Computing and Informatics, Vol. 34, 2015, 77–98

DISTRIBUTED COMPUTATION OF GENERALIZED ONE-SIDED CONCEPT LATTICES ON SPARSE DATA TABLES

Peter Butka

Department of Cybernetics and Artificial Intelligence
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
e-mail: peter.butka@tuke.sk

Jozef Pócs

Department of Algebra and Geometry
Palacký University Olomouc
17. listopadu 12, 771 46 Olomouc, Czech Republic
&
Mathematical Institute, Slovak Academy of Sciences
Grešákova 6, 040 01 Košice, Slovakia
e-mail: pocs@saske.sk

Jana Pócsová

BERG Faculty
Institute of Control and Informatization of Production Processes
Technical University of Košice
Boženy Němcovej 3, 043 84 Košice, Slovakia
e-mail: jana.pocsova@tuke.sk

Abstract. In this paper we present the study on the usage of distributed version of the algorithm for generalized one-sided concept lattices (GOSCL), which provides a special case for fuzzy version of data analysis approach called formal concept

analysis (FCA). The methods of this type create the conceptual model of the input data based on the theory of concept lattices and were successfully applied in several domains. GOSCL is able to create one-sided concept lattices for data tables with different attribute types processed as fuzzy sets. One of the problems with the creation of FCA-based models is their computational complexity. In order to reduce the computation times, we have designed the distributed version of the algorithm for GOSCL. The algorithm is able to work well especially for data where the number of newly generated concepts is reduced, i.e., for sparse input data tables which are often used in domains like text-mining and information retrieval. Therefore, we present the experimental results on sparse data tables in order to show the applicability of the algorithm on the generated data and the selected text-mining datasets.

Keywords: One-sided concept lattices, distributed algorithm, formal concept analysis, sparse data, text-mining

Mathematics Subject Classification 2010: 06A15, 06B99, 68T30

1 INTRODUCTION

The large amount of available data and the needs for their analysis brings up the challenges to the area of data mining. It is evident that methods for different analysis should be more effective and understandable. One of the conceptual data mining methods, called formal concept analysis (FCA) [13], is an exploratory data analytical approach which identifies conceptual structures (concept lattices) among data sets. FCA has been found useful for analysis of data in many areas like knowledge discovery, data/text mining, information retrieval, etc. The standard approach to FCA provides the method for analysis of object-attribute models based on the binary relation (where object has/has-not particular attribute). The extension of the classic approach is based on some fuzzifications for which object-attribute models describe the relationship between objects and attributes as fuzzy relations. From the well-known approaches for fuzzification we could mention an approach of Bělohlávek [2], Krajči [24], Popescu [30], the approach based on the multi-adjoint concept lattices [25, 26], and also work by one of the authors generalizing the other approaches [29]. A nice survey and comparison of some existing approaches to fuzzy concept lattices is presented also in [3].

In practical applications, so-called one-sided concept lattices are interesting, where usually objects are considered as a crisp subsets (as in classical FCA) and attributes are processed as fuzzy sets. In case of one-sided concept lattices, there is a strong connection with clustering (cf. [19]). As it is known, clustering methods produce subsets of a given set of objects, which are closed under intersection, i.e., closure system on the set of objects. Since one-sided concept lattice approach

produces also closure system on the set of objects, one can see one-sided concept lattice approaches as a special case of hierarchical clustering. Several one-sided approaches to FCA were already defined, we mention papers of Krajči [23], Yahia and Jaoua [4], work of Jaoua and Elloumi on Galois lattices of real relations [20], or an approach based on the variable threshold concept lattices by Zhang, Ma and Fan [34]. The approaches mentioned allow only one type of attribute (i.e., truth degrees structure) to be used within the input data table. In our previous paper [5] (see also [7] and [17] for connection with other FCA approaches) we have introduced the necessary theoretical details of the algorithm for creation of model called GOSCL (generalized one-sided concept lattice), which provides a special one-sided case of the fuzzy concept lattice based on the more general approach of Pócs [29] and is able to work with the input data tables with different types of attributes, e.g., binary, quantitative (values from the interval of reals), ordinal (scale-based), nominal, etc. The details regarding its implementation (in Java) and some real data examples were described in [8]. Let us note, that from the mathematical point of view, a framework for the GOSCL theory is provided by partially ordered sets with its possible extensions, cf. [12, 15, 16].

One of the problems of the methods for construction of one-sided concept lattices is computational complexity, which can be (in the worst case) exponential. In crisp cases several works exist, where problem of parallel/distributed creation of FCA models was analyzed, i.e., Krajca, Outrata and Vychodil designed the algorithm for parallel or distributed computing of fixpoints of Galois connections and described their results in several papers (cf. [21, 22, 27]), also Xu, Frein, Robson and Foghlu provided similar distributed algorithm based on the Map-Reduce framework in [33]. In order to analyze our case with generalized one-sided concept lattices, we have studied several aspects of GOSCL complexity. In [9] we have shown that for fixed input data table and attributes the time complexity of GOSCL is asymptotically linear with the increasing number of objects. This is based on the fact that after some time (which is specific for the input context) new concepts are added linearly to the increment of objects. Moreover, in [10] we have analyzed the significant reduction of computation times of the algorithm for sparse input data tables (i.e., input tables with many zero values). However, in order to achieve the objective to produce large-scale concept lattices on the real data, which can be then used in information retrieval tasks or text-mining analysis, it is possible to extend our approach and re-use distributed computing paradigm based on the data distribution [18]. In [6] we have designed distributed version of GOSCL algorithm based on the decomposition of input context to several subtables with separated (and disjoint) subsets of rows, i.e., data table is decomposed to several smaller tables (using recursive bisection-based method), for which small lattices are created and these are then iteratively combined (by a defined merging procedure) to one large concept lattice for the whole data table. The main aim of this paper is to extend the original work and provide experimental results on selected real datasets in order to analyze the algorithm on real sparse data tables. Therefore, we will recall the necessary details from the original paper and extend it according to new analysis, i.e., beyond the

experiments with randomly generated inputs (for specified sparseness of the data) we also provide experiments on real text-mining datasets, both based on newspaper articles (Reuters-21578, Times60) preprocessed into vector-based representation which was used for the preparation of the input data table for the experiments.

In the following section we recall necessary details for the definition of generalization of one-sided concept lattices and standard algorithm for their creation, as well as some notes on complexity of GOSCL which motivate us to design the distributed version of the algorithm. Section 3 is devoted to the introduction of distributed approach for creation of a model, also with a detailed description of the algorithm. In the following section experiments are described, i.e., we provide experiments on randomly generated inputs with different sparseness and also experiments with the real text-mining datasets.

2 GENERALIZED ONE-SIDED CONCEPT LATTICES

In the first two subsections we provide necessary details about the fuzzy generalization of classical concept lattices, so called generalized one-sided concept lattices, and algorithm for their creation, which were theoretically introduced in [5] and practical details on implementation in Java were described in [8]. In the last subsection we provide some notes on the complexity of our algorithm for creation of GOSCL on general and sparse data, that lead (as a motivation for reduction of computation times) to the design of the distributed version in the way presented in Section 3 of this paper.

2.1 Theoretical Introduction to GOSCL

The crucial role in the mathematical theory of fuzzy concept lattices play special pairs of mappings between complete lattices, commonly known as Galois connections. Hence, we provide necessary details regarding Galois connections and related topics. For lattice theory we will use the terminology and the notation as Grätzer in [14].

