HAL
open science

Order-preserving Biclustering Based on FCA and Pattern Structures

Nyoman Juniarta, Miguel Couceiro, Amedeo Napoli

To cite this version:

Nyoman Juniarta, Miguel Couceiro, Amedeo Napoli. Order-preserving Biclustering Based on FCA and Pattern Structures. Annalisa Appice; Michelangelo Ceci; Corrado Loglisci; Giuseppe Manco; Elio Masciari; Zbigniew W. Ras. Complex Pattern Mining: New Challenges, Methods and Applications, Springer Series on Studies in Computational Intelligence, pp.47-62, 2019, 978-3-030-36617-9. 10.1007/978-3-030-36617-9_4 . hal-02181585

HAL Id: hal-02181585

https://inria.hal.science/hal-02181585v1

Submitted on 12 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Order-preserving Biclustering Based on FCA and Pattern Structures

Nyoman Juniarta, Miguel Couceiro, and Amedeo Napoli

Abstract Biclustering is similar to formal concept analysis (FCA), whose objective is to retrieve all maximal rectangles in a binary matrix and arrange them in a concept lattice. FCA is generalized to more complex data using pattern structure. In this article, we explore the relation of biclustering and pattern structure. More precisely, we study the order-preserving biclusters, whose rows induce the same linear order across all columns.

1 Introduction

CrossCult (`http://www.crosscult.eu`) is a European project whose idea is to support the emergence of a European cultural heritage by allowing visitors in different cultural sites (e.g. museum, historic city, archaeological site) to improve the quality of their visit by using adapted computer-based devices and to consider the visit at a European level. Such improvement can be accomplished by studying, among others, the possibility to build a dynamic recommendation system. This system should be able to produce a relevant suggestion on which part of a cultural site may be interesting for a specific user/visitor.

Given U as the set of previous users and I as the set of items, one approach for producing such suggestion is by retrieving a set of similar users in U. There are different techniques of calculating the similarity between any two users. From a dataset of item ratings, we can interpret $u_1 \in U$ as similar to $u_2 \in U$ if they have similar order of preference. For example, suppose that $rating(u_x, i_y)$ is the rating given by user $u_x \in U$ to item $i_y \in I$. We can consider that u_1 is similar to u_2 if $rating(u_1, i_1) > rating(u_1, i_2)$ and $rating(u_2, i_1) > rating(u_2, i_2)$ (i.e. both of them prefer i_1 over i_2). Furthermore, their similarity is stronger if they give similar

N. Juniarta, M. Couceiro, A. Napoli
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, e-mail: {nyoman.juniarta,miguel.couceiro,amedeo.napoli}@loria.fr

rating order over a larger set of items, e.g. if $rating(u_1, i_1) > rating(u_1, i_2) > \cdots > rating(u_1, i_{10})$ and $rating(u_2, i_1) > rating(u_2, i_2) > \cdots > rating(u_2, i_{10})$. The task of finding a set of similar users based on this similarity can be regarded as the task of finding order-preserving (OP) biclusters in a numerical matrix.

In this article we explore two approaches of finding OP biclusters. The first is based on partition pattern structure, while the second is the application of sequence pattern structure. Both approaches originate from formal concept analysis (FCA), since it has a related objective with biclustering: finding submatrices whose cells have “similar" value. By using these approaches, we can theoretically enumerate all possible OP biclusters in a numerical matrix, and arrange them in a hierarchical structure.

OP bicluster is first defined in Ben et al. [2] to find a highly statistically significant OPSM in a gene expression dataset. The dataset is represented as a numerical matrix with genes as rows, experiments as columns, and each cell represents the expression level of a given gene under a given experiment. In such a matrix, an OPSM corresponds to a subset of genes that induce the same linear ordering of a subset of experiments.

The OP bicluster discovery problem is also a special case of sequential pattern mining [8]. A row in an $m \times n$ matrix can be regarded as a sequence of n items, and the whole matrix corresponds to a dataset of m sequences. This sequence dataset is extremely dense, since all possible items are present in each sequence (except missing value). Furthermore, many existing sequential pattern mining methods try to find patterns that satisfy a minimum support threshold [10, 12, 13, 14]. This means that longer sequences are difficult to be recovered, since shorter sequences (hence smaller OPSMs) dominate the results. This problem has been studied in [3] by proposing “rare" sequential pattern mining. We will also show that it can be used to obtain OP biclusters from a numerical matrix.

This article is organized as follows. First, we explains the background and examples of biclustering in Sect. 2. Sect. 3 describes the basic definitions of FCA and pattern structure. Our first approach is explained in Sect. 4, while the second approach is in Sect. 5. Some results of our experiments are described in Sect. 6. In the end, we discuss our conclusions and some future works in Sect. 7.

2 Order-preserving biclusters

In this section, we recall the basic background and discuss illustrative examples of biclustering, especially order-preserving biclusters, as described in [9].

We consider a dataset composed of a set of objects G, each of which has values over a set of attributes M. Therefore, in a numerical dataset (G, M), the value of m for object g is written as $m(g)$.

