An Incremental Algorithm for Computing n-Dimensional Concepts

Alexandre Bazin, Giacomo Kahn, Olivier Raynaud

An Incremental Algorithm for Computing n-Dimensional Concepts

Alexandre Bazin, Giacomo Kahn[1], Olivier Raynaud[1,2]

(1) Université Clermont Auvergne, LIMOS, BP 10448, F-63000 CLERMONT-FERRAND, FRANCE
(2) CNRS, UMR 6158, LIMOS, F-63178 AUBIERE, FRANCE
contact@alexandrebazin.com, giacomo.kahn@isima.fr, raynaud@isima.fr

Abstract

The problem of computing maximal rectangles (concepts) from a binary relation (context) has received much attention. The same cannot be said of the general, n-dimensional case. In this work, we propose the first output-polynomial incremental algorithm for computing the n-concepts of an n-context.

Keywords:
Polyadic Concept Analysis, Incremental Algorithm, Output-polynomial Algorithm

1. Introduction

Formal concept analysis represents lattices by means of formal contexts, *i.e.*, bidimensional crosstables. In this framework, the elements of a lattice are maximal rectangles full of crosses called concepts. This naturally finds applications in any domain dealing with data in its most basic object-attributes form [9, 11]. In recent years, probably motivated by the increasing complexity of the data manipulated in the real world, the attention has begun to shift to the multidimensional generalisation of contexts: n-contexts [6, 7, 10, 12, 13].

With those multidimensional n-contexts come the associated notions of n-concepts and n-lattices as generalisations of, respectively, concepts and lattices. Unfortunately, useful and entertaining as they are, these structures have not yet found the attention they deserve. Most of the literature on the subject is concerned with the three-dimensional case [1, 4, 8] and they have not been extensively studied in higher dimensions and even lack basic algorithms for $n > 3$. Most notably, for the crucial task of computing the n-concepts of an n-context, only one algorithm exists, DATA-PEELER [3].

The question of finding a reasonably close upper bound to the number of n-concepts in an n-context is still open. In [2], the author gives an example of a class of 3-lattices with 3^k elements but no results exist on attainable numbers for $n > 3$. In practice, even relatively small, common datasets can be difficult to handle. This is the reason why we would like to be able to adapt to changes in an n-context without recomputing all of its n-concepts. In order to do so, and because no algorithm exists, we propose an output-polynomial incremental algorithm for updating the set of n-concepts of an n-context after it has been extended by a new element on one of its dimensions.

This work is structured as follows. Section 2 presents the definitions of the necessary notions as well as the notations used throughout this work. Section 3 outlines the properties needed to validate the algorithm. Finally, Section 4 contains the algorithm itself and an analysis of its complexity.

2. Definitions and Notations

2.1. Definitions

This section contains the formal definitions of n-contexts and n-concepts. We refer the reader to [12] for an explanation of their relation to the notion of n-lattice that is not further developed here. An introduction to the 2-dimensional case can be found in [5].

Definition 1. *An n-*context *is an $(n+1)$-tuple $\mathcal{C} = (S_1, \ldots, S_n, R)$ in which S_i, $i \in \{1, \ldots, n\}$, is a set called a* dimension *and R is an n-ary relation between the dimensions.*

An n-context can be represented by a $|S_1| \times \cdots \times |S_n|$ crosstable as illustrated in Fig. 1. For obvious reasons, all our example figures will be 3-contexts.

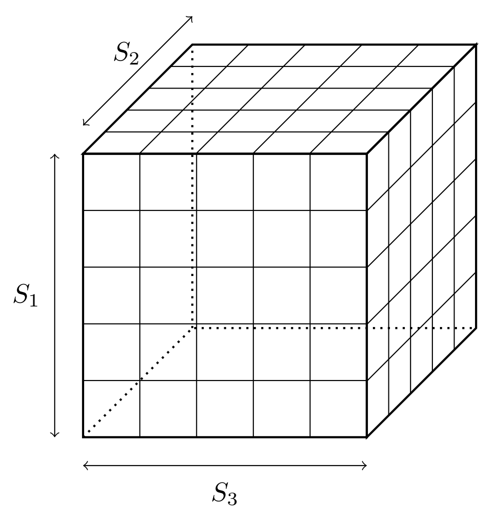

Figure 1: Visual representation of a 3-context without its crosses.