Let $(P, \leq)$ and $(Q, \leq)$ be complete lattices and let $\varphi\colon P \to Q$ and $\psi\colon Q \to P$ be maps between these lattices. Such a pair (φ, ψ) of mappings is called a *Galois connection* if the following condition is fulfilled:

$$p \leq \psi(q) \quad \text{if and only if} \quad \varphi(p) \geq q.$$

Galois connections between complete lattices are closely related to the notion of closure operator and closure system. Let L be a complete lattice. By a *closure operator* in L we understand a mapping $c\colon L \to L$ satisfying:

1. $x \leq c(x)$ for all $x \in L$,
2. $c(x_1) \leq c(x_2)$ for $x_1 \leq x_2$,
3. $c(c(x)) = c(x)$ for all $x \in L$ (i.e., c is idempotent).

As next we describe mathematical framework for one-sided concept lattices. We start with the definition of generalized formal context.

A 4-tuple $\big(B, A, \mathsf{L}, R\big)$ is said to be a *one-sided formal context* (or *generalized one-sided formal context*) if the following conditions are fulfilled:

1. B is a non-empty set of objects and A is a non-empty set of attributes.
2. $\mathsf{L} : A \to \mathsf{CL}$ is a mapping from the set of attributes to the class of all complete lattices. Hence, for any attribute a, $\mathsf{L}(a)$ denotes the complete lattice, which represents structure of truth values for attribute a.
3. R is generalized incidence relation, i.e. $R(b, a) \in \mathsf{L}(a)$ for all $b \in B$ and $a \in A$. Thus, $R(b, a)$ represents a degree from the structure $\mathsf{L}(a)$ in which the element $b \in B$ has the attribute a.

Then the main aim is to introduce a Galois connection between classical subsets of the set of all objects $\mathbf{P}(B)$ and the direct products of complete lattices $\prod_{a\in A} \mathsf{L}(a)$ which represents a generalization of fuzzy subsets of the attribute universe A. The direct product of lattices is a lattice under componentwise operations consisting of the cartesian product of any non-empty family of lattices. Let us remark that usually fuzzy subsets are considered as functions from the given universe U into real unit interval $[0, 1]$ or more generally as a mappings from U into some complete lattice L. In our case the generalization of fuzzy subsets is straightforward, i.e., to the each element of the universe (in our case the attribute set A) there is assigned the different structure of truth values represented by complete lattice $\mathsf{L}(a)$.

Now we provide basic results about one-sided concept lattices.

Let $\big(B, A, \mathsf{L}, R\big)$ be a generalized one-sided formal context. Then we define a pair of mapping $^{\perp} : \mathbf{P}(B) \to \prod_{a\in A} \mathsf{L}(a)$ and $^{\top} : \prod_{a\in A} \mathsf{L}(a) \to \mathbf{P}(B)$ as follows:

$$X^{\perp}(a) \;=\; \bigwedge_{b\in X} R(b, a), \tag{1}$$

$$g^{\top} \;=\; \{b \in B : \forall a \in A,\ g(a) \le R(b, a)\}. \tag{2}$$

Let $\big(B, A, \mathsf{L}, R\big)$ be a generalized one-sided formal context. Then a pair $(^{\perp}, ^{\top})$ forms a Galois connection between $\mathbf{P}(B)$ and $\prod_{a\in A} \mathsf{L}(a)$.

Now we are able to define generalized one-sided concept lattice. For formal context $\big(B, A, \mathsf{L}, R\big)$ we denote the set $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$ as the set of all pairs (X, g), where $X \subseteq B$, $g \in \prod_{a\in A} \mathsf{L}(a)$, satisfying

$$X^{\perp} = g \quad \text{and} \quad g^{\top} = X.$$

Set X is usually referred as *extent* and g as *intent* of the concept (X, g).

Further, we define a partial order on $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$ as follows:

$$(X_1, g_1) \le (X_2, g_2) \text{ iff } X_1 \subseteq X_2 \text{ iff } g_1 \ge g_2.$$

Let $\big(B, A, \mathsf{L}, R\big)$ be a generalized one-sided formal context. Then $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$ with the partial order defined above forms a complete lattice, where

$$\bigwedge_{i\in I}(X_i, g_i) = \Big(\bigcap_{i\in I} X_i,\ \big((\bigvee_{i\in I} g_i)^{\top}\big)^{\perp}\Big)$$

and

$$\bigvee_{i\in I}(X_i, g_i) = \Big(\big((\bigcup_{i\in I} X_i)^{\perp}\big)^{\top},\ \bigwedge_{i\in I} g_i\Big)$$

for each family $(X_i, g_i)_{i\in I}$ of elements from $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$.

2.2 Algorithm for Creation of GOSCL

Now we briefly describe an incremental algorithm for creating one-sided concept lattices. By the incremental algorithm we mean the algorithm which builds the model from the input data incrementally row by row, where in every step the current model from already processed inputs is a correct concept lattice for a subtable containing only rows already processed (and the addition of the new increment means to provide another row and update the last model, i.e. concept lattice). Let $\big(B, A, \mathsf{L}, R\big)$ be a generalized one-sided formal context, i.e., the input data is in the form of data table with objects from the set B, attributes from the set A and data table values written in table R (with values only from lattices for particular attributes). We will use the following notation. For $b \in B$ we denote by $R(b)$ an element of $\prod_{a\in A}\mathsf{L}(a)$ such that $R(b)(a) = R(b, a)$, i.e., $R(b)$ represents b-th row in data table R. Further, let 1_L denote the greatest element of $L = \prod_{a\in A}\mathsf{L}(a)$, i.e., $1_L(a) = 1_{\mathsf{L}(a)}$ for all $a \in A$.

Algorithm 1 Algorithm GOSCL

Require: generalized context $\big(B, A, \mathsf{L}, R\big)$
Ensure: set of all concepts $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$

$L \leftarrow \prod_{a\in A}\mathsf{L}(a)$ ▷ Direct product of attribute lattices
$I \leftarrow \{1_L\}$ ▷ $I \subseteq L$ will denote set of intents
$\mathcal{C}\big(B, A, \mathsf{L}, R\big) \leftarrow \emptyset$
for all $b \in B$ **do**
  $I^* \leftarrow I$ ▷ I^* represents "old" set of intents
  for all $g \in I^*$ **do**
    $I \leftarrow I \cup \{R(b) \wedge g\}$ ▷ Generation of new intent
  end for
end for
for all $g \in I$ **do**
  $\mathcal{C}\big(B, A, \mathsf{L}, R\big) \leftarrow \mathcal{C}\big(B, A, \mathsf{L}, R\big) \cup \{(g^{\top}, g)\}$
end for
return $\mathcal{C}\big(B, A, \mathsf{L}, R\big)$ ▷ Output of the algorithm

The correctness of the Algorithm 1 can be shown according to the following facts. Evidently, $\mathcal{C}$ is the smallest closure system in L containing $\{R(b) : b \in B\}$. Since $R(b) = (\{b\})^{\perp}$, we obtain $\mathcal{C} \subseteq \left(2^B\right)^{\perp}$. Conversely, if $g = X^{\perp} \in \left(2^B\right)^{\perp}$, then $g = \bigwedge_{b \in X}(\{b\})^{\perp} = \bigwedge_{b \in X} R(b) \in \mathcal{C}$. Hence $\mathcal{C} = \left(2^B\right)^{\perp}$.

