Computing Concept Lattices from Very Sparse Large-Scale Formal Contexts

Lenka Pisková and Tomáš Horváth

University of Pavol Jozef Šafárik, Košice, Slovakia
{lenka.piskova,tomas.horvath}@upjs.sk

Abstract. This paper introduces a new algorithm for computing concept lattices from very sparse large-scale formal contexts (input data) where the number of attributes per object is small. The algorithm consists of two steps: generate a diagram of a formal context and compute the concept lattice of the formal context using the diagram built in the previous step. The algorithm is experimentally evaluated and compared with algorithms AddExtent and CHARM-*L*.

1 Introduction

A well-known issue in the area of Formal Concept Analysis (FCA) is its computational complexity, i.e. the number of concepts grows exponentially with the amount of data, which are in the form of a binary relation. In our research, we were motivated by large-scale data sets where the maximum number of attributes per object is small. Examples of such data are drug prescriptions databases, jewelery shopping transactions, book store transactions, etc. All these relations (tables) share similar characteristics, namely that there are hundreds of thousands or millions of objects (rows) and thousands of attributes/items (columns) but the data is very sparse, i.e. despite a very large number of items, a drug prescription contains usually only a few drugs, people do not buy dozens of jewels at once as well as they buy only a few books during a visit of the bookstore, etc. We present an algorithm for computing formal concepts from very sparse large-scale data in this paper and show some of its characteristics.

1.1 Formal Concept Analysis [8]

A *formal context* is a triple (X, Y, R) consisting of a set X of objects, a set Y of attributes and an incidence relation $R \subseteq X \times Y$ between them. We write $(x, y) \in R$ to express that the object x has the attribute y. For a set $A \subseteq X$ of objects and a set $B \subseteq Y$ of attributes we define $A^{\uparrow_R} = \{y \in Y : (\forall x \in A)(x, y) \in R\}$ and $B^{\downarrow_R} = \{x \in X : (\forall y \in B)(x, y) \in R\}$. $A^{\uparrow_R}$ is the set of attributes common to the objects in A and $B^{\downarrow_R}$ is the set of objects which have all the attributes in B. A *formal concept* of (X, Y, R) is a pair (A, B) where $A \subseteq X, B \subseteq Y, A^{\uparrow_R} = B$ and $B^{\downarrow_R} = A$. A and B are called the *extent* and the *intent* of (A, B), respectively. The set of all concepts of (X, Y, R) is denoted by $\mathcal{B}(X, Y, R)$. $A \subseteq X$ $(B \subseteq Y)$

is an extent (intent) if and only if $A^{\uparrow_R \downarrow_R} = A$ ($B^{\downarrow_R \uparrow_R} = B$). We define a partial order $\leq$ on $\mathcal{B}(X, Y, R)$ by $(A_1, B_1) \leq (A_2, B_2)) \Leftrightarrow A_1 \subseteq A_2$ (equivalently, $B_1 \supseteq B_2$). The set of all concepts of (X, Y, R) ordered by $\leq$ constitutes the *concept lattice* $(\mathcal{B}(X, Y, R), \leq)$ of (X, Y, R).
Example 1: Table 1 shows a formal context, which induces 11 formal concepts.

Table 1. The formal context (left) and all formal concepts of the context (right)

	J	K	L	M	N
1	×		×	×	
2	×			×	
3		×	×	×	
4		×			×
5	×	×			

Formal concepts		
$(\{1,2,3,4,5\}, \emptyset)$	$(\{5\}, \{J, K\})$	$(\{1\}, \{J, L, M\})$
$(\{1,2,5\}, \{J\})$	$(\{1,2\}, \{J, M\})$	$(\{3\}, \{K, L, M\})$
$(\{3,4,5\}, \{K\})$	$(\{4\}, \{K, N\})$	$(\emptyset, \{J, K, L, M, N\})$
$(\{1,2,3\}, \{M\})$	$(\{1,3\}, \{L, M\})$	

1.2 Related Work

The major problem in computing formal concepts is that some concepts are computed multiple times which significantly slows down the computation. One possible solution is to generate concepts in a specific order, e.g. Ganter's NextClosure algorithm [7] is based on the lexicographical ordering of formal concepts.

Recently, attention has been paid to the CbO algorithm [11] leading to various modifications of it [2, 3, 10]. The order of attributes in a formal context (input table) can yield different CbO trees, and the number of concepts that are computed multiple times is different. The algorithms based on CbO significantly outperform other algorithms. In the competition[1] between algorithms for mining formal concepts, FCbO [10] took first place and the runner-up was In-Close [2]. Next, we focus on algorithms that generate all formal concepts and Hasse diagrams of concept lattices [4–6, 9, 13, 14, 16, 17, 19].

Lindig's algorithm [13] utilizes the idea that if (A, B) is a concept, then for each $x \in X \setminus A$ it holds that $((A \cup \{x\})^{\uparrow_R \downarrow_R}, (A \cup \{x\})^{\uparrow_R})$ is a concept which is greater than (A, B) and deals with the fact that $((A \cup \{x\})^{\uparrow_R \downarrow_R}, (A \cup \{x\})^{\uparrow_R})$ is not necessary an upper neighboor of (A, B).

The algorithm proposed in [16] constructs a lexicographic tree of concepts used to build the diagram graph of a concept lattice. It has the smallest computational complexity, but experiments have shown that other algorithms outperform it [12]. The incremental version of the algorithm was proposed in [17].

Another approach to solve the problem that some concepts are generated multiple times is to search for a new concept in the space of already computed concepts, but this is time consuming. One way to speed up the computation is to reduce the search space by dividing the set of concepts into disjoint sets.

The incremental algorithm proposed by Godin [9] divides the set of concepts into buckets according to the cardinality of their intents. The main idea of the

[1] `http://www.upriss.org.uk/fca/fcaalgorithms.html`

algorithm is that if $(C,D) = \inf\{(A,B) \in \mathcal{B}_{(X,Y,R)} : D = B \cap \{x\}^{\uparrow_R}\}$ and there is no concept of the form $(E,D) \in \mathcal{B}_{(X,Y,R)}$, then (A,B) is the generator of a new concept $(C,D) = (A \cup \{x\}, B \cap \{x\}^{\uparrow_R})$.

The drawback of the Godin algorithm is that it iterates through almost all concepts from $\mathcal{B}_{(X,Y,R)}$ so it would be appropriate to restrict the set of concepts. AddIntent [14] deals with this problem outperforming other algorithms for building lattices ([4, 7, 9, 15, 16]) except for sparse contexts where Bordat algorithm [4] is the fastest one and AddIntent [14] a close second.

