Restrictions on Concept Lattices for Pattern Management

Léonard Kwuida[1], Rokia Missaoui[2], Beligh Ben Amor[2], Lahcen Boumedjout[2], Jean Vaillancourt[2]

[1] Zurich University of Applied Sciences
kwuida@gmail.com
[2] Université du Québec en Outaouais
{rokia.missaoui,benb03}@uqo.ca
{lahcen.boumedjout,jean.vaillancourt}@uqo.ca

Abstract. This paper addresses the problem of pattern management in the framework of formal concept analysis using restrictions on objects or attributes of a given data set. Patterns are pieces of information/knowledge with a concise description that can be obtained from data using data mining techniques. These can be clusters or bi-clusters, implications or association rules, and so on. Even for relatively small data collections, the set of discovered patterns could be very large and therefore difficult to handle. In this paper we propose efficient methods to conduct the projection of a concept lattice on a set of attributes. The selection of a concept lattice on a set of objects is done dually.

1 Introduction

Pattern discovery and management refers to a set of activities related to the extraction, description, manipulation and storage of patterns in a similar (but more elaborated) way as data are managed by database applications. In pattern management and inductive databases [1, 6, 9, 7], patterns are knowledge artifacts (e.g., association rules, clusters) extracted from data using data mining procedures (generally run in advance), and retrieved upon user's request. A pattern is then a concise and semantically rich representation of raw data. An example of a pattern could be a cluster that represents a set of transactions with the common bought items or an association rule which states that whenever we buy cheese we tend to buy crackers too.

In many information system applications, users tend to be drowning in data and even in patterns while they are actually interested in a very limited set of knowledge pieces. Moreover, the scope of patterns to explore differs from one user to another and changes over time. Finally, one is frequently interested in an exploratory and iterative process of data mining (DM) to discover patterns under different scenarios and different hypotheses. To that end, we propose to define two main algebraic operators : selection and projection on concept lattices.

In this paper we handle the problem of pattern management by describing two algebraic operations commonly used in relational queries, namely projection and selection. These operations will be applied to data sets (tables or formal contexts) and concept sets/lattices. The problem of projection can be stated as follows:

Given a set of patterns $\mathcal{P}$ and a subset N of attributes in a table T, produce the corresponding set of patterns when the analysis is restricted to the attributes in N.

Dually, the problem of selection is: *given a set of patterns $\mathcal{P}$ and a formula φ of T, produce the corresponding set of patterns when the analysis is restricted to those tuples satisfying the condition φ.*

The rest of the paper is organized as follows. In Section 2 we introduce the basic notions of formal concept analysis and describe existing work about the general topic of pattern management. Sections 3 and 4 present different ways to conduct a projection on a set of attributes, namely projection on contexts and concept lattices. An experimental study is given in Section 5.

2 Background and Related Work

Formal Concept Analysis [4] has been successfully used for conceptual clustering and rule mining. A formal context is a triple $\mathbb{K} := (G, M, I)$ where G, M and I stand for a set of objects, a set of attributes, and a binary relation between G and M respectively. For $A \subseteq G$ and $B \subseteq M$ we define

$$A' := \{a \in M \mid oIa \,\forall o \in A\} \quad \text{and} \quad B' := \{o \in G \mid oIa \,\forall a \in B\},$$

the set of attributes common to objects in A and the set of objects sharing all the attributes in B. The mapping (denoted by $'$) between the powerset of G and the powerset of M defines a Galois connection, and the induced closure operators (on G and M) are denoted by $''$. A formal concept c is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A = B'$ and $B = A'$. A is called the extent of c (denoted by ext(c)) and B its intent (denoted by int(c)). Intents are then closed subsets of attributes and extents are closed subsets of objects. In the closed *itemset* mining framework [8, 12], G, M, A and B correspond to the notion of transaction database, set of items (products), closed *tidset* and closed *itemset* respectively. The set of all concepts of $\mathbb{K}$ is denoted by $\mathfrak{B}(\mathbb{K})$. Ordered by $(A, B) \leq (C, D) :\iff A \subseteq C$, it forms a complete lattice[3] called the concept lattice of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$. Our working example is on Figure 1.