Let us remark that algorithm step for creation of the product of the attribute lattices $\prod_{a \in A} \mathsf{L}(a)$ can be done in various ways and it is up to a programmer. For example, it is not necessary to store all elements of $\prod_{a \in A} \mathsf{L}(a)$, but it is sufficient to store only particular lattices $\mathsf{L}(a)$, since lattice operations are calculated componentwise.

2.3 Notes on the Complexity of GOSCL

In this subsection we recall some notes on the complexity of GOSCL algorithm, which are useful for our case.

First, in [9] we have shown that for general input data tables with the fixed number and types of attributes the complexity is the function of the number of objects. It can be easily shown using the following idea. Let (B, A, L, R) be a finite generalized one-sided formal context, i.e., we will assume that the following numbers $n = |B|$, $m = |A|$ and $N = \max\{|\mathsf{L}(a)| : a \in A\}$ are finite. The considered complexity of the GOSCL algorithm will be the function of the input formal context and we provide the upper bounds of this complexity function $h(B, A, \mathsf{L}, R)$ as a function of the number n of objects.

Computing the infimum of any two elements in the particular lattice $\mathsf{L}(a)$ can be done in at most $2 \cdot N$ steps (it corresponds to the searching in the two dimensional array with $N \times N$ entries). Thus we will assume that the operation of infimum in the lattice $\prod_{a \in A} \mathsf{L}(a)$ is processed in at most $2 \cdot N \cdot m$ steps, therefore there exists a constant independent on the number n of objects.

Considering the complexity of GOSCL according to the number of objects n, the most time consuming part of the presented algorithm is `for` cycle (between steps 4 and 9 in Algorithm 1) for creating the closure system I in $\prod_{a \in A} \mathsf{L}(a)$. In every iteration of the cycle there are possibly $|I|$ new objects, thus in the worst case the total number of elements in I will be two-times bigger then before. This yields that the closure system I is created at most in $2^n \cdot (2 \cdot N \cdot m)$ steps and the considered complexity is $h(B, A, \mathcal{L}, R) \in O(2^n)$. On the other side, there are examples of generalized one-sided formal context containing n elements with the resulting concept lattice with 2^n elements, hence there is needed at least 2^n steps for creating the concept lattice. As a consequence we obtain $h(B, A, \mathsf{L}, R) \in \Theta(2^n)$. Note, that $h(n) \in O(2^n)$ means $h(n) \leq k \cdot 2^n$ for some $k \in \mathbb{R}$ as $n \to \infty$ and $h(n) \in \Theta(2^n)$ denotes $k_1 \cdot 2^n \leq h(n) \leq k_2 \cdot 2^n$ for some $k_1, k_2 \in \mathbb{R}$ as $n \to \infty$.

In our fixed case it is reasonable to consider constant number of attributes with the fixed complete lattices representing their truth values structure. Hence, we will assume that each input context contains m_0 constant attributes with $N_0 = \max\{|\mathsf{L}(a)| : a \in A\}$. Let us remark that N_0 denotes the cardinality of the largest

complete lattice assigned to attributes. The numbers m_0 and N_0 do not depend on the number n of objects, which is considered as input variable. In this case we obtain the upper bound for cardinality of $\prod_{a\in A} \mathsf{L}(a)$ as $N_0^{m_0}$. Hence, for cardinality of the created closure system I we obtain

$$|I| \leq \left| \prod_{a\in A} \mathsf{L}(a) \right| \leq N_0^{m_0}.$$

This yields that `for` cycle for creation of I runs in linear time, therefore considered complexity of algorithm for restricted context is in the class $O(n)$. Therefore, for any finite context the complexity function becomes linear after some time, which depends on the number of attributes, their complexity and the variation of values within the objects.

The second important note is that linearization effect is visible earlier (or becomes more valuable) for a sparse data table. In [10] we presented the experimental study on the effect of reduction of computation times for different sparseness of the input data. As we have shown there, whenever sparseness of data is higher, the number of concepts after every step is lower, and therefore also number of comparisons is lower, i.e., the computation time decreases with the higher sparseness of the input data.

Our motivation is to apply GOSCL in domains similar to text-mining and information retrieval, which are usually very sparse (e.g., often less than 0.1% of values is non-zero). In this cases reduction of computation becomes very useful. Also, the usage of division of objects into smaller groups, for which it is even more faster to create their local FCA models, can have some benefits for additional reduction of computation times, if these are then merged together using the procedure which removes unnecessary combinations (removes already contained concepts in merging step). These facts lead us to define a distributed version of the algorithm for creation of GOSCL, which is introduced in the next section.

3 DISTRIBUTED ALGORITHM FOR GENERALIZED ONE-SIDED CONCEPT LATTICES

In this section we provide the theoretical details regarding the computation of GOSCL model in distributed manner. It means that division into partitions is defined. The main theorem is proved, which shows that concept lattices created for the partitions are equivalent to the concept lattice created for the whole input formal context. Then the algorithm for our approach to the distribution is provided. It is based on division of the data table into binary-like tree of starting subsets (with their smaller concept lattices), which are then combined in pairs (using a specified merge procedure) until the final concept lattice is available.

Let $\pi = \{B_i\}_{i\in I}$ be a partition of object set, i.e., $B = \bigcup_{i\in I} B_i$ and $B_i \cap B_j = \emptyset$ for all $i \neq j$. This partition indicates the division of objects in considered object-attribute model, where R_i defines several sub-tables containing the objects from B_i

and which yields several formal contexts $\mathcal{C}_i = (B_i, A, \mathsf{L}, R_i)$ for each $i \in I$. Consequently we obtain the system of particular Galois connections $({}^{\perp_i}, {}^{\top_i})$, between $\mathbf{P}(B_i)$ and $\prod_{a\in A} \mathsf{L}(a)$. Next we describe how to create Galois connection between $\mathbf{P}(B)$ and $\prod_{a\in A} \mathsf{L}(a)$. We use the method of creating Galois connections applicable for general fuzzy contexts, described in [29]. Let $X \subseteq B$ be any subset and $g \in \prod_{a\in A} \mathsf{L}(a)$ be any tuple of fuzzy attributes. We define

$$\uparrow\big(X\big)(a) = \bigwedge_{i\in I} (X \cap B_i)^{\perp_i} \quad \text{and} \quad \downarrow(g) = \bigcup_{i\in I} g^{\top_i} \tag{3}$$

Theorem 1. Let $\pi = \{B_i\}_{i\in I}$ be a partition of object set. Then the pair of mappings $(\uparrow, \downarrow)$ defined by (3) and the pair $({}^{\perp}, {}^{\top})$ defined by (1) represent the same Galois connection between $\mathbf{P}(B)$ and $\prod_{a\in A} \mathsf{L}(a)$.