FCA and Frequent Itemset Mining are two interconnected areas. The problem of computing (intents of) all formal concepts is equivalent to the problem of mining all closed frequent itemsets using a minimum support equal to zero [18]. The Titanic algorithm [19] for building lattices was inspired by the Apriori algorithm [1] for mining frequent itemsets.

Finally, CHARM (and CHARM-L) [20] explores the itemset and tidset space over an IT-tree and exploits some properties of itemset-tidset pairs for fast identification of frequent closed itemsets. The algorithm uses a vertical data representation called diffsets. To our knowledge, CHARM-L is the only algorithm for generating frequent closed itemsets along with their lattice structure.

2 The Algorithm

For the purposes of describing the algorithm it is useful to view each set of objects $A \in 2^X$ as a word in the alphabet X. We can define a lexicographic order on the set 2^X. This allows us to write $\{x_1,\ldots,x_k\} \in 2^X$ as a word $x_1 \ldots x_k$ where $x_1 < x_2 < \cdots < x_k$. Similarly for sets of attributes, we can write $y_1 \ldots y_l$ where $y_1 < y_2 < \cdots < y_l$ instead of $\{y_1,\ldots,y_l\} \in 2^Y$.

The basic structure the algorithm constructs is a diagram $\mathcal{D}_{(X,Y,R)}$, an example of which is shown in Fig. 1. Each node of the diagram (element of $\mathcal{N}_{(X,Y,R)}$) is a pair $(B^{\downarrow_R}, B)$ consisting of a set of objects $B^{\downarrow_R} \subseteq X$ and a set of attributes $B \subseteq Y$ such that $B \subseteq \{x\}^{\uparrow_R}$ for some $x \in X$. Observe that all nodes in $\mathcal{D}_{(X,Y,R)}$ are different from each other, but there are nodes having the same object parts in $\mathcal{D}_{(X,Y,R)}$. For each node $(B^{\downarrow_R}, B)$ it holds that $B^{\downarrow_R} \neq \emptyset$ except for the so-called bottom node $(\emptyset^{\uparrow_R\downarrow_R}, \emptyset^{\uparrow_R})$ (the node $(\emptyset, JKLMN)$ in Fig. 1) which is added to the diagram at the end of the construction of $\mathcal{D}_{(X,Y,R)}$ in case $\emptyset^{\uparrow_R\downarrow_R} = \emptyset$ (otherwise, the node $(\emptyset^{\uparrow_R\downarrow_R}, \emptyset^{\uparrow_R})$ is in the diagram already).

After adding the bottom node $(\emptyset^{\uparrow_R\downarrow_R}, \emptyset^{\uparrow_R})$ to the diagram of a formal context, this diagram contains all formal concepts of the formal context. Obviously, the diagram can still contain nodes that are not formal concepts, e.g. $(1, JL)$.

The set of nodes $\mathcal{N}_{(X,Y,R)}$ is partially ordered by the $\leq$ relation. The partial order relation $\leq$ on $\mathcal{N}_{(X,Y,R)}$ is defined as follows: $(A_1,B_1) \leq (A_2,B_2)$ if and only if $B_1 \supseteq B_2$. Next, we define the cover relation on $\mathcal{N}_{(X,Y,R)}$ for $\leq$. For two nodes $(A_1,B_1),(A_2,B_2) \in \mathcal{N}_{(X,Y,R)}$, if $(A_1,B_1) \leq (A_2,B_2)$ and there is no node $(A_3,B_3) \in \mathcal{N}_{(X,Y,R)}$ distinct from both (A_1,B_1) and (A_2,B_2) such that $(A_1,B_1) \leq (A_3,B_3) \leq (A_2,B_2)$, then (A_1,B_1) is called a lower cover (or a child) of (A_2,B_2) and (A_2,B_2) is called an upper cover (or a parent) of (A_1,B_1).

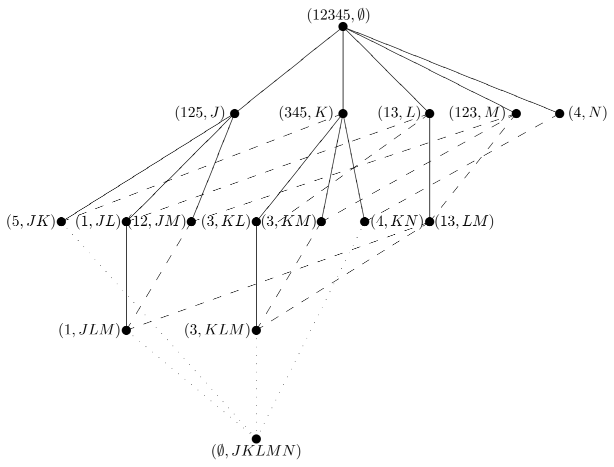

Fig. 1. The diagram $\mathcal{D}_{(X,Y,R)}$ of the formal context in Table 1. Edges (elements of $\mathcal{E}_{(X,Y,R)}$) represent the cover relation between nodes (elements of $\mathcal{N}_{(X,Y,R)}$).

Each (oriented) edge (element of $\mathcal{E}_{(X,Y,R)}$) of the diagram represents the cover relation, i.e. there is an edge between the nodes $(A_1, B_1), (A_2, B_2)$ in the diagram $\mathcal{D}_{(X,Y,R)}$ if and only if (A_1, B_1) is a lower cover of (A_2, B_2). After the node $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})$ is added to $\mathcal{N}_{(X,Y,R)}$ (provided that $\emptyset^{\uparrow_R \downarrow_R} = \emptyset$), it is connected by an edge to all nodes that do not have any child in the diagram, what is indicated by dotted lines.

Considering all nodes $(B^{\downarrow_R}, B) \in \mathcal{N}_{(X,Y,R)}$ such that $B^{\downarrow_R} \neq \emptyset$ and only the solid lines between the nodes, the diagram in Fig. 1 becomes a tree. All nodes at a certain level (depth) of the tree have the same cardinality of attribute parts. This tree is in fact a lexicographic tree similar to the one used by CHARM[2] [20].

The algorithm we present uses a two step approach:

1. Generate a diagram $\mathcal{D}_{(X,Y,R)}$ from the formal context (X, Y, R)
2. Construct the concept lattice $(\mathcal{B}(X, Y, R), \leq)$ of (X, Y, R) using $\mathcal{D}_{(X,Y,R)}$

2.1 Algorithm to Generate $\mathcal{D}_{(X,Y,R)}$

We first present how to modify the diagram (the set of nodes and the set of edges) of a formal context when introducing a new object into the context.

