Three Related FCA Methods for Mining Biclusters of Similar Values on Columns

Mehdi Kaytoue, Victor Codocedo, Jaume Baixeries, Amedeo Napoli

Three Related FCA Methods for Mining Biclusters of Similar Values on Columns

Mehdi Kaytoue[1], Victor Codocedo[2], Jaume Baixieres[3], and Amedeo Napoli[2]

[1] Université de Lyon. CNRS, INSA-Lyon, LIRIS. UMR5205, F-69621, France.
[2] LORIA (CNRS - Inria Nancy Grand Est - Université de Lorraine), B.P. 239, F-54506, Vandœuvre-lès-Nancy.
[3] Universitat Politècnica de Catalunya. 08032, Barcelona. Catalonia.

Corresponding author: mehdi.kaytoue@insa-lyon.fr

Abstract. Biclustering numerical data tables consists in detecting particular and strong associations between both subsets of objects and attributes. Such biclusters are interesting since they model the data as local patterns. Whereas there exists several definitions of biclusters, depending on the constraints they should respect, we focus in this paper on *biclusters of similar values on columns*. There are several *ad hoc* methods for mining such biclusters in the literature. We focus here on two aspects: *genericity* and *efficiency*. We show that Formal Concept Analysis provides a mathematical framework to characterize them in several ways, but also to compute them with existing and efficient algorithms. The proposed methods, which rely on pattern structures and triadic concept analysis, are experimented and compared on two different datasets.

Keywords: biclustering, triadic concept analysis, pattern structure

1 Introduction

Biclustering has attracted a lot of attention for many years now, as it was used in an extensive way for mining biological data [7]. Given a data-table with objects as rows and attributes as columns, the goal is to find "sub-tables", or pairs of both subsets of objects and attributes, such that the values in the subtables respect well-defined constraints or maximize a given measure [17].

There exist several types of biclusters depending on the relation the values should respect. For example, *constant biclusters* are subtables with equal values [12, 6, 17]. *Biclusters with similar values on columns* (BSVC) are subtables where all values are pairwise similar for each column [4, 17]. The latter can also be generalized to *biclusters of similar values* (BSV): any two values in the subtable are similar [2, 3, 12, 21]. Dozens of algorithms, mostly *ad hoc*, have been proposed for computing the different types of biclusters. In this paper, we are interested in possible extensions of the Formal Concept Analysis (FCA) formalism for achieving the problem of biclustering. This comes with two goals: (i) formalizing and understanding biclusters formation and structure, and (ii) reusing existing algorithms for genericity purposes.

Actually, the present paper is in continuation with the work of the authors on the use of pattern structures –an extension of FCA for mining complex data [8, 12]– for discovering functional dependencies in a crisp and a fuzzy settings [1], and as well on the adaptation of pattern structures to a specific biclustering task: the discovery of biclusters of type BSV [6, 11]. Moreover, the biclustering task is usually considered as a "'two-dimensional" (2D) process where biclusters are rectangles in a table verifying some prior constraints. It was one main idea of [11] to transpose the problem in a "three-dimensional" setting by using and adapting triadic concept analysis [16] to the biclustering task.

Here we follow the same line and we propose a new approach for discovering biclusters in a numerical dataset where biclusters have "similar values" w.r.t. their columns (type BSVC). This works is a new attempt to extend the capabilities of FCA and of pattern structures, in dealing with the important problem of biclustering. Actually, biclustering can be also considered in a (pure) numerical setting, where it is sometimes called coclustering [18] and where kernel or spectral methods are often used for achieving the task. Here we keep the discrete setting and more precisely an FCA-based setting.

The rest of this paper is organized as follows. In Section 2 we formally introduce the biclustering problem. Then, we recall in Section 3 the FCA basics that are necessary for developing our three methods in Section 4. We experiment with these methods and compare them by processing two real-world datasets in Section 5 before concluding.

2 Problem Definition

We introduce the problem of mining biclusters of similar values on columns, or simply biclusters when no confusion can be made. A numerical dataset is defined as a many-valued context in which biclusters are denoted as pairs of object and attribute subsets for which a particular similarity constraint holds.

Definition 1 (Many-valued context and numerical dataset). *A many-valued context consists in a quadruple (G, M, W, I) where G is a set of objects, M a set of attributes, W a set of attribute values, and $I \subseteq G \times M \times W$ a ternary relation. An element $(g, m, w) \in I$, also written $m(g) = w$ or $g(m) = w$, can be interpreted as: w is the value taken by the attribute m for the object g. The relation I is such that $g(m) = w$ and $g(m) = v$ implies $w = v$.*

In the present work, W is a set of numbers and $\mathcal{K}_{num} = (G, M, W, I)$ denotes a numerical dataset, i.e. a many-valued context where W is a set of numbers.