Proof. Let $X \subseteq B$ be any subset of object set and $a \in A$ be an arbitrary attribute. Using (3) we obtain

$$\uparrow\big(X\big)(a) = \bigwedge_{i\in I} (X \cap B_i)^{\perp_i} = \bigwedge_{i\in I} \left(\bigwedge_{b\in X\cap B_i} R_i(b,a) \right) = \bigwedge_{i\in I} \left(\bigwedge_{b\in X\cap B_i} R(b,a) \right).$$

Since π is the partition of the object set B, we have $X = \bigcup_{i\in I}(X \cap B_i)$. Involving the fact, that meet operation in lattices is associative and commutative, we obtain

$$X^{\perp}(a) = \bigwedge_{b\in X} R(b,a) = \bigwedge_{b\in \bigcup_{i\in I}(X\cap B_i)} R(b,a) = \bigwedge_{i\in I} \left(\bigwedge_{b\in X\cap B_i} R(b,a) \right),$$

which yields $\uparrow(X) = X^{\perp}$ for each $X \subseteq B$.

Similarly, we have

$$\downarrow(g) = \bigcup_{i\in I} g^{\top_i} = \bigcup_{i\in I} \{b \in B_i : \forall a \in A,\ g(a) \le R_i(b,a)\}.$$

Easy computation shows that this expression is equal to

$$\{b \in B : \forall a \in A,\ g(a) \le R(b,a)\} = g^{\top},$$

which gives $\downarrow(g) = g^{\top}$ for all elements $g \in \prod_{a\in A} \mathsf{L}(a)$. □

The presented theorem is the base for our distributed algorithm. The main aim is to effectively create the closure system in $\prod_{a\in A} \mathsf{L}(a)$ which represents the set of all intents. Of course, there are more options how to process the distribution itself, i.e., how to merge sub-models during the process. We have designed the version of distribution which divides the starting set of the objects into 2^N parts (from this section we use N as number of merging levels), for which local models are created

Algorithm 2 Distributed algorithm for GOSCL

Require: generalized context (B, A, L, R), $\pi = \{B_i\}_{i=1}^{2^N}$ – partition of B with 2^N parts
Ensure: merged concept lattice $\mathcal{C}(B, A, \mathsf{L}, R)$
1: **for** $i = 1$ to 2^N **do**
2: $\quad \mathcal{C}_i^{(N)} \leftarrow \mathrm{GOSCL}(B_i, A, \mathsf{L}, R_i)$ ▷ Application of Algorithm 1 on all subcontexts
3: **end for**
4: **for** $k = N$ down to 1 **do**
5: $\quad$ **for** $i = 1$ to 2^{k-1} **do**
6: $\quad\quad \mathcal{C}_i^{(k-1)} \leftarrow \mathrm{MERGE}(\mathcal{C}_{2i-1}^{(k)}, \mathcal{C}_{2i}^{(k)})$ ▷ Merging of adjacent concept lattices
7: $\quad$ **end for**
8: **end for**
9: **return** $\mathcal{C}\big(B, A, \mathsf{L}, R\big) \leftarrow \mathcal{C}_1^{(0)}$ ▷ Output of the algorithm

(in parallel) and then pairs of them are combined (in parallel) into 2^{N-1} concept lattices, and then this process continues, until we have only one concept lattice at the end. This output should be isomorphic to concept lattice created by the sequential run, because we have shown that previous theorem is valid and we only make partitions accordingly to the assumptions and proof of this theorem. It is easy to see that pair-based merging leads to the binary tree of sub-lattices, with N representing the number of merging levels. Now, we can provide the algorithm description. The Algorithm 2 is the distributed version (with the idea described here) and merge procedure is provided separately in Algorithm 3.

4 EXPERIMENTS

The experiments are divided into two parts. First, we provide the experiments on the randomly generated data with different sparseness. Next, the experiments with the real text-mining datasets (Reuters-21578 and Times60, newspaper articles) are provided, together with the necessary details on their preprocessing into vector representation used for the preparation of the input data table for GOSCL. The discussion on the results is provided in the last subsection.

4.1 Experiments with the Randomly Generated Input Data

For our first set of experiments with the distributed version of the algorithm we have produced randomly generated input formal contexts with the specified sparseness index of the whole input data table. The reason is simply in the natural characteristic of FCA-based algorithms in general, i.e., whenever the number of concepts (intents) still grows fast with the number of inputs, this data table has still too much different combinations of values of attributes in it (it is a problem especially important in case of fuzzy attributes, which we have here) and the merge procedure is not

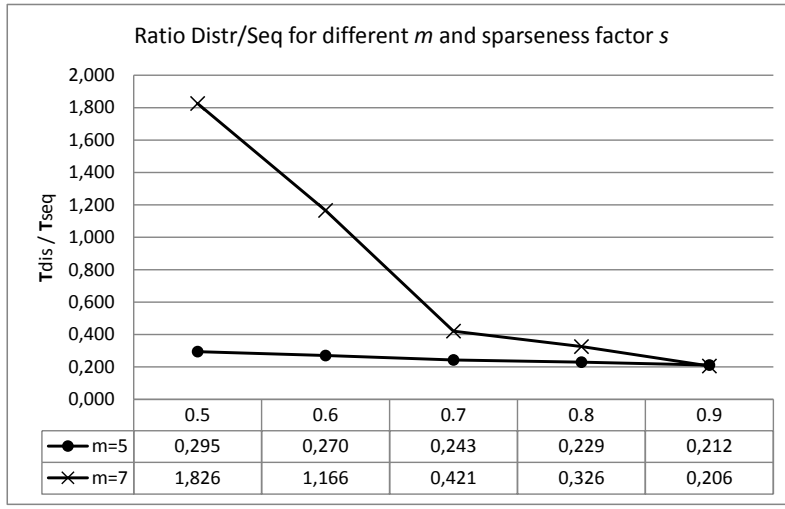
Algorithm 3 Procedure MERGE for merging two concept lattices

Require: two concept lattices $\mathcal{C}_1$ and $\mathcal{C}_2$
Ensure: MERGE$(\mathcal{C}_1, \mathcal{C}_2)$

$I_1, I_2 \leftarrow \emptyset$ ▷ I_1, I_2 will denote list of intents for $\mathcal{C}_1, \mathcal{C}_2$
for all $(X, g) \in \mathcal{C}_1$ **do**
  $I_1 \leftarrow I_1 \cup \{g\}$ ▷ Extraction of all intents from $\mathcal{C}_1$
end for
for all $(X, g) \in \mathcal{C}_2$ **do**
  if $g \notin I_1$ **then** ▷ Check for duplications of intents in I_1
    $I_2 \leftarrow I_2 \cup \{g\}$ ▷ Extraction of all intents from $\mathcal{C}_2$
  end if
end for
$I \leftarrow \emptyset$ ▷ Set of all newly created intents
for all $g \in I_2$ **do**
  for all $f \in I_1$ **do**
    $I \leftarrow I \cup \{f \wedge g\}$ ▷ Creation of new intent
  end for
end for
MERGE$(\mathcal{C}_1, \mathcal{C}_2) \leftarrow \mathcal{C}_1$
for all $g \in I$ **do**
  MERGE$(\mathcal{C}_1, \mathcal{C}_2) \leftarrow$ MERGE$(\mathcal{C}_1, \mathcal{C}_2) \cup \{(g^\top, g)\}$ ▷ Addition of new concepts
end for
return MERGE$(\mathcal{C}_1, \mathcal{C}_2)$ ▷ Output of the merging procedure

effective in order to get better times in comparison with the incremental algorithm. But in the cases with the set of intents for which number of concepts (intents) is strongly reduced, the provided approach to distribution will lead to the reduction of computation times (merge procedures will not need so much time to create their new lattices on new levels). Fortunately, as we are very interested in the application of GOSCL in text-mining domain, and sparse inputs generate much reduced concept lattices (more zeros will produce more similar combinations of values and less number of new intents), our distributed algorithm seems to be efficient mainly for very sparse input matrices.