In order to manipulate concept lattices in a similar way as relational tables, we take a joint database-FCA perspective (see [4]) by using operators similar in spirit to relational algebra operators (e.g., selection, projection and join) and by exploiting and adapting existing work related to operations on contexts and concept lattices to analyze and formalize such operators. In [11], the authors present an approach for lattice construction based on the apposition of two contexts $\mathbb{K}_1$ and $\mathbb{K}_2$ (having the same set of objects). Such operation is perceived as the opposite operation of the projection and identical to the relational join operation.

Recent studies on pattern management [1, 9] provide a uniform framework to data and pattern management and define links between data and pattern spaces through

[3] This is a poset in which every subset X has an infimum ($\bigwedge X$) and a supremum ($\bigvee X$). We set $a \wedge b := \bigwedge\{a, b\}$ and $a \vee b := \bigvee\{a, b\}$.

$\mathbb{K}$	a	b	c	d	e	f	g	h	S	T	U	V
1	×				×		×		×			
2	×				×	×		×				×
3	×	×			×	×	×		×			
4		×			×	×	×	×	×			×
5	×		×	×							×	
6	×	×	×	×						×	×	
7		×	×				×			×		
8		×	×	×			×			×		

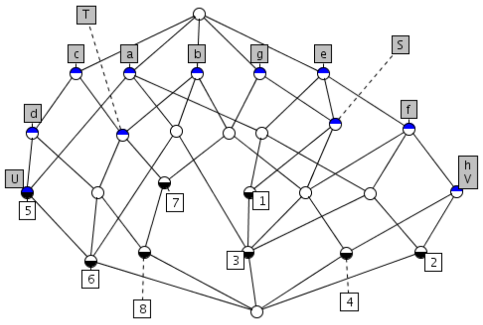

Fig. 1. A formal context (left) and its concept lattice (right). Concepts are nodes of the lattices. For each node the corresponding concept has as intent (resp. extent) the attributes (resp. objects) contained in the order filter (resp. ideal) generated by this node. For example the node labeled by the attribute S represents the concept $(\{1, 3, 4\}, \{e, g, S\})$.

bridging operations and cross-over queries such as finding data covered by a given pattern or identifying patterns related to a data set. Although many studies limit the management of patterns to association rules only, work conducted by Calders *et al.* [1], and Terrovitis *et al.* [9] cover different types of patterns. In [9], a pattern base management system is defined for storing, processing and querying patterns. Moreover, languages for pattern definition and manipulation are proposed, and temporal aspects of patterns are handled. In [1], a data mining algebra and a 3-World model are defined, as well as a small set of data mining primitive operators are proposed to further formulate complex queries. The proposed model includes three worlds: *D-World* for data definition and manipulation (e.g., projection, join), *I-World* for region (set of constraints) definition and manipulation, and *E-World* for operations on data contained in regions. In [2], a Data Mining Template Library is defined based on a generic data mining approach for controlling aspects of pattern mining through a set of properties. On the industry side, work was mainly done to design languages for pattern description, manipulation and exchange. This is the case of PMML and SQL/MM [9].

To the best of our knowledge, the only previous work on restricting concept sets via the projection or the selection is the one by Jeudy et al. [5] in which the authors define a graph representation similar in spirit to Hasse diagrams. The proposed procedure for projection requires four times traversals of the graph structure (as opposed to two times for the concept sorting algorithm) and hence is less efficient than our solution. The lattice resulting from the projection contains in [5] only the actual nodes while in our case it contains all the members of generated equivalence classes with a highlight on the maximal concept representatives. This could help "switch" easily from the projected lattice to the initial one (and vice versa) instead of reconstructing the initial lattice from scratch.

3 Pattern Restriction in Databases

A (relational) database is a collection of related tables that can be combined via relational operations to produce new ones. The main operations are selection, projection and join. In this paper we consider only the first two operations which are frequently used unary relational operations. We also assume that T is a (real or virtual) table with primary key values in G. We denote by M the set of attributes (other than the primary key) and by W the set of values of attributes in M. Then, the table T is a many-valued context (G, M, W, I), where I is a map from $G \times M$ to W such that $I(g, m) := m(g)$, i.e., the value of the attribute m for the tuple with key-value g.

