structure is a *sine qua non* condition for obtaining the *RI* basis. Thus, information conveyed by minimal generator set and partial order are of paramount importance.

In order to palliate the insufficiencies of existing frequent closed based algorithms, *i.e.*, the cost of the closure computation as well as neglecting the partial order construction, we will introduce a new algorithm called PRINCE. PRINCE highly reduces the cost of closure computation and generates the partially ordered structure, which makes it able to extract generic rule bases without coupling it with another algorithm.

4 Prince Algorithm

PRINCE takes as input an extraction context $\mathcal{K}$, the minimum threshold of support *minsup* and the minimum threshold of confidence *minconf*. It outputs the list of the frequent closed itemsets and their associated minimal generators as well as generic bases of association rules. Thus, PRINCE operates in three successive steps: (i) Minimal generator determination (ii) Partial order construction (iii) Extraction of generic rule bases.

4.1 Minimal Generator Determination

Following the "Test-and-generate" technique, PRINCE traverses the search space by level to determine the set of frequent minimal generators $\mathcal{FMG}_{\mathcal{K}}$ sorted by

decreasing support values as well as the negative border of minimal generators $\mathcal{GB}\mathrm{d}^{-}$[4][13]. In the second step, the set of frequent minimal generators will serve as a backbone to construct the *minimal generator lattice*. As in the case of TITANIC, the negative border will be used, in the second step, to determine the support of a non-minimal generator using the following proposition:

Proposition 1. *[7] Let $\mathcal{MG}_{\mathcal{K}}$=$\mathcal{FMG}_{\mathcal{K}} \cup \mathcal{GB}d^{-}$ be the set of minimal generators extracted from the context $\mathcal{K}$. If X is not a minimal generator then Supp(X) = min $\{Supp(g) \mid g \in \mathcal{MG}_{\mathcal{K}}$ and $g \subset X\}$.*

PRINCE uses, in this step, the same pruning strategies introduced in TITANIC namely *minsup*, the ideal order of the minimal generators and the estimated support.

4.2 Partial Order Construction

In this step, the frequent minimal generator set $\mathcal{FMG}_{\mathcal{K}}$ will form a *minimal generator lattice*, and this *without any access to the extraction context*. The main idea is how to construct the partial order without computing itemset closures, *i.e.*, how guessing the subsumption relation by only comparing minimal generators? To achieve this goal, the list of immediate successors[5] of each equivalence class will be updated in an iterative way. The processing of the frequent minimal generator set is done according to the order imposed in the first step (*i.e.*, by decreasing support values). Each frequent minimal generator g of size k ($k \geq 1$) is introduced into the minimal generator lattice by comparing it to the immediate successors of its (k-1)-subsets[6]. This is based on the isotony property of the closure operator [14]. Indeed, let g_1, a (k-1)-itemset, be one of the subsets of g, $g_1 \subseteq g \Rightarrow g_1'' \subseteq g''$. Thus, the equivalence class to which belongs g is a successor (not necessarily an immediate one) of the equivalence class to which belongs g_1.

While comparing g to the list of the immediate successors of g_1, noted L, two cases are to be distinguished. If L is empty then g is added to L. Otherwise, g is compared to the elements already belonging to L (*cf.* Proposition 2). The imposed order in the first step allows to distinguish only two cases sketched by Proposition 2 by replacing the frequent minimal generators X and Y by respectively g and one of the elements of L.

Proposition 2. *[15] Let X, $Y \in \mathcal{FMG}_{\mathcal{K}}$, $\mathcal{C}_X$ and $\mathcal{C}_Y$ their respective equivalence classes:*

a. If $Supp(X) = Supp(Y) = Supp(X \cup Y)$ then X and Y belong to the same equivalence class.

b. If $Supp(X) < Supp(Y)$ and $Supp(X) = Supp(X \cup Y)$ then $\mathcal{C}_X$ (resp. $\mathcal{C}_Y$) is a successor (resp. predecessor) of $\mathcal{C}_Y$ (resp. $\mathcal{C}_X$).

