Mining complex data and biclustering using formal concept analysis

Nyoman Juniarta

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Fouille de données complexes et biclustering avec l'analyse formelle de concepts

THÈSE

présentée et soutenue publiquement le 18 décembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Nyoman Juniarta

Composition du jury

Président :	Florence Le Ber	Directrice de recherche, Université de Strasbourg/ENGEES
Rapporteurs :	Marc Plantevit	Maître de conférences, HDR, Univ. Claude Bernard Lyon 1
	Henry Soldano	Maître de conférences, HDR, Université Paris 13
Examinateurs :	Peggy Cellier	Maître de conférences, INSA Rennes
	Sara C. Madeira	Professeure associée, Université de Lisbonne
	Mohamed Nadif	Professeur, Université Paris 5
Directeur :	Amedeo Napoli	Directeur de recherche, CNRS
Co-directeur :	Miguel Couceiro	Professeur, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

In the beginning, I would like to thank my supervisor Amedeo Napoli for considering me capable enough of doing a thesis, and my first co-supervisor Chedy Raïssi, who was substituted by my second co-supervisor Miguel Couceiro halfway through the thesis. All three of them were very helpful and patient to me during those "formative" years.

I also thank my monitoring committee Bart Lamiroy and Denis Jouvet for providing me useful guides and advices during the thesis. Toward the end of thesis, I would like to thank Marc Plantevit and Henry Soldano for reviewing my manuscript and giving detailed comments and suggestions. Moreover, I thank my thesis' examiners Florence Le Ber, Peggy Cellier, Sara C. Madeira, and Mohamed Nadif for having constructive discussions with me during the defense, in spite of the nontrivial problems about train strikes combined with internet connection issue.

Concerning the funding of this thesis, I thank Région Grand Est and the CrossCult project, particularly Ioanna Lykourentzou for showing great interests in my thesis.

I started, prepared, and finished this thesis in the Orpailleur team. Therefore, I would like to thank some members (and ex-members) of this great team. First, I thank Bernard Maigret, Abdelkader Ouali, Vincent Leroux, Victor Codocedo, Mehdi Kaytoue, and Jane Hung for their valuable scientific collaborations. Then, I also thank Patrice Ringot for helping me through the difficult technical part of many experiments. Last but not least, I would like to mention those who helped me in any other unpredictable ways: Justine Reynaud, Kevin Dalleau, Lauréline Nevin, Quentin Brabant, Tatiana Makhalova, Alexandre Bazin, Guilherme Alves, Laurine Huber, Nacira Abbas, Claire Theobald, Giorgios Zervakis, and Laura Zanella.

Je dédie cette thèse
à ma famille java-balinaise

Contents

General introduction

Knowledge discovery in database

In recent years, data are collected much faster. Beside the quantities, the variability and complexity of data also grows at a dramatic pace. Consequently, there is a need to construct tools that can help us to better understand the data. This is the subject of Knowledge Discovery in Database (KDD), which takes data as input and returns patterns or models to be interpreted.

As shown in Fig. 1, KDD is composed by some basic steps [32]. The first step in KDD is understanding the domain of the problem and identifying which kind of information is expected from this problem. Second, in the *selection* process, a set of data is selected, which will be used in KDD. The result is a target dataset. In the third step, this target dataset undergoes a *preprocessing* step. It can be noise removal or missing values management. Fourth, we prepare the data set by creating a representation of it, by *transformation*. This can be done by eliminating or transforming some variables.

The next step is *data mining* tasks applied to the *transformed data*. First, the expected information matched with a specific data mining method (e.g. classification, clustering, regression, summarization, etc.). Having selected a method, an algorithm is determined and data mining process is executed. From this step, we can retain some significant *patterns* discovered in the data. The last step is *interpretation*. Having the patterns extracted, we have to interpret them, by visualisation for example. The retained patterns are then reported to concerning parties, or used directly. We can also return to some previous steps to enhance the results.

In this thesis, we are interested in knowledge discovery method applied to complex data. More precisely, this thesis focuses on the data mining and data transformation steps in KDD.