Example. A tabular representation of a numerical dataset is given in Table 1: objects $G = \{g_1, g_2, g_3, g_4, g_5\}$ are represented by rows while attributes $M = \{m_1, m_2, m_3, m_4\}$ are represented by columns. $W = \{0, 1, 2, 6, 7, 8, 9\}$ and we have for example $g_2(m_4) = 9$.

	m_1	m_2	m_3	m_4
g_1	1	2	2	8
g_2	2	1	2	9
g_3	2	1	1	2
g_4	1	0	7	6
g_5	6	6	6	7

Fig. 1. A numerical dataset

Definition 2 (Biclusters with similar values on columns). *Given a numerical dataset (G, M, W, I), a pair (A, B) (where $A \subseteq G, B \subseteq M$) is called a bicluster of similar values on columns when the following statement holds:*

$$\forall g, h \in A, \forall m \in B, m(g) \simeq_\theta m(h)$$

where $\simeq_\theta$ is a similarity relation: $\forall w_1, w_2 \in W, \theta \in [0, max(W) - min(W)]$, $w_1 \simeq_\theta w_2 \iff |w_1 - w_2| \leq \theta$. A bicluster (A, B) is maximal if $\nexists g \in G \backslash A$ such that $(A \cup \{g\}, B)$ is a bicluster, and $\nexists m \in M \backslash B$ such that $(A, B \cup \{m\})$ is a bicluster.

Example. In Table 1, with $\theta = 1$, we have that $(A, B) = (\{g_1, g_2\}, \{m_1, m_2, m_3\})$ is a bicluster. Indeed, consider each attribute of B separately: the values taken by the objects A are pairwise similar. However, (A, B) is not maximal, since we have that both $(A \cup \{g_3\}, B)$ and $(A, B \cup \{m_4\})$ are also biclusters. Then, $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_1, g_2\}, \{m_1, m_2, m_3, m_4\})$ are both maximal.

Problem (Biclustering). Given a numerical dataset (G, M, W, I) and a similarity parameter θ, the goal of biclustering is to extract the set of all maximal biclusters (A, B) respecting the similarity constraint.

Remark. It should be noticed that in the formal definition, the similarity parameter is the same for all attributes. It is possible however to use a different parameter for each attribute without changing neither the problem definition or its resolution. For real-world datasets, one can choose different similarity parameters θ_m ($\forall m \in M$), but also can normalize/scale the attribute domains and use a single similarity parameter θ.

3 Basics on Formal Concept Analysis

In this paper, we show how our biclustering problem can be formalized and answered in FCA in different ways: (i) using standard FCA [9], (ii) using pattern structures [8], and (iii) using triadic concept analysis [16]. We recall below the basics of each approach.

Dyadic Concept Analysis. Let G be a set of objects, M a set of attributes and $I \subseteq G \times M$ be a binary relation. The fact $(g, m) \in I$ is interpreted as "g has attribute m". The two following derivation operators $(\cdot)'$ are defined:

$$A' = \{m \in M \mid \forall g \in A : gIm\} \qquad for\ A \subseteq G,$$
$$B' = \{g \in G \mid \forall m \in B : gIm\} \qquad for\ B \subseteq M$$

which define a Galois connection between the powersets of G and M. For $A \subseteq G$, $B \subseteq M$, a pair (A, B) such that $A' = B$ and $B' = A$, is called a *(formal) concept*. Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I). For a concept (A, B) the set A is called the *extent* and the set B the *intent* of the concept.

Triadic Concept Analysis. A triadic context is given by (G, M, B, Y) where G, M, and B are respectively called sets of objects, attributes and conditions, and $Y \subseteq G \times M \times B$. The fact $(g, m, b) \in Y$ is interpreted as the statement "Object g has the attribute m under condition b". A (triadic) concept of (G, M, B, Y) is a triple (A_1, A_2, A_3) with $A_1 \subseteq G$, $A_2 \subseteq M$ and $A_3 \subseteq B$ satisfying the two following statements: (i) $A_1 \times A_2 \times A_3 \subseteq Y$, $X_1 \times X_2 \times X_3 \subseteq Y$ and (ii) $A_1 \subseteq X_1$, $A_2 \subseteq X_2$ and $A_3 \subseteq X_3$ implies $A_1 = X_1$, $A_2 = X_2$ and $A_3 = X_3$. If (G, M, B, Y) is represented by a three dimensional table, (i) means that a concept stands for a 3-dimensional rectangle full of crosses while (ii) characterizes component-wise maximality of concepts. For a triadic concept (A_1, A_2, A_3), A_1 is called the extent, A_2 the intent and A_3 the modus. To derive triadic concepts, two pairs of derivation operators are defined. The reader can refer to [16] for their definitions which are not necessary for the understanding of the present work.