Definition 2. *An* n-concept *of* $\mathcal{C} = (S_1, \ldots, S_n, R)$ *is an* n*-tuple* $(X_1, \ldots, X_n)$ *such that* $\prod_{i \in \{1,\ldots,n\}} X_i \subseteq R$ *and, for all* $i \in \{1, \ldots, n\}$*, there is no* $k \in S_i \setminus X_i$ *such that* $\{k\} \times \prod_{j \in \{1,\ldots,n\} \setminus \{i\}} X_j \subseteq R$*.*

In other words, an n-concept is a box full of crosses (up to the permutation of the elements of the dimensions) that is maximal in the sense that it cannot be extended on any single dimension.

In the Fig. 2 example, five 3-concepts are present: $(\{1,2\},\{1,2,3\},\{1\})$, $(\{1\},\{1\},\{1,2\})$, $(\{2\},\{3\},\{1,3\})$, $(\emptyset,\{1,2,3\},\{1,2,3\})$ and $(\{1,2\},\emptyset,\{1,2,3\})$.

	1	2	3	1	2	3
1	x	x		x		
2	x			x		
3	x			x		x
		1			2	

Figure 2: An example of a $2 \times 3 \times 3$ 3-context.

The set of all the n-concepts of an n-context together with the n quasi-orders $\lesssim_i$ corresponding to the inclusion relations on the subsets of each dimension form a complete n-lattice [12]. Additionally, every complete n-lattice is isomorphic to the one formed by the n-concepts of a particular n-context.

2.2. Notations

In the remainder of this work, the first component/dimension of an n-concept/context will be called the *height* while the other components/dimensions will be called the *width*.

Let $\mathcal{C} = (S_1, \ldots, S_n, R)$ be an n-context, $\mathcal{C}_e = (S_2, \ldots, S_n, R_e)$ be an $(n-1)$-context, $\mathcal{K}$ be a set of n-concepts and $C = (X_1, \ldots, X_n)$ be an n-concept of $\mathcal{C}$. We will use the following notations:

- $\mathcal{T}(\mathcal{C})$ denotes the set of all n-concepts of $\mathcal{C}$.
- $\mathcal{H}_H(\mathcal{C})$ denotes the set of all n-concepts of the form $(H, Y_2, \ldots, Y_n) \in \mathcal{T}(\mathcal{C})$.
- $\mathcal{C}^e = (S_1 \cup \{e\}, S_2, \ldots, S_n, R^e)$ with $R^e = R \cup \{(e, r_2, \ldots, r_n) \mid (r_2, \ldots, r_n) \in R_e\}$ denotes the n-context constructed by "glueing" $\mathcal{C}_e$ on the bottom of $\mathcal{C}$.
- $\mathcal{P}(C, \mathcal{C}_e) = \{(x_2, \ldots, x_n) \in R_e \mid \forall i \in \{2, \ldots, n\}, x_i \in X_i\}$ denotes the *projection* of a concept C on $\mathcal{C}_e$.
- $\mathcal{P}(\mathcal{K}, \mathcal{C}_e) = \bigcup_{K \in \mathcal{K}} \mathcal{P}(K, \mathcal{C}_e)$ denotes the union of the projections on $\mathcal{C}_e$ of the n-concepts in $\mathcal{K}$.

$\mathcal{C}^e$

	$\mathcal{C}$						$\mathcal{C}_e$		
	1	2	3	1	2	3	1	2	3
1	x	x		x			x		
2	x			x			x	x	
3	x			x		x			x
		1			2			e	

Figure 3: The 3-context $\mathcal{C}^e$ resulting from glueing $\mathcal{C}_e$ to our example 3-context $\mathcal{C}$.