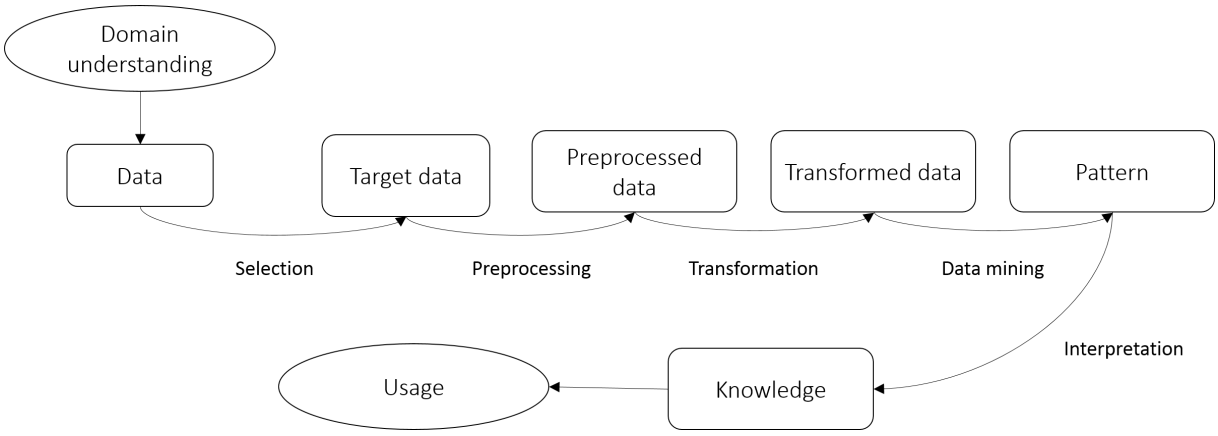


Figure 1: Diagram of some basic steps in KDD.

We had the opportunity to work on several types of complex data, namely numerical, sequential, and molecular datasets. In the data mining step, our objectives are to discover some biclusters in numerical data and to mine interesting patterns in sequential data. Formal concept analysis –which itself can be considered as a KDD method [23]– is used for completing these objectives since it is naturally related to biclustering in the sense that both of them discover submatrices indicating some regularities. Furthermore, it was shown that FCA and its extension, pattern structures, are suitable for mining complex sequential datasets [15, 19]. Concerning the data transformation step, our objective is to reduce the dimensionality of a molecular dataset by eliminating some features. This is performed by identifying some associations among features, which in turn can be used to decide which features are redundant and can be eliminated.

Contributions

As explained before, we worked on three kinds of complex data, which are gene expression data (numerical), visitor trajectory data (sequential), and antibacterial data (molecular). Several data mining methods based on FCA are used for discovering biclustering in gene expression data and for mining patterns in visitor trajectory data. In addition to data mining step, we worked on an important step in KDD which is feature selection using log-linear analysis to identify association among features/attributes. Therefore, this thesis can be summarized in three main contributions. We begin by presenting biclustering and showing how FCA is well-adapted to support several important types of biclustering. Then we discuss how FCA and pattern structures can be applied to mine sequences, which are then used to characterize visitor in a trajectory dataset. Finally, we discuss the feature selection step when working on a complex data such as antibacterial data. Each contribution is detailed as follows.

Biclustering

Biclustering with FCA is the first contribution of this thesis, working with the gene expression data as numerical dataset. Biclustering is connected to clustering. The objective of clustering is to discover some groupings of objects in a dataset by looking at their similarity across all of their features/attributes. This may be problematic, since some objects may share similarity with other objects across a subset of features only, which is particularly observed in gene expression datasets. This problem leads to the emergence of biclustering in gene expression datasets [18].

Given a matrix, formal concept analysis (FCA) is related to biclustering, in the sense that both methods try to find submatrices having regularities within their cells. FCA is focused on a binary matrix, hence it only finds constant-value biclusters. To discover other types of biclusters (as defined in [72]), some extensions of FCA are proposed. Partition pattern structures (PPS) were applied to find constant-column biclusters [20]. In this thesis, the possibility of finding other bicluster types is studied by defining the corresponding similarity operator between any two sets of objects used in PPS. Furthermore, interval pattern structures (IPS) were applied to discover similar-column biclusters [60]. The potential of IPS is studied to discover other types of biclusters. Contrasting with PPS, IPS allows to find less strict biclusters (e.g. similar column, similar row, non-exact additive biclustering, etc.) without scaling the numerical matrix. We applied our methods to the domain of bioinformatics, where the task of finding biclusters in a gene expression data matrix is still actively studied [43, 78, 81].