[4] An itemset belong to $\mathcal{GB}\mathrm{d}^{-}$ if it is an infrequent minimal generator and all its subsets are frequent minimal generators.

[5] By the term "immediate successor", we indicate a frequent minimal generator, unless otherwise specified.

[6] In the first step and for each candidate of size k, links towards its subsets of size (k-1) are stored during the check of the ideal order property.

During these comparisons and to avoid redundant closure computations, PRINCE introduces two complementary functions. These functions make it possible to maintain the concept of equivalence class throughout processing. To this end, each equivalence class $\mathcal{C}$ will be characterized by a *representative* itemset, which is the *first* frequent minimal generator introduced into the *minimal generator lattice*. Both functions are described below:

1. MANAGE-EQUIV-CLASS: This function is used if a frequent minimal generator, say g, is compared to the representative itemset of its equivalence class, say $\mathcal{R}$. The MANAGE-EQUIV-CLASS function replaces all occurrences of g by $\mathcal{R}$ in the immediate successor lists in which g was added. Then, comparisons to carry out with g will be made with $\mathcal{R}$. Thus, for each equivalence class, only its representative itemset appears in the lists of the immediate successors.
2. REPRESENTATIVE: This function makes it possible to find, for each frequent minimal generator g, the representative $\mathcal{R}$ of its equivalence class in order to complete the immediate successors of $\mathcal{C}_g$. This allows to manage only one list of the immediate successors for all frequent minimal generators belonging to the same equivalence class.

The pseudo-code of the second step is given by the GEN-ORDER procedure (Algorithm 1). Each entry, say g, in $\mathcal{FMG}_\mathcal{K}$ is composed by the following fields: (i)`support`: the support of g (ii) `direct-subsets`: the list of the (k-1)-subsets of g (iii)`immediate-succs`: the list of the immediate successors of g.

4.3 Extraction of Generic Rule Bases

In this step, PRINCE extracts the *informative association rules*. For this purpose, PRINCE finds the frequent closed itemset corresponding to each equivalence class by using Proposition 3.

Proposition 3. *[15] Let f and f_1 be two closed itemsets such that f covers f_1 in the Galois lattice $\mathcal{L}_{\mathcal{C}_\mathcal{K}}$. Let MG_f be the set of minimal generators of f. The closed itemset f can be composed as follows: $f = \cup\{g | g \in MG_f\} \cup f_1$.*

The traversal of the *minimal generator lattice* is carried out in an ascending manner from the equivalence class whose minimal generator is the empty set[7] (denoted $\mathcal{C}_\emptyset$) to the non subsumed equivalence class(es). If the closure of the empty set is not null, the informative exact rule between the empty set and its closure is then extracted. Having the partial ordered structure built, PRINCE extracts the informative approximative rules between the empty set and the frequent closed itemsets of the upper cover of $\mathcal{C}_\emptyset$. These closures are found, by applying Proposition 3, using the minimal generators of each equivalence class and the closure of the empty set. Equivalence classes forming the upper cover of $\mathcal{C}_\emptyset$ are stored which makes it possible to apply the same process to them. By the

[7] This class is called the Bottom element of the lattice [9]. The corresponding closure is calculated in the first step by collecting items having a support equal to the number of transactions in the extraction context.

```
1  Procedure: Gen-Order
   Data: - FMG_K.
   Results: - The elements of FMG_K partially ordered in the form of a
            minimal generator lattice.
2  Begin
3    Foreach (g ∈ FMG_K) do
4      Foreach (g_1 ∈ g.direct-subsets) do
5        R = Representative(g_1);
6        Foreach (g_2 ∈ R.immediate-succs) do
7          If (g.support = g_2.support = Supp(g ∪ g_2)) then
8            Manage-Equiv-Class(g,g_2); /*g,g_2 ∈ C_g and g_2 is the
             representative of C_g*/
9          Else if (g.support < g_2.support and g.support =
           Supp(g ∪ g_2)) then
10           g is compared with g_2.immediate-succs;
11           /*For the remainder of the element of
             R.immediate-succs, g is compared only with each g_3 |
             g_3.support > g.support;*/
12       If (∀ g_2 ∈ R.immediate-succs, C_g and C_{g_2} are incomparable)
         then
13         R.immediate-succs = R.immediate-succs ∪ {g};
14 End
```

