Information Sciences 181 (2011) 1989–2001

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Mining gene expression data with pattern structures in formal concept analysis

Mehdi Kaytoue [a], Sergei O. Kuznetsov [b,*], Amedeo Napoli [a], Sébastien Duplessis [c]

[a] Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA), Campus Scientifique, B.P. 70239, 54500 Vandœuvre-lès-Nancy, France
[b] State University Higher School of Economics, Pokrovskiy Bd. 11, 109028 Moscow, Russia
[c] UMR 1136, Institut National de la Recherche Agronomique (INRA), Nancy Université, Interactions Arbres/Micro-organismes, 54280 Champenoux, France

ARTICLE INFO

Article history:
Available online 13 August 2010

Keywords:
Formal concept analysis
Conceptual scaling
Numerical data
Pattern structures
Gene expression data

ABSTRACT

This paper addresses the important problem of efficiently mining numerical data with formal concept analysis (FCA). Classically, the only way to apply FCA is to binarize the data, thanks to a so-called scaling procedure. This may either involve loss of information, or produce large and dense binary data known as hard to process. In the context of gene expression data analysis, we propose and compare two FCA-based methods for mining numerical data and we show that they are equivalent. The first one relies on a particular scaling, encoding all possible intervals of attribute values, and uses standard FCA techniques. The second one relies on pattern structures without *a priori* transformation, and is shown to be more computationally efficient and to provide more readable results. Experiments with real-world gene expression data are discussed and give a practical basis for the comparison and evaluation of the methods.

1. Introduction

Numerous classification problems can be formalized by means of formal contexts. A context materializes a set of individuals (called objects), a set of properties (called attributes), and a binary relation usually represented by a binary table relating objects to attributes, where $(g,m) = \times$(a cross) if the object g has the property m [1,9]. Considering a pair of mappings between sets of all subsets of objects and attributes, called *Galois connection*, it is possible to derive for each object g the set of all attributes that apply to g. Similarly, it is possible to derive for each attribute m the set of all objects to which m applies. As a consequence, one may classify within *formal concepts* a set of objects sharing the same maximal set of attributes, and vice versa. Concepts are ordered within a lattice structure called *concept lattice* within the formal concept analysis (FCA) framework [9]. This mathematical structure supports potential knowledge discovery in databases that benefits of an important set of techniques for building, visualizing and interpreting concept lattices [9,20]. Concept lattices are represented by diagrams giving nice visualization of classes of objects of a domain. At the same time, the edges of these diagrams give essential knowledge about objects, by giving association rules between attributes describing the objects [21]. FCA can also be used for a number of purposes among which knowledge formalization and acquisition, ontology design, and information retrieval [36,38].

In real-world applications, e.g. in biology or chemistry, one rarely obtains binary data directly, *complex* and *heterogeneous* data involving numbers, graphs, intervals, etc., are more typical. To apply FCA-based methods to such data, the latter have to

* Corresponding author.
E-mail addresses: kaytouem@loria.fr (M. Kaytoue), skuznetsov@hse.ru (S.O. Kuznetsov), amedeo.napoli@loria.fr (A. Napoli), duplessi@nancy.inra.fr (S. Duplessis).

0020-0255/$ - see front matter © 2011 Published by Elsevier Inc.
doi:10.1016/j.ins.2010.07.007

be binarized, i.e. *scaled*. Many types of scaling are known in FCA literature [9], however, they do not always suggest the most efficient implementation right away, and there are situations where one would choose original data representation rather than scaled data [8]. Although scaling allows one to apply FCA tools, it may dramatically increase the complexity of computation and representation, and make worse the visualization of results.

Instead of scaling, one may work directly with initial data, i.e. complex object descriptions, defining so-called *similarity operators* which induce a semi-lattice on data descriptions. Several attempts were made for defining such semi-lattices on sets of graphs [8,17,18,22] and logical formulas [5,7] (see also [10,37] for FCA extensions). Indeed, if one is able to order object descriptions in complex data, e.g. with graph morphism when objects are described by labelled graphs, one may attempt to directly build a concept lattice from such data. In [8], a general approach called *pattern structures* was proposed, which allows one to apply standard FCA to any partially ordered data descriptions.

This paper addresses the problem of FCA-based classification of numerical data, where object descriptions are vectors of numbers, with pattern structures and a particular similarity operator. We focus on gene expression data (GED), where *gene expression profiles* represent the "behaviour" of genes in biological situations, and a situation corresponds to tissues at different time points or cellular loci (different organs, healthy or cancerous tissues, etc.). Genes with similar expression profiles are said to be co-expressed. It is now widely accepted that co-expressed genes interact together within the same biological process [34]. GED analysis is an important task and an active area of research involving mainly data-mining methods: clustering [14], biclustering [23,29]. FCA-based methods have been recently designed and applied in this domain [4,15,26].

For analysing GEDs by means of FCA, one needs to build a formal context from a GED, attribute values have to be discretized and intervals of entry values have to be considered as binary attributes, implying possible loss of actual data values [15]. In [9], *interordinal scaling* is defined and allows one to build a formal context that encodes all possible intervals of attributes values, without loss of information. However this scaling produces large and dense binary data, which are hard to process with existing FCA algorithms [20]. This is probably one of the reasons why this scaling has never been used for GED analysis. By contrast, the formalism of pattern structures, defined in full compliance with the FCA framework in [8], allows one to build a concept lattice without *a priori* scaling procedure. Accordingly, in this paper, we introduce an interval convexification as a similarity operator for ordering intervals within a semi-lattice, i.e. by taking the convex hull of any arbitrary set of intervals. However, this operation between complex descriptions of objects may be harder to process than classical set intersection and inclusion test after a scaling. Then, a challenging question arises for numerical data like GEDs: *should one scale numerical attributes?*

To discuss this question, we have experimented with both approaches, comparing their computational efficiency, the respective results and their representations. We show that both methods have equivalent outputs, but the method based on pattern structures is more computationally efficient than that based on interordinal scaling, and provides better readable and interpretable results.

The article is organized as follows. Firstly, gene expression data (GED) are presented in Section 2. Section 3 recalls the basics of FCA and shows how to build a concept lattice from GED using interordinal scaling to represent value intervals. An approach based on pattern structures for building an isomorphic lattice without transforming the data is detailed in Section 4. Experiments with a real-world GED are discussed in Section 5, giving also a practical basis for the comparison of the two methods. The conclusion suggests further research directions.

2. Gene expression data

Gene expression is the mechanism that produces a protein from a gene in two steps. Firstly transcription builds a copy of a gene called mRNA which is then translated into a protein. This mechanism differs in different biological situations: for each gene the concentration of mRNA and proteins depends on the current cell, time, etc. and reflects the behaviour of the gene. Indeed, biological processes of a living cell are based on chemical reactions and interactions between proteins and mRNA. Thus, it is important to measure and analyse mRNA and protein concentration to understand biological processes activated in a cell.

Using microarray biotechnology, the concentration of mRNA is measured into a numerical value called *gene expression value*, which characterizes the behaviour of a gene in a particular cell. Microarrays can monitor simultaneously the expression of a large number of genes, possibly the complete coding space of a genome. When several microarrays are considered, the expression value of a gene is measured in multiple situations or environments, e.g. different cells, time points, cells responding to particular environmental stresses, etc. This characterizes the behaviour of the gene w.r.t. all these situations and is represented by a vector of expression values called a *gene expression profile*.

A *gene expression data* (GED) is generally described as a table with n rows corresponding to genes and m columns corresponding to situations. A table entry is called an *expression value*. A row in the table denotes an *expression profile* associated to a gene. For example, in Table 1, the expression value of g_1 in the situation s_1 is 5. The expression profile of the gene g_1 is denoted by the vector $\langle 5,7,6\rangle$. In this paper, we consider the NimbleGen Systems Oligonucleotide Arrays technology[1]: expression values range from 0 (not expressed) to 65535 (highly expressed).