Sequence mining

The second contribution is related to the CrossCult project (http://www.crosscult.eu/), a European Project about cultural heritage. The general idea of CrossCult is to support the emergence of a European cultural heritage by allowing visitors in different locations (e.g. museum, city, archaeological site) to attentively contemplate their visit at a European level by using adapted computer-based devices. One objective of this project is to develop a mobile application to stimulate the reflection of history while users are visiting some cities or venues. This application registers the trajectory of each visitor while they interact with it, providing a kind of sequential dataset. A particular dataset is concerned with the trajectories of around 200 visitors in Hecht Museum in Haifa, Israel. Each trajectory is composed by sequence of visits, where each visit is represented by three components: start time, end time, and visited item. A novel method is then developed to understand these trajectories such that the visitor behaviors can be extracted according to four theoretical visiting patterns [62]. This approach is composed of two independent steps: the clustering and the mining of trajectories. Given that the trajectory dataset can be regarded as a sequential dataset, a proper sequence clustering method is used where the distance between any two sequences is obtained from the number of their common subsequences [29]. On the other hand, the mining of trajectory patterns is performed by two methods based on FCA [15, 19]. These patterns are then used to find the characteristic behavior of each cluster. This contribution may help the developer in providing recommendations to users/visitors for improving their visit and reflections.

Feature selection

The third contribution of this thesis is related to numerical pattern mining, especially the problem of feature selection as a data transformation step in KDD. This work is part on a larger research of discovering new antibacterial drugs, by classifying chemical molecules into two classes: antibacterial and non-antibacterial. In this thesis, we work on feature selection over chemoinformatics datasets where it is important in several aspects of drug design [98, 103]. Such datasets are usually represented as a matrix, where each row and column represents molecule and elements of the molecular structure respectively. Since a molecular structure has many properties, the matrix often has a high number of columns. This could increase the complexity and reduce the performance of any classification algorithm, causing the need of dimensionality reduction.

In feature selection, we start by the idea that a feature being highly correlated to another can be discarded, for obtaining a set of features that are not correlated. One way to detect correlation among attributes is given by log-linear analysis (LLA) [88]. The basic LLA examines every pair of attributes, and this is not suitable for a dataset with large number of attributes. This restriction can be solved by considering a particular sub-class of LLA: *decomposable* models [77]. This avoids the calculation of correlation of every attribute pair, and therefore reducing the complexity of LLA. The result of this feature selection approach is then tested in antibacterial datasets. An evaluation is based on how well the selected features perform in the antibacterial–nonantibacterial classification.

Thesis organization

This thesis is structured as follows.

- Chapter 1 presents the introduction of formal concept analysis and pattern structures, which will be used in the problem of sequence mining and biclustering.

The rest of this thesis is separated into two parts. The first part constitutes the first contribution about biclustering with three chapters as follows.

- Chapter 2 presents the problem of biclustering, the definition of current types of biclusters, and existing approaches to discover biclusters.
- Chapter 3 focuses on biclustering with partition pattern structures. First, a specific partition pattern structure –which was applied to constant-column and similar-column biclustering– is presented. Then, this approach is extended to other types of biclustering by adapting the description of each object and the similarity operation. This chapter is based on:
  - N. Juniarta, V. Codocedo, M. Couceiro, A. Napoli, *Biclustering Based on FCA and Partition Pattern Structures for Recommendation Systems* at the Proceedings of the 2018 International Workshop "What can FCA do for Artificial Intelligence?" @ 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence, pp. 105–116.
  - N. Juniarta, M. Couceiro, A. Napoli, *Application des Pattern Structures à la découverte de biclusters à changements de signes cohérents* at Conference Francophone sur l'Extraction et la Gestion des Connaissances (EGC) 2019, pp. 285–290.
  - N. Juniarta, V. Codocedo, M. Couceiro, M. Kaytoue, A. Napoli *Pattern Structures for Identifying Biclusters with Coherent Sign Changes* at the Supplementary Proceedings of the 2019 International Conference on Formal Concept Analysis, pp. 3–15.
  - N. Juniarta, M. Couceiro, A. Napoli, *Order-preserving Biclustering Based on FCA and Pattern Structures.* In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, Z.W. Ras (Eds), Complex Pattern Mining: New Challenges, Methods and Applications. Basel: Springer. (to be published)
  - N. Juniarta, M. Couceiro, A. Napoli, *A Unified Approach to Biclustering Based on Formal Concept Analysis* at Conférence sur la Gestion de Données – Principes, Technologies et Applications (BDA) 2019.
- Chapter 4 focuses on biclustering with interval pattern structures. Following the previous chapter, here we first explain the specific interval pattern structure which was built for discovering similar-column biclustering. Then, using alignment, we show that this approach can be extended to discover other types of biclusters, especially the additive and multiplicative biclusters. This chapter is based on:
  - N. Juniarta, M. Couceiro, A. Napoli, *A Unified Approach to Biclustering Based on Formal Concept Analysis and Interval Pattern Structures* at the 2019 International Conference on Discovery Science, pp. 51–60.