Algorithm 1: Gen-Order

same manner, Prince treats higher levels of the *minimal generator lattice* until reaching the maximal equivalence class(es). Due to lack of space, the pseudo-code of this step is omitted.

Example 1. Let us consider the extraction context $\mathcal{K}$ given by Figure 1 (Left) for *minsup*=2 and *minconf*=0.5. The first step allows the determination of the sorted set $\mathcal{FMG}_\mathcal{K}$ and the negative border of minimal generators $\mathcal{GB}\mathrm{d}^-$. $\mathcal{FMG}_\mathcal{K}$ = {(∅,5),(B,4),(C,4), (E,4), (A,3), (BC,3),(CE,3), (AB,2), (AE,2)} and $\mathcal{GB}\mathrm{d}^-$={(D,1)}. During the second step, Prince processes the element of $\mathcal{FMG}_\mathcal{K}$ by comparing each frequent minimal generator g, of size k ($k \geq 1$), with the lists of the immediate successors of its subsets of size (k-1). Since the list of immediate successors of the empty set is empty, B is added to ∅.`immediate-succs`. Then, C is compared to B. BC is a minimal generator, $\mathcal{C}_B$ and $\mathcal{C}_C$ are thus incomparable and C is added to ∅.`immediate-succs`. E is then compared to this list. By comparing E to B, E.`support` = B.`support` = Supp(BE) and E ∈ $\mathcal{C}_B$ whose B is the representative one. The Manage-Equiv-Class function is then applied by replacing the occurrences of E, in the immediate successor lists, by B (in this case, there is no occurrence) and by continuing comparisons with B instead of E (in this case, there are no more comparisons to make with E). At this moment of processing, we have ∅.`immediate-succs` =

{B,C}. A is compared to B. As AB $\in \mathcal{FMG}_{\mathcal{K}}$, $\mathcal{C}_B$ and $\mathcal{C}_A$ are incomparable. On the other hand, by comparing A with C, A.`support` < C.`support` and A.`support` = Supp(AC) and then $\mathcal{C}_A$ is a successor of $\mathcal{C}_C$. A is added to C.`immediate-succs` without any comparisons since it is still empty. BC is compared to the immediate successor lists of B and of C. Since, the list associated to B is empty, then BC is added. C has A as an immediate successor. A is then compared to BC and as BC.`support` = A.`support` but BC.`support` $\neq$ Supp(ABC), $\mathcal{C}_{BC}$ and $\mathcal{C}_A$ are incomparable and BC is then added to C.`immediate-succs`. CE is compared to the immediate successor lists of C and of E. The list associated to C contains A and BC. $\mathcal{C}_{CE}$ and $\mathcal{C}_A$ are then incomparable since CE.`support` = A.`support` but CE.`support` $\neq$ Supp(ACE). By comparing CE to BC, since CE.`support`=BC.`support`=Supp(BCE) then CE will be affected to the equivalence class of BC. The MANAGE-EQUIV-CLASS function is then applied. In particular, comparisons of CE to the immediate successors of $\mathcal{C}_E$ will be made with BC. As $\mathcal{C}_E$ has B as a representative itemset, BC is compared to the elements of B.`immediate-succs`. However, B.`immediate-succs` contains only BC and the comparisons stop. It is the same for AB and AE. Having the *minimal generator lattice* built, an ascending sweeping is carried out from the $\mathcal{C}_\emptyset$. As $(\emptyset)'' = \emptyset$, there is no informative exact rule related to $\mathcal{C}_\emptyset$. $\emptyset$.`immediate-succs`={B,C}. The frequent closed itemset associated to $\mathcal{C}_B$ is then found and is equal to BE. The informative approximative rule $\emptyset \Rightarrow BE$, of a support equal to 4 and a confidence equal to 0.8, will be extracted. It is the same for $\mathcal{C}_C$. Using the same process and from $\mathcal{C}_B$ and $\mathcal{C}_C$, the traversal of the *minimal generator lattice* is performed in an ascending way until extracting all the valid informative association rules. The resulting informative generic rules are sketched by Figure 1 (Right).