Let us suppose we are adding to a formal context (X, Y, R) a new object $x \notin X$ having attributes $\{y_1, \dots, y_n\}$. We do not assume any overlap of Y and $\{y_1, \dots, y_n\}$, i.e. $\{y_1, \dots, y_n\}$ can contain attributes not present in Y. Denote the

[2] The itemset-tidset search tree (IT-tree) of CHARM contains also nodes $(B^{\downarrow_R}, B)$ where $B^{\downarrow_R} = \emptyset$, e.g. $JKLM$ and KMN, however, by using some properties of IT-pairs, CHARM can quickly enumerate all closed frequent itemsets.

incidence relation between $\{x\}$ and $\{y_1, \ldots, y_n\}$ by $R_x \subseteq \{x\} \times \{y_1, \ldots, y_n\}$ and the new formal context with the object $\{x\}$ added by the triplet (X', Y', R'), where $X' = X \cup \{x\}$, $Y' = Y \cup \{y_1, \ldots, y_n\}$, and $R' \subseteq X' \times Y'$ such that $R' \cap (X \times Y) = R$, $R' \cap (\{x\} \times \{y_1, \ldots, y_n\}) \subseteq R_x$ and $R' \cap (X \times (\{y_1, \ldots, y_n\} \setminus Y)) = R' \cap (\{x\} \times (Y \setminus \{y_1, \ldots, y_n\})) = \emptyset$.

Observe that if $B \subseteq Y \setminus \{y_1, \ldots, y_n\} \neq \emptyset$, then $B^{\downarrow_{R'}} = B^{\downarrow_R}$, and if $B' \subseteq \{y_1, \ldots, y_n\} \setminus Y \neq \emptyset$, then $B'^{\downarrow_{R'}} = B'^{\downarrow_{R_x}} = \{x\}$. Otherwise, for any $B' \subseteq Y \cap \{y_1, \ldots, y_n\}$ it holds that $B'^{\downarrow_{R'}} = B'^{\downarrow_R} \cup \{x\}$. Further, if $A \subseteq X$, then $A^{\uparrow_{R'}} = A^{\uparrow_R}$. Obviously, $\{x\}^{\uparrow_{R'}} = \{x\}^{\uparrow_{R_x}} = \{y_1, \ldots, y_n\}$.

From the above presented ideas, for any $B' \subseteq \{y_1, \ldots, y_n\}$, if there is a node $(A, B) \in \mathcal{N}_{(X,Y,R)}$ such that $B' = B$, then the node $(A', B') \in \mathcal{N}_{(X',Y',R')}$ with the same attribute part $B' = B$ as the node (A, B) is a modified node enlarging the object part of (A, B) by the object x. Otherwise, if $B' \supset B$, then the node $(A', B') = (\{x\}, B') \in \mathcal{N}_{(X',Y',R')}$ is a new node. Finally, if $B \subseteq Y \setminus \{y_1, \ldots, y_n\} \neq \emptyset$, then the node (A, B) is old.

In addition, the edges of the diagram $\mathcal{D}_{(X',Y',R')}$ have to be updated. Each new node in $\mathcal{N}_{(X',Y',R')}$ can be connected by an edge only to modified or new nodes in $\mathcal{N}_{(X',Y',R')}$ (never to old nodes). If a node (A', B') is new in $\mathcal{N}_{(X',Y',R')}$, then (A', B') will be a child of a node (C', D') in the diagram $\mathcal{D}_{(X',Y',R')}$ if and only if $B' \supset D'$ and $|B'| = |D'| + 1$.

Remark 1. The new object x "adds" to the diagram $\mathcal{D}_{(X,Y,R)}$ a Boolean algebra, i.e. modified and new nodes in $\mathcal{D}_{(X',Y',R')}$ form a Boolean algebra.

Lemma 1. *Let (X, Y, R) be a formal context and (X', Y', R') be the formal context originated from (X, Y, R) by adding a new object $x \notin X$ having attributes $\{y_1, \ldots, y_n\}$. If $\mathcal{B}(X, Y, R) \subseteq \mathcal{N}_{(X,Y,R)} \cup \{(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})\}$, then $\mathcal{B}(X', Y', R') \subseteq \mathcal{N}_{(X',Y',R')} \cup \{(\emptyset^{\uparrow_{R'} \downarrow_{R'}}, \emptyset^{\uparrow_{R'}})\}$.*

Proof. $\mathcal{B}(X', Y', R')$ can be obtained from $\mathcal{B}(X, Y, R)$ by taking all formal concepts in $\mathcal{B}(X, Y, R)$ and modifying the extent of the formal concepts $(A, B) \in \mathcal{B}(X, Y, R)$ for which $B \subseteq \{y_1, \ldots, y_n\}$ by adding x. The concepts that remain intact are called old and the others, modified concepts in $\mathcal{B}(X', Y', R')$. In addition, new concepts are created. These new concepts are always in the form $(A \cup \{x\}, B \cap \{y_1, \ldots, y_n\})$ for some formal concept $(A, B) \in \mathcal{B}(X, Y, R)$. [9] Note that, obviously, $B \cap \{y_1, \ldots, y_n\} \subseteq \{y_1, \ldots, y_n\}$.

For any $B' \subseteq \{y_1, \ldots, y_n\}$ it holds that $(A', B') \in \mathcal{N}_{(X',Y',R')}$, i.e. (A', B') is modified or new node in $\mathcal{N}_{(X',Y',R')}$, and thus $x \in A'$ (see above).

From the assumption $\mathcal{B}(X, Y, R) \subseteq \mathcal{N}_{(X,Y,R)} \cup \{(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})\}$ and the previous considerations it follows that $\mathcal{B}(X', Y', R') \subseteq \mathcal{N}_{(X',Y',R')} \cup \{(\emptyset^{\uparrow_{R'} \downarrow_{R'}}, \emptyset^{\uparrow_{R'}})\}$. $\square$

The procedure *updateDiagram* depicted in Algorithm 1 computes the diagram $\mathcal{D}_{(X',Y',R')}$ of the formal context (X', Y', R') when adding a new object x having attributes $\{y_1, \ldots, y_n\}$ into the diagram $\mathcal{D}_{(X,Y,R)}$. The procedure accepts as its arguments a diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ of a formal context (X, Y, R), a new object x and attributes $\{y_1, \ldots, y_n\}$ of the object x. The procedure outputs the updated diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$.