For a subset N of M, a *projection* π_N on (G, M, W, I), gives the many-valued context (G, N, W_N, I_N) with $I_N : G \times N \to W_N \subseteq W$ such that $I_N(g, n) := I(g, n) = n(g)$. In practice $W_N = \bigcup\{n(g) \mid n \in N,\ g \in G\}$. Thus, conducting the projection π_N on a set $\mathcal{P}$ of patterns mined from (G, M, W, I) is equivalent to producing the corresponding patterns directly from (G, N, W_N, I_N). We denote by $\pi_N(\mathcal{P})$ the so-obtained set of patterns, and called it the projection of $\mathcal{P}$ on N. Given $\mathcal{P}$ from a database T, a naive and straightforward way to obtain $\pi_N(\mathcal{P})$ will be to conduct the projection π_N on (G, M, W, I), and then rerun all the steps done to produce $\mathcal{P}$, but this time on (G, N, W_N, I_N).

For a formula φ, the *selection* σ_φ produces a subcontext $(G_\varphi, M, W_\varphi, I_\varphi)$, where G_φ is the set of objects in G satisfying φ, and $I_\varphi : G_\varphi \times M \to W_\varphi \subseteq W$ with

$$I_\varphi(g, m) := I(g, m) \quad \text{and} \quad W_\varphi := \bigcup_{m \in M} \{m(g) \mid g \models \varphi\}.$$

Applied to a set of patterns $\mathcal{P}$ produced from (G, M, W, I), we get $\sigma_\varphi(\mathcal{P})$, the set of patterns that we should mine from $(G_\varphi, M, W_\varphi, I_\varphi)$ by applying the same techniques used to produce $\mathcal{P}$ from (G, M, W, I). As above, given $\mathcal{P}$ from a database T, one way to obtain $\sigma_\varphi(\mathcal{P})$ will be to conduct the selection σ_φ on (G, M, W, I), and then repeat all the steps done to produce $\mathcal{P}$, but this time on $(G_\varphi, M, W_\varphi, I_\varphi)$.

Conducting restrictions on raw data is however not always possible. Indeed, it may happen for some reason (e.g., storage constraint) that raw data from which the patterns were produced are no more available. Therefore, there is a need to explore means to either produce restricted patterns without returning to raw data, or explore the way to recover raw data from patterns. In this paper we focus on the first issue and investigate the way to produce restricted patterns directly from the already existing patterns.

The problem of restriction on patterns can be summarized in the commutativity of the diagram on Figure 2 below. Indeed, the analyst can get a pattern set from data using one of the two ways: (i) conduct a restriction on data followed by a data mining task, or (ii) handle a data mining task followed by a restriction on produced patterns.

In inductive databases and pattern management applications, there is no conceptual difference between data and patterns generated from that data: both can be stored and queried according to user's needs and preferences in a uniform way. In the upcoming sections, we show how a restriction of a lattice to a set of attributes can be conducted.

The problem of projection on concepts can be stated as follows: given a concept lattice $\mathfrak{B} := \mathfrak{B}(G, M, I)$ and a subset $N \subseteq M$ of attributes, find the concept lattice

$$\begin{tikzcd}
\text{Database} \arrow[r, shift left, "\text{mining}"] \arrow[d, "\text{Restriction}"'] & \text{Pattern set} \arrow[l, shift left, "\text{back to data}"] \arrow[d, "\text{Restriction}"] \\
\text{Database} \arrow[r, "\text{mining}"] & \text{Pattern set}
\end{tikzcd}$$

Fig. 2. Defining a restriction on patterns (right) from a restriction in databases (left).

$\Pi_N\mathfrak{B}$ when the analysis is restricted to the attributes in N. Similarly, the problem of selection on concepts is to find the corresponding concept lattice when the analysis is restricted to a subset of objects. Assuming that we have access to data from which the lattice was produced [4], one may perform the projection on the data and scale to get binary contexts and their corresponding concept lattices (*projection on contexts*). An alternative is to analyze the effect of the projection on the initial concept lattice and get $\Pi_N\mathfrak{B}(G, M, I))$ directly from $\mathfrak{B}(G, M, I)$ without returning to the raw data (*projection on concepts*). It is interesting to check in which cases a projection on contexts is more advantageous than a projection on concept lattices.

