An Incremental Algorithm for Computing n-concepts

Tatiana Makhalova[1] and Lhouari Nourine[2]

[1]National Research University Higher School of Economics, Kochnovsky pr. 3, Moscow 125319, Russia
[2]LIMOS, Université Blaise Pascal, BP 125, 62173 Aubière Cedex, France

tpmakhalova@hse.ru,nourine@isima.fr

Abstract. In this paper a new incremental algorithm for computing n-concepts is proposed. The time complexity of the algorithm is $O(|I|^2 \cdot In \cdot |\mathcal{B}_n|)$, where $|I|$ is the size of a context (an n-ary relation), $In = |K_1||\mathcal{B}_{n-1}|$ is the input, where K_1 is a set corresponding to an added dimension and $\mathcal{B}_{n-1}$ is the set of $(n-1)$-concepts. The output $\mathcal{B}_n$ of the algorithm is a set of n-concepts. The algorithm creates n-concepts (i.e. elements in $\mathcal{B}_n$) by merging sequentially $(n-1)$-concepts from $\mathcal{B}_{n-1}$ with the corresponding elements from the n-th dimension (i.e. set K_1).

1 Introduction

Mining of closed itemsets are widely used in many practical applications of data mining. Introduced by Rudolf Wille in 1982, formal concepts (i.e. dyadic closed itemsets) was expanded later to the triadic [9] and n-dimensional cases (n-concepts) [12]. Representation of data in the form of n-ary relations has been becoming more and more popular recently. Analysis of high-dimensional data can be more fruitful due to retaining some important information in complex structures. Despite increasing interest in polyadic concept analysis, the problem of the construction of high-dimensional concepts remains almost unexplored.

The large number of algorithms has been proposed for computing formal concepts. For a comparative study of algorithms see [8]. In recent times, some algorithms for computing triadic [4,5,11] and n-concepts [1] have been proposed as well as computing approximate triconcepts, i.e. triclusters [3,2]. Recall that in the worst-case the number of concepts can be exponential in the context size and that even counting them is a #P-complete problem [6,7], so we have at least the same complexity problems for triconcepts. However these algorithms are still not compared in a comprehensive manner.

In this paper, we propose an algorithm for computing n-concepts sequentially from $(n-1)$-concepts.

The paper is organized as follows. In Section 2 we describe the proposed approach and present the algorithm. In Section 3 we prove its properties. The time complexity is discussed in Section 4. Section 5 briefly concludes.

2 Strategy of Computing n-concepts

In this section we provide the main definitions and the idea of the algorithm.

Let us consider an n-ary context $\mathbb{K} = (K_1, K_2, ..., K_n, I^n)$. $\mathbb{K}_{K_1=a} \subseteq \mathbb{K}$ denotes a subcontext with fixed value $a \in K_1$, i.e. a context with the following relation $I^n_{K_1=a} = \{\{a, k_2, ..., k_n\} \mid k_2 \in K_2, ..., k_n \in K_n, (a, k_2, ..., k_n) \in I^n\}$. A subcontext where elements from the several dimensions are fixed is denoted in the same way. For example, a context where only elements $a_j \in K_j$ and $a_i \in K_i$ are fixed is denoted by $I^n_{K_i=a_i,K_j=a_j} = \{\{k_1, ..., k_{i-1}, a_i, k_{i+1}, ..., k_{j-1}, a_j, k_{j+1}, ..., k_n\} \mid$ $k_1 \in K_1, k_{i-1} \in K_{i-1}, k_{i+1} \in K_{i+1}, ..., k_{j-1} \in K_{j-1}, k_{j+1} \in K_{j+1}, k_n \in K_n,$ $(k_1, ..., k_{i-1}, a_i, k_{i+1}, ..., k_{j-1}, a_j, k_{j+1}, ..., k_n) \in I^n\}$.