Algorithm 1. Procedure *updateDiagram*$(\mathcal{D}, x, \{y_1, \dots, y_n\})$

```
1.  $i = 1$
2.  $array[0] \leftarrow \emptyset$
3.  for all $y \in \{y_1, \dots, y_n\}$ do
4.    for $j$ from 0 to $i - 1$ do
5.      $array[i + j] \leftarrow array[j] \cup \{y\}$
6.      $B \leftarrow array[i + j]$
7.      if there is $A$ such that $(A, B) \in \mathcal{N}$ then
8.        $A \leftarrow A \cup \{x\}$
9.      else
10.       $A \leftarrow \{x\}$
11.       add node $(A, B)$ to $\mathcal{N}$
12.       for all $(C, D) \in \mathcal{N}, B \supseteq D, |B| = |D| + 1$ do
13.         add edge $(A, B) \rightarrow (C, D)$ to $\mathcal{E}$
14.   $i \leftarrow i * 2$
```

The key step when processing the object x having attributes $\{y_1, \dots, y_n\}$ is the generation of all subsets of $\{y_1, \dots, y_n\}$ by using local variables *array* as a temporary storage for generated subsets of the set of attributes $\{y_1, \dots, y_n\}$.

After a new subset of $\{y_1, \dots, y_n\}$ is generated (line 5), the content of B is updated (line 6). Observe that B is the copy of this subset.The procedure checks if there is a node in the diagram $\mathcal{D}$ having B as its attribute part (line 7). The node is denoted by (A, B). If the test succeed, the node (A, B) is modified by augmenting A by the object x (line 8). If the test in line 7 fails, a new node $(\{x\}, B)$ is added to the diagram $\mathcal{D}$ (lines 10, 11) and it is connected to the corresponding nodes (lines 12-13).

Observe the order in which the subsets of $\{y_1, \dots, y_n\}$ are generated. If D is a subset of B, then D has been generated before B. Therefore, whenever a new node (A, B) is created, it can be connected to all nodes (C, D) such that $B \supseteq D$ and $|B| = |D| + 1$ (because all these nodes were created a step before).

Example 2: We illustrate Algorithm 1 on the following example: Consider the first two objects (rows) of the formal context in the table 1. The diagram $\mathcal{D}$ of the correspoding formal context contains the following 8 nodes: $N_1 = (12, \emptyset)$, $N_2 = (12, J)$, $N_3 = (1, L)$, $N_4 = (1, JL)$, $N_5 = (12, M)$, $N_6 = (12, JM)$, $N_7 = (1, LM)$, $N_8 = (1, JLM)$, and, the following 12 edges: $N_2 \rightarrow N_1$, $N_3 \rightarrow N_1$, $N_4 \rightarrow N_2$, $N_4 \rightarrow N_3$, $N_5 \rightarrow N_1$, $N_6 \rightarrow N_2$, $N_6 \rightarrow N_5$, $N_7 \rightarrow N_3$, $N_7 \rightarrow N_5$, $N_8 \rightarrow N_4$, $N_8 \rightarrow N_6$, $N_8 \rightarrow N_7$. When processing the third object (row), the following 4 new nodes are added: $N_9 = (3, K)$, $N_{10} = (3, KL)$, $N_{11} = (3, KM)$, $N_{12} = (3, KLM)$, and, the following 8 new edges are created: $N_9 \rightarrow N_1$, $N_{10} \rightarrow N_9$, $N_{10} \rightarrow N_3$, $N_{11} \rightarrow N_9$, $N_{11} \rightarrow N_5$, $N_{12} \rightarrow N_9$, $N_{12} \rightarrow N_3$, $N_{12} \rightarrow N_5$. In this step, the node N_1 is modified to $N_1 = (123, \emptyset)$, N_3 to $N_3 = (13, L)$, N_5 to $N_5 = (123, M)$ and N_7 to $N_7 = (13, LM)$.

Algorithm 1 can be easily used to generate the diagram $\mathcal{D}_{(X,Y,R)}$ of a formal context (X, Y, R) incrementally, i.e. processing objects of the formal context one by one. The function *generateDiagram* in Algorithm 2 accepts as its argument a formal context (X, Y, R) and outputs the diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ of (X, Y, R).

Algorithm 2. Function *generateDiagram*$((X, Y, R))$

```
 1. $\mathcal{N} \leftarrow \emptyset$
 2. $\mathcal{E} \leftarrow \emptyset$
 3. for all $x \in X$ do
 4.    $updateDiagram(\mathcal{D}, x, \{x\}^{\uparrow_R})$
 5. if $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R}) \notin \mathcal{N}$ then
 6.    $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})\}$
 7.    for all $(A, B) \in \mathcal{N}$ do
 8.       if there is no $(C, D) \rightarrow (A, B)$ in $\mathcal{E}$ then
 9.          add edge $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R}) \rightarrow (A, B)$ to $\mathcal{E}$
10. return $\mathcal{D}$
```

Initially, the diagram $\mathcal{D}$ is empty, i.e. contains no nodes nor edges (lines 1, 2). Then, *updateDiagram* is invoked repeatedly for each object $x \in X$ (lines 3 - 4). Finally, if there is the node $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})$ in $\mathcal{D}$, i.e. the test in line 5 fails, the diagram $\mathcal{D}$ is the diagram of the formal context (X, Y, R). Otherwise, i.e. the test in line 5 pass, the node $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R})$ is added to the diagram $\mathcal{D}$ (line 6) and it is connected to all nodes in $\mathcal{D}$ that do not have any child in $\mathcal{D}$ (lines 7 - 9).

Remark 2. The outcome of *generateDiagram* function in Algorithm 2 does not depend on the order in which the objects of the formal context are acquired.

Theorem 1. *The function generateDiagram in Algorithm 2 is correct, i.e. given a formal context* (X, Y, R)*, it stops after finitely many steps and returns the diagram* $\mathcal{D}_{(X,Y,R)}$ *that satisfies the following conditions:*

1. $\mathcal{D}_{(X,Y,R)}$ *contains all formal concepts of the formal context* (X, Y, R)*, i.e.* $\mathcal{B}(X, Y, R) \subseteq \mathcal{N}_{(X,Y,R)}$.
2. *The diagram* $\mathcal{D}_{(X,Y,R)}$ *is the Hasse diagram of a complete lattice.*

Proof. 1. Obviously, $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R}) \in \mathcal{N}_{(X,Y,R)}$. The theorem is therefore a direct consequence of Lemma 1.
2. We prove that for each $(A, B), (C, D) \in \mathcal{N}_{(X,Y,R)}$ there exists a greatest lower bound (a) and a least upper bound (b).