The second part represents the second contribution (about mining sequential dataset) and the third contribution (about feature selection) of this thesis. This part has two chapters as follows.

- Chapter 5 focuses on the task of describing visitor behaviors, using sequence clustering and FCA-based sequence mining. First, the formalization of sequence mining in FCA is presented. Then, combined with clustering of sequence using the count of "all common subsequences", the characterization of visitor trajectories is described. This chapter is based on:

- N. Juniarta, M. Couceiro, A. Napoli, C. Raïssi, *Sequential Pattern Mining using FCA and Pattern Structures for Analyzing Visitor Trajectories in a Museum* at the Proceedings of the 2018 International Conference on Concept Lattices and Their Applications, pp. 231–242.
- N. Juniarta, M. Couceiro, A. Napoli, C. Raïssi, *Sequence Mining within Formal Concept Analysis for Analyzing Visitor Trajectories* at the 2018 International Workshop on Semantic and Social Media Adaptation and Personalization, pp. 19–24.

- Chapter 6 focuses on the numerical pattern mining domain, where we study the application of attribute relation discovery to solve the problem of feature selection. The attribute relation discovery is performed using LLA, which is explained first. Then, we study the improvement of LLA based on chordal graph. Finally, we apply it to the antibacterial datasets and some publicly available datasets to understand the performance of the method. This chapter is based on:
  - A. Ouali, N. Juniarta, B. Maigret, A. Napoli, *A Feature Selection Method Based on Tree Decomposition of Correlation Graph* at the 2019 Workshop on Advances in Managing and Mining Large Evolving Graphs @ European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.

Chapter 1

Formal concept analysis and pattern structures

Contents

In this chapter, we introduce the formal concept analysis which will be used in the following chapters. First, we explain in Sect. 1.1 the basic definitions of order theory based on [36]. This theory is necessary to introduce the formal concept analysis in Sect. 1.2, and some algorithms for mining formal concepts in Sect. 1.3. In the end, pattern structures –a generalization of formal concept analysis– is presented in Sect. 1.4.

1.1 Order theory

A *binary relation* R between two sets M and N is a set of pairs (m, n) with $m \in M$ and $n \in N$. It can be written as mRn. Instead of two different sets, a binary relation can also be defined on a single set ($M = N$), and it is called a binary relation on M. An order on M is a binary relation on that set that satisfies the following conditions for all $x, y, z \in M$:

1. xRx (reflexivity),
2. xRy and $x \neq y \Rightarrow$ not yRx (antisymmetry),
3. xRy and $yRz \Rightarrow xRz$ (transitivity).

An order is often written as $\leq$. An ordered set is a pair $(M, \leq)$, with $\leq$ is an order on M. This type of set is sometimes denoted as *partially ordered set*, since it is possible that not every pair of elements is comparable. This means that there may exist two different elements x and y that are incomparable, where $x \not\leq y$ and $y \not\leq x$.

Table 1.1: A formal context with four objects and five attributes.

	m_1	m_2	m_3	m_4	m_5
g_1				×	×
g_2	×		×		
g_3	×	×	×	×	
g_4		×	×	×	

Consider a set C, a subset of an ordered set $(M, \leq)$. An element $m \in M$ is a lower bound of C if $\forall c \in C, m \leq c$. Dually, an element $n \in M$ is an upper bound of C if $\forall c \in C, c \leq n$. If there exists a single largest element among the set of all lower bounds of C, then this element is denoted the *infimum* of C. The *supremum* is defined dually.