In order to simplify the data preparation (generation) process, we have used one type of attribute in the table, i.e., scale-based (or ordinal attribute) with 5 possible values $\{0, 1, 2, 3, 4\}$, where 0 is the lowest 'zero' value (or bottom value of the attribute lattice) and 4 is the highest value (or top element of the attribute lattice). The values for the particular object are generated from this defined scale. For our experiments it is possible to use only one type of the attribute, because (without the loss of generality) according to the measured aspects (the influence of sparseness) it is not needed to analyze data tables with different types of attributes. Also, the experiments with the selected text-mining datasets (which are presented in Subsection 4.2) were done on the vector-based model with the attributes of the same type



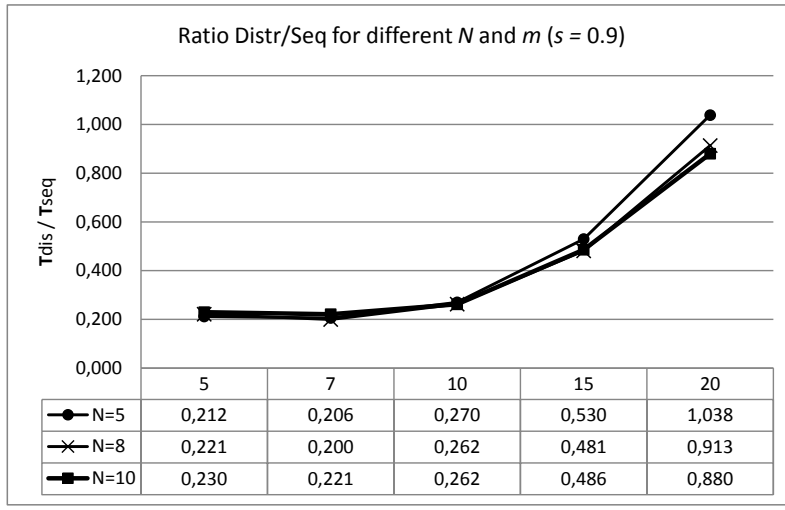


Figure 1. Experiments with the distributed version of GOSCL – up: analysis for different sparseness; bottom: analysis for different number of attributes and merging levels

(i.e., every term or word has the same type of characterization). The generation of the sparse data table is based on the sparseness factor $s \in [0, 1]$), which indicates the ratio of "zeros" (as a real number between 0 and 1) in data table, i.e., s indicates the level of sparseness of generated data table. For higher s, the number of zeros in input is also higher. The simple mechanism for the generation was used with the random number generator for decision on adding the zero (bottom elements of the attribute lattice) or some non-zero value according to the selected sparseness. The

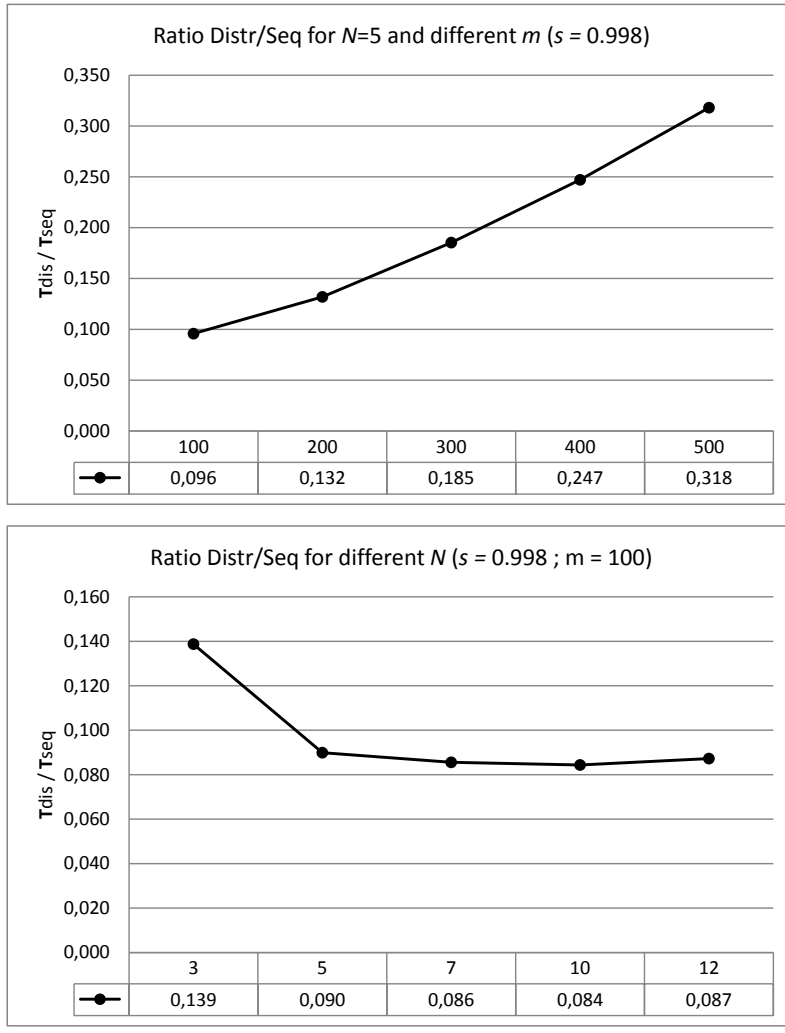


Figure 2. Experiments with the distributed version of GOSCL on very sparse inputs – up: analysis of reduction ratio with changing number of attributes; bottom: analysis of reduction ratio for different number of merging levels

generated data are then very close to the selected sparseness (especially for larger inputs). We can add that in text-mining domains the sparseness factor is very high (usually more than 0.998). The number of generated objects will be (for all our experiments with the generated data) 8 192, just to simplify its division to partitions which will be then easily merged in particular levels (2^{13} is 8 192).

Our first experiment with the generated datasets shows that the efficiency is different for different sparseness. Therefore, we have tried random contexts for 5 and 7 attributes and different sparseness (from 0.5 to 0.9). The results are shown in the upper part of the Figure 1. As we can see, with the higher sparseness of the data, the ratio of computation times between distributed version and sequential version (reduction ratio T_{dis}/T_{seq}) is lower (i.e., the reduction is better). Another experiment is based on the analysis of fixed sparseness (0.9), where the number of attributes is changing from 5 to 20 and also N (the number of levels in merging) has three different settings (5, 8 and 10). The result is shown in the bottom part of the Figure 1, where we can see that the number of merging levels (and therefore also number of starting partitions 2^n) has not a big influence on the reduction ratio. On the other hand, with the increasing number of attributes reduction the ratio should increase due to a higher number of intents produced by the combination of values for more attributes.

One of our interests is to analyze data from the text-mining domains, which are usually represented by the very sparse input tables. Therefore, we also add some experiments with very sparse inputs. First, we have analyzed the reduction ratio for fixed sparseness 0.998 and number of merging levels 5, where number of attributes is changing (from 100 to 500). As we can see in the upper part of the Figure 2, the reduction ratio increased with the number of attributes, but is still quite significant even for 500 attributes.