a First, we show that for each $(A, B), (C, D) \in \mathcal{N}_{(X,Y,R)}$ there exists $(E, F) \in \mathcal{N}_{(X,Y,R)}$ such that $(E, F) \leq (A, B)$ and $(E, F) \leq (C, D)$.
Let $F = B \cup D$. If $(E, F) \in \mathcal{N}_{(X,Y,R)}$, then $(E, F) \leq (A, B)$ and $(E, F) \leq (C, D)$ (because $F \supseteq B$ and $F \supseteq D$). Moreover, for each $(A, B), (C, D) \in \mathcal{N}_{(X,Y,R)}$ it holds if $(G, H) \leq (A, B)$ and $(G, H) \leq (C, D)$, then $(G, H) \leq (E, F)$ (if $H \supseteq B$ and $H \supseteq D$, then $H \supseteq B \cup D = F$). Otherwise, if $(E, F) \notin \mathcal{N}_{(X,Y,R)}$, then the greatest lower bound of (A, B) and (C, D) is $(\emptyset^{\uparrow_R \downarrow_R}, \emptyset^{\uparrow_R}) \in \mathcal{N}_{(X,Y,R)}$.

b Next, we show that for each $(A,B),(C,D) \in \mathcal{N}_{(X,Y,R)}$ there exists $(E,F) \in \mathcal{N}_{(X,Y,R)}$ such that $(A,B) \leq (E,F)$ and $(C,D) \leq (E,F)$.
Let $F = B \cap D$. Then, $(A,B) \leq (E,F)$ and $(C,D) \leq (E,F)$ ($B \supseteq F$ and $D \supseteq F$). Moreover, from the fact that $(A,B) \in \mathcal{N}_{(X,Y,R)}$ and $F \subseteq B$ it follows that $(E,F) \in \mathcal{N}_{(X,Y,R)}$. Further, for each $(A,B),(C,D) \in \mathcal{N}_{(X,Y,R)}$ it holds that if $(A,B) \leq (G,H)$ and $(C,D) \leq (G,H)$, then $(E,F) \leq (G,H)$ (if $B \supseteq H$ and $D \supseteq H$, then $F = B \cap D \supseteq H$).

The diagram $\mathcal{D}_{(X,Y,R)}$ is finite and, clearly, every finite lattice is complete. Therefore, $\mathcal{D}_{(X,Y,R)}$ is the Hasse diagram of a complete lattice. $\square$

2.2 Algorithm to Construct $(\mathcal{B}(X,Y,R), \leq)$ using $\mathcal{D}_{(X,Y,R)}$

The diagram $\mathcal{D}_{(X,Y,R)}$ contains all formal concepts of the formal context (X,Y,R) and possibly some other nodes (see Theorem 1). In this section we show that if we (properly) remove all non-concept nodes from $\mathcal{D}_{(X,Y,R)}$, then $\mathcal{D}_{(X,Y,R)}$ will be the Hasse diagram of the concept lattice $(\mathcal{B}(X,Y,R), \leq)$.

The identification of formal concepts in the diagram $\mathcal{D}_{(X,Y,R)}$ is simple. Naturally, formal concepts are these nodes (A,B) of $\mathcal{D}_{(X,Y,R)}$ for which $A^{\uparrow_R} = B$. The other way to identify whether a node (A,B) is the formal concept or not is by using the cover relation on the nodes of $\mathcal{D}_{(X,Y,R)}$ what we will see later.

First, we define an equivalence relation θ on the nodes of the diagram $\mathcal{D}_{(X,Y,R)}$.

Definition 1. *We define an equivalence relation θ on $\mathcal{N}_{(X,Y,R)}$ as follows: for nodes $(A,B),(C,D) \in \mathcal{N}_{(X,Y,R)}$ it holds $(A,B)\theta(C,D)$ if and only if $A = C$. The equivalence class of (A,B) is given by $[(A,B)]_\theta = \{(C,D) \in \mathcal{N}_{(X,Y,R)} : (A,B)\theta(C,D)\}$.*

All nodes from $\mathcal{N}_{(X,Y,R)}$ having the same object parts are in the same equivalence class, and, only one node in an equivalence class $[(A,B)]_\theta$, namely $(A, A^{\uparrow_R})$, i.e. $(A^{\uparrow_R\downarrow_R}, A^{\uparrow_R})$, is the formal concept of the formal context (X,Y,R).

Theorem 2. *Let (X,Y,R) be a formal context and let $(A,B) \in \mathcal{N}_{(X,Y,R)}$. Then, $(A, A^{\uparrow_R})$ is the smallest element of the equivalence class $[(A,B)]_\theta$ w.r.t. $\leq$.*

Proof. Obviously, $(A, A^{\uparrow_R}) \in \mathcal{N}_{(X,Y,R)}$. Next, we have to show that if $(C,D) \in [(A,B)]_\theta$, then $(A, A^{\uparrow_R}) \leq (C,D)$.

Let $(C,D) \in [(A,B)]_\theta$. Then, it holds that $A = C$. Moreover, from the fact that $(C,D) \in \mathcal{N}_{(X,Y,R)}$ it follows that $C = D^{\downarrow_R}$, and thus $A \subseteq D^{\downarrow_R}$. Since ${}^{\uparrow_R}$, ${}^{\downarrow_R}$ form a Galois connection [8], it holds $A^{\uparrow_R} \supseteq D$ which implies $(A, A^{\uparrow_R}) \leq (C,D)$. $\square$

The previous theorem allows us to identify non-concept nodes of the diagram $\mathcal{D}_{(X,Y,R)}$ using the cover relation on $\mathcal{N}_{(X,Y,R)}$ as follows: $(A,B) \in \mathcal{N}_{(X,Y,R)}$ is not the formal concept of (X,Y,R) if and only if (A,B) has a child having the same object part in the diagram $\mathcal{D}_{(X,Y,R)}$. Actually, it suffices to consider only the cardinality of object parts of the nodes (see Corollary 1).

Corollary 1. *Let (X, Y, R) be a formal context and let $(A, B) \in \mathcal{N}_{(X,Y,R)}$. $(A, B) \notin \mathcal{B}(X, Y, R)$ if and only if there exists $(C, D) \in \mathcal{N}_{(X,Y,R)}$ such that $|C| = |A|$ and (C, D) is a child of (A, B) in $\mathcal{D}_{(X,Y,R)}$.*

Proof. The claim follows from the fact that if $C \subseteq A$ and $|C| = |A|$, then $C = A$ (provided that C and A are finite sets). □

Now we are able to determine whether a node of the diagram is the formal concept or not. In the following, we describe how to remove a non-concept node from the diagram. More precisely, we show how to modify the set of edges of the diagram after a non-concept node is removed.

Let $\mathcal{D}_{(X,Y,R)}$ be the diagram of a formal context (X, Y, R) and let us suppose we are removing a node $(A, B) \notin \mathcal{B}(X, Y, R)$ from the diagram $\mathcal{D}_{(X,Y,R)}$. Denote by $\mathcal{D}'_{(X,Y,R)}$ the diagram constructed from the diagram $\mathcal{D}_{(X,Y,R)}$ by removing the node (A, B) and by updating the set of edges.

