Lattice-based biclustering using Partition Pattern Structures

Victor Codocedo, Amedeo Napoli

Lattice-based biclustering using Partition Pattern Structures

Victor Codocedo[1] and **Amedeo Napoli**[1]

Abstract. In this work we present a novel technique for exhaustive bicluster enumeration using formal concept analysis (FCA). Particularly, we use pattern structures (an extension of FCA dealing with complex data) to mine similar row/column biclusters, a specialization of biclustering when attribute values have coherent variations. We show how biclustering can benefit from the FCA framework through its robust theoretical description and efficient algorithms. Finally, we evaluate our bicluster mining approach w.r.t. a standard biclustering technique showing very good results in terms of bicluster quality and performance.

1 Introduction

Biclustering has become a fundamental tool for bioinformatics and gene expression analysis [4]. Different from standard clustering where objects are compared and grouped together based on their full descriptions, biclustering generates groups of objects based on a subset of their attributes, values or conditions. Thus biclusters are able to represent object relations in a local scale instead of the global representation given by an object cluster [12].

In this sense, biclustering has many elements in common with Formal Concept Analysis (FCA) [6]. In FCA objects are grouped together by the attributes they share in what is called a formal concept. Furthermore, formal concepts are arranged in a hierarchical and overlapping structure denominated a concept lattice. Hence a formal concept can be considered as a bicluster of objects and attributes representing relations in a local scale, while the lattice structure gives a description in the global scale.

FCA is not only analogous to biclustering, but has much to offer in terms of mining techniques and algorithms [10]. The concept lattice can also provide biclusters with an overlapping hierarchy which has been reported as an important feature for bicluster analysis [15]. Recently, some approaches considering the use of FCA algorithms to mine biclusters from a numerical data-table have been introduced showing good potential [8, 7]. In this work, we present a novel technique for lattice-based biclustering using the pattern structure framework [5], an extension of FCA to deal with complex data. More specifically, we propose a technique for mining biclusters with similar row/column values, a specialization of biclustering focused on mining attributes with coherent variations, i.e. the difference between two attributes is the same for a group of objects [12]. We show that, by the use of partition pattern structures [1], we can find high quality maximal biclusters (w.r.t. the mean squared error). Finally, we compare our approach with a standard constant row value algorithm [3], showing the capabilities and limitations of our approach.

In the remainder of this paper we introduce some background theory which supports our work in Section 2. Section 3 contains the formalization and description of our biclustering technique. Section 4 presents the experiments and initial findings of our approach. In Section 5 we discuss the state-of-the-art related to lattice-based biclustering. Finally, Section 6 concludes our article and presents some new perspectives of research.

2 Background Knowledge

2.1 Formal Concept Analysis

The basics of FCA are introduced in [6], but we recall some useful notions for the understanding of the paper. Let a formal context $\mathcal{K} = (\mathtt{G}, \mathtt{M}, \mathtt{I})$ be. a binary table where $\mathtt{G}$ is a set of objects, $\mathtt{M}$ a set of attributes, and $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M}$ an incidence relation indicating by $\mathtt{gIm}$ that the object $\mathtt{g}$ has the attribute $\mathtt{m}$. For $\mathtt{A} \subseteq \mathtt{G}$ and $\mathtt{B} \subseteq \mathtt{M}$, two derivation operators $(\cdot)'$ are defined as follows:

$' : \wp(\mathtt{G}) \longrightarrow \wp(\mathtt{M})$ with $\mathtt{A}' = \{\mathtt{m} \in \mathtt{M} \mid \forall \mathtt{g} \in \mathtt{A}, \mathtt{gIm}\}$

$' : \wp(\mathtt{M}) \longrightarrow \wp(\mathtt{G})$ with $\mathtt{B}' = \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{B}, \mathtt{gIm}\}$,

where $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ respectively denote the powersets of $\mathtt{G}$ and $\mathtt{M}$. The two derivation operators $'$ form a Galois connection[2] between $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ [6]. For a set of objects $\mathtt{A}$, $\mathtt{A}'$ is the set attributes which are common to all objects in $\mathtt{A}$. Analogously, for a set of attributes $\mathtt{B}$, $\mathtt{B}'$ is the set of objects having all attributes in $\mathtt{B}$. A *formal concept* is defined as a pair $(\mathtt{A}, \mathtt{B})$ where $\mathtt{A} \subseteq \mathtt{G}$, $\mathtt{B} \subseteq \mathtt{M}$, $\mathtt{A}' = \mathtt{B}$ and $\mathtt{B}' = \mathtt{A}$, $\mathtt{A}$ being the *extent* and $\mathtt{B}$ the *intent* of the formal concept.