Following the example in Fig. 3, the projections on $\mathcal{C}_e$ of the 3-concepts in $\mathcal{T}(\mathcal{C})$ are $\mathcal{P}((\{1,2\},\{1,2,3\},\{1\}),\mathcal{C}_e) = \{(1,1),(2,1)\}$, $\mathcal{P}((\{1\},\{1\},\{1,2\}),\mathcal{C}_e) = \{(1,1)\}$, $\mathcal{P}((\{2\},\{3\},\{1,3\}),\mathcal{C}_e) = \{(3,3)\}$, $\mathcal{P}((\emptyset,\{1,2,3\},\{1,2,3\}),\mathcal{C}_e) = R_e$ and $\mathcal{P}((\{1,2\},\emptyset,\{1,2,3\}),\mathcal{C}_e) = \emptyset$. Figure 4 shows the projection of a 3-concept of $\mathcal{C}$ onto $\mathcal{C}_e$.

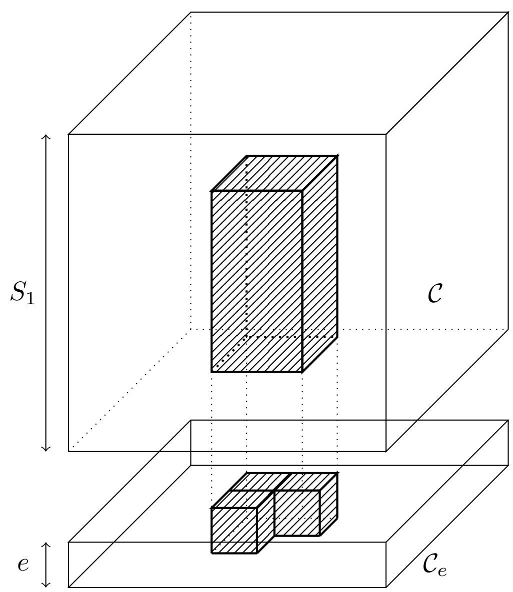

Figure 4: The $(n-1)$-context $(S_2,\ldots,S_n,\mathcal{P}(C,\mathcal{C}_e))$ is constructed from $\mathcal{C}_e$ by keeping only the crosses "under" C.

3. Important Properties

Our goal is to compute the n-concepts of an n-context incrementally. That is, we want to compute the n-concepts of $\mathcal{C}^e$ from those of $\mathcal{C}$. In this section, we present the properties that allow us to go from $\mathcal{T}(\mathcal{C})$ to $\mathcal{T}(\mathcal{C}^e)$.

Proposition 1. *Let* $(X_1, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$. *Then,* $\exists (X_1 \setminus \{e\}, Y_2 \ldots, Y_n) \in \mathcal{T}(\mathcal{C})$ *with* $X_i \subseteq Y_i$, $\forall i \in \{2, \ldots, n\}$.

Proof If $(X_1, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$, then $\prod_{i\in\{1,\ldots,n\}} X_i \subseteq R^e$. As such, we also have $X_1 \setminus \{e\} \times \prod_{i\in\{2,\ldots,n\}} X_i \subseteq R^e$. Consequently, $X_1 \setminus \{e\} \times \prod_{i\in\{2,\ldots,n\}} X_i \subseteq R$. Thus, in $\mathcal{C}$, $(X_1 \setminus \{e\}, X_2, \ldots, X_n)$ is a box full of crosses that is maximal for the first component as $(X_1, X_2, \ldots, X_n)$ would not otherwise be an n-concept. As it is not necessarily maximal on the other dimensions, there is an n-concept $(X_1 \setminus \{e\}, Y_2 \ldots, Y_n) \in \mathcal{T}(\mathcal{C})$ with $X_i \subseteq Y_i$, $\forall i \in \{2, \ldots, n\}$. $\square$

Corollary 1. *Let* $(X_1, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$. *If* $e \notin X_1$, *then* $(X_1, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C})$.

Proposition 1 states that each new n-concept of $\mathcal{C}^e$ can be constructed from at least one n-concept of $\mathcal{C}$ by adding e to its height and reducing its width.