4 Projection on Concept Lattices

For simplicity we will assume that M is a set of scaled attributes. Given a concept lattice $\underline{\mathfrak{B}}(G, M, I)$ (denoted by $\mathfrak{B}$) and a list of attributes $N \subseteq M$. The objective is to produce the concept lattice $\Pi_N(\mathfrak{B})$, where Π_N is the projection on the attributes in N.

The projection Π_N induces an equivalence relation on $\mathfrak{B}$ given by

$$(A_1, B_1) \simeq (A_2, B_2) :\iff B_1 \cap N = B_2 \cap N. \tag{1}$$

Each equivalence class has a greatest element that can be set as the representative of that class. The mapping

$$c : (A, B) \mapsto \max\{(U, V) \in \mathfrak{B} \mid V \cap N = B \cap N\} \tag{2}$$

is a closure operator on $\mathfrak{B}$. The intent of $c(A, B)$ is exactly $B \cap N$ and its extent is the one of the greatest concept in its equivalent class. Therefore, different scenarios can be considered to find the equivalence classes on the lattices. For example, starting from the top (or bottom), we can traverse the lattice level-wise and group together the elements of the same class. The link (covering relation) between class representatives is set up as follows: $c_i \prec c_j$ in $\Pi_N\underline{\mathfrak{B}}(G, M, I)$ iff there is a concept (A, B) in the equivalence class c_i and a concept (C, D) in the equivalence class c_j such that $(A, B) \prec (C, D)$ in $\underline{\mathfrak{B}}(G, M, I)$.

In the following subsections we present, analyze[5] and compare the performance of a selected set of methods for conducting a projection on concept lattices. The input is

[4] When the access to the initial formal context is no more possible, then its reconstruction is needed if we plan to apply the restriction to the context, and produce thereafter the lattice.

[5] The complexity analysis presented here is a preliminary step towards understanding why some methods perform better in a given configuration. Parameters taken into account include the number of concepts, the parents/children of a node and the size of equivalence classes.

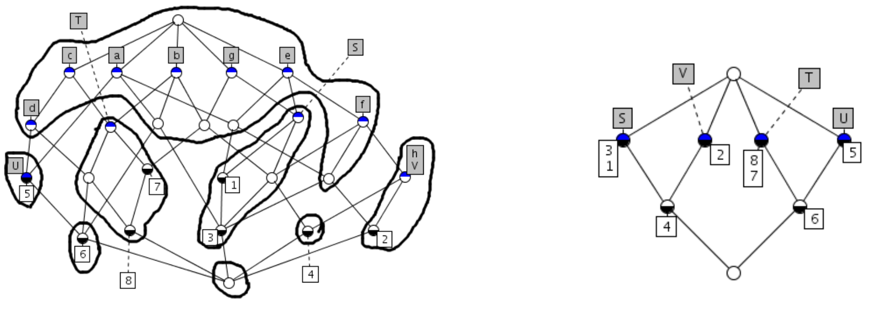

Fig. 3. Projection on $\{S, T, U, V\}$ of the concept lattice in Figure 1. On the left we can see equivalence classes marked on the initial lattice. On the right we note that each equivalence class is represented by a single node (behind which a whole class is attached).

a concept lattice $\underline{\mathfrak{B}}(G, M, I)$ and a set of attributes $N \subseteq M$. The output is a concept lattice $\Pi_N(\underline{\mathfrak{B}}(G, M, I))$. Dually, a selection σ_φ induces an equivalence relation on $\mathfrak{B}$ such that each equivalence class has a least element (class representative). Conducting a selection σ_φ is equivalent to conducting a projection of the dual lattice of $\mathfrak{B}$ on H, where H is the set of objects satisfying the formula φ.