The maximal sets of objects which are related to the maximal sets of attributes correspond to closed sets of the composition of both operators $'$, denoted $''$, for $\wp(G)$ and $\wp(M)$ respectively. The set $\mathfrak{B}(\mathtt{G}, \mathtt{M}, \mathtt{I})$ of all concepts from $\mathcal{K}$ is ordered by extent inclusion, denoted by $\leq_{\mathcal{K}}$, i.e. $(\mathtt{A}_1, \mathtt{B}_1) \leq_{\mathcal{K}} (\mathtt{A}_2, \mathtt{B}_2)$ when $\mathtt{A}_1 \subseteq \mathtt{A}_2$ (or dually $\mathtt{B}_2 \subseteq \mathtt{B}_1$). In this case we say that $(\mathtt{A}_1, \mathtt{B}_1)$ is the super-concept of $(\mathtt{A}_2, \mathtt{B}_2)$ and inversely, $(\mathtt{A}_2, \mathtt{B}_2)$ is the sub-concept of $(\mathtt{A}_1, \mathtt{B}_1)$. The *concept lattice* of $\mathcal{K}$ is denoted by $\underline{\mathfrak{B}}(\mathtt{G}, \mathtt{M}, \mathtt{I})$. For example, consider the formal context

[1] Laboratoire lorrain de recherche en informatique et ses applications (LORIA). Nancy, France.
victor.codocedo@inria.fr,amedeo.napoli@inria.fr

[2] A Galois connection is based on a dual adjunction between partially ordered sets.

in Table 1 containing 4 objects ($\mathtt{g}_1 - \mathtt{g}_4$) and 5 attributes ($\mathtt{m}_1 - \mathtt{m}_5$). The crosses in the formal context indicates the relations between objects and attributes (e.g. object $\mathtt{g}_1$ has attribute $\mathtt{m}_1$). Coloured cells show a maximal rectangle which corresponds to the formal concept with extent $\{\mathtt{g}_1, \mathtt{g}_2\}$ and $\{\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3\}$. The concept lattice in Figure 1 illustrates all the formal concepts found in the example.

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$	$\mathtt{m}_5$
$\mathtt{g}_1$	×	×	×		
$\mathtt{g}_2$	×	×	×	×	
$\mathtt{g}_3$		×	×	×	×
$\mathtt{g}_4$	×			×	×

Table 1: Formal context

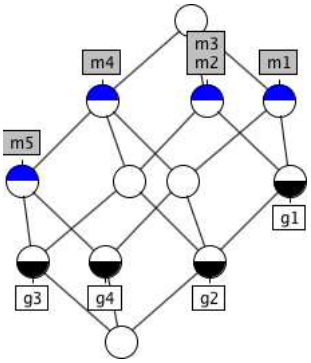

Figure 1: Concept Lattice.

The *object intent* is defined as $\mathtt{g}' = \{\ \mathtt{m} \in \mathtt{M} \mid \mathtt{gIm}\ \}$ for an object $\mathtt{g} \in \mathtt{G}$. Correspondingly, for $\mathtt{m} \in \mathtt{M}$, the *attribute extent* is defined as $\mathtt{m}' = \{\mathtt{g} \in \mathtt{G} | \mathtt{gIm}\}$. For a given object $\mathtt{g}$, the *object concept* is defined by $\gamma(\mathtt{g}) = (\mathtt{g}'', \mathtt{g}')$ (where $\mathtt{g}''$ stands for $(\mathtt{g}')'$). Dually, for a given attribute $\mathtt{m}$, the *attribute concept* is $\mu(\mathtt{m}) = (\mathtt{m}', \mathtt{m}'')$. Intuitively, the *object concept* is the smallest-extent concept in the lattice which includes the object. The "reduced notation" of a concept lattice consists on labelling object and attribute concepts with their respective objects/attributes. The concept lattice in 1 shows a concept lattice in reduced notation.

2.2 Biclustering

A numerical data-table is a matrix $\mathcal{M}$ where $\mathcal{M}_{ij}$ indicates the value of an object $\mathtt{g}_i \in \mathtt{G}$ w.r.t. the attribute $\mathtt{m}_j \in \mathtt{M}$ with $i \in [1..|\mathtt{G}|]$ and $j \in [1..|\mathtt{M}|]$ ($|\cdot|$ represents set cardinality). A bicluster of $\mathcal{M}$ is a submatrix $\mathcal{B}$ where each value $\mathcal{B}_{ij}$ satisfies a given restriction. According to [4, 12], there are five different restrictions which we summarize in Table 2.

Constant values	$\mathcal{B}_{ij} = c$	Within the submatrix, all values are equal to a constant $c \in \mathbb{R}$ ($\mathbb{R}$ indicates real values).
Constant row values	$\mathcal{B}_{ij}{=}c + \alpha_i$	Within the submatrix, all the values in a given row i are equal to a constant c and a row adjustment $\alpha_i \in \mathbb{R}$.
Constant column values	$\mathcal{B}_{ij}{=}c + \alpha_j$	Within the submatrix, all the values in a given column j are equal to a constant c and a column adjustment $\alpha_j \in \mathbb{R}$.
Coherent values	$\mathcal{B}_{ij}{=}$ $c + \alpha_i + \beta_j$	Within the submatrix, all the values in a given column j are equal to a constant c, a row adjustment α_i and a column adjustment β_j. Instead of addition, the model can also consider multiplicative factors.
Coherent evolution		Values in the submatrix induce a linear order.

Table 2: Types of biclusters.