A set of n-concepts corresponding to the context $\mathbb{K}$ is denoted by $\mathcal{B}_n$. $\mathcal{B}^{x_i}_{n-1}$ denotes a set of $(n-1)$-concepts of $\mathbb{K}_{K_1=x_i}$. $\mathbb{K}_{K_1=x_i}$ can be represented as an n-ary context where the first dimension consists of a single element, thus $\mathcal{B}^{\{a\}}_n = \mathcal{B}(\mathbb{K}_{K_1=\{a\}}) = \{(\{a\}, X_2, ..., X_n) \mid (X_2, ..., X_n) \in \mathcal{B}^a_{n-1}\}$. A set of n-concepts that corresponds to a subcontext $\mathbb{K}_{K_1=\{x_1,x_2,...,x_i\}}$, is denoted by $\mathcal{B}^{\{x_1,x_2,...,x_i\}}_n$.

Example Let us consider a context given on Fig.1 (a). The context is comprised of the following sets: $K_1 = \{1, 2, 3, 4\}$, $K_2 = \{a, b, c, d\}$, $K_3 = \{\alpha, \beta, \gamma, \delta\}$.

Subcontext $\mathbb{K}_{K_1=2}$ corresponds to the dyadic relation given in Fig. 1 (b).The set of dyadic concepts is $\mathcal{B}^2_2 = \{(\{a\}, \{\alpha, \beta\}), (\{a, b, c\}, \{\beta\}), (\{b, c\}, \{\beta, \gamma, \delta\})\}$.

The set of triadic concepts for subcontext $\mathbb{K}_{K_1=\{1,2\}}$ given on Fig. 1 (c) is following:

$$\mathcal{B}^{\{1,2\}}_2 = \{(\{1, 2\}, \{a\}, \{\alpha, \beta\}), (\{1\}, \{d\}, \{\beta, \gamma\}), (\{1\}, \{a, d\}, \{\beta\}), \\ (\{2\}, \{a, b, c\}, \{\beta\}), (\{2\}, \{\{b, c\}, \{\beta, \gamma, \delta\}))\}.$$

A subcontext with fixed elements from two dimensions in the triadic case takes the following form: $\mathbb{K}_{K_1=2,K_2=b} = \{\alpha, \beta, \gamma\}$.

During the recursive descent in step 8 of Algorithm 2 a particular value $a \in K_i$ form the next dimension $i \in [1, 2, ..., n-2]$ is fixed. At the end of the descent, for a subcontext $\mathbb{K}_{K_1=a_1,...,K_{n-2}=a_{n-2}}$, any algorithm for computing formal concepts can be applied.

During the recursive ascent (see step 9 of Algorithm 2) $(n-1)$-concepts from $\mathcal{B}^a_{n-1}$, $a \in K_1$ are merged. In the recursive ascent on the i-th level the algorithm builds $(n-i+1)$-concepts (the set of the concepts is denoted by $\mathcal{B}_n$ in Algorithm 2) using computed during the recursive descent $(n-i)$-concepts corresponding to a particular value $a \in K_1$. With the introduced notation, the states of $\mathcal{B}_n$ in Algorithm 2 is changed as follows: $\mathcal{B}^{\{\emptyset\}}_n$, $\mathcal{B}^{\{x_1\}}_n$ (derived from $\mathcal{B}^{x_1}_{n-1}$), $\mathcal{B}^{\{x_1,x_2\}}_n$ (derived by merging $\mathcal{B}^{x_1}_{n-1}$ and $\mathcal{B}^{x_2}_{n-1}$), ..., $\mathcal{B}^{\{x_1,x_2,...,x_{|K_1|}\}}_n$ (the result of the sequential merging of $\mathcal{B}^{x_j}_{n-1}$, where $j \in \{x_1, x_2, ..., x_{|K_1|}\}$).