4.1 Depth-first Search (DFS)

Conducting a depth-first scan for the projection of a concept lattice on a given set of attributes can be done as follows

1. Set the first class with the top element 1. Start with that element and follow a path to the bottom element 0. We get a path $1, a_1, a_2, \ldots, a_s, 0$. At each node a_i (going downwards), choose a child a_{i+1} and test if the two nodes are in the same class (by comparing the nodes a_i and a_{i+1}). If they do not belong to the same class, then create a new membership class for a_{i+1}.
2. Once the bottom element of the lattice is reached, go back to node a_s and repeat the following steps at each subsequent node o:
   – if the current node o has a child that is not yet marked, then move to this child, mark it, and find its equivalence class.
   – else go back one step (to the parent from which the node o was reached).
3. Stop the process once all nodes are marked.
   The links between equivalence classes are set up as the scan progresses.

To analyze the complexity of the procedure above, we consider the number of accesses to each node and the number of comparisons. Note that each node is visited at least twice (on the way down and back). If q is the number of equivalence classes, then there are in average $\frac{q}{2}$ comparisons to mark a node. In fact we have to check if the child node to classify does not belong to an existing class. In practice, using the hierarchy on concepts and the convexity[6] of equivalence classes, this comparison reduces to classes of neighboring nodes.

4.2 Breadth-first Search (BFS)

The idea here is to traverse the lattice from the top in a breadth-first manner, and assign a different color to a node only if it is the greatest element of its equivalence class. This avoids look-up into equivalence classes to see if the node has an already assigned color. This can drastically improve the performance of the algorithm, particularly if there are many classes. We assume that $\mathfrak{B}(G, M, I)$ is represented as follows: each node e has two lists $parent(e)$ and $child(e)$ containing respectively the upper neighbor and the lower neighbor of e. We proceed as follows:

1. Start with node $e := 1$ (top element) and mark the representative of the first class.
2. Move to each node in $child(e)$ and compare it with e. If it is not in the same class, then mark it as a new class. Actually, there is only one way to reach nodes in $child(e)$ from above.
3. Assume we are at a (marked) current node e. Then we compare it with the (unmarked) nodes in $child(e)$. Mark with the same color the equivalent nodes. If there is a node $g \in child(e)$ that is not equivalent to e, then check whether all nodes in $parent(g)$ are marked. If so, then a new color is assigned to this node. Else, g is not marked. In practice, for each node o we keep in $parent(o)$ only the parents that are not yet marked.
4. Repeat the previous step until e is the bottom.

To evaluate the complexity of this algorithm, we consider two parameters: the number of needed comparisons and the number of times each node is accessed. Each node o is visited exactly $\#parent(o) + 1$ times. Then, the overall access to nodes is

$$\sum_{o \in \mathfrak{B}} (\#parent(o) + 1) = \#\mathfrak{B} + \sum_{o \in \mathfrak{B}} \#parent(o).$$

If the average number of parents of nodes is p, then the overall number of node accesses is $(p + 1)n$, where n is the number of concepts in $\mathfrak{B}$. Each node o is compared with all its lower neighbors. Therefore, the total number of comparisons is pn and the overall complexity is $O(np)$.

4.3 Leading Bits Sort (LBS)

We have noticed that traversing the lattice to discover equivalence classes and group the concepts into these classes can be very time consuming. To speed up this process, we linearly order the nodes so that the equivalence classes can be obtained in a more efficient way. The benefit of this order is that it avoids a look up in the list of classes and each node is then visited only once to assign the color corresponding to its equivalence

[6] The equivalence classes are convex subsets of the lattice in the sense that if x and z are equivalent concepts and y is a concept between x and z, then x, y and z are also equivalent.

class. We choose the lectic order as in Next-closure [4, 3]. Each subset of attributes is represented by a sequence of bits (1/0) of fixed length $|M|$ (the cardinality of the set of attributes). The leading bits are labeled by the attributes in N on which we restrict the analysis. The remaining bits are labeled by the rest of the attributes (non-relevant to the present analysis). The lectic order on subsets of M states that A precedes B if the first position in which A and B differ contains 0 in A and 1 in B, (i.e. the first element, according to a predefined linear order on M, that distinguishes A and B, is in B). This is a linear strict order. If all intents are sorted with respect to the lectic order with leading bits headed by the (relevant) attributes in N, then the equivalent concepts/intents are necessarily consecutive. In this case we start from the first intent and move downwards until we get an intent whose leading bits differ from the current one, representing the current class. Then we start a new class and repeat this until we reach the last element of the list.