Proposition 2. *Let* $(X_1, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$ *with* $e \in X_1$, *then*

$$(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_{X_1\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e))) .$$

Proof Since $(X_1, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$, we know that $\prod_{i\in\{2,\ldots,n\}} X_i \subseteq R_e$. Moreover, according to Prop. 1, there is an n-concept $(X_1 \setminus \{e\}, Y_2, \ldots, Y_n) \in \mathcal{T}(\mathcal{C})$ with $X_i \subseteq Y_i$, $\forall i \in \{2, \ldots, n\}$, so $(X_2, \ldots, X_n)$ is a box full of crosses in $(S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_{X_1\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e))$.

Let us suppose it is not maximal and, thus, not an $(n-1)$-concept of $(S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_{X_1\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e))$. Without loss of generality, let us say that it can be extended on its first component: there is a $k \in S_2 \setminus X_2$ such that $\{k\} \times \prod_{i\in\{3,\ldots,n\}} X_i \subseteq \mathcal{P}(\mathcal{H}_{X_1\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e)$. This implies that $\mathcal{C}$ contains an n-concept $(X_1 \setminus \{e\}, Z_2, \ldots, Z_n)$ with $k \in Z_2$ and $X_i \subseteq Z_i$, $\forall i \in \{2, \ldots, n\}$. If this is the case, then $(X_1, X_2, \ldots, X_n)$ can be extended so it is not an n-concept of $\mathcal{T}(\mathcal{C}^e)$, which contradicts our premise. Thus, $(X_2, \ldots, X_n)$ is necessarily in $\mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_{X_1\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e)))$. $\square$

Proposition 2 states that the width of the new n-concepts of height $H \cup \{e\}$ can be found among the $(n-1)$-concepts of the $(n-1)$-context resulting from the projections on $\mathcal{C}_e$ of all the n-concepts of height H.

Proposition 3. *If* $(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_H(\mathcal{C}), \mathcal{C}_e)))$, *then there is an n-concept* $(Y, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$ *with* $H \cup \{e\} \subseteq Y$.

Proof If $(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_H(\mathcal{C}), \mathcal{C}_e)))$, then we have an n-concept $(H, Y_2, \ldots, Y_n) \in \mathcal{T}(\mathcal{C})$ with $X_i \subseteq Y_i$, $\forall i \in \{2, \ldots, n\}$. If, because the height is not maximal, $(H \cup \{e\}, X_2, \ldots, X_n) \notin \mathcal{T}(\mathcal{C}^e)$, then $\exists (Z, Z_2, \ldots, Z_n) \in \mathcal{T}(\mathcal{C})$ with $H \subseteq Z$ and, $\forall i \in \{2, \ldots, n\}$, $X_i \subseteq Z_i \subseteq Y_i$. Hence, $\prod_{i\in\{2,\ldots,n\}} X_i \subseteq \mathcal{P}((Z, Z_2, \ldots, Z_n), \mathcal{C}_e)$. If $(X_2, \ldots, X_n) \notin \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_Z(\mathcal{C}), \mathcal{C}_e)))$, then it is because it can be extended. Without loss of generality, let us say that it can be extended on its first component: $\exists k \in S_2 \setminus X_2$ such that $\{k\} \times \prod_{i\in\{3,\ldots,n\}} X_i \subset$

$\mathcal{P}(\mathcal{H}_Z(\mathcal{C}), \mathcal{C}_e))$. As such, we have that $Z \times \{k\} \times \prod_{i\in\{3,\ldots,n\}} X_i \subset R$. This implies that $H \times \{k\} \times \prod_{i\in\{3,\ldots,n\}} X_i \subset R$, which contradicts the fact that $(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_H(\mathcal{C}), \mathcal{C}_e)))$ as it would also be extended by k. Consequently, $(X_2, \ldots, X_n) \notin \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_Z(\mathcal{C}), \mathcal{C}_e)))$.

All in all, if $(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_H(\mathcal{C}), \mathcal{C}_e)))$, then either $(H \cup \{e\}, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$ or $(X_2, \ldots, X_n) \in \mathcal{T}((S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_Z(\mathcal{C}), \mathcal{C}_e)))$ for some $H \subset Z$. Eventually, because S_1 is finite, a maximal height Z such that $(Z \cup \{e\}, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$ will be reached. □

Proposition 3 implies that every $(n-1)$-concept that appears in the $(n-1)$-context resulting from the projection of a given height on $\mathcal{C}_e$ will eventually be the width of some n-concept.