Algorithm 3 iteratively constructs n-concepts by merging n-concepts from $\mathcal{B}^{\{x_1,x_2,...,x_{i-1}\}}_n$ with $(n-1)$-concepts from $\mathcal{B}^{x_i}_{n-1}$. On each call a new set of $(n-1)$-concepts, corresponding to a particular value $x_i \in K_1$, is added to a set of n-concepts. To add $(n-1)$-concepts to a set of n-concepts, each $(n-1)$-concept

	α				β				γ				δ			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
1	×				×			×				×				
2	×				×	×	×			×	×			×	×	
3			×	×	×	×	×			×	×					
4								×								

(a) A triadic context

	α	β	γ	δ
a	×	×		
b		×	×	×
c		×	×	×
d				

(b) Subcontext $\mathbb{K}_{K_1=2}$

	1				2			
	α	β	γ	δ	α	β	γ	δ
a	×	×			×	×		
b						×	×	×
c						×	×	×
d		×	×					

(c) Subcontext $\mathbb{K}_{K_1=\{1,2\}}$

Fig. 1. A triadic context and its subcontexts

$X = (X_2, ..., X_n) \in \mathcal{B}_{n-1}^{x_i}$ is expanded to $X = (\{x_i\}, X_2, ..., X_n)$ (see step 3 of Algorithm 3). A function *mark* assigns a binary label to a concept, a function $isFull(Y_1)$ checks whether $Y_1 \subseteq K_1$ contains all already considered elements $a \in K_1$.

Set-wise inclusion/exclusion relations between the corresponding sets A_2, ..., A_n of concepts from $\mathcal{B}_{n-1}^{x_i}$ and $\mathcal{B}_n$ are used to reduce the number of operations and to avoid the redundant concept computation.

Algorithm 4 checks whether an n-ary itemset $Z_1, ..., Z_n$ is closed by iterating over dimensions. For the selected dimension dim it iterates over elements from $K_{dim} \setminus Z_{dim}$ and checks elements from Z_i, $i \in \{1, ..., n\} \setminus \{dim\}$. It stops when an empty entry is found in the context for the first time or when all the dimensions dim are checked.

Algorithm 5 works with a trie. The trie stores concept $(A_1, ..., A_n)$ as a sequence of lexicographically ordered elements from A_1, ... A_n. A sequence of sets A_i is fixed, i.e. elements of A_i are closer to the root than elements of A_j for all $i < j$ and elements from each dimension i are lexicographically ordered within set A_i. To ensure the uniqueness of $(A_1, ..., A_n)$ it is sufficient to check the sequence of the lexicographically ordered elements from A_1 to A_{n-1}.

Algorithm 1: CreateSubcontext

Data: $a \in K_1, \mathbb{K} = (K_1, K_2, ..., K_n, I^n)$
Result: a subcontext $\mathbb{K}_{K_1=a}$ consisted of a set of $(n-1)$ tuples $\{(k_2, k_3, ..., k_n) \mid (a, k_2, k_3, ..., k_n) \in I^n\}$

```
begin
    return K_{K1=a} = (K_2, ..., K_n, I^n_{K1=a})
```

Example Let us consider how the algorithm works using a context from the running example. The algorithm consecutively computes triconcepts from the formal concepts corresponding to the following contexts: $\mathbb{K}_{K_1=1}$, $\mathbb{K}_{K_1=2}$, $\mathbb{K}_{K_1=3}$, $\mathbb{K}_{K_1=4}$. Parameters and results of the function "getNConcepts" in the execution order are given in Table 1.

Algorithm 2: ComputeConcepts

Data: $\mathbb{K} = (K_1, K_2, ..., K_n, I^n)$
Result: the set $\mathcal{B}_n = \{\{A_1, A_2, ..., A_n\} \,|A_i \subseteq K_i\}$ of n-concepts of $\mathbb{K}$

begin
    if $n = 2$ **then**
        return $ComputeFormalConcepts(\mathbb{K})$
    else
        $\mathcal{B}_n = \{\emptyset\}$
        for $a \in K_1$ **do**
            $\mathbb{K}_{K_1=a} = CreateSubcontext(a, \mathbb{K})$
            $\mathcal{B}^a_{n-1} = ComputeConcepts(\mathbb{K}_{K_1=a})$
            $\mathcal{B}_n = getNConcepts(\mathcal{B}_n, \mathcal{B}^a_{n-1}, a)$
        return $\mathcal{B}_n$

3 Properties of the Algorithm

Below it is proved that all concepts created by the algorithms are n-concepts (i.e. closed n-itemsets) of the context and all n-concepts of the context are generated by the algorithm. Algorithm 2 iteratively adds $a \in K_1$ to create n-closed concepts. To denote an intermediate result the following notation is used: an element $a \in K_1$ is denoted by x_i, thus already considered elements are denoted by $x_1, ..., x_{i-1}$, the current state of $\mathcal{B}_n$ is denoted by $\mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$. When element x_i is added to $\mathcal{B}_n$, $\mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$ changes its state to $\mathcal{B}_n^{\{x_1,...,x_i\}}$.