One may be interested in finding which subset of objects possesses the same values w.r.t. a subset of attributes. Regarding the matrix representation, this is equivalent to the problem of finding a submatrix that has a constant value over all of its elements

(example in Table 1a). This task is called biclustering with constant values, which is a simultaneous clustering of the rows and columns of a matrix.

Other than constant values, the bicluster approach also focused on finding other types of submatrices, as shown in Table 1. A bicluster with constant columns is a submatrix where each column has the same value, illustrated in Table 1b.

Table 1 Examples of (a) constant-value, (b) constant-columns, and (c) order-preserving (OP) biclusters

4	4	4	4
4	4	4	4
4	4	4	4
4	4	4	4

(a)

4	2	5	3
4	2	5	3
4	2	5	3
4	2	5	3

(b)

1	2	4	3
3	5	7	6
2	3	8	4
4	5	9	8

(c)

In this article, we are dealing with OP biclusters. In this type, each row induces the same linear order across all columns as defined in Definition 1.

Definition 1 Given a dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is an order-preserving (OP) bicluster iff $\forall g \in A$, there exist a sequence of columns $(m_1 \cdots m_{|B|})$ in B such that $m_i(g) < m_{i+1}(g)$ for all $1 \leq i \leq |B| - 1$.

For example, in the bicluster in Table 1c, each row follows *column1* < *column2* < *column4* < *column3*.

Biclustering shares many common elements with formal concept analysis (FCA). In FCA, from a binary matrix we try to find a maximal submatrix whose elements are 1. In other words, the objective of FCA is to identify maximal constant-value biclusters (but only for biclusters whose values are 1). Hence, a formal concept can be considered as a bicluster of objects and attributes. Furthermore, formal concepts are arranged in a concept lattice, that materializes the hierarchical relation among all biclusters. In the following section, we will recall some basic definitions of FCA and its generalization to pattern structures.

3 FCA and Pattern Structures

Formal concept analysis is a mathematical framework based on lattice theory and used for classification, data analysis, and knowledge discovery [6]. From a formal context, FCA detects all formal concepts, and arranges them in a concept lattice.

A formal context is a triple (G, M, I), where G is a set of objects, M is a set of attributes, and I is a binary relation between G and M, i.e. $I \subseteq G \times M$. If an object g has an attribute m, then $(g, m) \in I$. An example of a formal context is shown in Table 2, where $\times$ indicates the corresponding $(g, m) \in I$.

Between the sets G and M, there is a Galois connection that consists of two functions: $2^G \rightarrow 2^M$ and $2^M \rightarrow 2^G$. For a subset of objects $A \subseteq G$, A' is the set of

Table 2 A formal context

	m_1	m_2	m_3	m_4
g_1				×
g_2	×		×	
g_3	×	×	×	
g_4		×	×	×

attributes that are possessed by all objects in A, i.e.:

$$A' = \{m \in M | \forall g \in A, (g, m) \in I\}, \qquad A \subseteq G.$$

Dually, for a subset of attributes $B \subseteq M$, B' is the set of objects that have all attributes in B, i.e.:

$$B' = \{g \in G | \forall m \in B, (g, m) \in I\}, \qquad B \subseteq M.$$

A formal concept is a pair (A, B), where $A \subseteq G$ and $B \subseteq M$, and such that $A' = B$ and $B' = A$. For example, $\{g_2, g_3\}' = \{m_1, m_3\}$, and $\{m_1, m_3\}' = \{g_2, g_3\}$ in Table 2. They form a formal concept $(\{g_2, g_3\}, \{m_1, m_3\})$.

A formal concept (A, B) is a *subconcept* of (C, D) – denoted by $(A, B) \leq (C, D)$ – if $A \subseteq C$ (or equivalently $D \subseteq B$). A concept lattice can be formed using the $\leq$ relation which defines the partial order among concepts. If $(A, B) < (C, D)$ and there is no (E, F) such that $(A, B) < (E, F) < (C, D)$, then (A, B) is a lower neighbor of (C, D). For the context in Table 2, the formal concepts and their corresponding lattice are shown in Fig. 1.

FCA is restricted to specific datasets where each attribute is binary (e.g. has only yes/no value). For more complex values (e.g. numbers, strings, trees, graphs. . .), FCA is then generalized into pattern structures [5].

A pattern structure is a triple $(G, (D, \sqcap), \delta)$, where G is a set of objects, $(D, \sqcap)$ is a complete meet-semilattice of descriptions, and $\delta : G \rightarrow D$ maps an object to a description. The operator $\sqcap$ is a similarity operation that returns the common

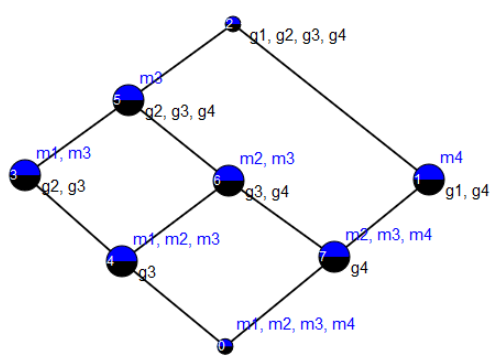

Fig. 1 Concept lattice for the formal context in Table 2

elements between any two descriptions. It implies that $c \sqcap d = c \Leftrightarrow c \sqsubseteq d$. A description can be a number, set, sequence, tree, graph, or other complex structure. In standard FCA (with set as description), $\sqcap$ corresponds to set intersection ($\cap$), i.e. $\{a, b, c\} \sqcap \{a, b, d\} = \{a, b\}$, and $\sqsubseteq$ corresponds to subset inclusion ($\subseteq$). In the case of sequence as a description, $\sqcap$ can be a set of common closed subsequences (SCCS) [3]. Similarly, $\sqsubseteq$ corresponds to subsequence inclusion ($\preceq$).