2.2.1 *Similar values instead of constant values*

When noise is present in a data-table, it is difficult to search for constant values. Several approaches have tackled this issue in different ways, e.g. by the use of evaluation functions [14], equivalence relations [2, 13] and tolerance relations [7]. The most common way is establishing a threshold $\theta \in \mathbb{R}$ to enable the similarity comparison of two different values $w_1, w_2 \in \mathbb{R}$. We say that $w_1 \simeq_\theta w_2$ (values are similar) iff $|w_1 - w_2| \leq \theta$. Thus, constant values are a special case of similar values when $\theta = 0$. Using this, we can redefine the first three types of biclusters as follows:

1. Similar values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kl}$.
2. Similar row/column values:
   (a) Similar row values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{il}$.
   (b) Similar column values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kj}$.

Example 1. *With $\theta = 1$, Table 3 shows in its upper left corner a bicluster with similar values (dark grey). The upper right corner represents a similar column bicluster (light grey). Lower left corner considering $\{\mathtt{g}_3, \mathtt{g}_4\}$ and $\{\mathtt{m}_1, \mathtt{m}_2\}$ (not marked in the table) represents a similar row bicluster.*

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$	$\mathtt{m}_5$
$\mathtt{g}_1$	1	2	2	1	6
$\mathtt{g}_2$	2	1	1	0	6
$\mathtt{g}_3$	2	2	1	7	6
$\mathtt{g}_4$	8	9	2	6	7

Table 3: Bicluster with similar values ($\theta = 1$).

	$\mathtt{m}_1$	$\mathtt{m}_3$
$\mathtt{g}_2$	2	1
$\mathtt{g}_3$	2	1

Table 4: Constant column bicluster.

2.3 The ties between FCA and biclustering

A bicluster $\mathcal{B}_{ij}$ has a direct correspondence with a set of objects $\mathtt{A}$ and a set of attributes $\mathtt{B}$, where $\mathtt{A} = \{\mathtt{g}_i\}$ and $\mathtt{B} = \{\mathtt{m}_j\}$ with i, j being rows and columns respectively in $\mathcal{B}_{ij}$. We can observe that the pair $(\mathtt{A}, \mathtt{B})$ resembles a formal concept. Actually, the ties between FCA and biclustering go beyond. If we consider the formal context to be a numerical data-table where crosses represent the value 1 and empty cells represent the value 0, then a formal concept is actually a constant value bicluster (where $\mathcal{B}_{ij} = 1$). Moreover, the concept lattice provides an overlapping hierarchy of a set of constant value biclusters [15], however incomplete as FCA can only deal with binary data, i.e. the biclusters where $\mathcal{B}_{ij} = 0$ are not present.

Usually, in FCA there are several techniques that allow data encoding to binary values by means of scaling [6, 7]. Nevertheless, this process introduces more parameters to the process [2] and increment its complexity [9, 1]. In the following, we discuss and propose a technique to overcome these issues.

3 Biclustering using partitions

Different from biclusters with constant values, constant columns tackle the problem of "attribute expression" throughout different objects. For example, consider that Table 3 represents the users (rows) ratings of a set of movies (columns).

Intuitively, we would like to find users which give the same rating for a given movie (constant column) while letting ratings be different across different movies (non-constant row). These kind of biclusters would contain users with the same taste in cinema.

Now, let us consider as an example the movie $\mathtt{m}_3$ (attribute) in Table 3. We can observe that ratings (values) for this movie "breaks" the set of users (objects) in two sets, namely $\{\mathtt{g}_1, \mathtt{g}_4\}$ (rated with value 2) and $\{\mathtt{g}_2, \mathtt{g}_3\}$ (rated with value 1). If we take a second movie, for example $\mathtt{m}_5$ we can search for the coincidences of how both movies "breaks" the space of users, which generates the subsets $\{\mathtt{g}_1\}, \{\mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_4\}$. In particular, we can see how the pair $(\{\mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{m}_3, \mathtt{m}_5\})$ corresponds to a bicluster with constant columns as depicted in Table 4.

Our approach for mining biclusters with similar row/column values is based on this "breaking" or "partitioning technique". More specifically, it relies on partitions patterns found in a numerical data-table through equivalence relations and FCA algorithms. The hierarchical and overlapping structure supporting the biclusters is just a consequence of the lattice-based modelling provided by FCA.

3.1 Formalizations

A partition $\mathtt{d} = \{p_i\}$ of a set $\mathtt{G}$ can be formalized as a collection of components p_i such as:

$$\bigcup_{p_i \in \mathtt{d}} p_i = \mathtt{G} \qquad p_i \cap p_j = \emptyset \; ; (p_i, p_j \in \mathtt{d}, i \neq j)$$

The space of all partitions is denoted as $\mathtt{D}$ [1]. A partition $\mathtt{d}_1 = \{p_i\}$ is a refinement of $\mathtt{d}_2 = \{p_j\}$ (or $\mathtt{d}_2$ is a coarsening of $\mathtt{d}_1$) iff $\forall\ p_i \in \mathtt{d}_1, \exists\ p_j \in \mathtt{d}_2,\ p_i \subseteq p_j$. A partition of $\mathtt{G}$ can be created from an equivalence relation $[\mathtt{g}]$. An equivalence relation is a reflexive, symmetric and transitive binary relation between elements in a set. For example, we can define the equivalence relation $[\mathtt{g}_i]_{\mathtt{m}_j}$ of an object w.r.t. an attribute as follows:

$$[\mathtt{g}_i]_{\mathtt{m}_j} = \{\mathtt{g}_k \in \mathtt{G} \mid \mathcal{M}_{ij} \simeq_\theta \mathcal{M}_{kj}\} \tag{1}$$