The properties 1-4 claim that all modifications of the concepts from $\mathcal{B}_n$ as well as construction of new ones result in a set of closed itemsets.

Property 1. If Y is *marked* (see step 16 and 28 in Algorithm 3) then $Y = (Y_1 \cup \{x_i\}, Y_2, ..., Y_n) \in \mathcal{B}_n^{\{x_1,...,x_i\}}$, where $(Y_1, Y_2, ..., Y_n) \in \mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$. If, in addition, $\forall x(x \in \{x_1, ..., x_{i-1}\})\ x \in Y_1$, then the intersection $(\{x_i\}, X_2, ..., X_n) \in \mathcal{B}_n^{\{a\}}$ with any other $(Z_1, Z_2, ..., Z_n) \in \mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$ gives unclosed subsets.

Proof Let us assume that Y is marked, but $Y \notin \mathcal{B}_n^{\{x_1,...,x_i\}}$. Then $\exists Z(Z \in \mathcal{B}_n^{\{x_1,...,x_i\}})$, such that $Y \subset Z$. Therefore, $(Y_1 \cup \{x_i\}) \subset Z_1 \Rightarrow x_i \in Z_1$, $Z_2 \subseteq$

Table 1. The sequence of “getNConcepts” calls for the running example

Input			**Output**
Value in K_1	$\mathcal{B}_n$	$\mathcal{B}^a_{n-1}$	$\mathcal{B}_n$
1	$\emptyset$	$(\{a\},\{\alpha,\beta\})$, $(\{d\},\{\beta,\gamma\})$, $(\{a,d\},\{\beta\})$	$(\{1\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$
2	$(\{1\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$	$(\{a\},\{\alpha,\beta\})$, $(\{b,c\},\{\beta,\gamma,\delta\})$, $(\{a,b,c\},\{\beta\})$	$(\{1,2\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$, $(\{2\},\{b,c\},\{\beta,\gamma,\delta\})$, $(\{2\},\{a,b,c\},\{\beta\})$
3	$(\{1,2\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$, $(\{2\},\{b,c\},\{\beta,\gamma,\delta\})$, $(\{2\},\{a,b,c\},\{\beta\})$	$(\{c,d\},\{\alpha\})$, $(\{b,c\},\{\beta,\gamma\})$, $(\{c\},\{\alpha,\beta,\gamma\})$, $(\{a,b,c\},\{\beta\})$	$(\{1,2\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$, $(\{2,3\},\{a,b,c\},\{\beta\})$, $(\{2\},\{b,c\},\{\beta,\gamma,\delta\})$, $(\{3\},\{c,d\},\{\alpha\})$, $(\{1,2,3\},\{a\},\{\beta\})$, $(\{2,3\},\{b,c\},\{\beta,\gamma\})$, $(\{3\},\{c\},\{\alpha,\beta,\gamma\})$
4	$(\{1,2\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$, $(\{2,3\},\{a,b,c\},\{\beta\})$, $(\{2\},\{b,c\},\{\beta,\gamma,\delta\})$, $(\{3\},\{c,d\},\{\alpha\})$, $(\{1,2,3\},\{a\},\{\beta\})$, $(\{2,3\},\{b,c\},\{\beta,\gamma\})$, $(\{3\},\{c\},\{\alpha,\beta,\gamma\})$	$(\{d\},\{\beta\})$	$(\{1,2\},\{a\},\{\alpha,\beta\})$, $(\{1\},\{d\},\{\beta,\gamma\})$, $(\{1\},\{a,d\},\{\beta\})$, $(\{2,3\},\{a,b,c\},\{\beta\})$, $(\{2\},\{b,c\},\{\beta,\gamma,\delta\})$, $(\{3\},\{c,d\},\{\alpha\})$, $(\{1,2,3\},\{a\},\{\beta\})$, $(\{2,3\},\{b,c\},\{\beta,\gamma\})$, $(\{3\},\{c\},\{\alpha,\beta,\gamma\})$, $(\{1,4\},\{d\},\{\beta\})$