If for any two elements $x, y \in M$, there exist the supremum $(x \vee y)$ and infimum $(x \wedge y)$, then this ordered set is a lattice. Furthermore, if the supremum and infimum also exist for any $X \subseteq M$, then this ordered set is a complete lattice.

1.2 Formal concept analysis

Formal concept analysis (FCA) is a mathematical framework based on lattice theory and used for classification, data analysis, and knowledge discovery [36]. From a formal context, FCA detects all formal concepts and the order among them in a concept lattice.

A formal context $\mathcal{K}$ 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 $(g, m) \in I$. A formal context can be represented as a matrix, an example is shown in Table 1.1.

In a formal context, FCA finds *maximal rectangles*. A rectangle is a submatrix whose cells are entirely ×. An example of such rectangle in Table 1.1 is the submatrix r_1 constructed by rows $\{g_3, g_4\}$ and columns $\{m_2, m_3\}$. Furthermore, a rectangle is maximal if there is no other larger rectangle that can be created by adding rows or columns to the current rectangle. Therefore, the rectangle r_1 is not maximal, since another larger rectangle r_2 can be formed by adding m_4 to the rows of r_1. On the other hand, the rectangle r_2, having rows $\{g_3, g_4\}$ and columns $\{m_2, m_3, m_4\}$, is maximal.

A maximal rectangle corresponds to a *formal concept* in FCA. Its set of rows and set of columns are called its *extent* and its *intent* respectively. Formal concepts can be found by introducing derivation operators for any subset of objects and any subset of attributes:

$$\begin{aligned} ' &: \wp(G) \to \wp(M), \\ ' &: \wp(M) \to \wp(G), \end{aligned} \tag{1.1}$$

where $\wp(G)$ and $\wp(M)$ represent the power set of G and M respectively. Given $A \subseteq G$ and $B \subseteq M$, these operators are defined by:

$$\begin{aligned} A' &= \{m \in M | \forall g \in A, (g, m) \in I\}, \\ B' &= \{g \in G | \forall m \in B, (g, m) \in I\}. \end{aligned} \tag{1.2}$$

In other words, A' is the set of attributes that are shared by all objects in A, while B' is the set of objects having all attributes in B. Therefore, these two operators form a Galois connection between $\wp(G)$ and $\wp(M)$ [36].

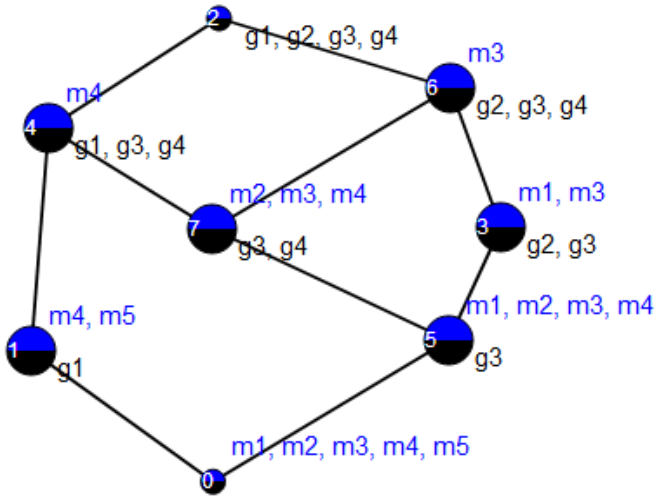


Figure 1.1: Concept lattice of the formal context in Table 1.1. The number inside each node is the concept number. The set written above and below each node represents the concept's intent and extent respectively. The diagram is generated using Latviz [4].

A formal concept is a pair (A, B) where $A' = B$ and $B' = A$. In the previous example, the rectangle r_2 is a formal concept since $\{g_3, g_4\}' = \{m_2, m_3, m_4\}$ and $\{m_2, m_3, m_4\}' = \{g_3, g_4\}$. This formal concept is written as $(\{g_3, g_4\}, \{m_2, m_3, m_4\})$. The set of all formal concepts $\mathfrak{B}(G, M, I)$ is partially ordered, given by relation $\leq_{\mathcal{K}}$:

$$(A_1, B_1) \leq_{\mathcal{K}} (A_2, B_2) \iff A_1 \subseteq A_2 \text{ (dually } B_2 \subseteq B_1). \tag{1.3}$$

$\mathfrak{B}(G, M, I)$ is a lattice, since for any two concepts, the supremum and the infimum always exist. Furthermore, it is also a complete lattice since there exist the supremum and infimum of any $\mathcal{B} \subseteq \mathfrak{B}(G, M, I)$.