[1] Details on this biotechnology can be found at www.nimblegen.com.

Table 1
Gene expression data.

	s_1	s_2	s_3
g_1	5	7	6
g_2	6	8	4
g_3	4	8	5
g_4	4	9	8
g_5	5	8	5

Stating that co-expressed genes interact together within the same biological process [34], various data-mining methods have applied to GEDs, among which clustering, biclustering and FCA-based methods (see related work in section 6). In the following, we use the FCA formalism to mine GEDs.

3. Mining GEDs by means of interordinal scaling

This section starts with classical definitions of FCA. Then a particular scaling for representing value intervals from numerical datasets is presented and illustrated for lattice-based classification of GEDs.

3.1. FCA: main definitions

Here we use standard definitions from [9]. Let G and M be arbitrary sets and $I \subseteq G \times M$ be an arbitrary binary relation between G and M. The triple (G,M,I) is called a formal context. Each $g \in G$ is interpreted as an object, each $m \in M$ is interpreted as an attribute. The fact $(g,m) \in I$ is interpreted as "g has attribute m". The two following derivation operators $(\cdot)'$:

$$A' = \{m \in M | \forall g \in A : gIm\} \quad \text{for } A \subseteq G,$$
$$B' = \{g \in G | \forall m \in B : gIm\} \quad \text{for } B \subseteq M,$$

define a Galois connection between the powersets of G and M. The operators $\{(\cdot)', (\cdot)'\}$ put in relation elements of the lattices $(2^G, \subseteq)$ of objects and $(2^M, \subseteq)$ of attributes and vice versa. A Galois connection induces a closure operator $(\cdot)''$ and realizes an one-to-one correspondence between all closed sets of objects and all closed sets of attributes, called concept extents and intents. 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 $(A_1, B_1) \leqslant (A_2, B_2) \iff A_1 \subseteq A_2 (\iff B_2 \subseteq B_1)$. 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.

3.2. Classical scalings for GEDs

To be represented in the form of a formal context numerical data require transformation called conceptual scaling. For each attribute, a scale is defined in accordance with the values of the attribute. The choice of a scale depends on data and research goals, and affects the size and interpretation of the resulting concept lattice.

Usually to apply FCA in GED analysis, an *l-cut* scaling is operated by using a single threshold l on expression values determined for each object [26,28,29]. Expression values greater than this threshold are said to be over-expressed and encoded by 1, otherwise by 0. Then formal concepts represent sets of genes simultaneously over-expressed.

In [15], a generalization with an interval-based scaling of numerical data is proposed, where interval number and size were chosen by experts. Given a set of genes G, a set of situations S and a set of ordered intervals T, $(g,(s,t)) \in I$, where $g \in G$, $s \in S$, $t \in T$ and I a binary relation, means that the expression value of the gene g is the interval of index t for the situation s. Formal concepts of the context $(G, S \times T, I)$ represent groups of genes whose expression values are in same intervals for a subset of situations (maybe for all situations). However these intervals are hard to determine adequately *a priori*.

3.3. Interordinal scaling for GEDs

Interordinal scaling defined in [9] can help describing all value intervals without loss of information. Let G be a set of genes, S a set of situations, $W \subset \mathbb{R}$ a set of expression values and I_1 a ternary relation defined on the Cartesian product $G \times S \times W$. The fact $(g,s,w) \in I_1$ or simply $g(s) = w$ means that gene g has expression value w for situation s (see for example Table 1). $\mathbb{K}_1 = (G, S, W, I_1)$ is called a many-valued context representing a GED. The objective is to extract formal concepts (A,B) from $\mathbb{K}_1$, where $A \subseteq G$ is a subset of genes sharing "similar values" of W, i.e. lying in a same interval. An appropriate binarization (scaling) technique is used to build a formal context $\mathbb{K}_2 = (G, S_2, I_2)$ called derived context of $\mathbb{K}_1$.

A scale is a formal context (cross-table), objects being the attributes of $\mathbb{K}_1$ and attributes being the derived ones of $\mathbb{K}_2$. As attributes do not take necessarily the same values, each of them is scaled separately. Let $W_s \subseteq W$ be the set of all values of

the attribute s. The following interordinal scale (see p. 42 in [9]) can be used to represent all possible intervals of attribute values:

$$\mathbb{I}_{W_s} = (W_s, W_s, \leqslant) | (W_s, W_s, \geqslant).$$

The operation of *apposition of two contexts* with identical sets of objects, denoted by $|$, returns the context with the same set of objects W_s and the set of attributes being the disjoint union of attribute sets of the original contexts. In our case, this operation is applied to two contexts $(W_s, W_s, \leqslant)$ and $(W_s, W_s, \geqslant)$. As W_s is composed of real numbers, the relations $\leqslant$ and $\geqslant$ are natural. Table 2 gives an example for $W_{s_1} = \{4, 5, 6\}$. The intents given by the interordinal scale are all possible value intervals.

Once a scale is chosen, conceptual scaling replaces each many-valued attribute of $\mathbb{K}_1$ with a set of binary attributes, resulting in the context $\mathbb{K}_2$. With interordinal scaling, each many-valued attribute s is replaced by $2{\cdot}|W_s|$ binary attributes with names "$s \leqslant w$" and "$s \geqslant w$", for all $w \in W_s$. For example, s_1 is replaced by $\{s_1 \leqslant 4, s_1 \leqslant 5, s_1 \leqslant 6, s_1 \geqslant 4, s_1 \geqslant 5, s_1 \geqslant 6\}$. Derived context $\mathbb{K}_2 = (G, S_2, I_2)$ is given in Table 3 for the attribute s_1 only. This transformation is applied without loss of information: the many-valued context can easily be reconstructed from the formal context. For example, derived attributes for $(g_1, s_1, 5)$ are $s_1 \leqslant 5$, $s_1 \leqslant 6$, $s_1 \geqslant 4$, $s_1 \geqslant 5$. The unique value in W_{s_1} respecting these predicates is 5 which is the original value.

The choice of an algorithm to build the concept lattice depends on the size and density of the formal context to process (see Section 4.6). Density of a formal context (G, M, I) is defined as the proportion of elements of I w.r.t. the size of the Cartesian product $G \times M$, i.e. density $d = |I|/(|G| \cdot |M|)$. In the case of interordinal scaling, density of derived context $\mathbb{K}_2$ is $d = \frac{\sum_{i=1}^{i \leqslant p} (|W_i|+1)}{2 \cdot \sum_{i=1}^{i \leqslant p} |W_i|}$, where p is the number of attributes in $\mathbb{K}_1$. When $|W|$ grows, d tends towards 50%. Moreover, the number

Table 2
Scale $\mathbb{I}_N := (N, N, \leqslant) | (N, N, \geqslant)$ for s_1, $N = \{4,5,6\}$.

	$s_1 \leqslant 4$	$s_1 \leqslant 5$	$s_1 \leqslant 6$	$s_1 \geqslant 4$	$s_1 \geqslant 5$	$s_1 \geqslant 6$
4	×	×	×	×		
5		×	×	×	×	
6			×	×	×	×

Table 3
$\mathbb{K}_2 = (G, S, I_2)$ for the attribute s_1.