Algorithm 3: getNConcepts

Data: $\mathcal{B}_n$ - the intermediate result, set of concepts of the context $\mathcal{B}^{a}_{n-1} = \{\{A_2, ..., A_n\} \mid A_i \subseteq K_i, i = 2, ..., n\}$ - the set of $(n-1)$-concepts of the subcontext $\mathbb{K}_{K_1=a}$, a

Result: $\mathcal{B}_n = \{\{A_1, A_2, ..., A_n\} \, | A_i \subseteq K_i\}$ the set of n-concepts of $\mathbb{K}$

begin
 for $(A_2, A_3, ..., A_n) \in \mathcal{B}^{a}_{n-1}$ **do**
  $\mathcal{B}^{\{a\}}_n = \mathcal{B}^{\{a\}}_n \cup (\{a\}, A_2, A_3, ..., A_n)$
 end
 if $\mathcal{B}_n = \emptyset$ **then**
  return $\mathcal{B}^{imp}_n$
 end
 else
  $\mathcal{B}^{x}_n = \{\emptyset\}$
  $\mathcal{CMP} = \{\emptyset\}$
  for $X = (\{a\}, X_2, ..., X_n) \in \mathcal{B}^{\{a\}}_n$ **do**
   $Z_1 = \{a\}$
   for $Y = (Y_1, Y_2, ..., Y_n) \in \mathcal{B}_n$ **do**
    if $X_2 = Y_2, \; ..., \; X_n = Y_n$ **then**
     $Y_1 = Y_1 \cup \{a\}$
     $mark(Y) = 1$
     if $isFull(Y_1)$ **then**
      $\mathcal{B}^{\{a\}}_n = \mathcal{B}^{\{a\}}_n \setminus X$
     end
    end
    else if $(X_2 \subseteq Y_2)\&...\&(X_n \subseteq Y_n)$ **then**
     $Z_1 = Z_1 \cup Y_1$
     $\mathcal{CMP} = \mathcal{CMP} \cup (Y, \{X_2, ...X_n\})$
    end
    else if $(X_2 \supseteq Y_2)\&...\&(X_n \supseteq Y_n)$ **then**
     $Y_1 = Y_1 \cup \{a\}$
     $\mathcal{CMP} = \mathcal{CMP} \cup (Y, \{X_2, ...X_n\})$
     $mark(Y) = 1$
    end
   end
   $\mathcal{B}^{x}_n = \mathcal{B}^{x}_n \cup \{(Z_1, X_2, ..., X_n)\}$
  end
  for $X = (X_2, ..., X_n) \in \mathcal{B}^{\{a\}}_n$ **do**
   $\mathcal{B}_{temp} = \{\emptyset\}$
   for $Y = (Y_1, Y_2, ..., Y_m) \in \mathcal{B}_n$ **do**
    if $mark(Y) = 0$ **then**
     if $(Y, X_2, ..., X_m) \notin \mathcal{CMP}$ **then**
      $Z_1 = x \cup Y_1$
      $Z_2 = Y_2 \cap X_2, ..., Z_n = Y_n \cap X_n$
      if $is\_closed(Z_1, Z_2, ..., Z_n)$ **then** `// The alg. 4`
       if $is\_unique(Z_1, Z_2, ..., Z_n)$ **then** `// The alg. 5`
        $\mathcal{B}_{temp} = \mathcal{B}_{temp} \cup \{Z\}$
       end
      end
     end
    end
   end
   $\mathcal{B}_n = \mathcal{B}_n \cup \mathcal{B}_{temp}$
  end
  $\mathcal{B}_n = \mathcal{B}_n \cup \mathcal{B}^{x}_n$
 end
end

$X_2, ..., Z_n \subseteq X_n$. Due to the assumption $(Y \subset Z)$ and steps 14 and 25 of Algorithm 3, $X_2 \subseteq Z_2, ..., X_n \subseteq Z_n$. These relations holds iff $X_2 = Z_2, ..., X_n = Z_n$. This is contrary to the properties of $\mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$, namely, the closedness and uniqueness of elements, thus our assumption is wrong, i.e. Y is closed.