Where $\mathcal{M}_{ij} \simeq_\theta \mathcal{M}_{kj} \iff |\mathcal{M}_{ij} - \mathcal{M}_{kj}| \leq \theta$ is a similarity relation as defined in Section 2.2. This allows us to create a partition mapping $\delta : \mathtt{M} \to \mathtt{D}$ which assigns an attribute with a partition over $\mathtt{G}$ such as:

$$\delta(\mathtt{m}_j) = \{[\mathtt{g}_i]_{m_j} \mid \mathtt{g}_i \in \mathtt{G}\} \tag{2}$$

Example 2. *From Table 3 we have* $[\mathtt{g}_1]_{\mathtt{m}_4} = [\mathtt{g}_2]_{\mathtt{m}_4} = \{\mathtt{g}_1, \mathtt{g}_2\}$, $[\mathtt{g}_3]_{m_4} = [\mathtt{g}_4]_{\mathtt{m}_4} = \{\mathtt{g}_3, \mathtt{g}_4\}$ *and* $\delta(\mathtt{m}_4) = \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}$.

As described in the beginning of this section, using partition operations (searching for coincidences) we can discover biclusters with constant column values (or similar column values, using equivalence relations). In the following, we define these operations in the space of partitions $\mathtt{D}$. We show that $\mathtt{D}$ is an ordered space. We also show that the space of attributes $\mathtt{M}$ and the space of partitions $\mathtt{D}$ are related through a Galois connection [10, 6] that can be exploited in order to mine all possible bicluster pairs with constant column values from a numerical data-table reusing the algorithmic machinery of FCA.

3.2 Partition Space

The space of all partitions $\mathtt{D}$ is a complete lattice [1, 13] where, for two elements $\mathtt{d}_1 = \{p_i\}$, $\mathtt{d}_2 = \{p_j\}$ with $i \in [1, |\mathtt{d}_1|]$ and $j \in [1, |\mathtt{d}_2|]$, the meet and join are defined by Equations 3 and 4 and the order between two partitions is determined by Equation 5.

$$\mathtt{d}_1 \sqcap \mathtt{d}_2 = \bigcup p_i \cap p_j \tag{3}$$

$$\mathtt{d}_1 \sqcup \mathtt{d}_2 = (\bigcup_{p_i \cap p_j \neq \emptyset} p_i \cup p_j)^+ \tag{4}$$

$$\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1 \tag{5}$$

Where we use $(\cdot)^+$ to denote closure for $\mathtt{d} \subseteq \wp(\mathtt{G})$ with components $p_i \subseteq \mathtt{G}$ such as:

$$\mathtt{d}^+ = \{p_i \in \mathtt{d} \mid \nexists p \in \mathtt{d}, p_i \subseteq p\}$$

Intuitively, this means that the closure only conserves the maximal components in $\mathtt{d}$ not included in any other component. The meet of two partitions corresponds to the coarsest common refinement of two partitions. As we will describe after, this is the operation that allows us to enumerate all the possible biclusters. The join, on the other hand, represents the finest coarsening of two partitions. We do not use it for practical purposes in this work. The order between partitions establishes a hierarchical structure in the space $\mathtt{D}$ where coarser partitions subsumes finer partitions.

Example 3. *From Table 3, with* $\delta(\mathtt{m}_1) = \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3\}, \{\mathtt{g}_4\}\}$, $\delta(\mathtt{m}_4) = \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}$ *and* $\delta(\mathtt{m}_5) = \{\{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}\}$ *we have:*

$$\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) = (\{\mathtt{g}_1 - \mathtt{g}_4\} \cap \{\mathtt{g}_1, \mathtt{g}_2\}) \cup (\{\mathtt{g}_1 - \mathtt{g}_4\} \cap \{\mathtt{g}_3, \mathtt{g}_4\})$$
$$\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) = \{\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}$$
$$\delta(\mathtt{m}_4) \sqcap \delta(\mathtt{m}_5) = \delta(\mathtt{m}_4) \iff \delta(\mathtt{m}_4) \sqsubseteq \delta(\mathtt{m}_5)$$
$$\delta(\mathtt{m}_1) \sqcup \delta(\mathtt{m}_4) = \{\{\mathtt{g}_1 - \mathtt{g}_3\}, \{\mathtt{g}_1 - \mathtt{g}_4\}, \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_4\}, \{\mathtt{g}_3, \mathtt{g}_4\}\}^+$$
$$= \{\mathtt{g}_1, \mathtt{g}_2, \mathtt{g}_3, \mathtt{g}_4\}$$

3.3 Partition Pattern Structures

The pattern structure framework is an extension of FCA proposed to deal with complex data [5]. Partition pattern structures are an instance of the pattern structure framework proposed to mine functional dependencies among attributes of a database [1] dealing with set partitions. In the following, we provide the specifics of partition pattern structures where the main definitions are given in [5].