Pattern Structures. Let G be a set of objects, let $(D, \sqcap)$ be a meet-semilattice of potential object descriptions and let $\delta : G \longrightarrow D$ be a mapping. Then $(G, (D, \sqcap), \delta)$ is called a *pattern structure*. Elements of D are called *patterns* and are ordered by a subsumption relation $\sqsubseteq$ such that given $c, d \in D$ one has $c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$. Within the pattern structure $(G, (D, \sqcap), \delta)$ we can define the following derivation operators $(\cdot)^{\square}$, given $A \subseteq G$ and a description $d \in (D, \sqcap)$:

$$A^{\square} = \prod_{g \in A} \delta(g) \qquad d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\}$$

These operators form a Galois connection between $(\wp(G), \subseteq)$ and $(D, \sqsubseteq)$. *(Pattern) concepts* of $(G, (D, \sqcap), \delta)$ are pairs of the form (A, d), $A \subseteq G$, $d \in (D, \sqcap)$, such that $A^{\square} = d$ and $A = d^{\square}$. For a pattern concept (A, d), d is called a *pattern intent* and is the common description of all objects in A, called *pattern extent*. When partially ordered by $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2$ $(\Leftrightarrow d_2 \sqsubseteq d_1)$, the set of all concepts forms a complete lattice called a *(pattern) concept lattice*.

Computing Concepts and Concept Lattices. Processing a formal context in order to generate its set of concepts can be achieved by various algorithms (see [15] for a survey and a comparison, see also itemset mining [19]). For processing pattern structures, such algorithms generally need minor adaptations. Basically, one needs to override the code for (i) computing the intersection of any two arbitrary descriptions, and (ii) test the ordering between two descriptions. Processing a triadic context is however not so direct and can be done with nested FCA algorithms [10] or dedicated data-mining algorithm [5].

Similarity relations in FCA. The notion of similarity can be formalized by a tolerance relation: a symmetric, reflexive but not necessarily transitive relation. The similarity relation $\simeq_\theta$ used for defining biclusters of similar values is a tolerance. Given W a set of numbers, any maximal subset of pairwise similar values is called a block of tolerance.

Definition 3. *A binary relation $T \subseteq W \times W$ is called a* tolerance *relation if:*
(i) $\forall x \in W$ xTx (reflexivity)
(ii) $\forall x, y \in W$ $xTy \rightarrow yTx$ (symmetry)

Definition 4. *Given a set W, a subset $K \subseteq W$, and a tolerance relation T on W, K is a* block of tolerance *if:*
(i) $\forall x, y \in K\ xTy$ *(pairwise similarity)*
(ii) $\forall z \notin K, \exists u \in K\ \neg(zTu)$ *(maximality)*

It is shown that tolerance blocks can be obtained from the formal context of a tolerance relation [14]. In the context $(W, W, \simeq_\theta)$, one can characterize all blocks of tolerance K (and only them) as formal concepts (K, K).

4 Mining biclusters of similar values on columns in FCA

The basic notions of FCA of the previous section allow us now to answer our biclustering problem in various ways with: (i) an original method using interval pattern structure, (ii) a recently introduced method using partition pattern structures [6], and (iii) an original method relying on triadic concept analysis. We emphasize the genericity of FCA to answer a data mining problem.

4.1 Interval Pattern Structure Approach

For a dataset $\mathcal{K}_{num} = (G, M, W, I)$, an interval pattern structure $(G, (D, \sqcap), \delta)$ is defined as follows [13]: the objects from G are described by vectors of intervals, where each dimension gives a range of values for an attribute $m \in M$ (following a canonical ordering of the dimensions, i.e. dimension i corresponds to attribute $m_i \in M$). Then, for $m \in M$, the semi-lattice of intervals $(D_m, \sqcap_m)$ is given by:

$$\begin{aligned} D_m &= \{[w_1, w_2] \mid \exists g, h \in G \text{ s.t. } m(g) = w_1 \text{ and } m(h) = w_2\} \\ [a,b] \sqcap_m [c,d] &= [min(a,c), max(b,d)] \\ c \sqcap_m d = c &\iff c \sqsubseteq_m d \\ [a,b] \sqsubseteq_m [c,d] &\iff [c,d] \supseteq [a,b] \end{aligned}$$

The description space $(D, \sqcap)$ of the interval pattern structure is a product of meet-semi-lattices $(D, \sqcap) = \times_{m \in M}(D_m, \sqcap_m)$ which is a semi-lattice.