The projection done this way can be divided into two steps. The sorting process with respect to the lectic order can be done in $O(n \times \ln(n))$, where n is the number of concepts in $\mathfrak{B}$. The marking of equivalence classes on $\mathfrak{B}$ is straightforward since there is one linear pass in the linearly sorted set of concepts. Thus, the overall process has a complexity of $O(n \times \ln(n))$. Note that in this case we do not assume any special representation for $\mathfrak{B}$. All we need is the list of concepts so that we can color the equivalence classes.

If user's access behavior (both in terms of data and patterns) is known, then the candidate attributes of the projection may be identified beforehand. If we choose Next-closure [4, 3] to compute the concepts, then the lectic order will be defined with the attributes in N at the first positions. This way, the sorting step is no more necessary since the intents are already sorted in lectic order with leading attributes in front and less interesting attributes at the end.

In practice the sorting step can be restricted to the $|N|$ leading bits. In fact, we can define a quasi-order $\sqsubseteq^N$ on the intent (subsets of M) as follows: we start with sorting the base set M in a linear order so that every element of N appears before every element of $M \setminus N$, for example $M = \{m_1 < \cdots < m_n < m_{n+1} < \cdots < m_k\}$, where $N = \{m_1, \cdots m_n\}$ is the set of attributes on which the projection is to be done. For two intents A and B, we say that $A \sqsubseteq^N B$ if and only if A and B have the same restriction on N or the first element (with respect to the linear order on the base set M) which distinguishes $A \cap N$ from $B \cap N$ is in B. The equivalence relation induced by $\sqsubseteq^N$ is exactly the one induced by Π_N (see Equation (1) above).

4.4 Bottom-up Search (BUS)

In this method, the idea is to iteratively remove (one by one) the attributes found in the complement of N and more precisely in $N'' \setminus N$. This is in fact the dual procedure of incremental lattice update procedures [10]. To accelerate the process, we start the exploration of the lattice (upwards from the bottom) with the most general concept c, whose intent contains N (i.e. $c = (N', N'')$). In fact, every concept of this lattice has an equivalent concept in the filter generated by c. Therefore we restrict the search space to $\uparrow c$.

There are two possibilities: if the concept c has exactly N as intent, then the ideal generated by c is exactly the equivalence class of 0. Moreover the output of the projection is the filter $\uparrow c$. If N is not an intent, then the attributes that are in $N'' \setminus N$ will be deleted one by one from the intent of concepts in the filter $\uparrow c$. The nodes that have a same intersection of their intent with N will collapse, leading to a unique concept. The links between nodes are updated as the exploration progresses. This methods is faster when the number of attributes to be removed is small or when N is an intent.

As an illustration, the projection of the lattice shown in Figure 1 on $N = \{a, b, f\}$ is done as follows: the most general concept whose intent is larger than N is identified: $c = (\{3\}, \{a, b, e, f, g, S\})$. The search space is now the filter $\uparrow c$ (with 14 concepts instead of 26 concepts if the search was to be done on the whole lattice). Then, the attributes in $N'' \setminus N = \{e, g, S\}$ will be removed one by one from the intents of the nodes in $\uparrow c$. While removing attribute e from node intents in $\uparrow c$, the concepts $(G, \emptyset)$ and $(\{1, 2, 3, 4\}, \{e\})$ collapse as well as $(\{1, 2, 3\}, \{a, e\})$ and $(\{1, 2, 3, 5, 6\}, \{a\})$. The node $(\{2, 3\}, \{a, e, f\})$ changes to $(\{2, 3\}, \{a, f\})$ but does no collapse with any other node of $\uparrow c$.

5 Empirical Study

In order to compare the performance of the proposed algorithms we have conducted a set of empirical tests on a Windows XP-based system with 3 GB memory and 1.9 GHz processor using a Java implementation of the algorithms. The tests aim to compare the execution time of the four proposed algorithms on four different sizes of concept lattices. Each concept lattice is produced from a formal context of 50 objects and 50 attributes with different densities. For space limitation, we show the results for only two lattices of 71 114 (density = 50%) and 234 946 concepts (density = 60%). The execution time of the projection on the initial context is also provided. Moreover, a variation in the size of the set N of projection attributes is considered in order to identify the cases when a projection on concepts is more efficient than a projection on contexts. Each experiment is conducted twice and the average values of execution time and memory consumption are stored.