$\mathcal{D}'_{(X,Y,R)}$ is constructed from $\mathcal{D}_{(X,Y,R)}$ as follows: First, the node (A, B) and all edges connecting this node to all its children (let S be a set of children of (A, B)) as well as its parents (let T be a set of parents of (A, B)) are removed from the diagram $\mathcal{D}_{(X,Y,R)}$. Clearly, for each $(S_1, S_2) \in S$ and $(T_1, T_2) \in T$ it holds that $(S_1, S_2) \leq (T_1, T_2)$. Possibly, (S_1, S_2) is the child of (T_1, T_2) (or (T_1, T_2) is the parent of (S_1, S_2)) in the diagram $\mathcal{D}'_{(X,Y,R)}$. If there exists a path that goes upward from (S_1, S_2) to (T_1, T_2) and does not pass through the node (A, B) in the diagram $\mathcal{D}_{(X,Y,R)}$, then (S_1, S_2) is not the child of (T_1, T_2) in the diagram $\mathcal{D}'_{(X,Y,R)}$. Otherwise, (S_1, S_2) is the child of (T_1, T_2) in $\mathcal{D}'_{(X,Y,R)}$, and thus the edge from (S_1, S_2) to (T_1, T_2) is added to the diagram $\mathcal{D}_{(X,Y,R)}$.

Lemma 2. *Let $\mathcal{D}_{(X,Y,R)}$ be the diagram of a formal context (X, Y, R), $(A, B) \in \mathcal{N}_{(X,Y,R)}$ and $(A, B) \notin \mathcal{B}(X, Y, R)$. Let $\mathcal{D}'_{(X,Y,R)}$ be the diagram constructed from $\mathcal{D}_{(X,Y,R)}$ by removing (A, B) as described above. There is a path that goes upward from a node (C, D) to a node (E, F) in $\mathcal{D}'_{(X,Y,R)}$ if and only if $(C, D) \leq (E, F)$.*

Proof. Let $((C, D) = P_1, \ldots, P_i = (E, F))$ be a path that goes upward from (C, D) to (E, F) in the diagram $\mathcal{D}'_{(X,Y,R)}$. We differentiate the two cases:
1. The path $(P_1, \ldots, P_i)$ does not pass through the node (A, B) in the diagram $\mathcal{D}_{(X,Y,R)}$, i.e. $P_j \neq (A, B)$ for each $j \in \{1, \ldots, i\}$. In this case the proposition follows from the assumption.
2. The path passes through the node (A, B) in $\mathcal{D}_{(X,Y,R)}$ (which is not contained in the diagram $\mathcal{D}'_{(X,Y,R)}$), i.e. there exists $j \in \{1, \ldots, i\}$ such that $P_j = (A, B)$. Then, the path also goes through a child (S_1, S_2) and a parent (T_1, T_2) of (A, B). It holds that either there exists a path that goes upward from (S_1, S_2) to (T_1, T_2) in $\mathcal{D}'_{(X,Y,R)}$, or the edge from (S_1, S_2) to (T_1, T_2) is added to the diagram when the diagram $\mathcal{D}'_{(X,Y,R)}$ is constructed, and thus the assertion holds. □

The procedure in Algorithm 3 removes a node (A, B) from the diagram $\mathcal{D}_{(X,Y,R)}$ of (X, Y, R) with adjusting the set of edges of the diagram. The procedure *removeNode* accepts as its arguments a diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ of a formal

Algorithm 3. Procedure *removeNode*$(\mathcal{D}, (A,B))$

```
1.  $S \leftarrow$ children of $(A,B)$ in $\mathcal{D}$
2.  $T \leftarrow$ parents of $(A,B)$ in $\mathcal{D}$
3.  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(A,B)\}$
4.  for all $(S_1,S_2) \in S$ do
5.     remove edge $(S_1,S_2) \rightarrow (A,B)$ from $\mathcal{E}$
6.  for all $(T_1,T_2) \in T$ do
7.     remove edge $(A,B) \rightarrow (T_1,T_2)$ from $\mathcal{E}$
8.  for all $(S_1,S_2) \in S$ do
9.     for all $(T_1,T_2) \in T$ do
10.       if there is no path from $(S_1,S_2)$ to $(T_1,T_2)$ in $\mathcal{D}$ then
11.          add edge $(S_1,S_2) \rightarrow (T_1,T_2)$ to $\mathcal{E}$
```

context (X,Y,R) and a node (A,B) of the diagram. The procedure outputs the modified diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$.

First, the node (A,B) and all edges connecting this node with its children as well as its parents are removed from the diagram $\mathcal{D}$ (lines 3 - 7). Then, if there is no path that goes upward from a child (S_1,S_2) to a parent (T_1,T_2) of (A,B) in $\mathcal{D}$, the egde from (S_1,S_2) to (T_1,T_2) is added to $\mathcal{D}$ (lines 8 - 11).

Algorithm 3 is used to construct the Hasse diagram of the concept lattice $(\mathcal{B}(X,Y,R),\leq)$ from the diagram $\mathcal{D}_{(X,Y,R)}$. The procedure *computeLattice* in Algorithm 4 accepts as its argument a diagram $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ of a formal context (X,Y,R) and computes the Hasse diagram of $(\mathcal{B}(X,Y,R),\leq)$.

Algorithm 4. Procedure *computeLattice*$(\mathcal{D})$

```
1.  for all $(A,B) \in \mathcal{N}$ do
2.     if $(A,B)$ has a child $(C,D)$ in $\mathcal{D}$ such that $|C| = |A|$ then
3.        removeNode$(\mathcal{D}, (A,B))$
```

The idea of *computeLattice* procedure is the following: Nodes of the diagram $\mathcal{D}$ are processed one by one (line 1). If the test in line 2 passes, i.e. a node (A,B) of the diagram $\mathcal{D}$ is not the formal concept of $\mathcal{B}(X,Y,R)$, then the node (A,B) is removed from $\mathcal{D}$ using *removeNode* procedure described above (line 3).

Theorem 3. *The procedure computeLattice in Algorithm 4 is correct, i.e. for a given diagram $\mathcal{D}_{(X,Y,R)}$, it stops after finitely many steps and computes the Hasse diagram of the concept lattice $(\mathcal{B}(X,Y,R),\leq)$.*

Proof. The diagram $\mathcal{D}_{(X,Y,R)}$ contains all formal concepts of the formal context according to Theorem 1. The statement follows from Corollary 1 (only non-concept nodes of $\mathcal{D}_{(X,Y,R)}$ are removed) and Lemma 2. □

2.3 Complexity of the Algorithm

In this section we analyze the worst-case time complexity of the algorithm.