5 Experimental Results

In this section, we shed light on PRINCE performances vs those of CLOSE, A-CLOSE and TITANIC algorithms. PRINCE was implemented in the C language using gcc version 3.3.1. All experiments were carried out on a PC with a 2.4 GHz Pentium IV and 512 MB of main memory (with 2 GB of Swap) and running SuSE Linux 9.0.

In all our experiments, all times reported are real times, including system and user times. Figure 2 shows execution times of PRINCE[8] algorithm compared to those of CLOSE, A-CLOSE and TITANIC algorithms[9].

- **Connect**: For this dense dataset, PRINCE performances are better than those of CLOSE, A-CLOSE and TITANIC for all *minsup* values. The Connect dataset is characterized by a great number of highly sized transactions. These characteristics complicate the task of CLOSE and A-CLOSE which have to carry

[8] For PRINCE algorithm, the value of *minconf* is set to 0.

[9] The authors are grateful to Yves Bastide who kindly provided source codes of CLOSE, A-CLOSE and TITANIC algorithms.

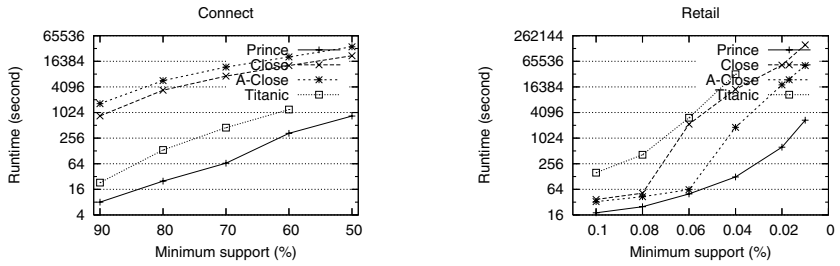


Fig. 2. PRINCE performances vs those of CLOSE, A-CLOSE and TITANIC

out expensive intersection computations. PRINCE and TITANIC avoid these expensive computations with an advantage for PRINCE. Indeed, PRINCE is favoured compared to TITANIC by a reduced cost of carried out comparisons for a frequent minimal generator compared with extension attempts carried out by TITANIC. Interestingly enough, for $minsup = 50\%$, the execution TITANIC could not come to an end for lack of memory space.

- **Retail**: For this sparse dataset, execution times of our algorithm are much more reduced than those of CLOSE, A-CLOSE, TITANIC algorithms. These performances can be explained by the enormous influence of the high number of items in the Retail dataset. Indeed, CLOSE is handicapped by a great number of candidates for which it is obliged to calculate closures, even though a high number of them is infrequent. The number of candidates affects also A-CLOSE performances because of the additional sweeping for each candidate as well as the cost of the closure computation step. On its side, TITANIC is considerably penalized by a great number of frequent items to consider in closure computations (for $minsup = 0.04\%$, 4643 items are frequent and the maximum size of a frequent minimal generator is only equal to 6 items). The execution of TITANIC stops, for support values lower than 0.004%, for lack of memory capacity. PRINCE performs better since the treatment to do for each frequent minimal generator is by far less expensive than that performed in the other algorithms.