Examples. In Table 1, $(\{g_1, g_2, g_3\}, \langle [1,2], [1,2], [1,2], [2,9] \rangle)$ is a pattern concept:

$$\begin{aligned} \delta(g_1) &= \langle [1,1], [2,2], [2,2], [8,8] \rangle \\ \{g_1, g_2, g_3\}^\square = \delta(g_1) \sqcap \delta(g_2) \sqcap \delta(g_3) &= \langle [1,2], [1,2], [1,2], [2,9] \rangle \\ \langle [1,2], [1,2], [1,2], [8,9] \rangle \sqsubseteq \langle [1,2], [1,2], [1,2], [2,9] \rangle & \\ \{g_1, g_2, g_3\}^{\square\square} &= \{g_1, g_2, g_3\} \end{aligned}$$

We now give the intuitive idea on how the interval pattern concept lattice can be used to characterize the biclusters. Consider first the concept $(A_1, d_1) = (\{g_1, g_2\}, \langle [1,2], [1,2], [1,2], [8,9] \rangle)$. Consider also a function $attr : D \to M$ which returns for an interval pattern the set of attributes whose interval is not larger than the θ parameter, for $d = \langle [a_i, b_i] \rangle$, $i \in [1, |M|]$: $attr(d) = \{m_i \in M | a_i \simeq_\theta b_i\}$. $(A_1, attr(d_1)) = (\{g_1, g_2\}, \{m_1, m_2, m_3, m_4\})$ is a maximal bicluster. Consider the interval pattern concept $(A_2, d_2) = (\{g_1, g_2, g_3\}, \langle [1,2], [1,2], [1,2], [2,9] \rangle)$: $(A_2, attr(d_2)) = (\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ is a maximal bicluster (with $\theta = 1$). This means that biclusters can be characterized thanks to pattern concepts.

Proposition 1. *Consider a numerical dataset (G, M, W, I) as an interval pattern structure $(G, (D, \sqcap), \delta)$. For any maximal bicluster (A, B), there exists a pattern concept (A, d) such that $(A, B) = (A, attr(d))$.*

Proof. To ease reading, the proof is given in an appendix. □

4.2 Partition pattern structure approach

A partition pattern structure is a pattern structure instance where the description space is given by a semi-lattice of partitions over a set X [2]. Formally, we have $(G, (D, \sqcap), \delta)$ where: $D = Part(X)$ and $d_1 \sqcap d_2 = \bigcup p_i \cap p_j$ where $p_i, p_j \subseteq X$, $p_i \in d_1$, $p_j \in d_2$. The semi-lattice is actually a complete lattice of set partitions in which the bottom element is not considered. In [1], we showed that the definition of $\sqcap$, and equivalently $\sqsubseteq$, needs a slight modification when $D = 2^{2^K}$, i.e. a description $d \in D$ is a set of subsets of X, and they do cover X (possibly with overlapping). In that case, we have that $d_1 \sqcap d_2 = max(\bigcup p_i \cap p_j)$ where $p_i, p_j \subseteq X$, $p_i \in d_1$, $p_j \in d_2$ and $max(.)$ returns the maximal sets w.r.t. inclusion.

Now we show that such a pattern structure can be constructed from a numerical dataset, and that the corresponding concepts allow to generate all maximal biclusters. From a numerical dataset (G, M, W, I), we build the structure $(M, (D, \sqcap), \delta)$ where $D = 2^{2^G}$. The description of an object[4] $m \in M$ is given by: $\delta(m) = \{p_1, p_2, ...\}$ where $p_1, p_2, .. \subseteq G$ and:

$$
\begin{aligned}
m(g_1) \simeq_\theta m(g_2), \forall g_1, g_2 \in p_i \quad &\text{(similarity)} \\
\nexists g_3 \in G \backslash p_i \text{ with } m(g_3) \simeq_\theta m(g_k), \forall g_k \in p_i \quad &\text{(maximality)} \\
\bigcup_i p_i = G \quad &\text{(covering)}
\end{aligned}
$$

In other words, each original attribute $m \in M$ is described by a family of subsets of G, where each one corresponds to a block of tolerance w.r.t. the values of attribute m. Let $(A, d = \{p_i\})$ be a partition pattern concept, it is easy to see how the pairs $bic_i = (p_i, A)$ are biclusters with rows $g \in p_i$ and columns $m \in A$[5]. While any $bic_i = (p_i, A)$ is a bicluster, it is not necessarily a maximal bicluster. Nevertheless, maximal biclusters can be identified using the concept lattice.

Proposition 2. *Consider a pattern concept $(A, d = \{p_i\})$. The bicluster $bic_i = (p_i, A)$ is maximal if there is no pattern concept $(C, \{p_i, ...\})$ with $A \subseteq C$.*

Proof. The proof to this proposition is very intuitive. Recall from Section 2 that the bicluster (p_i, A) is maximal if two conditions are met, namely $\nexists g \in G \backslash p_i$ such that $(p_i \cup \{g\}, A)$ is a bicluster and $\nexists m \in M \backslash A$ such that $(p_i, A \cup \{m\})$ is

[4] Object in the pattern structure; attribute in the numerical dataset.