Let $\mathtt{G}$ be a set of objects, $\mathtt{M}$ a set of attributes and $\mathcal{M}$ a data-table of numerical values where $\mathcal{M}_{ij}$ contains the value of attribute $\mathtt{m}_j \in \mathtt{M}$ in object $\mathtt{g}_i \in \mathtt{G}$. The lattice of all partitions of $\mathtt{G}$ is determined by $(\mathtt{D}, \sqcap)$ wherein a partition mapping function $\delta : \mathtt{M} \to \mathtt{D}$ assigns an attribute w.r.t. an equivalence relation $[\mathtt{g}_i]_{\simeq_\theta} : \mathtt{G} \to \wp(\mathtt{G})$ as defined in Equation 1. Then, a *partition pattern structure* is determined by the triple $(\mathtt{M}, (\mathtt{D}, \sqcap), \delta)$ in which the following derivation operators for $\mathtt{B} \subseteq \mathtt{M}$ and $\mathtt{d} \in \mathtt{D}$ are defined:

$$\mathtt{B}^{\square} = \prod_{\mathtt{m} \in \mathtt{B}} \delta(\mathtt{m}) \tag{6}$$

$$\mathtt{d}^{\square} = \{\mathtt{m} \in \mathtt{M} \mid \mathtt{d} \sqsubseteq \delta(\mathtt{m})\} \tag{7}$$

Similarly to standard FCA, we have that $(\mathtt{B}, \mathtt{d})$ is a partition pattern concept (pp-concept) when $\mathtt{B}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{B}$ and that for two pp-concepts $(\mathtt{B}_1, \mathtt{d}_1)$ and $(\mathtt{B}_2, \mathtt{d}_2)$, the order between them is given by $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2) \iff (\mathtt{B}_1 \subseteq \mathtt{B}_2)$ or $(\mathtt{d}_2 \sqsubseteq \mathtt{d}_1)$. When $\mathtt{B}^{\square\square} = \mathtt{B}$, we ensure the maximality of pp-concept $(\mathtt{B}, \mathtt{d})$ and we call it a pp-concept. Pp-concepts determines biclusters as pairs $(p, \mathtt{B})$ where p is a component of the partition pattern. It should be noticed that to keep consistency with previous notation, we write biclusters as pairs $(p, \mathtt{B})$ (p represent rows and $\mathtt{B}$ represent columns), while pp-concepts are written inversely $(\mathtt{B}, \mathtt{d})$ ($\mathtt{B}$ is the extent and $\mathtt{d}$ is the intent of $(\mathtt{B}, \mathtt{d})$).

Proposition 1. *Let $(\mathtt{B}, \mathtt{d})$ be a pp-concept, then for any partition component $p \in \mathtt{d}$ each pair $(p, \mathtt{B})$ corresponds to a similar column value bicluster.*

The proof of this proposition is easy considering that each pair $(p, \mathtt{B})$ represents a submatrix the columns of which were selected using an equivalence relation, i.e. the values in the columns are similar w.r.t. $\simeq_\theta$.

We say that a bicluster $(p, \mathtt{B})$ is maximal iff adding an object to p or an attribute to $\mathtt{B}$ does not result in a bicluster, i.e. $(p \cup \{\mathtt{g}\}, \mathtt{B})$ and $(p, \mathtt{B} \cup \{\mathtt{m}\})$ are not biclusters.

Example 4. *Figure 2 shows a partition pattern concept lattice (pp-lattice) created from Table 3 using $\theta = 1$. The pp-lattice contains 4 pp-concepts, but they correspond to 6 maximal similar column value biclusters listed in Table 5.*

While pp-concepts are maximal (closed under $(\cdot)^{\square}$), biclusters corresponding to pairs $(p, \mathtt{B})$ are not always maximal. For instance, in Table 5 we have that bicluster 7 is not maximal because of bicluster 3. The same happens with bicluster 8 which is not maximal because of 4. This is due to the fact that pp-concepts are maximal w.r.t. the partitions and not w.r.t. the individual components of those partitions. Nevertheless, maximal biclusters are still easy to identify.

Proposition 2. *Let $(\mathtt{B}_1, \mathtt{d}_1), (\mathtt{B}_2, \mathtt{d}_2)$ be two pp-concepts such as $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2)$. Let $p \subseteq \mathtt{G}$ be a component of a partition. If $p \in \mathtt{d}_1$ and $p \notin \mathtt{d}_2$ then the bicluster corresponding to $(p, \mathtt{B}_1)$ is maximal.*

Proof. Given definitions in Equations 1, 6 and 7, we have that for $(\mathtt{B}_1, \mathtt{d}_1)$ and for any $\mathtt{g}_i \in p$, the following is true:

$$p = \bigcap_{\mathtt{m}_j \in \mathtt{B}_1} \{\mathtt{g}_k \in \mathtt{G} \mid \mathcal{M}_{ij} = \mathcal{M}_{kj}\} \tag{8}$$

Consequently, for any other object $\mathtt{g}_h \in \mathtt{G}$, such as $\mathtt{g}_h \notin p$, we have $\mathcal{M}_{ij} \neq \mathcal{M}_{hj}$. Hence, the pair $(p + \{\mathtt{g}_h\}, \mathtt{B})$ cannot be a bicluster.