Let n be the maximum number of attributes per object in a formal context (X, Y, R), i.e. $n = \max\{|\{x\}^{\uparrow_R}| : x \in X\}$. Observe that each node of $\mathcal{D}_{(X,Y,R)}$ has at most n parents and n is the depth of the diagram $\mathcal{D}_{(X,Y,R)}$.

The outer loops of Algorithm 1 (lines 3 and 4), i.e. generating the power set of $\{x\}^{\uparrow_R}$, require $O(2^n)$ time. Assuming that the test to check whether each generated set of attributes B is the attribute part of a node already present in the diagram (line 7) is done using a binary search, this part takes $O(n \log |Y|)$. Each new node (A, B) is connected to the corresponding nodes (lines 12 - 13) in time $O(n)$. Thus, the time complexity of Algorithm 1 is $O(2^n n^2 \log |Y|)$.

The complexity of the algorithm 2 is $O(|X| 2^n (n^2 \log |Y| + 1))$.

In Algorithm 3, all edges connecting the node (A, B) with its parents as well as its children are removed from the diagram $\mathcal{D}_{(X,Y,R)}$ (lines 4 - 7) in time $O(n + |Y|)$. The test whether there is a path (consisting of egdes of $\mathcal{D}_{(X,Y,R)}$) connecting a child (S_1, S_2) and a parent (T_1, T_2) of (A, B) in $\mathcal{D}_{(X,Y,R)}$ (lines 10 - 11) takes $O(n!)$. Thus, the time complexity of Algorithm 3 is $O(n!\,(n + |Y|))$.

The complexity of the algorithm 4 is $O(|X| 2^n\, n!\,(n + |Y|))$.

The maximum number of attributes per object and the total number of nodes of the diagram $\mathcal{D}_{(X,Y,R)}$ have significant impact on the efficiency of the algorithm.

3 Experimental Evaluation

We have first evaluated the proposed algorithm by comparing the number of formal concepts of a given context and the number of nodes of the diagram constructed by the algorithm. Definitely, the higher the average number of attributes per object in a formal context, the higher the number of nodes of the diagram generated by the algorithm that are not formal concepts (but due to the lack of space we omit the details of this experiment).

Next, we have experimentally compared the performance of the proposed algorithm against AddExtent (an attribute-incremental implementation of AddIntent [14]) that has been provided to us by one of its authors[3], and CHARM-L [20] that is publicly available. The experiments were conducted on the computing node with 16 cores equipped with 24 GB RAM memory running GNU/Linux openSUSE 12.1. We have run the experiments on randomly generated data and measured CPU times with outputs of algorithms turned off.

The results of experiments are depicted in Fig. 2 and Fig. 3. The names of the graphs in the figures indicate the total number of attributes, the average and maximum number of attributes per object in data sets, respectively.

Let us first compare how the algorithms perform on data sets in which the average number of attributes per object is 2 (top of Fig. 2 and Fig. 3). On these data sets, our algorithm is the best choice. The number of objects does not affect the performance of our algorithm much unlike AddExtent and CHARM-L.

Next, let us analyze the results of experiments on data sets with the average number of attributes per object set to 3 (middle of Fig. 2 and Fig. 3). If the total number of attributes is low (1000, 5000), our algorithm has similar performance

[3] We thank Sergei O. Obiedkov for providing the source code for AddExtent.

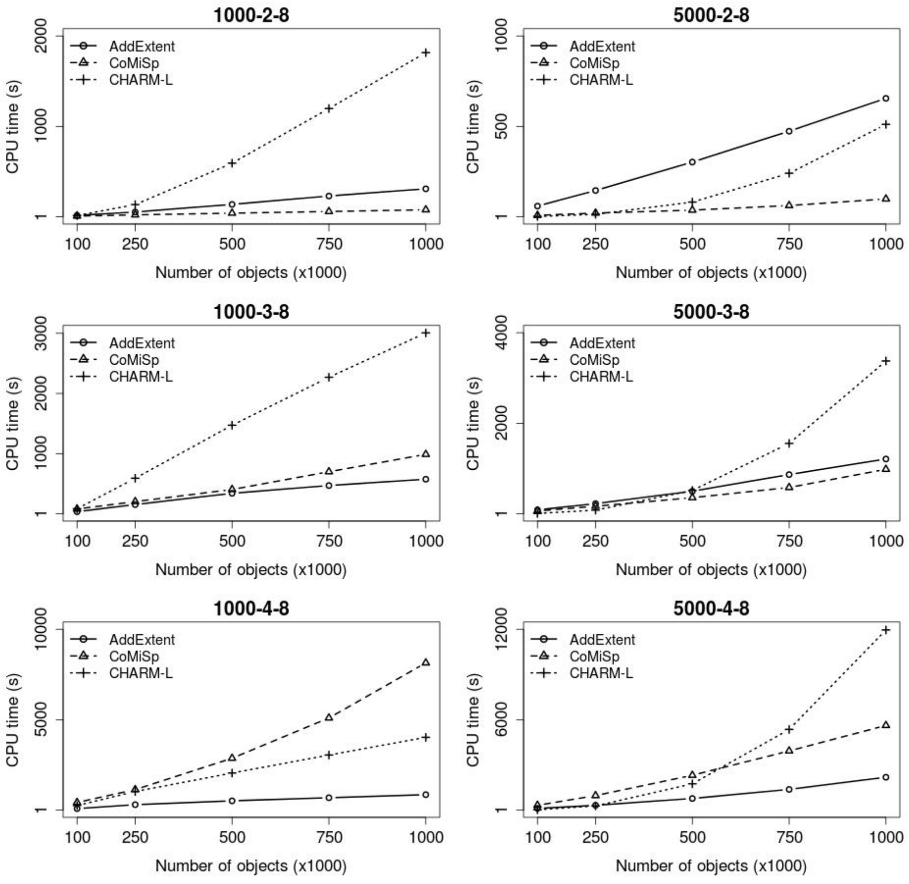

Fig. 2. The performance of algorithms (CPU times in seconds) on random data sets having 1000 attributes (left) and 5000 attributes (right). The average number of attributes per object is 2 (top), 3 (middle) and 4 (bottom), and the maximum number of attributes per object is set to 8.

as AddExtent. On data sets having the total number of attributes 10000 and 50000 (middle of Fig. 3), the behaviour of our algorithm is curious. Our algorithm outperforms both AddExtent and CHARM-*L* on all data sets except for those where the number of objects is 1000000.