	$s_1 \leqslant 4$	$s_1 \leqslant 5$	$s_1 \leqslant 6$	$s_1 \geqslant 4$	$s_1 \geqslant 5$	$s_1 \geqslant 6$
g_1		×	×	×	×	
g_2			×	×	×	×
g_3	×	×	×	×		
g_4	×	×	×	×		
g_5		×	×	×	×	

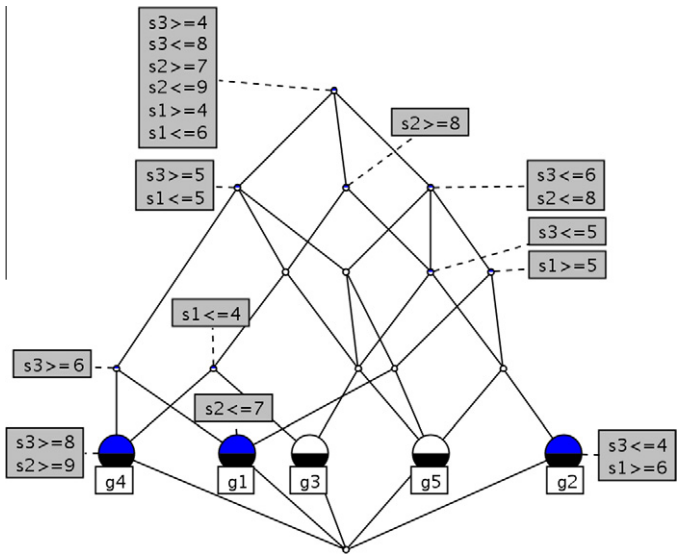

Fig. 1. Concept lattice of formal context $\mathbb{K}_2 = (G, S, I_2)$ drawn with the Concept Explorer software (http://conexp.sourceforge.net/).

of derived attributes is $2 \cdot \sum_{i=1}^{i \leqslant p} |W_i|$ and $|g'| = |W| + 1$ for all $g \in G$. This makes the derived contexts dense, large and difficult to process. For comparison, density of binary data in [29] does not exceed 6% and the number of derived attributes remains the same after scaling. As pointed in Section 4.6, these binary data show better computational properties, but biases are introduced.

Let us now consider the concept lattice of $\mathbb{K}_2$ given in Fig. 1. Concept extents near the Bottom concept contain a few genes, since the corresponding intents are related to the smallest intervals. The extent of the Top concept contains all genes and its intent corresponds to intervals of maximal size. The higher a concept lies in the diagram, the larger is the interval corresponding to its intent. Concepts near the Top are not interesting: they allow almost all possible values of attributes. The problem of selecting the best concepts in GED analysis is addressed in Subsection 5.3.

4. Mining GEDs by means of pattern structures

4.1. Introducing pattern structures

In this section, we present an alternative to scaling when a context includes many-valued attributes. This alternative is based on the idea of *pattern structures* [8] which was motivated by research on learning with labelled graphs and other complex descriptions [17,18].

Intuitively, the *similarity* of two sets of labelled graphs X and Y, denoted by $X \sqcap Y$, is given by the maximal common subgraphs of graphs from X and Y. Then a *graph pattern* may be defined as a set of graphs X such that $X \sqcap X = X$, i.e. X is "maximal" w.r.t. the similarity operation. It is easily seen that the operation $\sqcap$ is idempotent, associative and commutative. The similarity operation $\sqcap$ on sets of graphs is a sort of "attribute sharing", as in the binary case, where objects in extent share the maximal set of attributes in the corresponding intent. Denote by D the set of all graph patterns, then $(D, \sqcap)$ is a semi-lattice with infimum (meet) operator $\sqcap$. A natural subsumption order on graph patterns is given by $X \sqsubseteq Y \Leftrightarrow X \sqcap Y = X$.

More generally, a *pattern structure* is a triple $(G, (D, \sqcap), \delta)$ where G is a set of objects, $(D, \sqcap)$ is a meet-semi-lattice of object descriptions or *patterns*, and $\delta : G \to D$ is a mapping providing any object $g \in G$ with a description $d \in (D, \sqcap)$. As $(D, \sqcap)$ or equivalently $(D, \sqsubseteq)$ are semi-lattices, the following Galois connection, denoted by $\left\{(\cdot)^{\square}, (\cdot)^{\square}\right\}$, between $(2^G, \subseteq)$ and $(D, \sqsubseteq)$ gives rise to a complete lattice called the *pattern concept lattice* of $(G, (D, \sqcap), \delta)$ [8].

$$A^{\square} = \sqcap_{g \in A} \delta(g) \quad \text{for } A \subseteq G,$$
$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{for } d \in (D, \sqcap).$$

The first derivation operator takes a set of objects and returns a maximal description (pattern) shared by all objects. The second derivation operator takes a description and returns the maximal set of objects sharing this description.

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) the component d is called a *pattern intent* and is a description of all objects in A, called *pattern extent*. For a pattern structure $(G, (D, \sqcap), \delta)$, a pattern $d \in (D, \sqcap)$ is *closed* if $d^{\square\square} = d$. A set of objects $A \subseteq G$ is *closed* if $A^{\square\square} = A$. Obviously, pattern extents and intents are closed. When partially ordered by $(A_1, d_1) \leqslant (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow d_2 \sqsubseteq d_1)$, the set of all pattern concepts forms a complete lattice called a *pattern concept lattice*.

4.2. Intervals are patterns

To define a semi-lattice operation $\sqcap$ for intervals that would be analogous to the set-theoretic intersection or meet operator on sets of graphs, one should realize that "similarity" between two real numbers (between two intervals) may be expressed in the fact that they lie within some (larger) interval, this interval being the smallest interval containing both two.

Then, we choose to define the meet of two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$, as follows:

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)].$$

This operation can be viewed as a *convexification* of its arguments, as it returns the convex hull of two intervals. The choice of this operator seems natural to have a more general description when considering more objects, which would not be the case if considering a classical interval intersection as attribute values are numbers. The $\sqcap$ operator is idempotent, commutative, and associative. This means that the meet of several intervals is the smallest interval containing all intervals. Then, interval subsumption and interval inclusion are related as follows:

$$[a_1, b_1] \sqsubseteq [a_2, b_2] \Longleftrightarrow [a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \Longleftrightarrow [min(a_1, a_2), max(b_1, b_2)] = [a_1, b_1]$$
$$\Longleftrightarrow a_1 \leqslant a_2 \text{ and } b_1 \geqslant b_2 \Longleftrightarrow [a_1, b_1] \supseteq [a_2, b_2].$$

The definition of $\sqcap$ implies that smaller intervals subsume larger intervals that contain them. For example, with $D = \{[4,4], [5,5], [6,6], [4,5], [5,6], [4,6]\}$, the meet-semi-lattice $(D, \sqcap)$ is given in Fig. 2. The interval labeling a node is the meet of all intervals labeling its ascending nodes, e.g. $[4,5] = [4,4] \sqcap [5,5]$, and is also subsumed by these intervals, e.g. $[4,5] \sqsubseteq [5,5]$ and $[4,5] \sqsubseteq [4,4]$.

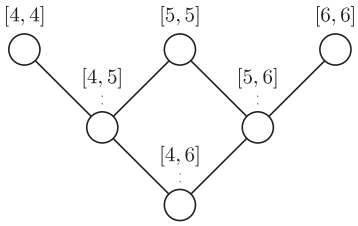

Fig. 2. Meet-semi-lattice $(D, \sqcap)$ with $D = \{[4,4], [5,5], [6,6], [4,5], [5,6], [4,6]\}$.

We have shown how intervals can be seen as patterns. Now we can define a pattern structure where each object is described by an interval. We show in the following how to generalize the process when considering vectors of intervals. Furthermore, this is exactly what we need for analysing GED where gene expression profiles are vectors of numbers (and $[a,a]$ is an interval for any $a \in \mathbb{R}$).