The concept lattice of the formal context in Table 1.1 is depicted in a Hasse diagram in Fig. 1.1. This diagram reflects the partial ordering of $\mathfrak{B}(G, M, I)$. Concept number 5 ($c_5 = (\{g_3\}, \{m_1, m_2, m_3, m_4\})$) is less than concept number 6 ($c_6 = (\{g_2, g_3, g_4\}, \{m_3\})$), written as $c_5 \leq_{\mathcal{K}} c_6$. Therefore, there is a path going down from c_6 to c_5. Meanwhile, c_3 and c_7 is not comparable, since neither $c_3 \leq_{\mathcal{K}} c_7$ nor $c_7 \leq_{\mathcal{K}} c_3$. Therefore, there is no path going entirely up or entirely down between these two concepts.

1.3 Formal concept miners

Various algorithms have been proposed to enumerate all formal concepts from a formal context and/or building the corresponding concept lattice. In this thesis, we used the Close by One and AddIntent algorithms since they can be easily adapted to pattern structures (will be explained later). We also explain here In-Close2 algorithm, which was used in the literature for FCA-based biclustering.

1.3.1 Close by One

Close by One (CbO) was proposed in [63], and detailed here in Algo. 1. The time complexity of CbO is $O(|G|^2|M||L|)$ [65], where L is the number of concepts. It starts from a single object, then adding objects one by one according to the lexicographical order of objects. When an object is added to a current set, the algorithm checks the set's closure (the $(.)''$ operation).

The process of concepts generation of the formal context in Table 1.1 is illustrated as a tree in Fig. 1.2. Each node which is not crossed corresponds to a formal concept. First, the set $\{g_1\}$ and its closure $\{g_1\}$ is checked, and kept as a concept $(\{g_1\}, \{m_4, m_5\})$. Then, the next object is added to the previous closure, forming the set $\{g_1, g_2\}$ with closure $\{g_1, g_2, g_3, g_4\}$. Since, there is no more object can be added to the preceding closure (line 12 of the algorithm), this branch is terminated.

The node with $\{g_1, g_4\}$ is not kept as a concept, since it is not canonical. It tries to generate a concept $(\{g_1, g_3, g_4\}, \{m_3\})$, but this concept is already generated previously, from the node $\{g_1, g_3\}$. This checking is done in the line 10 of the algorithm. The closure of the set $\{g_1, g_4\}$ is $\{g_1, g_3, g_4\}$, containing g_3 that is lexicographically less than another object in the set (g_4).

```
1  CloseByOne:
2  L := ∅
3  foreach g ∈ G do
4  |  Process({g}, g, (g'', g'))
5  end
6  L is the set of concepts
7
8
9  Process(A, g, (C, D)):
10 if {h|h ∈ C\A and h < g} = ∅ then
11 |  L := L ∪ {(C, D)}
12 |  foreach f ∈ {h|h ∈ G\C and g < h} do
13 |  |  Z := C ∪ {f}
14 |  |  Y := D ∩ {f}'
15 |  |  X := Y'
16 |  |  Process(Z, f, (X, Y))
17 |  end
18 end
```

Algorithm 1: Close by One algorithm. The input is a formal context (G, M, I), while the output is the set of formal concepts $\mathfrak{B}(G, M, I)$, denoted as L in the algorithm.

1.3.2 AddIntent

The result of CbO is the set of concepts $\mathfrak{B}(G, M, I)$. If the lattice of the concepts is also expected, then we need another approach. AddIntent [73] is an algorithm that enumerates the set of concepts and generates its lattice. This algorithm, shown in Algo. 2–3, is a bottom-up approach. In general, each time an object is added, the algorithm calculates the possible intersections between the object's intent and the intents of all other concepts in the current lattice. Then, the algorithm generates new concepts based on these intersection and connects them accordingly to the existing concepts.