[5] In order to keep consistency with the previous notation, biclusters are written inversely as partition pattern concepts.

a bicluster, The first condition holds for bic_i given the maximality condition of the tolerance block p_i; The second follows from the proposition declaration. □

Example. The numerical dataset (G, M, W, I) given in Table 1 can be turned into a pattern structure as follows with $\theta = 1$:

$$\delta(m_1) = \{\{g_1, g_2, g_3, g_4\}\{g_5\}\} \quad \delta(m_2) = \{\{g_2, g_3, g_4\}\{g_1, g_2, g_3\}\{g_5\}\}$$
$$\delta(m_3) = \{\{g_1, g_2, g_3\}\{g_4, g_5\}\} \quad \delta(m_4) = \{\{g_4, g_5\}\{g_1, g_5\}\{g_1, g_2\}\{g_3\}\}$$

Indeed, each component of a description is a maximal set of objects having pairwise similar values for a given attribute. The pattern concept lattice is given in Figure 2. We remark that (i) any concept corresponds to a bicluster, (ii) some of them correspond to a maximal bicluster, and most importantly, (iii) any maximal bicluster can be found as a concept. For example, from the concept $(A_1, d_1) = (\{m_3, m_4\}, \{\{g_1, g_2\}, \{g_4, g_5\}, \{g_3\}\})$ we obtain the following biclusters: $bic_1 = (\{g_1, g_2\}, \{m_3, m_4\})$ and $bic_2 = (\{g_4, g_5\}, \{m_3, m_4\})$. Whereas bic_2 is a maximal bicluster bic_1 is not since we have that $(A_2, d_2) = (\{m_1, m_2, m_3, m_4\}, \{\{g_1, g_2\}, \{g_3\}, \{g_4\}, \{g_5\}\})$ with $(A_2, d_2) \leq (A_1, d_1)$. In turn, $bic_3 = (\{g_1, g_2\}, \{m_1, m_2, m_3, m_4\})$ is a maximal bicluster.

Remark. It is noticeable that an equivalent formal context can be built. By equivalent, we mean that the concept lattices produced by both structures are isomorphic. To obtain this formal context, we use a slight modification of the data transformation of [9] (pp. 92): $(M, \mathbb{B}_2(G), I)$ st. $(m, (g, h)) \in I \iff m(g) \simeq_\theta m(h)$. The concept lattice is equivalent to the pattern concept lattice [2], and thus it can be used in the same way to get maximal biclusters. In our running example, such context is given in Table 1, and its associated concept lattice is given in Figure 2 (right), a lattice isomorphic to the one raised from the pattern structure (left). The proof can be done in a similar manner as it is done in [2].

	(g_1, g_2)	(g_1, g_3)	(g_1, g_4)	(g_1, g_5)	(g_2, g_3)	(g_2, g_4)	(g_2, g_5)	(g_3, g_4)	(g_3, g_5)	(g_4, g_5)
m_1	×	×	×		×	×		×		
m_2	×	×			×	×		×		
m_3	×	×			×					×
m_4	×									×

Table 1. Formal context

4.3 Triadic Concept Analysis Approach

We present another original result: any maximal bicluster of similar values is characterized as a triadic concept. The triadic context is derived from the numerical dataset by encoding the tolerance relation between the values.

Proposition 3. *Given a numerical dataset (G, M, W, I), consider the derived triadic context given by (M, G, G, Y) s.t. $(m, g_1, g_2) \in Y \iff m(g_1) \simeq_\theta m(g_2)$.*

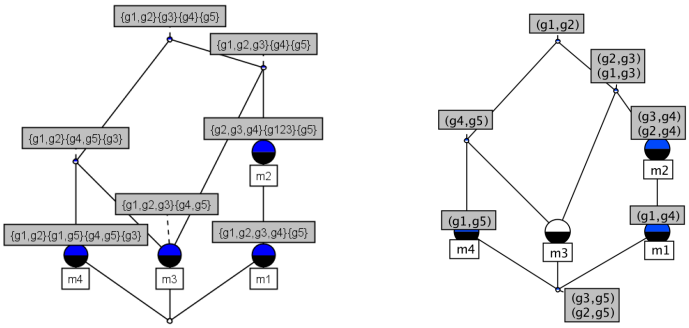

Fig. 2. Pattern concept lattice on the left side, concept lattice of the right side.