4.3. Interval vectors are patterns

We call an *interval vector* a p-dimensional vector of intervals. When e and f are vectors of p intervals, we write $e = \langle [a_i, b_i] \rangle_{i \in [1,p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$. The similarity operation $\sqcap$ is defined by the meet of corresponding components for vector of the same size (knowing that the order of the components is canonical):

$$e \sqcap f = \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \iff e \sqcap f = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]}.$$

Therefore, interval vectors are partially ordered by:

$$e \sqsubseteq f \iff \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]} \iff [a_i, b_i] \sqsubseteq [c_i, d_i], \quad \forall i \in [1,p],\ i \in [1,p],$$

meaning that each interval $[a_i, b_i]$ of e is subsumed by the corresponding interval $[c_i, d_i]$ of f. For example, $\langle [2,4], [2,6] \rangle \sqsubseteq \langle [4,4], [3,4] \rangle$ as $[2,4] \sqsubseteq [4,4]$ and $[2,6] \sqsubseteq [3,4]$.

4.4. Mining a GED as a pattern structure

GED in Table 1 can be formalized as a pattern structure $(G, (D, \sqcap), \delta)$ where $G = \{g_1, \ldots, g_5\}$ and D is a set of interval vectors or three-dimensional vectors, where each component corresponds to an attribute of the table. For example, $\delta(g_1) = \langle [5,5], [7,7], [6,6] \rangle$, where $[a,a]$ stands for any $a \in \mathbb{R}$. When $A \subseteq G$ is a set of objects and $d \in (D, \sqcap)$ is an interval vector, $A^{\square}$ returns an interval vector composed, for each dimension, of the smallest interval containing all intervals in the description of each object in A, i.e. their convex hull. On the other hand, $d^{\square}$ returns the set of objects being described for each dimension by an interval included in the corresponding interval of d.

For example, with data of Table 1, we have:

$$\begin{aligned} \{g_1, g_2\}^{\square} &= \sqcap_{g \in \{g_1, g_2\}} \delta(g) = \delta(g_1) \sqcap \delta(g_2) = \langle [5,5], [7,7], [6,6] \rangle \sqcap \langle [6,6], [8,8], [4,4] \rangle \\ &= \langle [5,5] \sqcap [6,6], [7,7] \sqcap [8,8], [6,6] \sqcap [4,4] \rangle = \langle [5,6], [7,8], [4,6] \rangle, \end{aligned}$$

$$\langle [5,6], [7,8], [4,6] \rangle^{\square} = \{g \in G \mid \langle [5,6], [7,8], [4,6] \rangle \sqsubseteq \delta(g)\} = \{g_1, g_2, g_5\}.$$

Obviously, g_1 and g_2 belong to $\langle [5,6], [7,8], [4,6] \rangle^{\square}$. g_5 also belongs to this set because $\langle [5,6], [7,8], [4,6] \rangle \sqsubseteq \delta(g_5)$.

Then, the pair $(A, d) = (\{g_1, g_2, g_5\}, \langle [5,6], [7,8], [4,6] \rangle)$ is a pattern concept meaning that $A^{\square} = d$ and $A = d^{\square}$. The set of all pattern concepts gives rise to a pattern concept lattice (see Fig. 3).

4.5. Concept lattice and pattern concept lattice

The following proposition establishes an isomorphism between the concept lattice of $\mathbb{K}_I$ with the relation $\mathbb{I}_{W_s} = (W_s, W_s, \leqslant) \mid (W_s, W_s, \geqslant)$, resulting from the interordinal scaling as defined in Section 3.3, and the pattern concept lattice of $(G, (D, \sqcap), \delta)$.

Proposition 1. *Let $A \subseteq G$, then statements 1 and 2 are equivalent:*

1. *A is an extent of the pattern structure $(G, (D, \sqcap), \delta)$ and $A^{\square} = \langle [\underline{m}_i, \bar{m}_i] \rangle_{i \in [1,p]}$, where $\underline{m}_i$ and $\bar{m}_i$, respectively, denote the minimum and maximum of values of the objects in A for the ith attribute.*
2. *A is a concept extent of the context $\mathbb{K}_I$ so that for all $i \in [1,p]$ $\underline{m}_i$ is the largest number n such that the attribute $s_i \geqslant n$ is in A' and $\bar{m}_i$ is the smallest number n such that the attribute $s_i \leqslant n$ is in A'.*

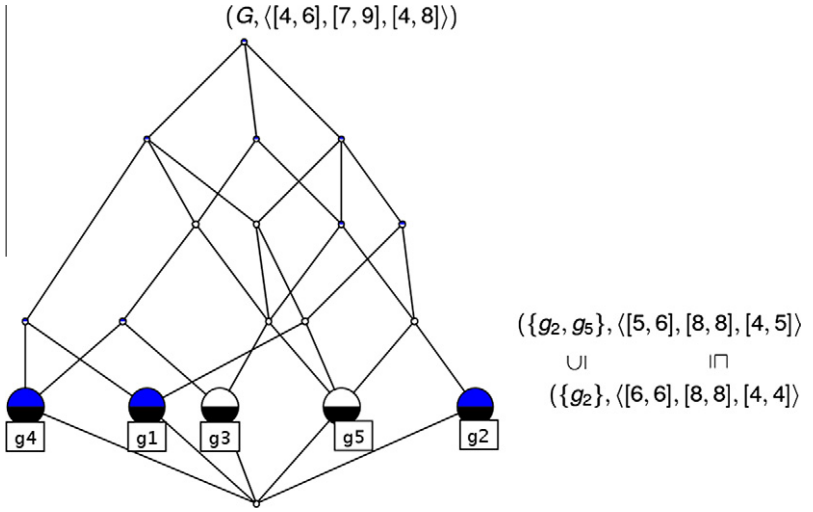

Fig. 3. Pattern concept lattice of pattern structure from Table 1.

Proof. $1 \to 2$ Let $A \subseteq G$ be a pattern extent. Given $\delta_i(g)$ the mapping that returns the ith interval of the vector describing object g. Since $A^{\square} = \langle [\underline{m}_i, \bar{m}_i] \rangle_{i \in [1,p]}$, for every object $g \in A$ one has $\underline{m}_i \leqslant \delta_i(g) \leqslant \bar{m}_i$ and there are objects $g_1, g_2 \in A$ such that $\delta_i(g_1) = \underline{m}_i$, $\delta_i(g_1) = \bar{m}_i$. Hence, in context K_I one has

$$A' = \cup_{i \in [1,p]} \{s_i \geqslant n_{\min}, \ldots, s_i \geqslant n_1, s_i \leqslant n_2, \ldots, s_i \leqslant n_{\max}\},$$

where

$$n_{\min} \prec \cdots \prec n_1 \leqslant n_2 \prec \cdots \prec n_{\max},$$

and $n_1 = \underline{m}_i$, $n_2 = \bar{m}_i$. Hence, $\underline{m}_i$ is the largest number n such that the attribute $s_i \geqslant n$ is in A' and $\bar{m}_i$ is the smallest number n such that the attribute $s_i \leqslant n$ is in A'. Suppose that A is not an extent of K_I. Hence, $A \subset A''$ and there is $g \in A'' \backslash A$ and $g' \supseteq A'$. This means that for all i $\underline{m}_i \leqslant \delta_i(g) \leqslant \bar{m}_i$. Therefore, $g \in A^{\square\square}$ and $A \neq A^{\square\square}$, a contradiction. The proof $2 \to 1$ is similar. □

Consider an example of pattern concept: $(\{g_1,g_2,g_5\},\langle[5,6],[7,8],[4,6]\rangle)$, the equivalent concept of the interordinally scaled context is $(\{g_1,g_2,g_5\},\{s_1 \leqslant 6, s_1 \geqslant 4, s_1 \geqslant 5, s_2 \geqslant 7, s_2 \leqslant 8, s_2 \leqslant 9, s_3 \leqslant 6, s_3 \leqslant 8, s_3 \geqslant 4\})$. Pattern intents are concise representations of concept intents. Therefore, concept intents are long descriptions, which can be turned to pattern intents by a simple syntactic post-processing.