In Figure 4 the x-axis represents the percentage of the projection attributes and the y-axis expresses the execution time in seconds (right) or the memory usage in megabytes (left) in logarithmic scale for five projection procedures: projection on context (CTX), depth-first search (DFS), breadth-first scan (BFS), leading bits concept sort (LBS), and bottom-up search (BUS). One can see that conducting the projection of a concept lattice onto a set N of attributes through a bottom-up exploration with node pruning becomes an interesting alternative as the number of attributes in N increases and mainly when their proportion exceeds 50% of the initial set of attributes. In fact the number of attributes to be removed decreases when the number of projection attributes increases. The two top-down algorithms (DFS and BFS) have similar performance values, with a slight advantage in favor the breath-first scan. The leading bits concept sort is more efficient than DFS and BFS. The gap between the three algorithms increases as the number of the projection attributes increases and the density increases (leading to a larger number of children/parents per node). As one can expect, the projection of a

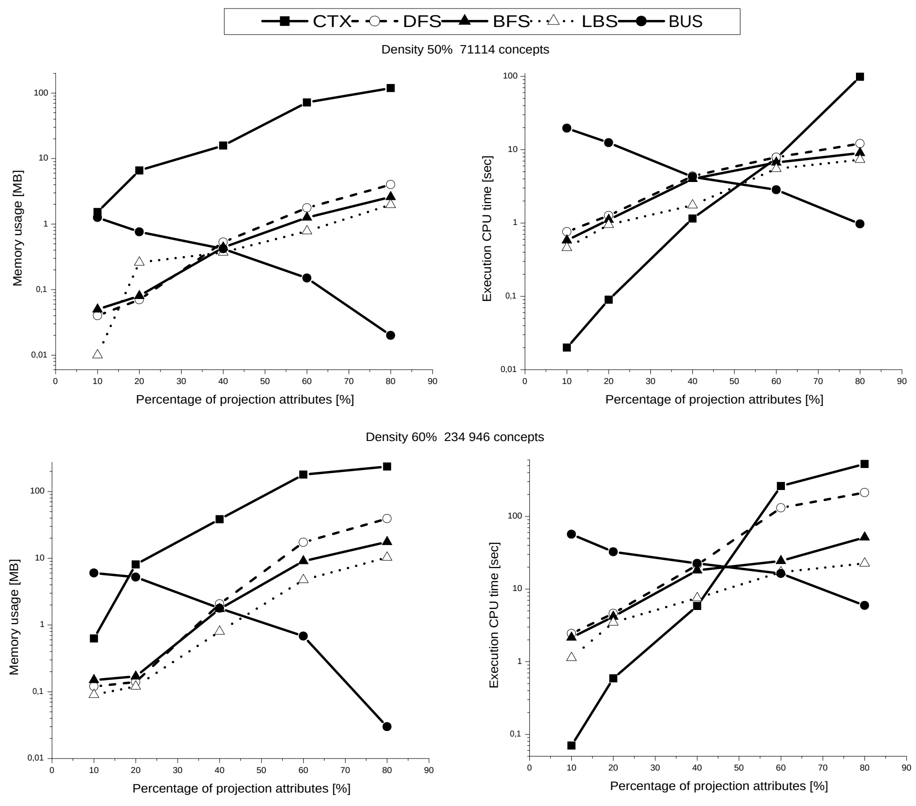

Fig. 4. Execution time and memory usage of the projection procedures.

formal context onto a set N of attributes is more efficient than the LBS procedure when the size of N is small (less than 40%) and the context density is significant. In fact, if the number of attributes on which the projection is to be done is small, then it makes sense to conduct the projection on the context since the later will be small and the corresponding concept lattice could be constructed easily and quickly. However, it becomes drastically inefficient as N and the density of the initial formal context increase.