There is a one-to-one correspondence between the set of all maximal biclusters (A, B)*, the set of all triadic concepts* (B, A, A) *of the derived context.*

Proof. Consider a maximal bicluster (A, B). We have that $\forall g, h \in A : m(g) \simeq_\theta m(h) \iff m \in B$, if and only if (by the definition of Y) $(B, A, A) \subseteq Y$. We now take $(B', A', A') \subseteq Y$ such that $B \subseteq B'$ and $A \subseteq A'$. Since (A, B) is a maximal bicluster, we have that for any pair of objects $g, h \in A'$ and $m \in B'$ such that $g(m) \simeq_\theta h(m)$, implies that $g, h \in A$ and $m \in B$. Let (B, A, A) be a triadic concept. We have that for any pair of objects $g, h \in A$ and $m \in B$ we have that $g(m) \simeq_\theta h(m)$, this is, that $\forall g, h \in A : g(m) \simeq_\theta h(m) \iff m \in B$, which is the alternative definition of maximal bicluster. □

Example. Taking again $\theta = 1$, the triadic context derived from the numerical dataset from Table 1 is given in Table 2. An example of triadic concept is: $(\{m_3, m_2, m_1\}, \{g_1, g_3, g_2\}, \{g_1, g_2, g_3\})$ which is in turn the maximal bicluster $(\{g_1, g_3, g_2\}, \{m_3, m_2, m_1\})$.

5 Experiments

We experiment with the different FCA methods introduced in the previous section. We report preliminary results in two aspects: efficiency (running time) and compactness (number of concepts) to discuss the strengths and weaknesses of the different methods.

m_1	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×	×	
g_2	×	×	×	×	
g_3	×	×	×	×	
g_4	×	×	×	×	
g_5					×

m_2	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×		
g_2	×	×	×	×	
g_3	×	×	×	×	
g_4		×	×	×	
g_5					×

m_3	g_1	g_2	g_3	g_4	g_5
g_1	×	×	×		
g_2	×	×	×		
g_3	×	×	×		
g_4				×	×
g_5				×	×

m_4	g_1	g_2	g_3	g_4	g_5
g_1	×	×			×
g_2	×	×			
g_3			×		
g_4				×	×
g_5	×			×	×

Table 2. Triadic context derived from Table 1 thanks to $\simeq_1$.

Data and experimental settings. The first dataset, "Diagnosis"[6], contains 120 objects with 8 attributes. The first attribute provides temperature information of a given patient with a range $[35.5, 41.5]$ (numerical). For this attribute we used $\theta = 0.1$ and then $\theta = 0.3$. The other 7 attributes are binary ($\theta = 0$). The second dataset, "dataSample_1.txt", is provided with the BiCat software[7]. It contains 420 objects and 70 numerical attributes with range $[-5.9, 6.7]$. We used $\theta = 0.05$ for all attributes. We provide results in Table 3 for the three different FCA methods discussed in this article, namely interval pattern structure (IPS), tolerance blocks/partition pattern structures (TBPS) and triadic concept analysis (TCA). We also report on the use of standard FCA using the discretization technique discussed at the end of Section 4.2 (FCA). We also discuss the computing of clarified contexts, given that it can dramatically reduce the size of the context while keeping the same concept lattice (FCA-CL). A context is clarified when there exists neither two objects with the same description, or two attributes shared by the same set of objects.

For the methods based on FCA and pattern structures (IPS, TBPS), we used a C++ version of the AddIntent algorithm [20][8]. No restrictions were imposed over the size of the biclusters. The TCA method was implemented using DataPeeler [5]. All the experiments were performed using a Linux machine with Intel Xeon E7 running at 2.67GHz with 1TB of RAM.

Discussion. Results in Table 3 show that for the Diagnosis dataset, the clarified context using standard FCA (FCA-CL) is the best of the five methods w.r.t. execution time while for the BicAt sample 1, the best is TCA. Times are expressed as the sum of the time required to create the input representation of the dataset for the corresponding technique and its execution. In the case of FCA and FCA-CL, the pre-processing can be as high as the time required for applying the AddIntent algorithm. However, for large datasets such as the BicAt example, this times can be ignored. It is also worth noticing that the pre-processing depends on the chosen θ value, hence for each different θ configuration, a new pre-processing task has to be executed. This is not the case for interval and partition pattern structures the pre-processing of which is linear

[6] http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
[7] http://www.tik.ee.ethz.ch/sop/bicat/
[8] https://code.google.com/p/sephirot/

	Diagnosis				BicAt sample 1	
	$\theta = 0.3$		$\theta = 0.1$		$\theta = 0.05$	
Technique	Time [s] Preproc + Exec.	#Concepts	Exec. Time [s] Preproc + Exec.	#Concepts	Exec. Time [s] Preproc + Exec.	#Concepts
FCA	0.11 + 0.335	**98**	0.11 + 0.291	**88**	2.3 + 2,220	**476,950**
FCA-CL	**0.11 + 0.02**	**98**	**0.11 + 0.011**	**88**	2.3 + 2,220	**476,950**
TCA	0.04 + 33.3	3,322	0.04 + 31.34	2,127	**3.17 + 360**	741,421
IPS	0.011 + 0.303	928	0.001 + 0.178	301	0.02 + 2,340	722,442
TBPS	0.011 + 1.76	**98**	0.001 + 0.411	**88**	0.02 + 5,340	**476,950**