The Galois connection for a pattern structure $(G, (D, \sqcap), \delta)$ is defined as:

$$A^{\diamond} = \bigsqcap_{g \in A} \delta(g), \qquad A \subseteq G,$$
$$d^{\diamond} = \{g \in G | d \sqsubseteq \delta(g)\}, \qquad d \in D.$$

A pattern concept – similar to a standard formal concept – is a pair (A, d), $A \subseteq G$ and $d \in D$, where $A^{\diamond} = d$ and $d^{\diamond} = A$.

Table 2 can be regarded as a pattern structure with $G = \{g_1 \cdots g_4\}$, while the description of each object is a set of attributes. For example, $\delta(g_2) = \{m_1, m_3\}$ and $\delta(g_3) = \{m_1, m_2, m_3\}$. Their similarity is a set intersection, i.e. given $A = \{g_2, g_3\}$, $A^{\diamond} = \delta(g_2) \sqcap \delta(g_3) = \{m_1, m_3\}$. Furthermore, with $d = \{m_1, m_3\}$, we have $d \sqsubseteq \delta(g_2)$ and $d \sqsubseteq \delta(g_3)$. Therefore $d^{\diamond} = \{g_2, g_3\}$ and the pair $(\{g_2, g_3\}, \{m_1, m_3\})$ is a pattern concept.

The set of all pattern concepts also forms a lattice. When applying pattern structure to the task of finding biclusters, the lattice can be useful to obtain the hierarchical structure of a set of biclusters. In the following sections, we will describe two approaches of finding OP biclusters: based on partition pattern structure (Sect. 4) and based on sequence pattern structure (Sect. 5).

4 Finding Biclusters Using Partition Pattern Structure

In this section, first we recall the partition pattern structure (pps) detailed in [4]. In pps, the pattern structures is $(M, (D, \sqcap), \delta)$, where attributes are considered as objects, explained in Sect. 4.1. The description of each attribute $m \in M$ is a partition of objects according to the values on the given attribute. We will then propose an extension of this approach to perform OP biclustering in Sect. 4.2.

4.1 Partition Pattern Structure

As explained in [1, 4], a partition d of a set G is a collection of subsets of G ($d = \{p_i\}$, $p_i \subseteq G$), such that:

$$\bigcup_{p_i \in d} p_i = G \quad \text{and} \quad p_i \cap p_j = \emptyset \quad \text{whenever} \quad i \neq j. \tag{1}$$

The set of all partitions is denoted as D, and it will become the description of attributes in the pattern structure $(M, (D, \sqcap), \delta)$. The function δ maps an attribute to a partition of objects, i.e. $\delta : M \to D$.

Table 3 A dataset with 4 objects and 5 attributes

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	3	1	7
g_2	1	2	4	2	7
g_3	2	5	4	5	3
g_4	2	5	4	5	7

The whole set of attributes can be partitioned (respecting Eq. 1) according to the values of an attribute. To do that, we need an equivalence relation – which is reflexive, symmetric, and transitive – among objects. The equivalence relation $[g_i]_{m_j}$ of an object g_i w.r.t. an attribute m_j is:

$$[g_i]_{m_j} = \{g_k \in G | m_j(g_i) = m_j(g_k)\}. \tag{2}$$

Given an attribute m_j, the relation $[g_i]_{m_j}$ splits G into equivalence classes. It satisfies Eq. 1, such that the classes cover the whole set of objects and there is no intersection between any two different classes. These classes can be regarded as partition elements. Therefore a partition mapping is defined as $\delta : M \to D$, such as:

$$\delta(m_j) = \{[g_i]_{m_j} | g_i \in G\}. \tag{3}$$

For example, from Table 3:

$$\begin{aligned}
&[g_1]_{m_1} = [g_2]_{m_1} = \{g_1, g_2\} \\
&[g_3]_{m_1} = [g_4]_{m_1} = \{g_3, g_4\} \\
&\delta(m_1) = \{[g_1]_{m_1}, [g_2]_{m_1}, [g_3]_{m_1}, [g_4]_{m_1}\} = \{\{g_1, g_2\}, \{g_3, g_4\}\}.
\end{aligned}$$

The meet and join of two partitions $d_1 = \{p_i\}$ and $d_2 = \{p_j\}$ are defined as:

$$d_1 \sqcap d_2 = \{\{p_i \cap p_j\} | p_i \in d_1 \text{ and } p_j \in d_2\} \tag{4}$$

$$d_1 \sqcup d_2 = (\{\{p_i \cup p_j\} | p_i \in d_1 \text{ and } p_j \in d_2 \text{ and } p_i \cap p_j \neq \emptyset\})^+ \tag{5}$$

where $(.)^+$ is a closure that preserves only the maximal components in d. For example, given $\delta(m_1) = \{\{g_1, g_2\}, \{g_3, g_4\}\}$ and $\delta(m_4) = \{\{g_1\}, \{g_2\}, \{g_3, g_4\}\}$, we have $\delta(m_1) \sqcap \delta(m_4) = \{\{g_1\}, \{g_2\}, \{g_3, g_4\}\}$, and $\delta(m_1) \sqcup \delta(m_4) = \{\{g_1, g_2\}, \{g_3, g_4\}\}$.