In the last experiment, we have used the same sparseness of the data (0.998) with 100 attributes and analyzed the reduction ratio for different merging levels (N). The result is shown in the bottom part of the Figure 2, where we can see now more evidently that the higher number of levels can help in better times, but it seems that it is not needed to make very small partitions (e.g., for $N = 12$, which leads to starting with two-object sets, the ratio is a little higher than for the previous values). Probably, the best number of merging levels can be estimated based on the real datasets and then used for better results.

4.2 Experiments with the Selected Text-Mining Datasets

Now we are able to describe the experiments with the real text-mining datasets. For this case we have selected two collections of documents, both containing newspaper articles written in English language.

The first one is Reuters collection known as Reuters-21578 (ModApte split), which contains Reuters articles published in 1987. The collection is available publicly in SGML format and for our experimental purposes it was transformed into the XML. ModApte version is split into training and testing subsets. Both subsets contain documents in 90 different categories. From this collection we have used only documents from the training part containing 7 769 articles.

The second one is Times60 collection, which contains articles from the Times newspaper from 1960's. It is relatively small dataset with 420 documents on different topics like international relationships, economics, political situation and history of

different countries and regions, Vietnam war, etc. We have used this dataset due to the fact that it is not too large and it is well-known to us from our previous work in the text-mining area.

4.2.1 Preparation of the Input Data Tables from Textual Datasets

For the preparation of the data from selected datasets which are comparable to the experiment settings we used preprocessing steps like tokenizing, stemming, frequency-based term filtering and stop-words filtering. Then basic TF-IDF scheme (see for example [32] for more details on this topic) was used for weighting of terms in documents and vectors of documents were normalized. The Java-based library (designed and implemented also by one of the authors) for the support of representation and processing of textual documents called JBOWL (Java Bag-Of-Words Library [1]) was used for this process. In order to simplify the experiments setup, we have also did pruning of very small non-zero values and discretization of the weights from the preprocessing tool using the histogram of their distribution in the vector-based model. Then our implementation of GOSCL was able to reuse the context defined in its required format (similar for comparison with the previous experiments).

For the Reuters dataset we had input context with 3 031 attributes, where preprocessing step produces two different input contexts after discretization and pruning (according to the analysis of the histogram of values):

1. with the sparseness factor 0.999 and 5 different values for attribute,
2. with the sparseness factor 0.9995 and 4 different values for attribute.

Similarly, for the Times dataset we had input context with 1 924 attributes, where preprocessing produces two different input data tables (after similar process of discretization and pruning):

1. with the sparseness factor 0.999 and 4 different values for attribute,
2. with the sparseness factor 0.999 and also 4 different values of attributes.

4.2.2 Results of the Experiments

In the first experiments on the selected datasets we have analyzed the reduction ratio for the Reuters dataset. Due to higher number of attributes we did only tests for maximum of 1 000 input documents. For our first experiments we have shown the reduction ratio according to the number of objects (n from 100 to 1 000, with the step between particular experiments 100 objects) and different sparseness (depending on the preprocessing steps, s with values 0.999 and 0.9995). Experiments were repeated ten times and final data describes the average values (this was used also for other experiments). The result is shown in the upper part of the Figure 3. As we can see, the reduction ratio grows with the number of objects, but it has linear tendency with

the higher number of objects. Also it seems that even small difference in sparseness can be important factor for the better reduction ratio.

In the second experiments we tested the influence of merging levels on the reduction ratio for Reuters documents, i.e., we used again two sparseness (preprocessing) settings of data (0.999 and 0.9995) with fixed number of objects (500) for different merging levels (N from 3 to 7 for five different settings). The result is shown in the bottom part of the Figure 3, where we can see that similarly to the experiments on random data the higher number of levels can help for better times.

Another experiments were related to the Times dataset, i.e., we have analyzed the reduction ratio for a smaller collection of documents containing documents from the Times newspapers. Similarly to Reuters, we have shown the reduction ratio according to the number of objects (n from 50 to 400, step 50) and different sparseness (depending on preprocessing steps, s with values 0.998 and 0.999). The result is shown on the upper part of the Figure 4. As we can see, the results are not absolutely similar to the Reuters dataset (there are more changes of tendencies in the graph of reduction ratio), but it can be specific to the dataset, as well as a quite lower number of objects within the experiment with this dataset could be an important factor.

In the last experiment, we present the reduction ratio for the Times documents according to different merging levels, i.e., we again used two sparseness (preprocessing) settings of the data (0.998 and 0.999) with fixed number of objects (300) for different merging levels (N for different settings from 2 to 6). The result is shown in the bottom part of the Figure 4. The difference for different merging levels is relatively small, but it has similar characteristic as for the other experiments on the random and Reuters datasets.

4.3 Short Discussion on the Results and Future Work Ideas

At the end of this section we should say that provided distributed approach seems to be applicable in very sparse domains and can lead to computation time reduction, but there are also several limitations. Whenever the sparseness is not very high, this type of distributed algorithm will not be able to cut down computation time significantly. The presented experiments with the selected datasets (especially the larger Reuters dataset) proved the same behavior of the algorithm according to the randomly generated data, but sparseness of the input data table from the real datasets shows its importance and the reduction ratio is quite sensitive to this parameter. The number of merging levels is not a very sensitive parameter and it is only good to stay with middle values, e.g., number of levels which lead to less then 50 objects in starting subsets.

On the other side, the presented work can be viewed as a proof of concept and used as a motivation to implement more effective algorithms for the distribution of the GOSCL computation, which can be used in the future for the creation of large-scale concept lattices from textual datasets or other sparse-based domains [28]. According to the presented algorithm and experiments, we can summarize several ideas for the future work:

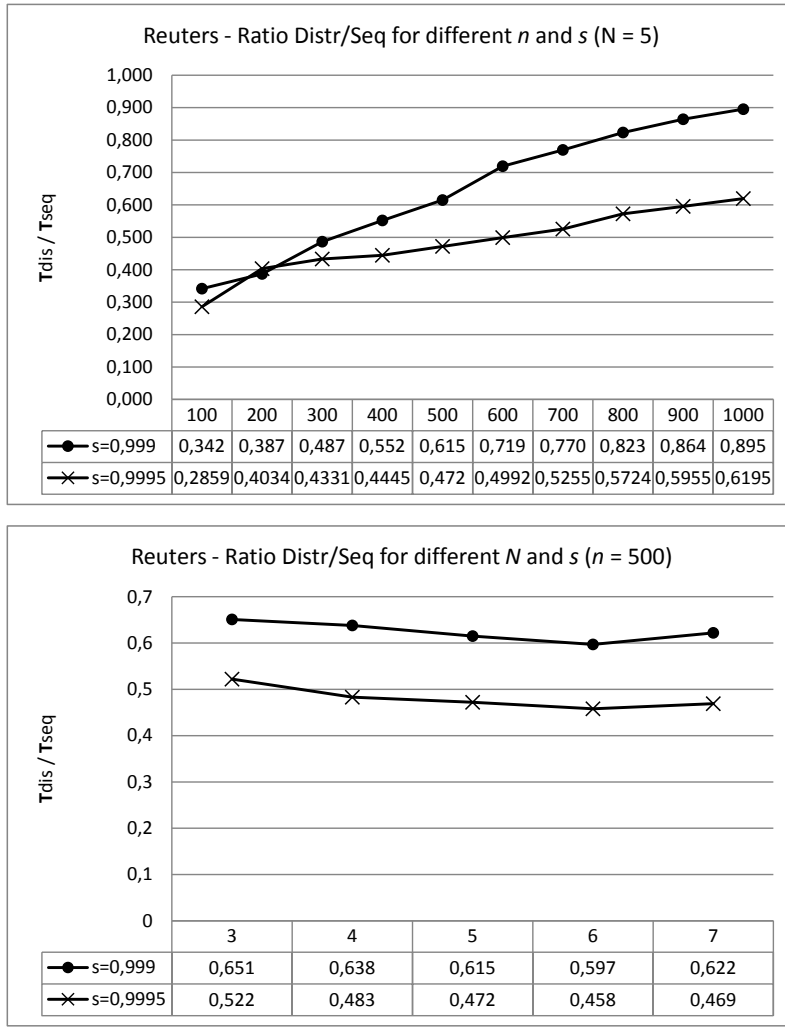
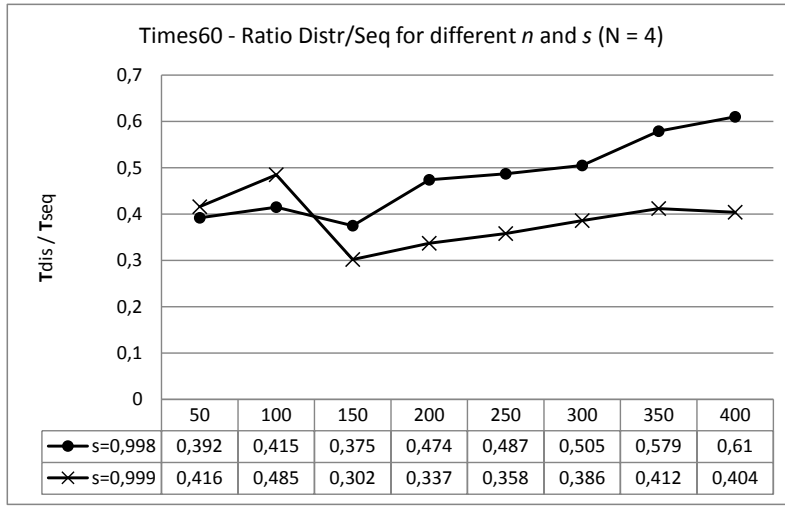


Figure 3. Experiments with the distributed version of GOSCL on Reuters dataset – up: analysis of reduction ratio for different number of objects and sparseness; bottom: analysis of reduction ratio for different number of merging levels and sparseness

- The distribution can be done in various ways and more effective scheme can be used in the future experiments (e.g., unbalanced subsets for more effective merging and reduction of processors required, heuristic-based selection of objects in starting subsets according to the potentially better reduction, etc.).
- Several less important aspects were identified during experiments with the current implementation of the distributed algorithm, like a possibility to run some sub-parts of the steps independently on 'free' processors (which were used in



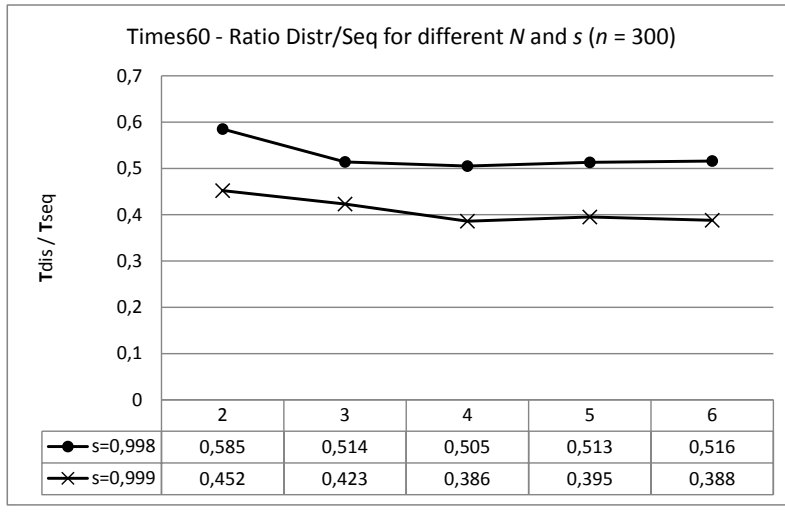


Figure 4. Experiments with the distributed version of GOSCL on Times dataset – up: analysis of reduction ratio for different number of objects and sparseness; bottom: analysis of reduction ratio for different number of merging levels and sparseness

previous level, but are unused in the new one), pre-computation of some values, advanced control of already used combinations of intents, etc.

- The implementation of Map-Reduce architecture (or similar paradigms) could be important in the future, together with inclusion of the knowledge from previous implementation of fast algorithms for crisp FCA.
- As we have already shown in [11], it is possible to introduce a specialized sparse-based implementation of the algorithm for GOSCL, which leads to a further

significant reduction of the computation time. Therefore, it will be interesting in the future to implement also distributed/parallel version of GOSCL algorithm based on such specialized sparse-based methods.

In the future we also want to analyze the usage of presented algorithm (and its possible extensions) in other data analysis tasks in some hybrid architectures (implementations), where FCA is used as a part of the analysis combined with other machine learning methods, especially some clustering methods, or production of well-known association rules or classification schemes based on the concept hierarchies acquired using GOSCL.

5 CONCLUSIONS

In the presented work we have introduced distributed algorithm for the creation the FCA-based model known as Generalized One-Sided Concept Lattice and its usage for sparse input data tables. As we proved in the provided theorem, it is possible to produce the merged concept lattice from the smaller ones, which were individually created for disjoint subsets of objects. For merging the particular concept lattices we have introduced a simple merging procedure, based on a partition similar to binary tree (lists are smallest concept lattices from the start of the distribution and the root is merged final lattice). In the part related to our testing we provided the experiments on the two types of input data. First, we shown the experiments on the randomly generated data in order to prove the basic potential of the approach for its usage on sparse input data tables. Then we provided the experiments on the selected text-mining datasets (Reuters, Times60). The results proved a similar behavior of the algorithm also on the real data, but we were also able to find some limitations and ideas for possible research extensions (impulses for the future work), which are presented at the end of the paper. In the future we also want to apply this algorithm (and its extensions) in other similar domains and data analysis tasks with the sparse input data.

Acknowledgments

The first author was supported by the Slovak VEGA Grant No. 1/1147/12 and by the KEGA Grant No. 025TUKE-4/2015. The second author was supported by the Slovak VEGA Grant 2/0028/13 and by the ESF Fund CZ.1.07/2.3.00/30.0041. The third author was supported by the Slovak Research and Development Agency under contract APVV-0482-11; by the Slovak VEGA Grant 1/0529/15 and by the KEGA grant No. 040TUKE-4/2014.

REFERENCES

[1] Bednár, P.—Butka, P.—Paralič, J.: Java Library for Support of Text Mining and Retrieval. ZNALOSTI 2005, Proceedings of the 4th Annual Conference, Stará Lesná, Slovakia, 2005, pp. 162–169.

[2] Bělohlávek, R.: Lattices of Fixed Points of Fuzzy Galois Connections. Mathematical Logic Quarterly, Vol. 47, 2001, No. 1, pp. 111–116.