From Propositions 1, 2 and 3, we can deduce how to compute the new n-concepts that contain e in their height and recognize which n-concepts from $\mathcal{T}(\mathcal{C})$ will remain in $\mathcal{T}(\mathcal{C}^e)$. Indeed, if $C = (X_1, \ldots, X_n) \in \mathcal{T}(\mathcal{C})$ and $\mathcal{P}(C, \mathcal{C}_e) = \prod_{i\in\{2,\ldots,n\}} X_i$, then Proposition 2 implies that $(X_1 \cup \{e\}, X_2, \ldots, X_n) \in \mathcal{T}(\mathcal{C}^e)$, which means that C is not maximal anymore.

4. Computing the n-Concepts

4.1. Algorithm

Using Propositions 1, 2 and 3, we are able to propose an algorithm for computing $\mathcal{T}(\mathcal{C}^e)$ from $\mathcal{T}(\mathcal{C})$ and $\mathcal{C}_e$. First, for every height H corresponding to at least an n-concept, we compute $\mathcal{H}_H(\mathcal{C})$ and the $(n-1)$-context $(S_2, \ldots, S_n, \mathcal{P}(\mathcal{H}_H(\mathcal{C}), \mathcal{C}_e))$. For each such $(n-1)$-context, we compute its set of $(n-1)$-concepts. These $(n-1)$-concepts are then transformed into n-concept candidates by adding e to the height that generated their $(n-1)$-context. Finally, they are checked for maximality on their height and added to the new set of concepts if they have to be. The removal of old concepts that are no longer maximal occurs during the computation of their projection on $\mathcal{C}_e$.

Proposition 4. *Algorithm 1 ends and returns each n-concepts of $\mathcal{C}^e$ exactly once.*

Proof The dimensions of the contexts are finite. The set of n-concepts of an n-context is finite too and so is the number of possible heights. The algorithm goes through each height and projects each n-concept once. Thus, Algorithm 1 ends.

Proposition 2 ensures that every n-concept $(H \cup \{e\}, X_2, \ldots, X_n)$ is found when the loop processes the height H so every n-concept is found at least once. As an n-concept $(H \cup \{e\}, X_2, \ldots, X_n)$ can only be generated for the height H, every n-concept is added to the output only once. □

Algorithm 1 $NCremental(\mathcal{T}(\mathcal{C}), \mathcal{C}_e)$

Require: $\mathcal{T}(\mathcal{C})$ and an $(n-1)$-context $\mathcal{C}_e$
Ensure: $\mathcal{T}(\mathcal{C}^e)$
  $R \leftarrow \mathcal{T}(\mathcal{C})$
  for all different heights H **do**
    $P \leftarrow \emptyset$
    for each $C = (X_1, \ldots, X_n)$ in $\mathcal{H}_H(\mathcal{C})$ **do**
      $P \leftarrow P \cup \mathcal{P}(C, \mathcal{C}_e)$
      if $\mathcal{P}(C, \mathcal{C}_e) = \prod_{i=\{2,\ldots,n\}} X_i$ **then**
        $R \leftarrow R \setminus \{C\}$
      end if
    end for
    Compute $\mathcal{T}((S_2, \ldots, S_n, P))$
    for each $(X_2, \ldots, X_n)$ in $\mathcal{T}((S_2, \ldots, S_n, P))$ **do**
      if $(H \cup \{e\}, X_2, \ldots, X_n)$ is an n-concept **then**
        $R \leftarrow R \cup (H \cup \{e\}, X_2, \ldots, X_n)$
      end if
    end for
  end for
  return R