With respect to memory usage (see Figure 4), the procedures LBS, DPS, BFS and CTX have a similar behavior in the sense that they need an increasing size of memory as the number of projection attributes increases. However, LBS and BFS are less memory demanding than DFS and CTX. Like execution time values, memory usage for the bottom-up search decreases as the number of projection attributes increases.

6 Conclusion

As indicated earlier, the present work is intimately related to the general and important problem of pattern management in inductive databases and knowledge discovery applications. In this paper we have described the general operation of restriction of concept lattices in order to restrict the analysis of a pattern set to either a set of objects or a set of attributes. We focused on the projection of a pattern set onto a set of attributes but the work can be easily adapted to handle a selection on a set of objects. We have proposed, implemented and tested a set of procedures for the projection of concept lattices on a set of attributes and analyzed their performance both theoretically and empirically. The first three methods scan the lattice in a top-down manner without any graph pruning using either a depth-first search, a breadth-first scan or exploiting concept sorting. The fourth one uses a bottom-up search and node pruning to focus on most promising concepts. As we mentioned before the methods perform better that the one in [5]. The proposed procedure in [5] for projection requires four times traversals of the graph structure (as opposed to two times for the concept sorting algorithm) and hence is less efficient than our procedures. Although our solution needs more space to store all the nodes of the initial lattice (see the two alternate representations of the output in Figure 3), it has the merit to help "switch" easily from the projected lattice to the initial one (and vice versa) instead of reconstructing the initial lattice from scratch. Due to space limitation we could not include the detailed steps of the algorithms. The problem we have considered in this paper can be seen as a preliminary step towards the more general following issue:

> Given a set of patterns $\mathcal{P}$ (*e.g.*, implications) and a restriction $\mathfrak{r}$, is it convenient to return to the source data in the formal context K and conduct the restriction $\mathfrak{r}$ on it or apply $\mathfrak{r}$ directly on $\mathcal{P}$?

Acknowledgment

The second and last authors acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC). All the authors would like to thank the anonymous referees for their very helpful comments and suggestions.

References

1. Toon Calders, Laks V. S. Lakshmanan, Raymond T. Ng, and Jan Paredaens. Expressive power of an algebra for data mining. *ACM Trans. Database Syst.*, 31(4):1169–1214, 2006.
2. Vineet Chaoji, Mohammad Al Hasan, Saeed Salem, and Mohammed Javeed Zaki. An integrated, generic approach to pattern mining: data mining template library. *Data Min. Knowl. Discov.*, 17(3):457–495, 2008.
3. Bernhard Ganter. Two basic algorithms in concept analysis. In Léonard Kwuida and Baris Sertkaya, editors, *ICFCA*, volume 5986 of *Lecture Notes in Computer Science*, pages 312–340. Springer, 2010.
4. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., 1999. Translator-C. Franzke.

5. Baptiste Jeudy, Christine Largeron, and François Jacquenet. A model for managing collections of patterns. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 860–865, New York, NY, USA, 2007. ACM.
6. Heikki Mannila. Theoretical frameworks for data mining. *SIGKDD Explorations*, 1(2):30–32, 2000.
7. Rokia Missaoui, Léonard Kwuida, Mohamed Quafafou, and Jean Vaillancourt. Algebraic operators for querying pattern bases. *CoRR*, abs/0902.4042, 2009.
8. Nicolas Pasquier, Rafik Taouil, Yves Bastide, Gerd Stumme, and Lotfi Lakhal. Generating a condensed representation for association rules. *J. Intell. Inf. Syst.*, 24(1):29–60, 2005.
9. Manolis Terrovitis, Panos Vassiliadis, Spiros Skiadopoulos, Elisa Bertino, Barbara Catania, Anna Maddalena, and Stefano Rizzi. Modeling and language support for the management of pattern-bases. *Data Knowl. Eng.*, 62(2):368–397, 2007.
10. Petko Valtchev and Rokia Missaoui. Building concept (galois) lattices from parts: Generalizing the incremental methods. In *ICCS*, pages 290–303, 2001.
11. Petko Valtchev, Rokia Missaoui, and Pierre Lebrun. A partition-based approach towards constructing galois (concept) lattices. *Discrete Math.*, 256(3):801–829, 2002.
12. Mohammed Javeed Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining*, Arlington, VA, USA, April 11-13, 2002.