The order between any two partitions is given by the subsumption relation:

$$d_1 \sqsubseteq d_2 \Longleftrightarrow d_1 \sqcap d_2 = d_1. \tag{6}$$

Therefore, $\delta(m_4) \sqsubseteq \delta(m_1)$ since $\delta(m_1) \sqcap \delta(m_4) = \delta(m_4)$.

Given a set of attributes M, a set of partitions D, and a mapping δ, a partition pattern structure is determined by the triple $(M, (D, \sqcap), \delta)$. Notice that in this triple, M is regarded as the set of objects, where the description of an object is a partition. A pair (A, d) is then called a partition pattern concept (pp-concept) iff $A^{\square} = d$ and $d^{\square} = A$, where:

$$A^{\square} = \bigsqcap_{m \in A} \delta(m) \qquad A \subseteq M \tag{7}$$

$$d^{\square} = \{m \in M | d \sqsubseteq \delta(m)\} \qquad d \in D. \tag{8}$$

For example, given $A = \{m_3, m_5\}$ in Table 3, we get $A^{\square} = \delta(m_3) \sqcap \delta(m_5) = \{\{g_1\}, \{g_3\}, \{g_2, g_4\}\}$. Dually, given $d = \{\{g_1\}, \{g_3\}, \{g_2, g_4\}\}$, we get $d^{\square} = \{m_3, m_5\}$ since $d \sqsubseteq \delta(m_3)$ and $d \sqsubseteq \delta(m_5)$. Therefore, $(\{m_3, m_5\}, \{\{g_1\}, \{g_3\}, \{g_2, g_4\}\})$ is a pp-concept. All pp-concepts from Table 3 are hierarchically illustrated as a lattice in Figure 2.

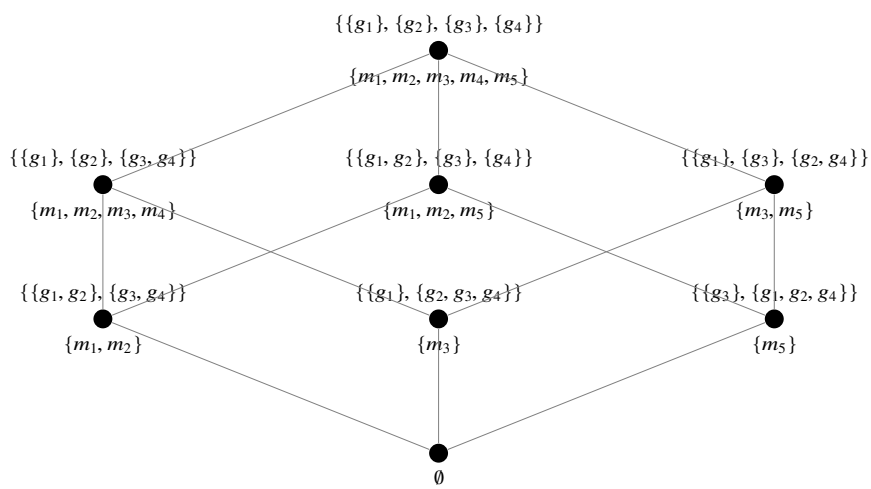

Fig. 2 Partition pattern lattice for Table 3.

4.2 OP Biclustering Using Partition

In this subsection, we will explain the possible application of partition pattern structure to discover OP biclusters as defined in Definition 1.

Consider the dataset given by Table 4, with the set of attributes $G = \{g_1, g_2, g_3, g_4, g_5\}$. For the task of finding OP biclusters, we introduce the notation r_y^x as a pair of attributes m_x and m_y, $x < y$, and R is the set of all possible r_y^x. That is, from n attributes, there will be C_2^n pairs.

Table 4 A dataset with 5 objects and 5 attributes

	m_1	m_2	m_3	m_4	m_5
g_1	1	2	3	4	5
g_2	4	2	1	5	3
g_3	2	3	4	1	5
g_4	5	4	2	3	1
g_5	2	1	5	4	3

Table 5 Some examples of partitions over Table 4

Pair	Partition
r_2^1	$\{\{g_1, g_3\}, \{g_2, g_4, g_5\}\}$
r_3^1	$\{\{g_1, g_3, g_5\}, \{g_2, g_4\}\}$
r_4^1	$\{\{g_1, g_2, g_5\}, \{g_3, g_4\}\}$
r_3^2	$\{\{g_1, g_3, g_5\}, \{g_2, g_4\}\}$
r_5^2	$\{\{g_1, g_2, g_3, g_5\}, \{g_4\}\}$