4.2. Complexity

At the time of writing, the only known bound for the number of n-concepts of an n-context $(S_1, \ldots, S_n, R)$ is $\prod_{i\in\{1,\ldots,n\}\setminus\{k\}} 2^{|S_i|}$ with $k = argmax_{k\in\{1,\ldots,n\}}|S_k|$. As per Proposition 3, an $(n-1)$-concept appearing in an $(n-1)$-context resulting from the projections of the n-concepts of a given height is always the width of some n-concept in $\mathcal{C}^e$. As the number of heights is bounded by both $2^{|S_1|}$ and $|\mathcal{T}(\mathcal{C})|$, such an $(n-1)$-concept cannot appear in more than $min(2^{|S_1|}, |\mathcal{T}(\mathcal{C})|)$ $(n-1)$-contexts. Hence, the algorithm will not manipulate more than $min(2^{|S_1|}, |\mathcal{T}(\mathcal{C})|) \times |\mathcal{T}(\mathcal{C}^e) \setminus \mathcal{T}(\mathcal{C})|$ $(n-1)$-concepts.

Computing $\mathcal{H}_H(\mathcal{C})$ for every possible H - i.e. regrouping concepts by height - can be done using a sorting algorithm in $O(|\mathcal{T}(\mathcal{C})| \times log|\mathcal{T}(\mathcal{C})|)$. Computing the projection of an n-concept $(X_1, \ldots, X_n)$ on $\mathcal{C}_e$ is done in $O(\prod_{i\in\{2,\ldots,n\}} |X_i|)$. The algorithm that computes $\mathcal{T}(\mathcal{C}^e)$ from $\mathcal{T}(\mathcal{C})$ and $\mathcal{C}_e$ is thus in $O(|\mathcal{T}(\mathcal{C})| \times (log|\mathcal{T}(\mathcal{C})| + \prod_{i\in\{2,\ldots,n\}} |S_i|) + L \times (K + |\mathcal{T}(\mathcal{C}^e) \setminus \mathcal{T}(\mathcal{C})| \times M))$ where L is the number of possible heights, K the complexity of computing the $(n-1)$-concepts of an $(n-1)$-context and M the complexity of checking whether an n-tuple is an n-concept.

Computing the n-concepts of an n-context from scratch can be done with Algorithm 1 by starting with a single n-concept $(\emptyset, S_2, \ldots, S_n)$ and adding the $|S_1|$ layers one by one. Hence, as adding a layer cannot reduce the number of concepts, computing $\mathcal{T}(\mathcal{C})$ from scratch is in $O(|S_1| \times Q)$ where Q is the complexity of Algorithm 1 with $|\mathcal{T}(\mathcal{C})|$ n-concepts as input. A 1-context (S_1, R) contains a single 1-concept that can be computed in $O(|S_1|)$.

As mentioned previously, the $(n-1)$-concepts manipulated by the algorithm are the widths of n-concepts so they are maximal. As such, only the height has to be tested for maximality before we decide whether an n-tuple is an n-concept. The complexity of doing so depends on the information available. If we know $\mathcal{C}$, deciding whether $(H, X_2, \ldots, X_n)$ is an n-concept can easily be done in $O(|S_1 \setminus X_1| \times \prod_{i \in \{2,\ldots,n\}} |X_i|)$ by checking whether a layer that is not in H contains the width. If we do not have $\mathcal{C}$, we can use the fact that $(H, X_2, \ldots, X_n)$ is an n-concept if and only if $H \setminus \{e\}$ is the greatest height (for the inclusion) for which $(X_2, \ldots, X_n)$ is an $(n-1)$-concept in $\mathcal{P}(\mathcal{H}_{H\setminus\{e\}}(\mathcal{C}), \mathcal{C}_e)$. By sorting the concepts by their heights H_i into any linear order such that $H_i \leq H_j \Leftrightarrow H_j \subseteq H_i$, a new n-concept is created as soon as its width is found for the first time.

We can check whether a concept $(X_2, \ldots, X_n)$ is already the width of a concept of greater height by storing old concepts in a hashtable structure. As the X_i can be represented as bit-vectors, they can be seen as integers from 0 to $2^{|X_i|}$. Hashtable structures can ensure an average $O(1)$ cost for the look-up of a concept. However, as the universe of keys is $2^{X_2} \times \cdots \times 2^{X_n}$, and the number of stored concept big, in order to keep the load factor under a certain threshold, one would need to allocate $O(2^{|X_2|} \times \cdots \times 2^{|X_n|})$ in memory space.