6 Conclusion

In this paper, we proposed a new algorithm, called PRINCE, for an efficient extraction of frequent closed itemsets and their respective minimal generators as well as the generic rule bases. To this end, PRINCE builds the partial order contrary to the existing algorithms. A main characteristic of PRINCE algorithm is that it relies only on minimal generators to build the underlying partial order. Carried out experiments outlined that PRINCE largely outperforms existing "Test-and-generate" algorithms of the literature for both dense and sparse contexts. In the near future, we plan to add other constraints [16] to PRINCE algorithm so that the number of generic rules will be reduced while keeping the most interesting rules for the user.

References

1. Goethals, B., Zaki, M.J.: FIMI'03: Workshop on frequent itemset mining implementations. In: FIMI Repository: http://fimi.cs.helsinki.fi/. (2003)
2. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Journal of Information Systems **24** (1999) 25–46
3. Wille, R.: Restructuring lattices theory: An approach based on hierarchies of concepts. I. Rival, editor, Ordered Sets, Dordrecht-Boston (1982) 445–470
4. BenYahia, S., Nguifo, E.M.: Approches d'extraction de règles d'association basées sur la correspondance de Galois. In Boulicault, J.F., Cremilleux, B., eds.: Revue d'Ingénierie des Systèmes d'Information (ISI), Hermès-Lavoisier. Volume 3–4. (2004) 23–55
5. Pasquier, N., Bastide, Y., Touil, R., Lakhal, L.: Discovering frequent closed itemsets. In Beeri, C., Buneman, P., eds.: Proceedings of 7th International Conference on Database Theory (ICDT'99), LNCS, volume 1540, Springer-Verlag, Jerusalem, Israel. (1999) 398–416
6. Pei, J., Han, J., Mao, R., Nishio, S., Tang, S., Yang, D.: Closet: An efficient algorithm for mining frequent closed itemsets. In: Proceedings of the ACM-SIGMOD DMKD'00, Dallas, Texas, USA. (2000) 21–30
7. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Journal on Knowledge and Data Engineering (KDE) **2** (2002) 189–222
8. Zaki, M.J., Hsiao, C.J.: Charm: An efficient algorithm for closed itemset mining. In: Proceedings of the 2nd SIAM International Conference on Data Mining, Arlington, Virginia, USA. (2002) 34–43
9. Ganter, B., Wille, R.: Formal Concept Analysis. Springer-Verlag (1999)
10. Bastide, Y., Pasquier, N., Taouil, R., Lakhal, L., Stumme, G.: Mining minimal non-redundant association rules using frequent closed itemsets. In: Proceedings of the International Conference DOOD'2000, LNAI, volume 1861, Springer-Verlag, London, UK. (2000) 972–986
11. BenYahia, S., Cherif, C.L., Mineau, G., Jaoua, A.: Découverte des règles associatives non redondantes : application aux corpus textuels. Revue d'Intelligence Artificielle (special issue of Intl. Conference of Journées francophones d'Extraction et Gestion des Connaissances (EGC'2003)), Lyon, France **17** (2003) 131–143
12. Kryszkiewicz, M.: Concise representations of association rules. In Hand, D.J., Adams, N., Bolton, R., eds.: Proceedings of Exploratory Workshop on Pattern Detection and Discovery in Data Mining (ESF), 2002, LNAI, volume 2447, Springer-Verlag, London, UK. (2002) 92–109
13. Kryszkiewicz, M.: Concise representation of frequent patterns based on disjunction-free generators. In: Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM), San Jose, California, USA. (2001) 305–312
14. Davey, B., Priestley, H.: Introduction to Lattices and Order. Cambridge University Press (2002)
15. Hamrouni, T., BenYahia, S., Slimani, Y.: Prince : Extraction optimisée des bases gnriques de règles sans calcul de fermetures. In: Proceedings of the International Conference INFORSID , Inforsid Editions, Grenoble, France. (2005) 353–368
16. Bonchi, F., Lucchese, C.: On closed constrained frequent pattern mining. In: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04), Brighton, UK. (2004) 35–42