Using the same definition of partition as Equation 1, we define a new partition mapping $\delta : R \to D$, such as:

$$\delta(r_y^x) = \{[g_i]_{r_y^x} | g_i \in G\}. \tag{9}$$

The $[g_i]_{r_y^x}$ is the equivalence relation of an object w.r.t. a pair of attributes:

$$[g_i]_{r_y^x} = \{g_k \in G | \arg\max_j(m_j(g_i)) = \arg\max_j(m_j(g_k)), j \in \{x, y\}\}. \tag{10}$$

For example, from Table 4:

$$[g_1]_{r_2^1} = [g_3]_{r_2^1} = \{g_1, g_3\}$$
$$[g_2]_{r_2^1} = [g_4]_{r_2^1} = [g_5]_{r_2^1} = \{g_2, g_4, g_5\}.$$

In other words, δ maps a pair of attributes to a partition according to the pair's comparison. For example, $\delta(r_2^1) = \{\{g_1, g_3\}, \{g_2, g_4, g_5\}\}$ because $m_2 > m_1$ for g_1 and g_3; and $m_1 > m_2$ for g_2, g_4, and g_5. Some pairs and their partitions are listed in Table 5.

Given a set of attribute pairs R, a set of partitions D, and the mapping function δ, a partition pattern structure for finding OP biclusters is determined by the triple $(R, (D, \sqcap), \delta)$. A concept is a pair (B, d) such that $B^\square = d$ and $d^\square = B$, where:

$$B^\square = \bigsqcap_{r \in B} \delta(h) \qquad B \subseteq R \tag{11}$$

$$d^\square = \{r \in R | d \sqsubseteq \delta(r)\}. \qquad d \in D \tag{12}$$

The meet and subsumption relation between two partitions follow Equation 4 and Equation 6 respectively.

Here, the extent of a pp-concept is a set of attribute pairs. We can obtain a OP bicluster in a concept if there is a "clique" among the attributes in the pairs, as described in Definition 2.

Definition 2 Consider a concept (B, d). There may exist a set of attributes $C \subseteq M$ such that for all $m, n \in C$, there exists $r_n^m \in B$ or $r_m^n \in B$. The set C is called a *clique* in B.

For example, consider the concept pc_1 with extent $\{r_2^1, r_3^1, r_3^2\}$ and intent $\{\{g_1, g_3\}, \{g_5\}, \{g_2, g_4\}\}$. Its extent forms a clique among m_1, m_2, and m_3, since all pairs of any two of those attributes are included.

If a concept (B, d) contains a set of attributes $C \subseteq M$ that forms a clique, then each pair in $\{(p, C) | p \in d\}$ corresponds to a OP bicluster. For example, from pc_1, we can obtain three OP biclusters:

- $(\{g_1, g_3\}, \{m_1, m_2, m_3\})$, which follows $m_1 < m_2 < m_3$;
- $(\{g_2, g_4\}, \{m_1, m_2, m_3\})$, which follows $m_3 < m_2 < m_1$; and
- $(\{g_5\}, \{m_1, m_2, m_3\})$, which follows $m_2 < m_1 < m_3$.

5 Finding Biclusters Using Sequence Pattern Structure

In this section, we will recall the characterization of sequential pattern mining using sequence pattern structure as proposed in [3]. Then, we show how the problem of finding OP biclusters can be solved using sequence pattern structure.

5.1 Sequence Pattern Structure

A sequence is an ordered list $\langle s_1 s_2 \ldots s_m \rangle$, where s_i is an itemset $\{i_1, \ldots, i_n\}$. The list $s^1 = \langle \{a, b\}\{a, c, d\} \rangle$ is a sequence of 2 itemsets. For simplicity, we follow the bar notation in [3] such that two consecutive itemsets are separated by a bar. Therefore, the sequence from the previous example can be written as $ab|acd$.

A sequence $s = s_1|s_2|\ldots|s_m$ is a subsequence of $t = t_1|t_2|\ldots|t_n$, denoted by $s \preceq t$, if there exist indices $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $s_j \subseteq t_{i_j}$ for all $j = 1 \ldots m$ and $m \leq n$. For example, the sequence $a|d$ is a subsequence of $ab|acd$, while sequence $c|d$ is not.

Given a sequence database S (example in Table 6), the set of all subsequences of elements in S is denoted as Q, where $q \in Q \iff \exists s \in S$ s.t. $q \preceq s$ $(S \subseteq Q)$. The objective of frequent sequential pattern mining is to retrieve all $q \in Q$ whose support is larger than a threshold. The support of a sequence – denoted as $\sigma(q)$ – w.r.t. S is the number of sequences in S which have q as a subsequence, or $\sigma(q) = |\{s \in S | q \preceq s\}|, q \in Q$. In Table 6, given $q^1 = a|c$, the support of q^1 is 2, since $q^1 \preceq s^1$ and $q^1 \preceq s^2$.