[3] Bělohlávek, R.—Vychodil, V.: What Is a Fuzzy Concept Lattice? Concept Lattices and Their Applications (CLA 2005), Olomouc, Czech Republic, 2005, pp. 34–45.

[4] Ben Yahia, S.—Jaoua, A.: Discovering Knowledge from Fuzzy Concept Lattice. In: A. Kandel, M. Last, and H. Bunke (Eds.): Data Mining and Computational Intelligence, Physica-Verlag, 2001, pp. 167–190.

[5] Butka, P.—Pócs, J.: Generalization of One-Sided Concept Lattices. Computing and Informatics, Vol. 32, 2013, No. 2, pp. 355–370.

[6] Butka, P.—Pócs, J.—Pócsová, J: Distributed Version of Algorithm for Generalized One-Sided Concept Lattices. Studies in Computational Intelligence, Vol. 511, 2014, pp. 119–129.

[7] Butka, P.—Pócs, J.—Pócsová, J.: On Equivalence of Conceptual Scaling and Generalized One-Sided Concept Lattices. Information Sciences, Vol. 259, 2014, pp. 57–70.

[8] Butka, P.—Pócsová, J.—Pócs, J.: Design and Implementation of Incremental Algorithm for Creation of Generalized One-Sided Concept Lattices. Proceedings of 12th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2012), Budapest, Hungary, 2011, pp. 373–378.

[9] Butka, P.—Pócsová, J.—Pócs, J.: On Some Complexity Aspects of Generalized One-Sided Concept Lattices Algorithm. Proceedings of 10th IEEE Jubilee International Symposium on Applied Machine Intelligence and Informatics (SAMI 2012), Herľany, Slovakia, 2012, pp. 231–236.

[10] Butka, P.—Pócsová, J.—Pócs, J.: Experimental Study on Time Complexity of GOSCL Algorithm for Sparse Data Tables. Proceedings of 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2012), Timisoara, Romania, 2012, pp. 101–106.

[11] Butka, P.—Pócsová, J.—Pócs, J.: Comparison of Standard and Sparse-Based Implementation of GOSCL Algorithm. Proceedings of 13th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2012), Budapest, Hungary, 2012, pp. 67–71.

[12] Chajda, I.—Halaš, R.—Kühr, J.: Every Effect Algebra Can Be Made Into a Total Algebra. Algebra Universalis, Vol. 61, 2009, No. 2, pp. 139–150.

[13] Ganter, B.—Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin 1999.

[14] Grätzer, G.: Lattice Theory: Foundation. Springer, Basel 2011.

[15] Halaš, R.: Annihilators and Ideals in Ordered Sets. Czechoslovak Mathematical Journal, Vol. 45, 1995, No. 1, pp. 127–134.

[16] Halaš, R.—Lihová, J.: On Weakly Cut-Stable Maps. Information Sciences, Vol. 180, 2010, No. 6, pp. 971–983.

[17] Halaš, R.—Pócs, J.: Generalized One-Sided Concept Lattices with Attribute Preferences. Information Sciences, Vol. 303, 2015, pp. 50–60.

[18] Janciak, I.—Sarnovsky, M.—Min Tjoa, A.—Brezany, P.: Distributed Classification of Textual Documents on the Grid. Lecture Notes in Computer Science, Vol. 4208, 2006, pp. 710–718.

[19] Janowitz, M. F.: Ordinal and Relational Clustering. World Scientific Publishing Company, Hackensack, NJ, 2010.

[20] Jaoua, A.—Elloumi, S.: Galois Connection, Formal Concepts and Galois Lattice in Real Relations: Application in a Real Classifier. The Journal of Systems and Software, Vol. 60, 2002, pp. 149–163.

[21] Krajca, P.—Outrata, J.—Vychodil, V.: Parallel Algorithm for Computing Fixpoints of Galois Connections. Annals of Mathematics and Artificial Intelligence, Vol. 59, 2010, No. 2, pp. 257–272.

[22] Krajca, P.—Vychodil, V.: Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework. Proceedings of the 8th International Symposium on Intelligent Data Analysis (IDA 2009), Lyon, France, 2009, pp. 333–344.

[23] Krajči, S.: Cluster Based Efficient Generation of Fuzzy Concepts. Neural Network World, Vol. 13, 2003, No. 5, pp. 521–530.

[24] Krajči, S.: A Generalized Concept Lattice. Logic Journal of IGPL, Vol. 13, 2005, No. 5, pp. 543–550.

[25] Medina, J.—Ojeda-Aciego, M.—Ruiz-Calviño J.: Formal Concept Analysis via Multi-Adjoint Concept Lattices. Fuzzy Sets and Systems, Vol. 160, 2009, pp. 130–144.

[26] Medina, J.—Ojeda-Aciego, M.: On Multi-Adjoint Concept Lattices Based on Heterogeneous Conjunctors. Fuzzy Sets and Systems, Vol. 208, 2012, pp. 95–110.

[27] Outrata, J.—Vychodil, V.: Fast Algorithm for Computing Fixpoints of Galois Connections Induced by Object-Attribute Relational Data. Information Sciences, Vol. 185, 2012, No. 1, pp. 114–127.

[28] Paralič, J.—Richter, C.—Babič, F.—Wagner, J.—Raček, M.: Mirroring of Knowledge Practices based on User-Defined Patterns. Journal of Universal Computer Science, Vol. 17, 2011, No. 10, pp. 1474–1491.

[29] Pócs, J.: Note on Generating Fuzzy Concept Lattices via Galois Connections. Information Sciences, Vol. 185, 2012, No. 1, pp. 128–136.

[30] Popescu, A.: A General Approach to Fuzzy Concepts. Mathematical Logic Quarterly, Vol. 50, 2004, No. 3, pp. 265–280.

[31] Roman, S.: Lattices and Ordered Sets. Springer, New York, NY, 2009.

[32] Salton, G.—McGill, M. J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York, NY, 1986.

[33] Xu, B.—de Frein, R.—Robson, E.—Foghlu, M.: Distributed Formal Concept Analysis Algorithms Based on an Iterative MapReduce Framework. Lecture Notes in Computer Science, Vol. 7278, 2012, pp. 292–308.

[34] Zhang, W. X.—Ma, J. M.—Fan, S. Q.: Variable Threshold Concept Lattices. Information Sciences, Vol. 177, 2007, No. 22, pp. 4883–4892.

Peter Butka received his Ph.D. degree from the Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University in Košice in 2010. Since 2006 he is working as a researcher and Assistant Professor at the Faculty of Electrical Engineering and Informatics or at the Faculty of Economics. His research interests include text/data mining, formal concept analysis, knowledge management, semantic technologies, and information retrieval.

Jozef Pócs received his Ph.D. degree from the Mathematical Institute of the Slovak Academy of Sciences in 2008. Since 2007 he has been working as research fellow at the Mathematical Institute of the Slovak Academy of Sciences in Košice. Currently he also works as post-doctoral research fellow at Palacký University, Olomouc. His research interests include abstract algebra and application of algebraic methods to information sciences.

Jana Pócsová received her Ph.D. degree from Pavol Jozef Šafárik University in Košice in 2009. Since 2009 she has been working as Assistant Professor at BERG Faculty of Technical University in Košice. Her research interests include data mining, formal concept analysis, and teaching mathematics.