Let $\mathtt{B}_2 = \mathtt{B}_1 + \{\mathtt{m}_j\}$ for any $\mathtt{m}_j \in \mathtt{M}$, we show that $(p, \mathtt{B}_2)$ cannot be a cluster by contradiction. Let $(p, \mathtt{B}_2)$ be a bicluster. Then, there exists the pp-concept $(\mathtt{B}_2, \mathtt{B}_2^{\square})$ such as $p \in \mathtt{B}_2^{\square}$. If it does, then it is necessarily a direct super concept of $(\mathtt{B}_1, \mathtt{d}_1)$. However, this contradicts the definition $p \notin \mathtt{B}_2^{\square}$. □

Intuitively, maximal biclusters can be found in a kind of "attribute concept" of the corresponding partition component (see Section 2.1) (e.g. consider the maximal bicluster 6 in Table 5. It corresponds to the pp-concept labelled with $\mathtt{m}_4$ in Figure 2 and we can appreciate that $\{\mathtt{g}_3, \mathtt{g}_4\}$ is not present in the intent of its superconcept. The opposite happens with bicluster 7 in the concept labelled with $\{\mathtt{m}_1, \mathtt{m}_2\}$ in the lattice.). Indeed, partition pattern structures resembles a standard FCA process where the attribute set is multi-dimensional. In [1], it is shown how, through the use of transitive closures, the partition pattern structure is isomorphic to a binary formal context where the "scaling" procedure (encoding from many-valued data) generates a quadratic number of objects. In general, this cannot be applied on real datasets. Modelling a numerical dataset as a pattern structure is regarded as the best way of proceeding in these kind of situations [5, 9].

3.4 Mining biclusters

The calculation of the pp-concepts was implemented using AddIntent (the algorithm is described in [16]) for calculating a lattice of formal concepts. The algorithm was modified in order to obtain maximal biclusters from a numerical datatable. An important step to the process is the calculation of $\delta(\mathtt{m}_j)$ for a given $\mathtt{m}_j \in \mathtt{M}$, i.e. the initial object partition for a given column.

Calculating initial partitions: The equivalence relation described in Section 3.1 in the set of objects $\mathtt{G}$ given an attribute $\mathtt{m}_j$ is calculated using a graph-based method proposed in [13]. A complete graph $(\mathtt{G}, \mathtt{E})$ with edges $e_{ik} \in \mathtt{E}$ holding the distances $e_{ik} = |\mathcal{M}_{ij} - \mathcal{M}_{kj}|$ is created. Next, all edges $e_{ik} \geq \theta$ are removed. Then, two objects are equivalent if they belong to the same connected component. While this method is effective, it is also expensive (complexity $\mathcal{O}(|\mathtt{M}||\mathtt{G}^2|)$) and problematic in the sense that it generates false equivalences (e.g. with $\theta = 1$, we have $1 \simeq_\theta 2$ and $2 \simeq_\theta 3 \implies 1 \simeq_\theta 3$ which is not true).

A more appropriate method is to consider disjoint equivalence blocks. An equivalence block is an interval v over a range of values W (analogous to tolerance blocks described in [10]). For a given attribute $\mathtt{m}_j$, we define a set of disjoint equivalence blocks V_j where any two intervals $v_1, v_2 \in V_j$ have an empty intersection $v_1 \cap v_2 = \emptyset$. An object $\mathtt{g}_i$ belongs to the interval $v = [l, r]$ with $l, r \in W$ iff $l \leq \mathcal{M}_{ij} \leq r$ (actually, this holds for closed intervals). Two objects $\mathtt{g}_i, \mathtt{g}_k$ are equivalent for V_j (given $\mathtt{m}_j$) iff they belong to the same interval $v \in V_j$. Each interval generates an equivalence class. This method avoids false equivalences as the one just described while the calculation of equivalence classes only has a complexity of $\mathcal{O}(|\mathtt{M}||\gamma||\mathtt{G}|)$, where γ is the number of equivalence classes created. In general, equivalence blocks can be predefined by a user or calculated from W given γ. In our experiments, we consider both, pre-defined equivalence blocks and using a γ value.

Breaking the lattice: As discussed at the end of Section 3.3, for a given pp-concept $(\mathtt{B}, \mathtt{d})$, not every pair $(p, \mathtt{B})$ with $p \in \mathtt{d}$ represents a bicluster. Through the use of Proposition 2, we can easily discover biclusters within the pp-lattice while it is being calculated. Experimental data has shown that less than 20% of the pp-concepts within the pp-lattice actually hold a maximal bicluster. This has given room for an optimization

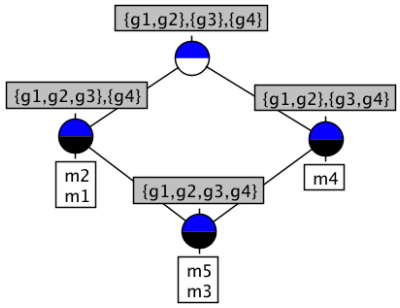

Figure 2: Partition pattern concept lattice.