Table 3. Number of concepts and execution times (pre-processing + addIntent run)

w.r.t. the number of objects (it is actually, just a change of format). We can also appreciate a more compact representation of the biclusters by the use of partition pattern structures (TBPS) and its formal context versions (FCA and FCA-CL). While TBPS is the slowest of the five methods, it is also the cheapest one in terms of the use of machine resources, more specifically RAM. TCA is the more expensive method in terms of machine resources and data representation, however this yields results faster. Interval pattern structures are in the middle as a good trade-off of compactness and execution time.

For this initial experimentation we have not reported the number of maximal biclusters nor the bicluster extraction algorithms that can be implemented for each different technique, but only in the FCA techniques themselves. Regarding the number of maximal biclusters, this is the same for each technique since all of them are bicluster enumeration techniques, i.e. all possible biclusters are extracted. Hence, the difference among techniques is not given by the number of maximal biclusters extracted, but by the number of formal concepts found and their post-processing complexity to extract the maximal biclusters from them. In general, it is easy to observe from Propositions 1, 2 and 3 that the post-processing of TCA is linear w.r.t. the number of triadic concepts found, while for TPS is linear w.r.t. the number of interval pattern concepts times the number of columns of the numerical dataset squared and for TBPS is linear w.r.t. the number of *super-sub* concept relations in the tolerance block pattern concept lattice. Nevertheless, different strategies for bicluster extraction can be implemented for each technique rendering the comparison unfair. For example, in [6] an optimization is proposed regarding biclustering using partition pattern structures (which can be easily adapted to TBPS) which cuts in half its execution time by breaking the structure of the lattice. Similar strategies for IPS and TCA could also be implemented but are still a matter of research.

6 Conclusion

Biclustering is an important data analysis task that is used in several applications such as transcriptome analysis in biology and for the design of recommender systems. Biclustering methods produce a collection of local patterns that are easier to interpret than a global model. There are several types of biclusters and corresponding algorithms, *ad hoc* most of the time. In this paper, our main contribution shows how the *biclusters of similar values on columns* can be characterized or generated from formal concepts, pattern concepts and triadic concepts. Bringing back this problem of biclustering into formal concept analysis settings allows the usage of existing and efficient algorithms without any modifications. However, and this is among the perspectives of research, several optimizations can be made. For example, with the triadic method, one should not generate both concepts (A, B, C) and (A, C, B): they are redundant since only concepts with $B = C$ correspond to maximal biclusters.

References

1. J. Baixeries, M. Kaytoue, and A. Napoli. Computing similarity dependencies with pattern structures. In M. Ojeda-Aciego and J. Outrata, editors, *CLA*, volume 1062 of *CEUR Workshop Proceedings*, pages 33–44. CEUR-WS.org, 2013.
2. J. Baixeries, M. Kaytoue, and A. Napoli. Characterizing Functional Dependencies in Formal Concept Analysis with Pattern Structures. *Annals of Mathematics and Artificial Intelligence*, pages 1–21, Jan. 2014.
3. J. Besson, C. Robardet, L. D. Raedt, and J.-F. Boulicaut. Mining bi-sets in numerical data. In S. Dzeroski and J. Struyf, editors, *KDID*, volume 4747 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2007.
4. A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In P. E. Bourne, M. Gribskov, R. B. Altman, N. Jensen, D. A. Hope, T. Lengauer, J. C. Mitchell, E. D. Scheeff, C. Smith, S. Strande, and H. Weissig, editors, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, August 19-23, 2000, La Jolla / San Diego, CA, USA*, pages 75–85. AAAI, 2000.
5. L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *TKDD*, 3(1), 2009.
6. V. Codocedo and A. Napoli. Lattice-based biclustering using Partition Pattern Structures. In *21st European Conference on Artificial Intelligence (ECAI)*, 2014.
7. A. V. Freitas, W. Ayadi, M. Elloumi, J. Oliveira, J. Oliveira, and J.-K. Hao. *Biological Knowledge Discovery Handbook: Preprocessing, Mining, and Postprocessing of Biological Data*, chapter Survey on Biclustering of Gene Expression Data. John Wiley & Sons, Inc., 2013.
8. B. Ganter and S. O. Kuznetsov. Pattern structures and their projections. In *ICCS '01: Proceedings of the 9th International Conference on Conceptual Structures*, pages 129–142. Vol. 2120, Springer-Verlag, 2001.
9. B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.
10. R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Trias - an algorithm for mining iceberg tri-lattices. In *ICDM*, pages 907–911, 2006.
11. M. Kaytoue, S. O. Kuznetsov, J. Macko, and A. Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, 70(1-2), 2014.
12. M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Biclustering numerical data in formal concept analysis. In P. Valtchev and R. Jäschke, editors, *ICFCA*, volume 6628 of *LNCS*, pages 135–150. Springer, 2011.
13. M. Kaytoue, S. O. Kuznetsov, A. Napoli, and S. Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Information Science*, 181(10):1989–2001, 2011.
14. S. O. Kuznetsov. Galois connections in data analysis: Contributions from the soviet era and modern russian research. In B. Ganter, G. Stumme, and R. Wille, editors, *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 196–225. Springer, 2005.
15. S. O. Kuznetsov and S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Intell.*, 14(2-3):189–216, 2002.
16. F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In *ICCS*, volume 954 of *LNCS*, pages 32–43. Springer, 1995.
17. S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