4.6. Computation

Many algorithms for generating formal concepts from a formal context are compared in [20]. Experimental results highlight *Norris*, *CloseByOne* and *NextClosure* algorithms as the best algorithms when the context is dense and large, which is the case of interordinally derived formal contexts (shown in Section 3). Worst-case upper bound time complexity of the three algorithms for computing a set of formal concepts from a formal context G, M, I is $O(|G|^2 \cdot |M| \cdot |L|)$ with G the set of genes, M the set of attributes (here the set of attributes of the scaled context), and L the set of generated concepts.

To compute interval pattern concepts, the selected FCA algorithms *Norris*, *CloseByOne*, and *NextClosure*, need only slight modifications. The worst-case time complexity of computing the set of interval patterns is $O(|G|^2 \cdot p \cdot |L|)$, where p is the number of components in interval vectors, i.e. the number of numerical attributes in the original numerical data.

In both cases, the sets G and L are the same, thus relative efficiency of processing both data representations depends on the number of different attribute values in the original many-valued numerical context.

We now propose an adaptation of the *CloseByOne* algorithm for processing pattern structures such as vectors of intervals. This algorithm is the most efficient in our case (see Section 5) and the original pseudo-code for processing a formal context is given in Algorithms 1 and 2. It generates all concepts in a bottom-up way (from minimal to maximal extents). It considers objects one by one starting from the minimal one w.r.t. a linear order $<$ on G, e.g. a lexical order on object labels. Then it adds the next object w.r.t $<$ and applies the closure operator $(\cdot)''$ to generate the next concept. Testing $<$ on the obtained extent helps to easily determine if a concept extent was already generated. Finally, recursiveness of the algorithm induces a tree structure on the set of all concepts. More details on this algorithm can be found in [20,33]. To adapt this algorithm for pattern structures, one has to replace each call to a $(\cdot)'$ operator by a call to the corresponding $(\cdot)^{\square}$ operator. Then, computing $A^{\square}$ for a set $A \subseteq G$ is realized by taking *min* (respectively *max*) of all left (respectively right) limits of the intervals of each object

description. For a pattern $d \in (D,\sqcap)$, $d^{\square}$ is computed by testing for each object $g \in G$ if each interval of its description is included in the corresponding interval of d.

Algorithm 1 Close by one

```
1:  L = ∅
2:  for each g ∈ G
3:      process ({g},g,(g',g'))
4:  L is the concept set
```

Algorithm 2 Process $(A,g,(C,D))$ with $C = A''$ and $D = A'$ and $<$ the lexical order on object names

```
    if {h|h ∈ C\A and h < g} = ∅ then
2:      L = L ∪ {(C,D)}
        for each f ∈ {h|h ∈ G\C and g < h}
4:        Z = C ∪ {f}
          Y = D ∩ {f'}
6:        X = Y'
           process (Z,f,(X,Y))
8:  end if
```

By contrast, it is not that simple to adapt a depth-first search algorithm such as Charm that searches for closed sets [39]. For this algorithm, the binary representation of descriptions and their storage of particular data structures are essential.

5. Biological experiments

This section shows how pattern structures are used for extracting biological information from a real-world GED and how they outperform interordinally scaled contexts in terms of processing time.

5.1. A real-world GED

Biologists at the UMR IAM (INRA) study interactions between fungi and trees. They published the complete genome sequence of the fungus *Laccaria bicolor* [24]. This fungus lives in symbiosis with many trees of boreal and temperate forests. The fungus forms a mixed organ on tree roots and is able to exchange nutrients with its host in a specific symbiotic structure called ectomycorrhiza, contributing to a better tree growth and enhancing forest productivity. On the other hand, the plant repays its symbiotic partner by providing carbohydrates, allowing the fungus to complete its biological cycle by producing fruit-bodies (e.g. mushrooms). It is thus of major interest to understand how the symbiosis performs at the cellular level. The genome sequence of *Laccaria bicolor* contains more than 20,000 genes [24]. The study of their expression in various biological situations helps to understand their roles and functions in the biology of the fungus. Microarray techniques enable to compare expression values of all the genes between contrasted situations like free-living cells of the fungus (i.e. mycelium), cells engaged in the symbiotic association (i.e. ectomycorrhiza), and specialized cells forming the fruit-body structure (i.e. mushroom). *Laccaria bicolor* gene expression data is available at the Gene Expression Omnibus of the National Center for Biotechnology Information (NCBI).[2] It is composed of 22,294 genes in lines and five various biological situations in columns, reflecting cells of the organism in various stages of its biological cycle, i.e. free-living mycelium (situation FLM), symbiotic tissues (situations MP and MD) or fruiting bodies (situations FBe and FBl).

5.2. Preprocessing the data

First, a selection from the 22,294 genes is processed. Indeed, a gene that shows similar expression values in all situations presents less interest to the biologist than a gene with high differences of expression. One gene with a constant expression does not indicate a particular contribution to a cellular process (although its expression *per se* can be sufficient to participate to the process). Besides, significant changes in gene expression may reflect a role in a biological process and such genes help the biologist to draw hypotheses.

Filtering the genes consists in removing genes having no significant difference of expression across all situations. For each couple of situation, a t-test is performed and a p-value is attributed. If the p-value >0.05 (cut-off classically applied in biology) for all couples of situations then the current gene is removed from the dataset. The CyberT tool[3] was used to filter the

[2] www.ncbi.nlm.nih.gov/geo/ as series GSE9784.
[3] Available at http://cybert.microarray.ics.uci.edu.

dataset and obtain 11,930 genes. Another classical pre-processing in GED analysis is to transform expression values using log_2. Indeed, it allows the capture of small expression values into intervals that should be larger for high expression values. Finally, for making computation possible, a last pre-processing consists in rounding log_2 expression values to one digit after the comma, recalling that the more there are different attribute values, the more they are concepts.

5.3. Methodology of mining GED

Before extracting concepts from the GED defined above, we should remark that, given the definition of $\sqcap$ as a convexification of intervals, the following property of an (interval vector) pattern concept lattice is obvious. The lowest concepts w.r.t. $\leqslant$ are generally composed of pattern extents with few objects and "precise" descriptions, i.e. whose pattern intent is composed of "small" intervals. Then, the higher a concept is, the more elements there are in its extent, and the more intervals of its intent are large. For example, the Top concept, i.e. the highest concept w.r.t. $\leqslant$, has an extent containing all objects, and an intent composed of the largest intervals subsumed by all respective intervals of the data. In the example, Top = $(G, \langle[4,6],[7,9],[4,8]\rangle)$. However, the main goal of GED analysis is extracting homogeneous groups of genes, i.e. groups of genes having similar expression values. Therefore, descriptions of homogeneous groups should be composed of intervals with "small" sizes where $size([a,b]) = b - a$.

Consider a parameter max_{size} that specifies the maximal admissible size of any interval composing an interval vector. Then pattern concepts of interest have pattern intents $d = \langle[a_i,b_i]\rangle_{i\in[1,p]} \in (D,\sqcap)$ satisfying the constraint: $\exists i \in [1,p](b_i - a_i) \leqslant max_{size}$, for any $a, b \in \mathbb{R}$. A stronger constraint would be $\forall i \in [1,p](b_i - a_i) \leqslant max_{size}$, meaning that only concepts representing genes with "similar" expression values in at least one or all biological situations are retained. Therefore, two values are said to be similar if their difference does not exceed max_{size}. Since both constraints are monotone (if an intent does not satisfy it, then a subsumed intent does not satisfy it either), the subsets of patterns satisfying any of these constraints are order ideals (w.r.t. subsumption on intervals $\sqsubseteq$) of the lattice of pattern intents. In terms of computation, this means that only some lower part of the pattern lattice is computed, with patterns satisfying the constraints. *CloseByOne* can easily consider these constraints as it generates concepts from minimal to maximal extents.