Proposition 5. *Algorithm 1 enumerates the elements of $\mathcal{T}(\mathcal{C}^e)$ in output-polynomial time.*

Proof The number of possible heights is less than $min(2^{|S_1|}, \mathcal{T}(\mathcal{C}))$. Checking whether an n-tuple is an n-concept can be done in time polynomial in the size of the n-context. From Proposition 3, we know that the number of $(n-1)$-concepts in an $(n-1)$-context resulting from projections is bound by the size of the output. As such, computing the n-concepts of an n-context can be done in output-polynomial time if and only if computing the $(n-1)$-concepts of the projections can also be done in output-polynomial time. Computing all the 1-concepts of a 1-context can be done in polynomial time so computing the 2-concepts of a 2-context can be done in output-polynomial time. Recursively, it is easy to see that computing the n-concepts of an n-context can also be done in output-polynomial time. □

As it is possible to have none of the $(n-1)$-concepts of the $(n-1)$-context corresponding to the projections of the n-concepts of a given height produce new n-concepts, the algorithm might have to consider $|\mathcal{T}(\mathcal{C}^e)| - |\mathcal{T}(\mathcal{C})|$ objects before adding to the output. Hence, the delay is output-polynomial.

5. Conclusion

Algorithm 1 computes the set of n-concepts of an n-context incrementally but does not allow for the retrieval of the order structure of the subjacent n-lattice. As new n-concepts are constructed by adding only a single element to the height of another, it is easy to retrieve a spanning tree of the covering relation of the partial-order $(X_1, \ldots, X_n) \leq_1 (Y_1, \ldots, Y_n) \Leftrightarrow X_1 \subseteq Y_1$. However, the same cannot be said for $\leq_i$, $\forall i \in \{2, \ldots, n\}$. While applications requiring only the spanning tree of $\leq_1$ are likely to exist, it would be interesting to be able to compute the whole lattice, either by modifying Algorithm 1 or through a new approach.

Acknowledgments

This research was partially supported by the European Union's "*Fonds Européen de Développement Régional (FEDER)*" program.

Bibliography

[1] K. Biedermann. An Equational Theory for Trilattices. *algebra universalis*, 42(4):253–268, 1999.

[2] Klaus Biedermann. Powerset Trilattices. In *Conceptual Structures: Theory, Tools and Applications, 6th International Conference on Conceptual Structures, ICCS '98, Montpellier, France, August 10-12, 1998, Proceedings*, pages 209–224, 1998.

[3] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Closed Patterns Meet n-ary Relations. *TKDD*, 3(1):3:1–3:36, 2009.

[4] Bernhard Ganter and Sergei A. Obiedkov. Implications in Triadic Formal Contexts. In *Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS 2004, Huntsville, AL, USA, July 19-23, 2004. Proceedings*, pages 186–195, 2004.

[5] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations.* Springer, 1999.

[6] Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. TRIAS - An Algorithm for Mining Iceberg Tri-lattices. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 907–911, 2006.

[7] Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, and Amedeo Napoli. Biclustering Meets Triadic Concept Analysis. *Ann. Math. Artif. Intell.*, 70(1-2):55–79, 2014.

[8] Fritz Lehmann and Rudolf Wille. A Triadic Approach to Formal Concept Analysis. In *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings*, pages 32–43, 1995.

[9] Gerd Stumme. Efficient Data Mining Based on Formal Concept Analysis. In *Database and Expert Systems Applications, 13th International Conference, DEXA 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, pages 534–546, 2002.

[10] Diana Troanca. Conceptual Visualization and Navigation Methods for Polyadic Formal Concept Analysis. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4034–4035, 2016.

[11] Petko Valtchev, Rokia Missaoui, and Robert Godin. Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges. In *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, pages 352–371, 2004.

[12] George Voutsadakis. Polyadic Concept Analysis. *Order*, 19(3):295–304, 2002.

[13] Di Wang and Jun Ma. Recommendation Based on Frequent N-adic Concepts. In *Web Technologies and Applications - 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings*, pages 318–330, 2014.