18. N. Rogovschi, L. Labiod, and M. Nadif. A spectral algorithm for topographical co-clustering. In *IJCNN*, pages 1–6. IEEE, 2012.
19. T. Uno, M. Kiyomi, and H. Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In R. J. B. Jr., B. Goethals, and M. J. Zaki, editors, *FIMI*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
20. D. van der Merwe, S. Obiedkov, and D. Kourie. AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In P. Eklund, editor, *Concept Lattices*, volume 2961 of *LNCS*, pages 205–206. Springer, Berlin/Heidelberg, 2004.
21. R. Veroneze, A. Banerjee, and F. J. V. Zuben. Enumerating all maximal biclusters in real-valued datasets. *CoRR*, abs/1403.3562, 2014.

7 Appendix: Proof of proposition 1

We introduce notations, before to recall and prove Proposition 1 that relates maximal biclusters to interval pattern concepts of a pattern structure. The intuition lies in the relation between the set of attributes M of $(G, M, W, I))$ in an interval pattern structure $(G, (D, \sqcap), \delta)$. Let $d = \langle [a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n] \rangle \in D$ be a pattern interval in an interval pattern structure $(G, (D, \sqcap), \delta)$, where $|M| = n$. For any $m_i \in M$, we define: $d(m_i) = [a_i, b_i]$. and $|d(m_i)| = |a_i - b_i|$.

Definition 5. *Let d be a pattern in an interval pattern structure $(G, (D, \sqcap), \delta)$. The function* $\text{attr} : D \mapsto M$ *is defined as:* $\text{attr}(d) = \{m \in M \mid |d(m)| \leq \theta\}$.

Definition 6. *Let $A \subseteq G$ be a set of objects and $m \in M$ an attribute. We define: $A(m) = \{g(m) \mid g \in B\}$. For instance, in Table 1, if $A = \{g_1, g_2, g_3\}$, then, $A(m_4) = \{2, 8, 9\}$.*

Proposition 4. *For $A \subseteq G$, we have that, for all $m_i \in M$:*

$$A^{\square} = \langle [min(A(m_1)), max(A(m_1))], \ldots, [min(A(m_n)), max(A(m_n))] \rangle$$

Proof. Since the operation $\sqcap$ is associative and commutative, we have that

$$A^{\square} = \bigsqcap_{g_i \in A} g_i = \langle [min(A(m_1)), max(A(m_1))], \ldots, [min(A(m_n)), max(A(m_n))] \rangle$$

□

Now we reformulate and prove the Proposition 1.

Proposition 5. *Consider a numerical dataset (G, M, W, I) as an interval pattern structure $(G, (D, \sqcap), \delta)$. For any maximal bicluster (A, B), we define: $d = A^{\square}$. Then: 1. $B = \text{attr}(d)$ and 2. (A, D) is a pattern concept in $(G, (D, \sqcap), \delta)$.*

Proof. 1. $B = \text{attr}(d)$. We prove that $m \in \text{attr}(b) \leftrightarrow m \in B$. Since $B = A^{\square}$, then, by the definition of maximal bicluster we have that $\forall m \in M : m \in B \leftrightarrow |A(m)| \leq \theta$, if and only if $|min(A(m)) - max(A(m))| \leq \theta$ if and only if (by the definition of d) $m \in \text{attr}(d)$. □

2. We need to prove that $A = d^{\square}$ and that $A^{\square} = d$. $A^{\square} = d$ holds by the definition of d. As for $A = d^{\square}$, we take $g \in d^{\square}$, which means that $\forall m \in M : g(m) \in d(m)$, also if $m \in B$, which implies that $g \in A$ by definition of maximal bicluster.