The *CloseByOne* algorithm was run on the resulting pattern structure with $max_{size} = 0.35$. A concept is retained if it describes at least seven co-expressed genes in at least five situations, i.e. the intent has at least five intervals whose size do not exceed the max_{size} parameter. Indeed, let us recall that concepts near the Bottom, i.e. in the lowest levels of the concept lattice, are composed of a few genes described by small intervals. Processing time was about 2 min and returns 2,120 concepts (hardware details are given in next section).

5.4. Biological results and interpretation

Here we present two extracted patterns selected as grouping genes with high expression levels in the fruit-bodies situations, whereas their expression remains similar between the mycelium and symbiosis situations (Fig. 4). These patterns have been extracted from the whole list of 2,120 patterns for the following characteristic: in both cases, the expression levels measured are about two times higher in the fruit-body compared to the other situations. It indicates that these genes correspond to biological functions of importance at this stage. The expression measured in the mycelium and symbiosis situations tends to indicate that these genes are also involved in general cellular processes as they are already expressed in all situations.

The pattern in Fig. 4 (left) contains seven genes, of which only three possess a putative cellular function assignment based on similarity in international gene databases at NCBI. Interestingly, these genes all encode enzymes involved in distinct metabolic pathways. A gene encodes a 1-pyrroline-5-carboxylate dehydrogenase which is involved in amino-acid metabolism, another corresponds to an acyl-coA dehydrogenase, involved in fatty acid metabolism and a last gene encodes a transketolase, an enzyme involved in the pentose phosphate pathway of carbohydrate metabolism. All these metabolic functions are

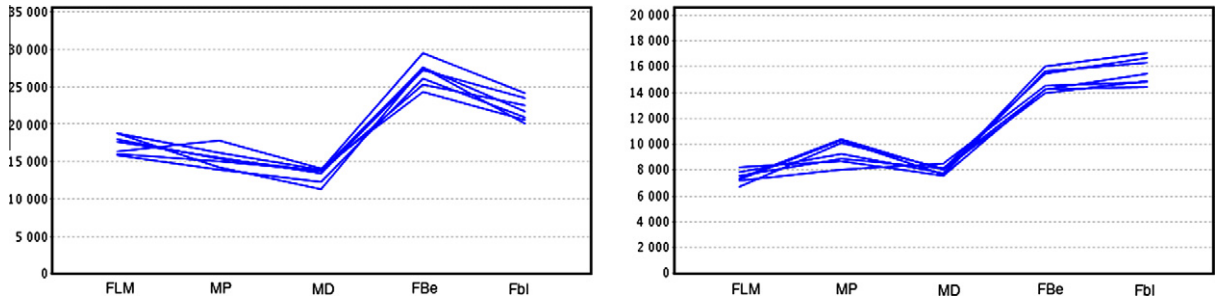

Fig. 4. Graphical visualization of two extracted concepts. *X*-axis is composed of situations, *Y*-axis is the expression values axis. Each line denotes the expression profile of a gene in the concept extent. Values are taken before the logarithmic transformation.

essential for the fungus and reflect that the fruit-body is a highly active tissue. The fruit-body is a specific fungal organ that differentiate in order to produce spores and that further ensure spore dispersal in nature [30]. Previous gene expression analyses of the fruit-body development conducted in the ectomycorrhizal fungus *Tuber borchii* also reported the strong induction of several genes involved in carbon and nitrogen metabolisms [13] as well as in lipid metabolism [32]. The present results are consistent with these observations and supports an important mobilization of nutrient sources from the mycelium to the fruit-body. It seems obvious that the primary metabolism requires to be adapted to use these sources in order to properly build spores and provide spore-forming cells with nutrients [30].

The pattern on Fig. 4 (right) also contains seven genes, of which only three possess a putative biological function. Interestingly, one of these genes encodes one pseudouridylate synthase, an enzyme involved in nucleotide metabolism that might also be involved in remobilization of fungal components from the mycelium to spore-forming cells and spores. The two other genes encode a cytoskeleton protein (actin) and a protein related to autophagy (autophagy-related 10 protein), a process that can contribute to the recycling of cellular material in developing tissues. Both functions participate in re-constructive cellular processes [30], which is consistent with the involvement of metabolic enzymes in remobilization of fungal resources towards the new organ in development.

Analysis of these two patterns that present a high expression level in the fruit-body situation is highly informative, confirms existing knowledge in the field and highlights the importance of remobilization in the developing organ. These co-expressed genes share related roles in a particular process. This could indicate that they are under the control of common regulators of gene expression. Interestingly, these patterns also contained a total of eight genes of unknown functions, i.e. for which no functional assignment was possible in international gene databases. There were four genes encoding hypothetical proteins with a homology in databases but no detailed function and 4 genes not previously described in fungi or other organism and which are considered specific to *Laccaria bicolor*. There are about 30% of such genes specific to this fungus and these may play specific roles in the biology of this soil fungus [24]. All these genes show consistent profiles with those encoding metabolic functions. Thus, these genes are interesting investigation leads as they may contain new enzymes not previously described of the pathways or eventual regulator of the cellular process. Altogether, these results contribute to a better understanding of the molecular processes underlying the fruit-body development.

As stated earlier, the expression of these genes was not specific to this biological situation. Their expression levels was already high in the mycelium and the symbiotic tissue indicating that these processes are essential not only to the fruit-body development but also to general cellular processes as previously described in expression studies of the tree-fungus symbiosis development [31].

5.5. Performance and efficiency study

Here we compare time performance of three algorithms for mining pattern structures of interval vectors (Section 4) and equivalent interordinally scaled contexts (Section 3). We have implemented the *Norris*, *NextClosure*, and *CloseByOne* algorithms, for both processing formal contexts and pattern structures. We have added the Charm algorithm [12] that extracts closed itemsets, i.e. concept intents in a formal context. FCA algorithms have been implemented in original versions as described in [20]. These algorithms are run within the Coron System [35].[4] All implementations are in Java: sets of objects and binary attributes are described with the BitSet class and interval descriptions with standard double arrays. The experiments were carried out on an Intel Core2 Quad CPU 2.40 Ghz machine with 4 GB RAM running under Ubuntu 8.10.

We began to compare algorithms on the data presented in biological experiments, i.e. from a many-valued context (G, S, W, I_1) where $|G| = 10{,}225$ and $|S| = 5$. Even by reducing the number of attribute values, computation is infeasible. Indeed we do not consider here constraints like the maximal interval size. Then we randomly selected samples of the data, by increasing the number of objects. As attribute values are real numbers with about five digits after the comma, the size of W is large. In the worst-case, $|W| = |G| \times |S|$, i.e. each attribute value is different in the dataset. This implies very large formal contexts to process and a large number of concepts. The execution times for this case are shown in Table 4. The *Norris* algorithm shows the best results in formal contexts, meeting conclusions of [20] for large and dense contexts. However, *CloseByOne* performs better for pattern structures, and most importantly is the only one able to compute a very large collection of concepts. When strongly reducing the size of W by rounding attribute values to the integer, i.e. $|W| \ll |G| \times |S|$, the *Charm* algorithm outperforms the others. The *Norris* algorithm is still the best FCA algorithm in formal contexts and *CloseByOne* is the best in pattern structures (see Table 5).

To sum up, we can say the following: When the number of different attribute values w.r.t. $|G| \times |S|$ is low, computing concepts from formal contexts is the most efficient solution. For large datasets with many different attribute values, it is much more efficient to compute with interval pattern structures. One explanation is that for formal concepts the concept intent representation is a bit string whose length increases with the growth of $|W|$. Object descriptions in pattern structure are arrays of constant size w.r.t. $|W|$.