Finally, let us consider the performance of algorithms on data sets where the average number of attributes per object is 4 (bottom of Fig. 2 and Fig. 3). AddExtent outperforms both other algorithms on data sets where the total number of attributes is 1000. If the total number of attributes is increased to 5000, our algorithm performs better than CHARM-*L*, but AddExtent still outperforms our algorithm. If the total number of attributes is 10000 or 50000 and the maximum number of attributes per object is lowered to 6 (bottom of Fig. 3), our algorithm performs better than AddExtent and have similar performance as CHARM-*L*.

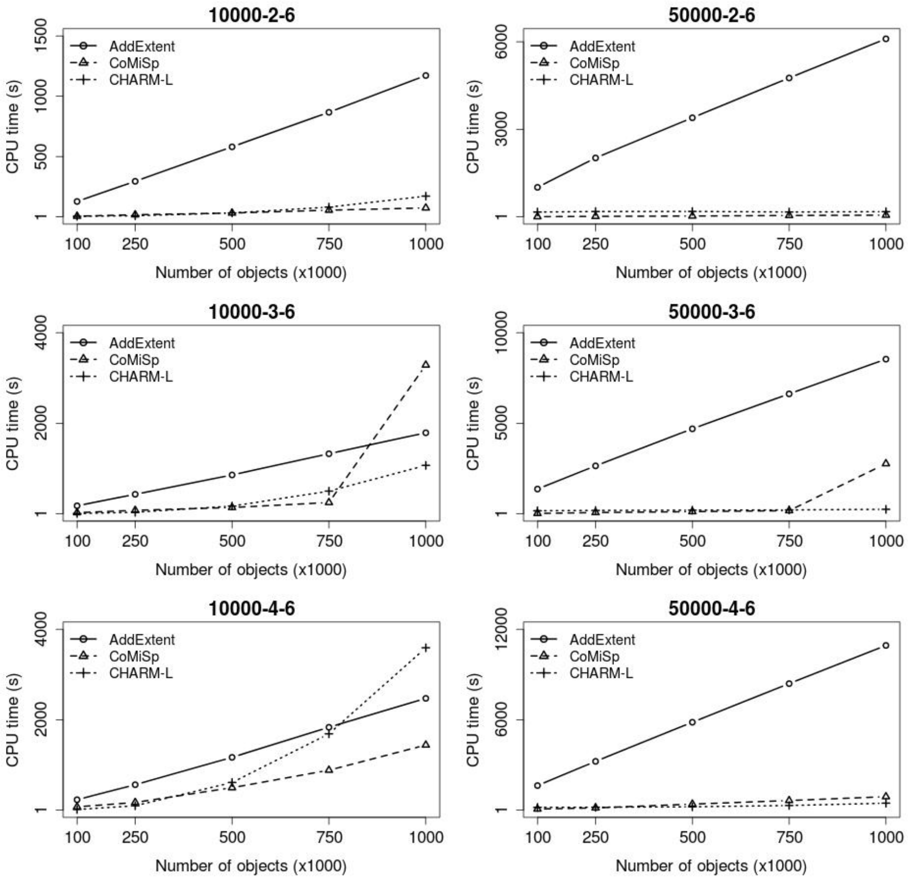

Fig. 3. The performance of algorithms (CPU times in seconds) on random data sets having 10000 attributes (left) and 50000 attributes (right). The average number of attributes per object is 2 (top), 3 (middle) and 4 (bottom), and the maximum number of attributes per object is set to 6.

4 Conclusion

We presented an algorithm for mining formal concepts from very sparse large-scale data. We identified some issues for our future work, the most important of which are (i) to identify some speed-up possibilities of our algorithm both on theoretical as well as implementation side, (ii) provide more exhaustive experiments on larger – and possibly real – datasets and (iii) investigate the way of parallelization of our algorithm. Even if some work remains to fine-tune our algorithm, we believe that it is worth of further investigation.

Acknowledgements. This publication is the result of the Project implementation: University Science Park *TECHNICOM* for Innovation Applications Supported by Knowledge Technology, *ITMS: 26220220182*, supported by the Research & Development Operational Programme funded by the ERDF and partially supported by the projects Center of knowledge and information systems in Košice *CEZIS, ITMS: 26220220158* and *VEGA 1/0832/12*.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993)
2. Andrews, S.: In-close, a fast algorithm for computing formal concepts. In: Supplementary Proceedings of ICCS. CEUR WS, vol. 483 (2009)
3. Andrews, S.: In-close2, a high performance formal concept miner. In: Andrews, S., Polovina, S., Hill, R., Akhgar, B. (eds.) ICCS 2011. LNCS (LNAI), vol. 6828, pp. 50–62. Springer, Heidelberg (2011)
4. Bordat, J.P.: Calcul pratique du trelis de galois dune correspondance. Math. Sci. Hum. 96 (1986)
5. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: Proc. of the 10th Conf. on Machine Learning (1993)
6. Chein, M.: Algorithme de recherche des sous-matrices premiéres dune matrice. Bull. Math. Soc. Sci. Math. R. S. Roumanie 13 (1969)
7. Ganter, B.: Two basic algorithms in concept analysis. Tech. Rep. FB4 No. 831 (1984)
8. Ganter, B., Wille, R.: Formal concept analysis. Springer (1999)
9. Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on galois (concept) lattice. Computational Intelligence 11 (1995)
10. Krajča, P., Outrata, J., Vychodil, V.: Advances in alg. based on CbO. In: CLA 2010 (2010)
11. Kuznetsov, S.O.: A fast algorithm for computing all intersections of objects in a finite semi-lattice. Aut. Docum. and Math. Linguistics 27(5), 11–21 (1993)
12. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Jour. of Exp. and Theor. Art. Intelligence 14(23) (2002)
13. Lindig, C.: Fast concept analysis. In: Working with Conceptual Structures - Contributions to ICCS 2000 (2000)
14. van der Merwe, D., Obiedkov, S., Kourie, D.: AddIntent: A new incremental algorithm for constructing concept lattices. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 372–385. Springer, Heidelberg (2004)
15. Norris, E.M.: An algorithm for computing the maximal rectangles in a binary relation. Revue Roumaine de Mathmatiques Pures et Appliqués 23(2), 243–250 (1978)
16. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Information Processing Letters 71, 199–204 (1999)
17. Nourine, L., Raynaud, O.: A fast incremental algorithm for building lattices. Jour. of Exp. and Theor. Art. Intelligence 14, 217–227 (2002)

18. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Pruning Closed Itemset Lattices for Association Rules. In: Actes Bases De Données Avancées Bda 1998, Hammamet, Tunisie, pp. 177–196 (1998)
19. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with titanic. Knowl. and Data Eng. 42(2), 189–222 (2002)
20. Zaki, M.J., Hsiao, C.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE TKDE 17(4), 462–478 (2005)