If, in addition, $Y_1 = \{x_1, ..., x_{i-1}\}$, then any other concepts $Z = (Z_1, ..., Z_n) \in \mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$ are met the following conditions: $Z_1 \subset Y_1 \cup \{x_i\}$ and there exists Z_j such that $Y_j \subset Z_j$. Since $Z_l \cap X_l \subset X_l = Y_l$, one can not obtain $(Z_1 \cup \{x_1\}, Z_2 \cap X_2, ...Z_n \cap X_n) \not\subseteq (Y_1 \cup \{x_1\}, X_2, ..., X_n)$. It was required to prove.

Property 2. If $Z = (Z_1, X_2, ..., X_n)$ has been updated in step 22 (and added in step 31), then $Z \in \mathcal{B}_n^{\{x_1,...,x_i\}}$.

Proof Let us assume that $Z \notin \mathcal{B}_n^{\{x_1,...,x_i\}}$, i.e. it is not closed and $\exists Y = (Y_1, ..., Y_n)$ $(Y \in \mathcal{B}_n^{\{x_1,...,x_i\}})$, such that $Z \subset Y$. Since $(X_2, ..., X_n) \in \mathcal{B}_{n-1}^{x_i}$, then Z is not closed iff $\exists y(y \in Y_1)$, such that $y \notin Z_1$. The last inference is impossible, since Algorithm 3 ensures presence of all elements $y \in Y_1$, such that $X_2 \subseteq Y_2, ..., X_n \subseteq Y_n$. We get the contradiction (i.e. $y \in Z_1$) to the assumption that Z is unclosed. It is wrong and $Z \in \mathcal{B}_n^{\{x_1,...,x_i\}}$.

Property 3. The modifications in step 26 produce a concept (i.e. closed it*emset*).

Proof Let us assume that we have updated Y, i.e. we have got $Y^{mod} = (Y_1 \cup x_i, Y_2, ..., Y_n)$ (in the listing x_i is denoted by a) and $Y \in \mathcal{B}_n^{\{x_1,...,x_{i-1}\}}$, but $Y^{mod} \notin \mathcal{B}_n^{\{x_1,...,x_i\}}$. Hence, $Y^{mod} = Y_1$, that contradicts to our assumption and $Y^{mod} \in \mathcal{B}_n^{\{x_1,...,x_i\}}$.

Property 4. If $Y \in \mathcal{B}_{temp}$, constructed in steps 35-43, then Y is closed and unique.

The closedness and uniqueness of concepts is checked implicitly by Algorithms 4 and 5, respectively.

As it was shown before, the algorithm produces n-ary itemsets which are closed. Property 5 claims that the algorithm computes all concepts and does not miss any closed itemsets.

Property 5. Algorithm 3 computes all n-concepts and only them, i.e. $X \in \mathcal{B}_n \Leftrightarrow X \in \mathcal{B}(\mathbb{K})$.

Proof It has been proved above that $X \in \mathcal{B}_n \Rightarrow X \in \mathcal{B}(\mathbb{K})$.

Let us prove that $X \in \mathcal{B}(\mathbb{K}) \Rightarrow X \in \mathcal{B}_n$ by the mathematical induction.

Basis: For any fixed values $k_1, k_2, ..., k_{n-2}$ from the corresponding subcontext

$$\mathbb{K}_{K_1=k_1,K_2=k_2,...,K_{n-2}=k_{n-2}} \subseteq \mathbb{K}$$

we get for all $X \in \mathcal{B}_2^{k_1,k_2,...,k_{n-2}} \Rightarrow X \in \mathcal{B}(\mathbb{K}_{K_1=k_1,K_2=k_2,...,K_{n-2}=k_{n-2}})$, since a correct algorithm for formal concepts is used.