[4] The Coron System is freely available at http://coron.loria.fr and also integrates a tool for applying interordinal scaling to numerical data.

Table 4
Generation time in both data representations (no projection).

Datasets							
$\|G\|$	10	20	30	40	50	75	100
$\|W\|$	50	100	150	199	249	374	252
Density (%)	51.00	50.50	50.33	50.25	50.20	50.13	50.20
Generation time in formal contexts (in ms)							
Charm	60	916	16,469	N/A	N/A	N/A	N/A
Next closure	5	145	1299	12,569	68,969	N/A	N/A
Norris	**2**	**90**	**609**	**5180**	**28,831**	N/A	N/A
Close by one	3	106	906	7944	41,238	N/A	N/A
Generation time in pattern structures (in ms)							
Next closure	6	100	763	5821	35,197	N/A	N/A
Norris	6	172	1982	15,522	83,837	N/A	N/A
Close by one	**2**	**85**	**585**	**3094**	**18,320**	**1,004,073**	**2,288,200**
Concept set L							
$\|L\|$	280	9587	78,173	455,008	1,857,725	40,325,176	64,571,385

Table 5
Generation time in both data representations. Attribute values are rounded.

Datasets							
$\|G\|$	25	50	75	100	125	150	200
$\|W\|$	34	37	44	53	58	62	66
Generation time for formal contexts (in ms)							
Density (%)	51.47	51.35	51.14	50.94	50.86	50.81	50.76
Charm	55	**154**	**184**	**243**	**394**	**936**	**1856**
Next closure	100	933	3333	22,973	30,854	78,790	593,416
Norris	**38**	320	861	2697	5954	15,359	46,719
Close by one	84	483	2424	8452	22,173	59,070	227,432
Generation time for pattern structures (in ms)							
Next closure	59	372	1924	6215	15,417	42,209	143,501
Norris	44	479	2602	7243	16,257	40,991	109,814
Close by one	**40**	**220**	**1084**	**3832**	**9289**	**23,989**	**89,804**
Concept set L							
$\|L\|$	1165	5928	23,962	48,176	73,463	163,316	252,515

6. Related work

Numerous methods have been designed since the end of 90's allowing the discovery and description of biological processes in living cells [34]. These methods may be divided into three categories: *clustering*, *biclustering*, and *FCA-based methods*. Units extracted by these methods, i.e. *clusters*, *biclusters*, and *formal concepts*, characterize biological processes. These methods are unsupervised or supervised by domain knowledge. We restrict our discussion to unsupervised methods, as there is generally a few knowledge units available when dealing with GEDs of species whose genome was very recently sequenced (this is the case of the plant species *Laccaria bicolor* considered in Section 5). We also do not discuss evolutionary computation of clusters, see e.g. [11].

Clustering methods group genes into *clusters* w.r.t. a *global* similarity, e.g. based on Euclidean distance, of their expression profiles. Here, "global" means that the similarity is computed for whole numerical vectors representing gene expression profiles. Then, clustering may fail to detect biological processes activated in some situations only [23].

To overcome this limitation, biclustering algorithms have been suggested [6,23]. *Biclusters* in a GED are defined as groups of genes having similar expression values in a same group of situations, but not necessarily all. However, it is known that most of the genes are involved in several processes [34], i.e. biclusters should overlap. Then, it becomes difficult to extract homogeneous biclusters based for example on subtables of a GED and respecting given constraints as their number grows exponentially. If constraints are more "heuristic-like", then interesting patterns may be missed [4]. This is one of the reasons why only a few biclustering algorithms allow overlapping of biclusters [23]. Our approach is close to that described in [27] where descriptions of gene clusters are sought as interval vectors of gene expression values. The authors of [27], however, do not use the semi-lattice on intervals for systematically generating all interesting clusters of this kind, but adhere to local optimum probabilistic approach by randomly generating a maximal gene expression bicluster at each iteration of a greedy algorithm that constructs a cover of the set of all genes. The authors of [27] do not compare their approach to equivalent approaches that use binarization of numerical values.

For a binary object-attribute table, e.g. Table 2, a bi-cluster is a set of objects having the same, or almost the same, set of attributes. In [29], authors proposed the *Bi-Max* biclustering algorithm, which extracts *inclusion-maximal biclusters* as

follows. Given a set G of genes, a set M of situations, and a binary table $\{e_{gm}\}_{g\in G,\ m\in M}$ such that $e_{gm} = 1$ or $e_{gm} = 0$, the pair $(A,B) \in 2^G \times 2^M$ is called an inclusion-maximal bicluster if and only if (1) $\forall a \in A,\ b \in B : e_{ab} = 1$ and (2) $\nexists (C,D) \in 2^G \times 2^M$ with (a) $\forall c \in C,\ \forall d \in D : e_{cd} = 1$ and (b) $A \subseteq C \wedge B \subseteq D \wedge (C,D) \neq (A,B)$. Stated in this way, a bicluster is nothing else than a formal concept.

Formal concept analysis (FCA) can be viewed as a kind of binary biclustering method. It provides means for extracting patterns from a binary relation, namely *formal concepts*. In application to GED analysis, concept extents are maximal sets of genes related to a common maximal set of situations (not necessarily all). The ordering of concepts among a complete lattice makes overlapping of concepts natural. Then a complete enumeration of patterns respecting some constraints like closure and minimal frequency is possible [4,15]. Indeed, the subsets of patterns satisfying these constraints is an order ideal of the lattice of patterns. Actually, in this paper, we payed particular attention to scaling problems, such as boundary problems, and we proposed monotone constraints to retain best concepts for a GED analysis.

7. Conclusion

In this paper, we addressed the problem of efficiently mining numerical data with techniques based on formal concept analysis (FCA). The standard way of dealing with numerical data in FCA is based on scaling. However, the data may be scaled in a lot of different ways leading to different results and interpretations. Most importantly, this usually leads either to loss of information and precision, or to huge and dense binary datasets difficult to process.

In the context of gene expression data analysis, we have compared two mathematically equivalent methods for processing numerical data. The first one uses interordinal scaling and classical FCA algorithms. It encodes all possible intervals of attribute values in a formal context that is processed with classical FCA algorithms. The second method relies on pattern structures: it builds a concept lattice directly from the original data. We proved that both resulting concept lattices are isomorphic. Most importantly, pattern structures offer more concise representations, better scalability, and better readability of the (pattern) concept lattice. Thus, we gave elements for answering the challenging question, *should one scale numerical attributes?*. We also showed substantial results for GED analysis, highlighting the important potential of pattern structures as a biclustering technique. It remains now to compare this method with other gene expression data mining techniques across a systematic comparative study.

Among other directions of further research, one may involve domain knowledge. The semi-lattice of descriptions $(D,\sqcap)$ may be viewed as an attribute hierarchy, where domain knowledge may be encoded, e.g. in some dimensions of a pattern vector. Domain knowledge can be given by text annotations on genes, e.g. [25], for which a similarity operation $\sqcap$ can be defined.

Considering the similarity operation $\sqcap$ as interval convexification generates too many patterns: some patterns and their sub-patterns w.r.t. $\sqsubseteq$ may describe almost the same set of genes, i.e. a few genes differs in their extents. Concept stability was introduced in [19] for measuring this phenomena. In this paper, we solved the problem of un-interesting patterns thanks to a monotone constraint. Recently, we embedded tolerance relations in pattern structures to produce only concepts with similar objects, w.r.t. a distance on their values [16]. Furthermore, one can also be interested to use pattern structures in fuzzy settings, although FCA has already been extended in [2,3] where an object is associated with an attribute with a truth degree.

Finally, and most importantly, the use of interval pattern structures should be of great interest for mining association rules in numerical data.