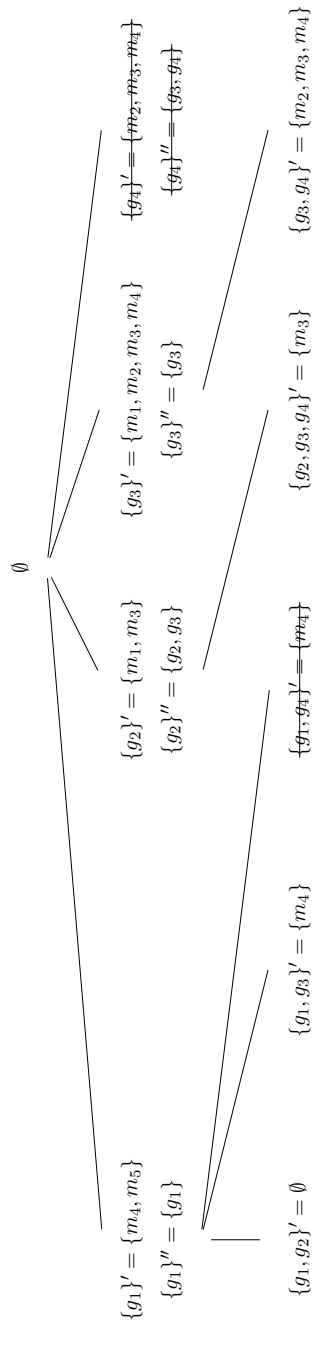


Figure 1.2: Generations of concepts of the formal context in Table 1.1 by Close by One algorithm. The nodes are generated depth-first.

```
1 CreateLatticeIncrementally(G, M, I):
2 BottomConcept := (∅, M)
3 L := BottomConcept
4 foreach g ∈ G do
5 |   ObjectConcept = AddIntent(g′, BottomConcept, L)
6 |   Add g to the extent of ObjectConcept and all concepts above
7 end
```

Algorithm 2: The procedure to generate concept lattice, using AddIntent algorithm.

As shown in Algo. 2, AddIntent starts with a lattice with the bottom concept $(\emptyset, M)$. Then, the algorithm calls `AddIntent` function (Algo. 3) for each object $g \in G$, with g' as `intent`. The function first looks for the `GeneratorConcept` (line 2) from which `NewConcept` may be generated. The function `GetMaximalConcept` finds the most general concept whose intent $\supseteq$ *intent*. The parents of `NewConcept` can be found among the parents of `GeneratorConcept`. For every parent of `GeneratorConcept` (line 8), the algorithm goes up until it finds a concept whose intent is a subset of `intent` (done recursively in line 9–10). This concept is added to `NewParents` after checking whether it is comparable to the existing `NewParents` (line 13–24). After that, the `NewConcept` is added between any concept in `NewParents` and its `GeneratorConcept`.

An example of AddIntent concept generations is illustrated in Fig. 1.3, working with the formal context in Table 1.1. Consider Fig. 1.3c, which is the lattice after the addition of g_2. The next object to be added is g_3, with $\{g_3\}' = \{m_1, m_2, m_3, m_4\}$ as `intent`. The `GeneratorConcept` for this object is c_0, having two parents c_1 and c_3. From c_1, the algorithm finds c_2 as the new parents, and inserts a new concept c_4 between c_1 and c_2. Then from c_3, the algorithm finds c_3 and c_4 as the new parents, thus inserts a new concept c_5 between c_0 and both of c_3 and c_4.

1.3.3 In-Close and In-Close2

In-Close2 algorithm [8] and its predecessor In-Close [7] are based on CbO. Both of them start from the pair $(G, \emptyset)$ and adding attributes one by one to discover other concepts. To see the difference between In-Close and CbO, here we will consider that In-Close start from the pair $(\emptyset, M)$ and it adds objects one by one.

The running of In-Close algorithm is illustrated in Fig. 1.4. The algorithm starts from the top concept $(\emptyset, M)$, and adds objects by their lexicographical order. Each time an object is added to the current extent A, the algorithm calculates the new intent A' by doing the intersection. There are three outcomes of the new intent:

1. O_1: Emptyset,
2. O_2: A proper subset of the previous intent,
3. O_3: Same as the previous intent.

Consider the node where c_2 is generated in Fig. 1.4, after the addition of g_1. The algorithm then separately adds g_2, g_3, and g_4; hence three branches in the figure. The new intent from the addition of g_2 is $\emptyset$, corresponding to O_1. In this case, the algorithm stops exploring this branch. The new intent from the addition of g_3 is a proper subset of the previous intent $\{m_4, m_5\}$, corresponding to O_2. Therefore, the new pair $(\{g_1, g_3\}, \{m_4\})$ is accepted as a new concept c_3. Then, the algorithm continues this branch, adding g_4. The new intent is the same with the