Bicluster	Elements	Maximal?
1	$(\{g_1, g_2, g_3, g_4\}, \{m_3, m_5\})$	✓
2	$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3, m_5\})$	✓
3	$(\{g_4\}, \{m_1, m_2, m_3, m_4, m_5\})$	✓
4	$(\{g_1, g_2\}, \{m_1, m_2, m_3, m_4, m_5\})$	✓
5	$(\{g_3\}, \{m_1, m_2, m_3, m_4, m_5\})$	✓
6	$(\{g_3, g_4\}, \{m_3, m_4, m_5\})$	✓
7	$(\{g_4\}, \{m_1, m_2\})$	✗
8	$(\{g_1, g_2\}, \{m_3, m_4, m_5\})$	✗

Table 5: Biclusters with similar values from pp-lattice.

of the AddIntent algorithm based on pruning the pp-lattice from pp-concepts not holding maximal biclusters. While this actually breaks the structure of the concept lattice, it keeps the order among the remaining pp-concepts, meaning that the output of AddIntent remains the same, while dramatically decreasing its computational time. This optimization has been included in the implementation of our biclustering algorithm.

Complexity: Aside the possible optimizations to the AddIntent algorithm, the fact remains that enumerating all possible biclusters from a data-table, from a large dataset, is computationally intractable [4, 12], since the number of biclusters grows exponentially w.r.t. the number of objects and attributes. For example, the complexity of the AddIntent algorithm is $\mathcal{O}(|\mathcal{L}| \cdot |\mathtt{G}^2| \cdot |\mathtt{M}|)$ where $|\mathcal{L}|$ is the number of concepts in the lattice, which can grow up to $2^{|\mathtt{G}|}$ [16].

Methods to overcome this issue are still an open research problem in FCA, as well as in other disciplines. A popular technique in FCA is the use of minimal supports for formal concept mining. This allows to cut from the lattice a set of formal concepts which are regarded as not representative and hence, not important for analysis purposes. The analogous of minimal support for biclustering is a minimal number of rows or columns required for a bicluster. This restriction has been used in several exhaustive bicluster enumeration algorithms to avoid exponential complexity [12]. In our approach, we set a minimal length for each equivalence class which we call σ. Each equivalence class such as $|[\mathtt{g}]_{\mathtt{m}_j}| \leq \sigma$ is ignored and not included the pp-concept intent.

4 Experiments

The first experiment shows the effects of the optimization discussed in the previous section. We used a subset of the dataset called MovieLens 100k[3] of movie ratings containing 943 users and 50 movies (out of a total of 1682) using $\sigma = 1$ (all bicluster sizes) and the predefined set of equivalence blocks $[1, 2][3, 3][4, 5]$. The dataset contains user ratings for movies which range from 1 to 5. When information is not available, the matrix contains 0 which we disregard (we do not mine biclusters with columns equal to 0). The dataset contained 16532 similar column biclusters. The basic AddIntent algorithm processes a single object per iteration. The experiment consists in inserting between AddIntent iterations a lattice pruning process while measuring the execution time. The dataset size was reduced to let $\sigma = 1$ and mine every possible bicluster. Results are shown in Figure 3. The solid horizontal line represents the execution time without optimization (30.5 seconds). While initially, the execution time doubles the non-optimized version (for a lattice prune each AddIntent iteration), later the time quickly stabilizes around half the time the non-optimized version. Best time is found for 40 iterations (15 seconds).

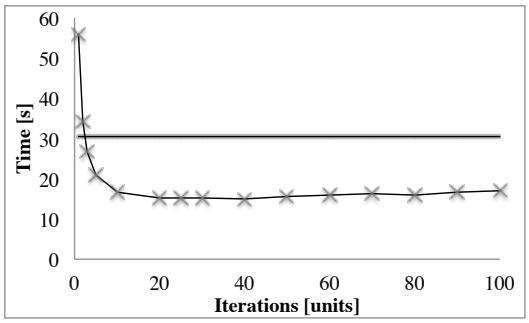

Figure 3: AddIntent Iterations per prune vs Execution time

The optimization affects the number of intent intersections performed by AddIntent. When the lattice is pruned, there are not as many intents to intersect as there were originally. However, pruning the lattice is an expensive task and adds overhead to the algorithm. The correct balance of this trade-off leads to dramatic improvements in the performance (twice in the experiments), however further experimentation in different numerical data-tables are needed to draw more conclusions regarding its setting.

The second experiment was performed over an example dataset provided with the system BicAt[4] containing 419 objects and 70 attributes. We measure the performance of our approach mining similar row biclusters compared with Cheng and Church's algorithm (CC) [3]. CC tries to find a determined number of biclusters with a maximum threshold for the mean squared error δ. Results are shown in Table 6. Parameters for pp-lattice are number of equivalence blocks γ and minimal number of columns in the cluster σ. CC was executed as provided by BicAt and other parameters were left as system's default.

Results show a general better performance of our approach which is able to mine more than four million maximal biclusters from the dataset in less time than CC calculates only ten thousands. In terms of minimal squared error (MSE), our approach gets smaller scores which induces better quality biclus-

[3] `http://grouplens.org/datasets/movielens/`

[4] `http://www.tik.ee.ethz.ch/sop/bicat/`

	Time [s]	Biclusters [Kunits]	Parameters	MSE Max	Max Size [cells]
PPL	451	901	γ=20, σ=10	0.016	209
PPL	27	36	γ=10, σ=30	0.032	372
PPL	306	390	γ=10, σ=25	0.037	442
PPL	3,404	4,471	γ=10, σ=30	0.041	462
PPL	253	314	γ=5, σ=50	0.259	1,173
CC	418	1	$\delta = 0.5$	3.2	17,752
CC	416	1	$\delta = 0.3$	2.81	17,752
CC	4,018	10	$\delta = 0.5$	4.92	17,752

Table 6: Comparison between CC and pp-lattice bicluster algorithm.

ters. CC is able to find larger biclusters compared to our approach given the top-down strategy which implements. While larger biclusters can be found with our approach by decreasing the number of equivalent classes (γ), this is done at the cost of increasing the MSE as shown in Table 6. Compared to CC, our approach is better on finding many high quality and rather small biclusters inducing specialized associations among objects. CC is better at creating a global map of the entire data-table by finding larger biclusters.