Algorithm 4: is_closed

Data: An n-ary itemset $(Z_1, ..., Z_n)$
Result: Boolean value: true, if the itemset is closed; false, otherwise

```
begin
    closed = True
    dim = 1
    while (dim < n) and closed do
        for a ∈ K_dim \ A_dim do
            contain_all = True
            for z_1 ∈ Z_1, ..., z_{dim-1} ∈ Z_{dim-1}, z_{dim+1} ∈ Z_{dim+1}, ..., z_n ∈ Z_n do
                if (z_1, ...z_{dim-1}, a, z_{dim+1}, ..., z_n) = 0 then
                    contain_all = False
            if contain_all then
                closed = False
        dim = dim + 1
    retrun closed
```

Inductive step: Show that if for any $x \in K_1$ and the subcontext $\mathbb{K}_{K_1=x} \subseteq \mathbb{K}$ it is true that $\forall x \forall X (x \in K_1) \, (X \in \mathcal{B}(\mathbb{K}_{K_1=x})) \Rightarrow X \in \mathcal{B}^x_{n-1}$, then $\forall Z \, (Z \in \mathcal{B}(\mathbb{K})) \Rightarrow Z \in \mathcal{B}_n$.

Accordingly to the algorithm, $\forall x \exists Z (x \in K_1)(Z \in \mathcal{B}_n)$, such that $x \in Z_1$ and all concepts of $\mathcal{B}(\mathbb{K}_{K_1=x})$ present in $\mathcal{B}_n$ with maximal possible set X_1.

The intersection $X \in \mathcal{B}^{x_i}_{n-1}$ with each elements of $\mathcal{B}^{\{x_1,...,x_{i-1}\}}_n$ (closed concepts of the previous step) results in the creation of all possible combinations of $(X_2, ..., X_n)$ with other already existed elements. Since on each iteration a new element $x_i \in K_1$ is added to $\mathcal{B}^{\{x_1,...,x_{i-1}\}}_n$, we will get all possible combinations $(Y_2, ..., Y_n)$ for all possible subsets of $\{x_1, ..., x_i\} \subseteq K_1$. As the result, we obtain all maximal $Z \in \mathcal{B}_n$.

4 Complexity of the Algorithm

In this section the time complexity of Algorithm 3 is discussed. The input of the algorithm is a set $\{(\{a\}, A_2, ..., A_n) \mid a \in K_1\}$ with the space complexity $inp = O(|K_1||\mathcal{B}_{n-1}|)$, the output is $\mathcal{B}_n$ with the space complexity $O(|\mathcal{B}_n|)$, and $|\mathcal{B}_n| \leq |K_1||\mathcal{B}_{n-1}|$.

It should be noted here that as in [10] a trie is used to store concepts and to improve the time complexity. One can ensure the presence of only unique elements by storing $(n-1)$ sets in a trie, since any two closed n-dimensional itemsets have at most $n-2$ similar subsets. The space complexity of this structure will be huge, to be precise $O(2^{|K_1|+|K_2|+...+|K_{n-1}|})$. Using this structure we need

Algorithm 5: is_unique

Data: An n-ary itemset $(Z_1, ..., Z_n)$, a trie $\mathcal{T}$, computed on $K_1 \times K_2 \times ... \times K_{n-1}$

Result: Boolean value: **True**, if the itemset is already exist; **False**, otherwise

```
begin
    if Z ∈ T then
        retrun False
    else
        retrun True
```

$O(|K_1|...|K_{n-1}|)$ to check the uniqueness (Algorithm 5), to add new element to the trie one needs $O(|K_1|...|K_{n-1}|)$ and to modify an element (to expand the first set) one needs $O(2|K_1|...|K_{n-1}|)$.

The number of pairwise intersections of $(n-1)$ elements of $\mathcal{B}_n$ and $\mathcal{B}^a_{n-1}$ is reduced by utilizing *mark* labels, $\mathcal{CMP}$. The absence of unclosed and non-unique elements is ensured by applying Algorithm 4 and 5 to a subset of generated concepts.