Sequential pattern mining in is defined with respect to a particular sequence pattern structure, and it can retrieve “rare" sequences (small support but having larger number of itemsets) [3]. This pattern structure relies on the notion of closed sequences. From a set of sequences d, its set of closed sequences is:

$$d^+ = \{q^i \in d | \nexists q^j \in d \text{ s.t. } q^i \prec q^j\}. \tag{13}$$

Table 6 An example of sequence database

s	Sequence
s^1	$ab\|acd$
s^2	$a\|d\|c$
s^3	$ac\|d\|b$

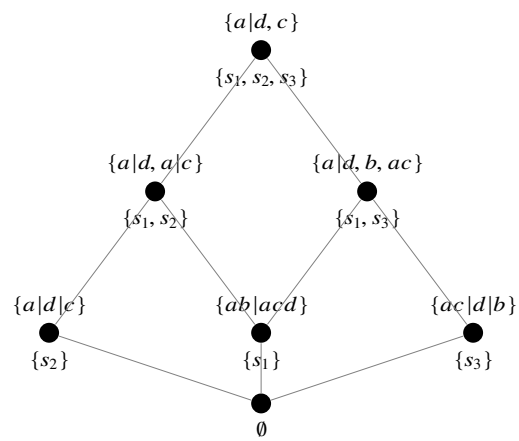

Fig. 3 Sequence pattern lattice for Table 6.

In other words, d^+ is all sequences in d who are not subsequence of other sequence in d. Then, the sequence pattern structure is defined as a triple $(S, (D, \sqcap), \delta)$, where $\delta : Q \to D$ is a mapping from a sequence q to its description $\delta(q)$. Here, a description is a set of closed sequences. As example, $\delta(s^1) = \{ab|acd\}$ in Table 6.

The similarity operator between any two descriptions d^1 and d^2 is the similarity between each sequence in d^1 and each sequence in d^2, or:

$$d^1 \sqcap d^2 = \left\{ \bigcup_{q^i \in d^1, q^j \in d^2} q^i \wedge q^j \right\}^+ . \tag{14}$$

The $\wedge$ operator returns the set of closed sequences that are subsequences of both q^i and q^j, or:

$$q^i \wedge q^j = \{q \in Q | q \preceq q^i \text{ and } q \preceq q^j\}^+ . \tag{15}$$

For example, suppose that we have $d^1 = \{a|c|d, a|b|c\}$ and $d^2 = \{a|d|c, a|b|c|d\}$. Each of these descriptions has two sequences. Their similarity is:

$$\begin{aligned} d^1 \sqcap d^2 &= \{a|c|d \wedge a|d|c, a|c|d \wedge a|b|c|d, a|b|c \wedge a|d|c, a|b|c \wedge a|b|c|d\}^+ \\ &= \{a|c, a|d, a|c|d, a|c, a|b|c\}^+ \\ &= \{a|c|d, a|b|c\}. \end{aligned}$$

The space of descriptions is denoted by D, and is a partially ordered set of descriptions. A description $d^1 \in D$ can be *subsumed by* ($\sqsubseteq$) $d^2 \in D$, iff:

$$d^1 \sqsubseteq d^2 \Longleftrightarrow \forall q^i \in d^1 : \exists q^j \in d^2 \text{ s.t. } q^i \preceq q^j. \tag{16}$$

This means that d^1 is subsumed by d^2 iff every sequence in d^1 is a subsequence of at least one sequence in d^2.

With $T \subseteq S$ and $d \in D$, a pair (T, d) is a sequence pattern concept iff $T^\square = d$ and $d^\square = T$, where:

$$T^\square = \bigsqcap_{s \in T} \delta(s) \tag{17}$$

$$d^\square = \{s \in S | d \sqsubseteq \delta(s)\}. \tag{18}$$

As example, $(\{s^1, s^3\}, \{a|d, b, ac\})$ is a sequence pattern concept from Table 6. All concepts from this table are shown in Figure 3.

5.2 OP Biclustering Using Sequence

The task of finding OP biclusters can be formulated as sequential pattern mining, and can be solved using sequence pattern structure. Consider again the dataset in Table 4. It can be regarded as a sequence database. Each $g \in \{g_1, \cdots g_5\}$ is a sequence of five items $m \in \{m_1, \cdots m_5\}$. The numbers in the matrix determine the order of each item in a given sequence. For example, the object g_2 has $m_3 < m_2 < m_5 < m_1 < m_4$, so the sequence of g_2 is $m_3|m_2|m_5|m_1|m_4$, and consequently $\delta(g_2) = \{m_3|m_2|m_5|m_1|m_4\}$.

A sequence pattern concept (T, d) has a set of objects (or sequences) as extent and a set of common subsequences as intent. Given a numerical table like Table 4, the intent is a set of sequence of attributes. A sequence $m_1| \cdots |m_n$ in the intent signifies that $m_1 < \cdots < m_n$ for all objects in the extent. As a result, from any sequence pattern concept (T, d), any pair of T and each sequence in d forms a OP bicluster. For example, consider the concept $(\{g_2, g_4, g_5\}, \{m_3, m_2|m_1, m_5|m_4\})$ from Table 4. Its intent has three sequences, so we have three OP biclusters:

- $(\{g_2, g_4, g_5\}, \{m_3\})$,
- $(\{g_2, g_4, g_5\}, \{m_1, m_2\})$, and
- $(\{g_2, g_4, g_5\}, \{m_4, m_5\})$.