5 Related Work

Biclustering techniques have usually been proposed for bioinformatics and gene expression analysis. A thorough description of these approaches can be found in [12] and a more recent one in [4]. Regarding FCA-based biclustering, two main techniques have been proposed. In [8], the authors present a technique based in interval pattern structures to mine similar value biclusters which was later revisited using Triadic Concept Analysis (TCA) in [7]. Our work shares many similarities with both of these approaches, however we focus on a different kind of biclustering, namely similar row/column value biclustering. Furthermore, we use a different framework and algorithms.

Lattice-based hierarchical clustering has been explored in [13] where the author proposes a technique analogous to single-linkage hierarchical clustering based in the Galois connection [6]. The author also introduces notions of "object" distance to support both, symbolic and numerical data. Our work extends from these notions to hierarchical biclustering and proposes an implementation not present in the aforementioned work.

Regarding exhaustive bicluster mining, in [2] the system NBS-miner (numerical bi-sets) is introduced for constant value biclustering which also deals with tolerance relations using a threshold for "object" similarity. In the same lines, in [14], an approach based on range support patterns mining is proposed. In this work, the authors propose a technique for exhaustive "similar" row biclustering using an evaluation function on the biclusters and the Apriori algorithm [11]. Both of these techniques present efficient ways for bicluster enumeration. Our approach differs in that it is able to find an overlapped hierarchical structure along with the biclusters.

6 Conclusions and research perspectives

In this work we have presented a novel technique for exhaustive similar row/column value biclustering based on FCA algorithms using partition pattern structures. We have shown the capabilities of the technique which is able to find a large number of high quality biclusters. Furthermore, biclusters are provided with an overlapping hierarchy based on a concept lattice structure. How to leverage current biclusters analysis techniques using the concept lattice is still a matter of research.

Partition pattern structures were initially proposed for functional dependencies mining [1] using association rules from pp-concepts. How these techniques may benefit from the current approach and the opposite, is an interesting subject which should be explored. Using other techniques of formal concept selection and filtering, and their associations with biclusters is another compelling aspect for a future work.

REFERENCES

[1] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli, 'Characterizing functional dependencies in formal concept analysis with pattern structures', *Annals of Mathematics and Artificial Intelligence*, (2014).

[2] Jérémy Besson, Céline Robardet, Luc Raedt, and Jean-François Boulicaut, 'Mining bi-sets in numerical data', in *Knowledge Discovery in Inductive Databases*, (2007).

[3] Yizong Cheng and George M. Church, 'Biclustering of expression data', in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, (2000).

[4] Adelaide Valente Freitas, Wassim Ayadi, Mourad Elloumi, Joséluis Oliveira, Joséluis Oliveira, and Jin-Kao Hao, *Survey on Biclustering of Gene Expression Data*, 2013.

[5] Bernhard Ganter and Sergei O. Kuznetsov, 'Pattern Structures and their projections', *Conceptual Structures: Broadening the Base*, (2001).

[6] Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis*, mathematic edn., 1999.

[7] Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, and Amedeo Napoli, 'Biclustering meets triadic concept analysis', *Annals of Mathematics and Artificial Intelligence*, (2013).

[8] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli, 'Biclustering numerical data in formal concept analysis', in *Formal Concept Analysis*, (2011).

[9] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli, 'Revisiting numerical pattern mining with formal concept analysis', *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, (November 2011).

[10] Sergei O. Kuznetsov, 'Galois connections in data analysis: Contributions from the soviet era and modern russian research', in *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, (2005).

[11] Sergei O Kuznetsov and Sergei Obiedkov, 'Comparing Performance of Algorithms for Generating Concept Lattices', *Journal of Experimental and Theoretical Artificial Intelligence*, (2002).

[12] Sara C. Madeira and Arlindo L. Oliveira, 'Biclustering algorithms for biological data analysis: A survey', *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, (January 2004).

[13] Sadaaki Miyamoto, 'Lattice-valued hierarchical clustering for analyzing information systems', in *Rough Sets and Current Trends in Computing*, volume 4259 of *Lecture Notes in Computer Science*, (2006).

[14] Gaurav Pandey, Gowtham Atluri, Michael Steinbach, Chad L. Myers, and Vipin Kumar, 'An association analysis approach to biclustering', in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2009).

[15] Gianvito Pio, Michelangelo Ceci, Corrado Loglisci, Domenica D'Elia, and Donato Malerba, 'A novel biclustering algorithm for the discovery of meaningful biological correlations between mirnas and mrnas', *EMBnet.journal*, **18**(A), (2012).

[16] Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie, 'Addintent: A new incremental algorithm for constructing concept lattices', in *Concept Lattices*, (2004).