The closedness verification (Algorithm 4) for an itemset $(A_1, ..., A_n)$ takes at most $O(|K_1 \setminus A_1||A_2|...|A_n| + |A_1||K_2 \setminus A_2|...|A_n| + ... + |A_1||A_2|...|K_n \setminus A_n|)$, more generally, $O(n|K_1||K_2|...|K_n|)$.

Thus, the time complexity of the algorithm is

$$O(\underbrace{|K_1||K_2|...|K_n||\mathcal{B}_n||\mathcal{B}_{n-1}|}_{\text{pairwise comparison of itemsets}}(CV + UV + TP)),$$

where $CV = |K_1||K_2|...|K_n|$, $UV = |K_1||K_2|...|K_{n-1}|$, $TU = |K_1||K_2|...|K_{n-1}|$ correspond to the number of operations for the closedness validation, the uniqueness validation and the trie update.

In terms of the input ($\mathbf{p} = |K_1||\mathcal{B}_{n-1}|$) and the output ($\mathbf{K} = |\mathcal{B}_n|$) the complexity takes the following form: $O(|K_2|...|K_n| \cdot \mathbf{p} \cdot \mathbf{K} \cdot (CV + UV + TU))$, or, more generally, $O(|I^n|^2\mathbf{p} \cdot \mathbf{K})$.

5 Conclusion

In this paper we have proposed a new incremental algorithm for computing n-concepts. The algorithm can be based on any algorithm for computing formal concepts. The algorithm recursively computes k-concepts, where $k \in \{3, ..., n\}$ and iteratively merges $k-1$-concepts to derive k-concepts. It has $O(|I^n|^2\mathbf{pK})$ time complexity, $\mathbf{p} = |K_1||\mathcal{B}_{n-1}|$ and $\mathbf{K} = |\mathcal{B}_n|$ are the size of the input and the output, respectively.

An important direction for future work is to compare the proposed algorithm with other algorithms for generating concepts of dimension $n \geq 3$.

Acknowledgment

Sections 1, 2 were made by T. Makhalova and supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia. The work of Lhouri Nourine was supported by ANR project Graphen ANR-15-CE40-0009.

References

1. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Data peeler: Contraint-based closed pattern mining in n-ary relations. In: SDM. vol. 8, pp. 37–48. SIAM (2008)
2. Ignatov, D.I., Gnatyshak, D.V., Kuznetsov, S.O., Mirkin, B.G.: Triadic formal concept analysis and triclustering: searching for optimal patterns. Machine Learning 101(1-3), 271 (2015)
3. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J., Zhukov, L.E.: Can triconcepts become triclusters? International Journal of General Systems 42(6), 572–593 (2013)
4. Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Trias–an algorithm for mining iceberg tri-lattices. In: null. pp. 907–911. IEEE (2006)
5. Ji, L., Tan, K.L., Tung, A.K.: Mining frequent closed cubes in 3d datasets. In: Proceedings of the 32nd international conference on Very large data bases. pp. 811–822. VLDB Endowment (2006)
6. Kuznetsov, S.O.: Interpretation on graphs and complexity characteristics of a search for specific patterns. Automatic Documentation and Mathematical Linguistics 24(1), 37–45 (1989)
7. Kuznetsov, S.O.: On computing the size of a lattice and related decision problems. Order 18(4), 313–321 (2001)
8. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence 14(2-3), 189–216 (2002)
9. Lehmann, F., Wille, R.: Conceptual Structures: Applications, Implementation and Theory: Third International Conference on Conceptual Structures, ICCS '95 Santa Cruz, CA, USA, August 14–18, 1995 Proceedings, chap. A triadic approach to formal concept analysis, pp. 32–43. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
10. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Information processing letters 71(5), 199–204 (1999)
11. Trabelsi, C., Jelassi, N., Yahia, S.B.: Scalable mining of frequent tri-concepts from folksonomies. In: Advances in knowledge discovery and data mining, pp. 231–242. Springer (2012)
12. Voutsadakis, G.: Polyadic concept analysis. Order 19(3), 295–304 (2002)