Therefore, we can obtain OP biclusters in a numerical matrix using sequence pattern structure.

6 Experiment

In this section, we first compare the runtime of the two methods for finding OP bicluster: partition pattern structure (pps) and sequence pattern structure (sps), both using AddIntent algorithm to generate all concepts and the corresponding lattice. Randomly generated matrices are used, where the value of each cell is between 1 and 100 following uniform distribution. We inspect the effect of the number of attributes on both methods, and we choose a small number of objects (10).

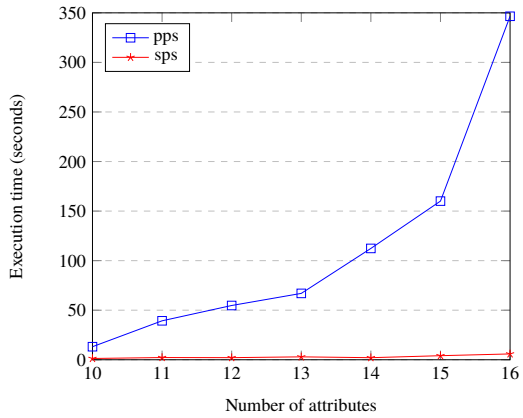

Fig. 4 Comparison of partition pattern structure (pps) and sequence pattern structure (sps) in the task of finding OP biclusters in matrices with 10 rows and varying number of columns.

The comparison of runtime are shown in Figure 4. It is shown that the execution time of partition-based mining of OP biclusters grows faster than sequence-based. Using partition pattern structure, from an $m \times n$ numerical matrix, a new $m \times C_2^n$ matrix is generated to compare every pair of columns. The partition is then performed in the new matrix. In this second approach, an $m \times n$ matrix is converted to a set of m sequences, each of them has n items. The first approach is more complex than the second, since the first creates a larger matrix before applying partition pattern structure.

We tested the pps-based approach to breast cancer dataset [7]. This dataset has 3226 rows (genes) and 21 columns (tissues). As shown in Table 7, these 21 tissues are composed by 7 brca1 mutations, 8 brca2 mutations, and 6 sporadic breast cancers. Since it has 21 columns, the pps-based approach will convert the dataset into a $3226 \times C_2^{21} = 3226 \times 210$ matrix. In order to reduce the computational complexity, we introduce a parameter θ. In calculation of the intent of any partition pattern concept, any partition component having the number of elements less than θ is discarded. For example, with $\theta = 3$, the partition $\{\{a, b, c\}, \{d, e\}\}$ becomes $\{\{a, b, c\}\}$. The number of concepts can be very large, so we provides the runtime until 10K concepts are obtained.

The result of experiment with varying θ is shown in Fig. 5. Here we see that in general, larger θ means more time to obtain 10K concepts. However, it does not necessarily mean that larger θ needs more time to finish the computation of the

Table 7 The columns in the breast cancer dataset in [7].

Column	Type	Column	Type	Column	Type
c_1	brca1	c_8	brca2	c_{15}	sporadic
c_2	brca1	c_9	brca2	c_{16}	sporadic
c_3	brca1	c_{10}	brca2	c_{17}	brca1
c_4	brca1	c_{11}	sporadic	c_{18}	brca2
c_5	brca1	c_{12}	sporadic	c_{19}	brca2
c_6	brca1	c_{13}	sporadic	c_{20}	brca2
c_7	brca2	c_{14}	sporadic	c_{21}	brca2

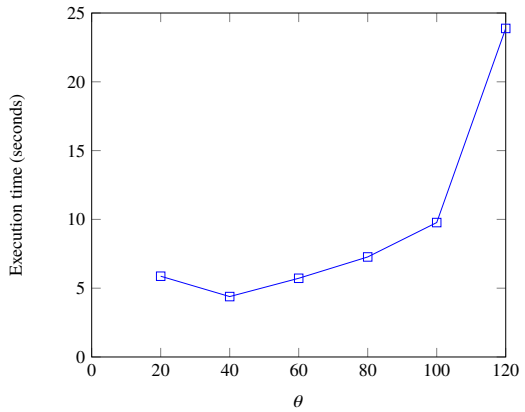

Fig. 5 Runtime of pps-based approach with AddIntent and varying θ in obtaining 10K concepts, applied to the breast cancer dataset. Any partition component having elements less than θ is discarded.

whole lattice. This is because lesser θ implies more concepts, hence faster to obtain 10K concepts. It can also be noted that for $\theta > 120$ (not shown in the figure), there are less than 10K concepts in the whole lattice.

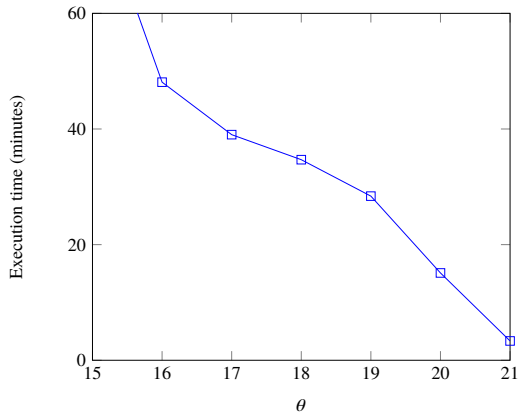

Fig. 6 Runtime of sps-based approach with AddIntent and varying ℓ, applied to the breast cancer dataset. Any sequence having length less than ℓ is discarded.