Acknowledgements

The second author was supported by the project of the Russian Foundation for Basic Research, Grant No. 08-07-92497-NTsNIL_a. This work was partially funded by the Contrat de Plan Etat – Région Lorraine: Modélisation, Information et Systèmes Numériques (2007–2013).

References

[1] M. Barbut, B. Monjardet, Ordre et Classification, Algèbre et Combinatoire, Hachette, Paris, 1970.
[2] R. Belohlávek, Fuzzy Relational Systems: Foundations and Principles, Kluwer Academic Publishers, 2002.
[3] R. Belohlavek, E. Sigmund, J. Zacpal, Evaluation of IPAQ questionnaires supported by formal concept analysis, Information Sciences 181 (10) (2011) 1774–1786.
[4] S. Blachon, R. Pensa, J. Besson, C. Robardet, J.-F. Boulicaut, O. Gandrillon, Clustering formal concepts to discover biologically relevant knowledge from gene expression data, In Silico Biology 7 (4–5) (2007) 467–483.
[5] L. Chaudron, N. Maille, Generalized formal concept analysis, in: B. Ganter, G.W. Mineau (Eds.), ICCS, LNCS, vol. 1867, Springer, 2000, pp. 357–370.
[6] Y. Cheng, G. Church, Biclustering of expression data, in: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISBM), 2000, pp. 93–103.
[7] S. Ferré, O. Ridoux, A logical generalization of formal concept analysis, in: B. Ganter, G.W. Mineau (Eds.), ICCS, LNCS, vol. 1867, Springer, 2000, pp. 371–384.
[8] B. Ganter, S.O. Kuznetsov, Pattern structures and their projections, in: H.S. Delugach, G. Stumme (Eds.), ICCS, LNCS, vol. 2120, Springer, 2001, pp. 129–142.
[9] B. Ganter, R. Wille, Formal Concept Analysis, mathematical foundations ed., Springer, 1999.
[10] A. Gély, R. Medina, L. Nourine, Representing lattices using many-valued relations, Information Sciences 179 (16) (2009) 2729–2739.
[11] E.R. Hruschka, R.J. Campello, L.N. de Castro, Evolving clusters in gene-expression data, Information Sciences 176 (13) (2006) 1898–1927.

[12] C.-J. Hsiao, M.J. Zaki, Efficient algorithms for mining closed itemsets and their lattice structure, IEEE Transactions on Knowledge and Data Engineering 17 (4) (2005) 462–478.
[13] I. Lacourt, S. Duplessis, S. Abba, P. Bonfante, F. Martin, Isolation and characterization of differentially expressed genes in the mycelium and fruit body of Tuber Borchii, Applied and Environmental Microbiology 68 (2002) 4574–4582.
[14] D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: a survey, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1370–1386.
[15] M. Kaytoue, S. Duplessis, A. Napoli, Using formal concept analysis for the extraction of groups of co-expressed genes, in: L.T.H. An, P. Bouvry, P.D. Tao (Eds.), MCO, CCIS, vol. 14, Springer, 2008, pp. 439–449.
[16] M. Kaytoue, Z. Assaghir, A. Napoli, S.O. Kuznetsov, Embedding tolerance relations in Formal Concept Analysis for classifying numerical data, in: Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM), ACM, 2010.
[17] S.O. Kuznetsov, JSM-method as a machine learning method, Itogi Nauki i Tekhniki, ser. Informatika 15 (1991) 17–50.
[18] S.O. Kuznetsov, Learning of simple conceptual graphs from positive and negative examples, in: J.M. Zytkow, J. Rauch (Eds.), PKDD, LNCS, vol. 1704, Springer, 1999, pp. 384–391.
[19] S.O. Kuznetsov, On stability of a formal concept, Annals of Mathematics and Artificial Intelligence 49 (1–4) (2007) 101–115.
[20] S.O. Kuznetsov, S.A. Obiedkov, Comparing performance of algorithms for generating concept lattices, Journal of Experimental and Theoretical Artificial Intelligence 14 (2002) 189–216.
[21] L. Lakhal, G. Stumme, Efficient mining of association rules based on formal concept analysis, in: B. Ganter, G. Stumme, R. Wille (Eds.), Formal Concept Analysis, LNCS, vol. 3626, Springer, 2005, pp. 180–195.
[22] M. Liquiere, Some links between formal concept analysis and graph mining, in: D.J. Cook, L.B. Holder (Eds.), Mining Graph Data, vol. 12, John Wiley and Sons, 2006, pp. 227–252.
[23] S. Madeira, A. Oliveira, Biclustering algorithms for biological data analysis: a survey, IEEE/ACM Transactions on Computational Biology and Bioinformatics 1 (1) (2004) 24–45.
[24] F. Martin, The genome of Laccaria bicolor provides insights into mycorrhizal symbiosis, Nature 452 (7183) (2008) 88–92 (69 co-authors wrote this paper).
[25] M. Molla, P. Andreae, J. Glasner, F. Blattner, J. Shavlik, Interpreting microarray expression data using text annotating the genes, Information Sciences 146 (1–4) (2002) 75–88.
[26] S. Motameny, B. Versmold, R. Schmutzler, Formal concept analysis for the identification of combinatorial biomarkers in breast cancer, in: R. Medina, S.A. Obiedkov (Eds.), ICFCA, LNCS, vol. 4933, Springer, 2008, pp. 229–240.
[27] T. Murali, S. Kasif, Extracting conserved gene expression motifs from gene expression data, Pacific Symposium on Biocomputing (2003) 77–88.
[28] R.G. Pensa, C. Leschi, J. Besson, J.-F. Boulicaut, Assessment of discretization techniques for relevant pattern discovery from gene expression data, in: M.J. Zaki, S. Morishita, I. Rigoutsos (Eds.), BIOKDD, 2004, pp. 24–30.
[29] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, A systematic comparison and evaluation of biclustering methods for gene expression data, Bioinformatics 22 (9) (2006) 1122–1129.
[30] S. Busch, G.H. Braus, How to build a fungal fruit body: from uniform cells to specialized tissue, Molecular Microbiology 64 (2007) 873–876.
[31] S. Duplessis, P.E. Courty, D. Tagu, F. Martin, Transcript patterns associated with ectomycorrhiza development in Eucalyptus Globulus and Pisolithus Microcarpus, New Phytologist 165 (2005) 599–611.
[32] S. Gabella, S. Abba, S. Duplessis, B. Montanini, F. Martin, P. Bonfante, Transcript profiling reveals novel marker genes involved in fruiting body formation in Tuber Borchii, Eukaryotic Cell 4 (2005) 1599–1602.
[33] S.O. Kuznetsov, A fast algorithm for computing all intersections of objects in a finite semi-lattice, Automatic Documentation and Mathematical Linguistics 27 (5) (1993) 11–21.
[34] R.B. Stoughton, Applications of DNA microarrays in biology, Annual Review of Biochemistry 74 (1) (2005) 53–82.
[35] L. Szathmary, Symbolic Data Mining Methods with the Coron Platform, Ph.D. Thesis in Computer Science, Univ. Henri Poincaré, France, 2006.
[36] P. Valtchev, R. Missaoui, R. Godin, Formal concept analysis for knowledge discovery and data mining: the new challenges, in: P.W. Eklund (Ed.), ICFCA, LNCS, vol. 2961, Springer, 2004, pp. 352–371.
[37] F.J. Valverde-Albacete, C. Peláez-Moreno, Extending conceptualisation modes for generalised formal concept analysis, Information Sciences 181 (10) (2011) 1888–1909.
[38] R. Wille, Why can concept lattices support knowledge discovery in databases?, Journal of Experimental and Theoretical Artificial Intelligence 14 (2–3) (2002) 81–92
[39] M. Zaki, Efficient algorithms for mining closed itemsets and their lattice structure, IEEE Transactions on Knowledge and Data Engineering 17 (4) (2005) 462–478.