We also tested the sps-based approach to the same dataset. Contrary to the previous experiment where we take only 10K concepts, here we calculate the runtime for the

computation of the whole lattice. To reduce the computational time, we introduced the parameter ℓ, which is the minimal length of any sequence. Therefore, in calculating the intent of any sequence pattern concept, any sequence having length less than ℓ is discarded. For example, with $\ell = 3$, an intent $\{a|b|c, a|d\}$ becomes $\{a|b|c\}$.

The result of this experiment is shown in Fig. 6. With larger ℓ, the computational time is reduced, since we will have less concepts. This is useful compared to the majority of existing sequential pattern miners. To obtain sequential pattern with larger length, they usually need to lower the *minimum support* parameter. This results in more patterns, and consequently larger computational time.

Concerning the biclusters, we found that the bicluster of size 15×2 is the widest (having most columns) bicluster with more than 1 row. It follows the sequence of columns $c_{19} < c_{20} < c_1 < c_{17} < c_{12} < c_{21} < c_{11} < c_9 < c_{16} < c_{14} < c_{10} < c_7 < c_{13} < c_5 < c_3$, which is present in gene #24638 and #291057. Regarding biclusters covering more than 2 rows, the widest biclusters are of size 10×3. They are statistically significant because at a random 3226×21 matrix, we can only expect a bicluster covering 1 row (as $3226/10! < 1$) exhibiting a certain ordering of 10 columns.

7 Conclusion

In this article, we propose two approaches of finding OP biclusters. The first approach is based on partition pattern structure, going from the idea that given two attributes m_i and m_j, the set of all objects G can be separated to two groups: those with $m_i < m_j$ and those with $m_i > m_j$. For this article, we do not consider yet the equality (the case with $m_i = m_j$). Then given that the task of finding OP biclusters is similar to the sequential pattern mining, the second approach is based on sequence pattern structure. This allows us to define a threshold, such that we can mine OP biclusters whose number of columns is larger than the threshold.

In general, both approaches generate a set of overlapping OP biclusters, a characteristic similar to FCA, where a set of overlapping submatrices is obtained. Furthermore, both the set of partition pattern concepts and the set of sequence pattern concepts are partially ordered and forms a lattice. This ordering can be studied to obtain the hierarchical structure of the generated OP biclusters. The hierarchical and overlapping characteristics are notably similar to HOCCLUS2 method in [11], although with different bicluster type. One of the differences between our method and HOCCLUS2 is that we obtain hierarchical and overlapping biclusters directly from a matrix, while HOCCLUS2 finds a set of non-overlapping biclusters from a matrix, and then identify overlapping and hierarchically organized biclusters.

The OP biclusters can be further studied to build a collaborative recommendation systems. One example is when we have a user-item matrix, where each cell shows the rating of an item given by a user. A OP bicluster from this matrix can be regarded as a set of users having similar order of preference over a set of items.

References

1. Baixeries, J., Kaytoue, M., Napoli, A.: Characterizing functional dependencies in formal concept analysis with pattern structures. Annals of Mathematics and Artificial Intelligence **72**, 129–149 (2014)
2. Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: the order-preserving submatrix problem. Journal of computational biology **10**(3-4), 373–384 (2003)
3. Codocedo, V., Bosc, G., Kaytoue, M., Boulicaut, J.F., Napoli, A.: A proposition for sequence mining using pattern structures. In: International Conference on Formal Concept Analysis. pp. 106–121. Springer (2017)
4. Codocedo, V., Napoli, A.: Lattice-based biclustering using partition pattern structures. In: Proceedings of the Twenty-first European Conference on Artificial Intelligence. pp. 213–218. IOS Press (2014)
5. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: International Conference on Conceptual Structures. pp. 129–142. Springer (2001)
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations (1999)
7. Hedenfalk, I., Duggan, D., Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Raffeld, M., et al.: Gene-expression profiles in hereditary breast cancer. New England Journal of Medicine **344**(8), 539–548 (2001)
8. Henriques, R., Madeira, S.C.: BicSPAM: flexible biclustering using sequential patterns. BMC bioinformatics **15**(1), 130 (2014)
9. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **1**(1), 24–45 (2004)
10. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on knowledge and data engineering **16**(11), 1424–1440 (2004)
11. Pio, G., Ceci, M., D'Elia, D., Loglisci, C., Malerba, D.: A novel biclustering algorithm for the discovery of meaningful biological correlations between micrornas and their target genes. BMC Bioinformatics **14**(7), S8 (2013)
12. Wang, J., Han, J.: BIDE: Efficient mining of frequent closed sequences. In: Data Engineering, 2004. Proceedings. 20th International Conference on. pp. 79–90. IEEE (2004)
13. Yan, X., Han, J., Afshar, R.: CloSpan: Mining: Closed sequential patterns in large datasets. In: Proceedings of the 2003 SIAM international conference on data mining. pp. 166–177. SIAM (2003)
14. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine learning **42**(1-2), 31–60 (2001)