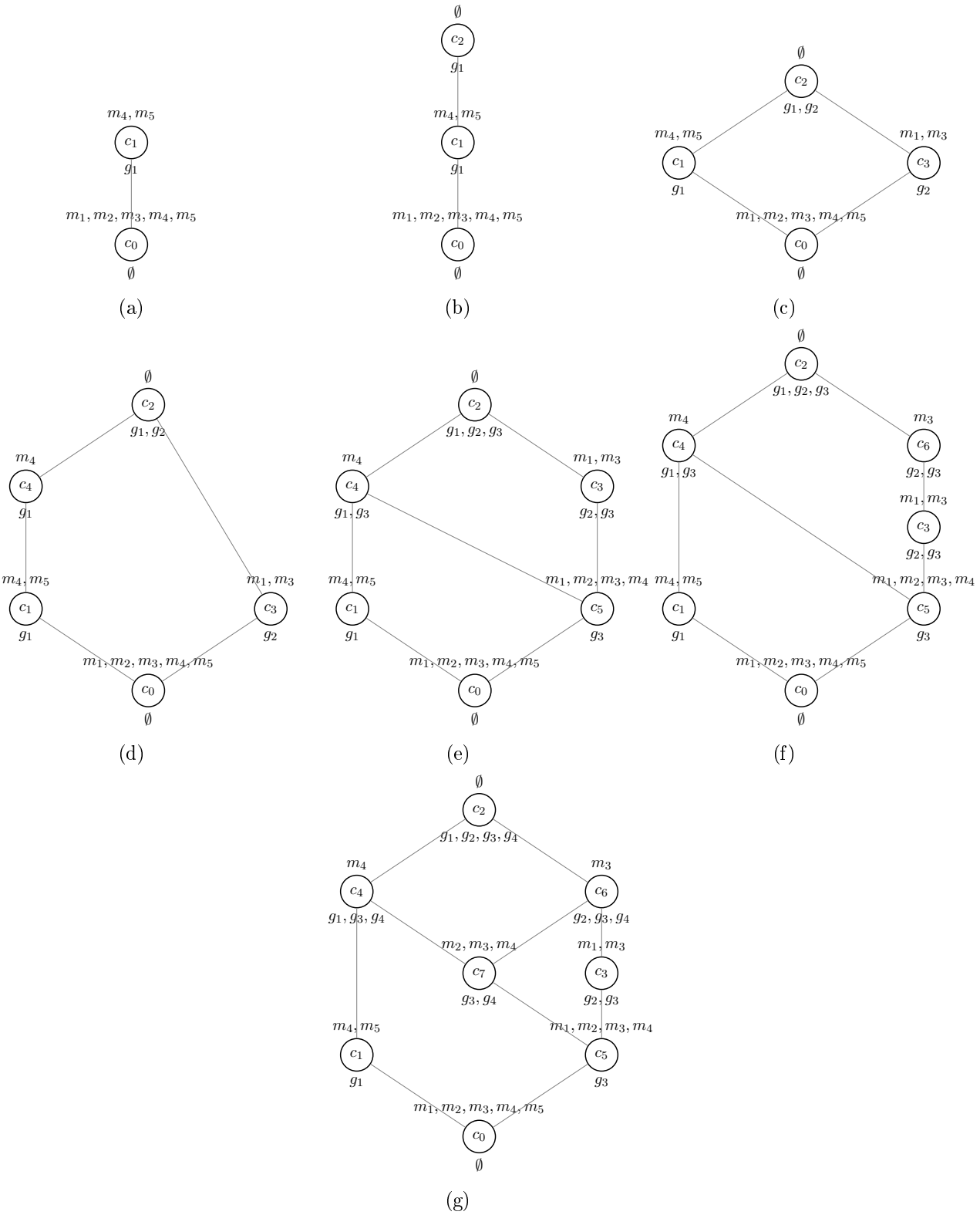


Figure 1.3: The generation of formal concepts using AddIntent for the formal context in Table 1.1, showing the steps after addition of g_1 (a) until the final lattice (g). The figures (b–c), (d–e), and (f–g) correspond to the additions of g_2, g_3, and g_4 respectively.

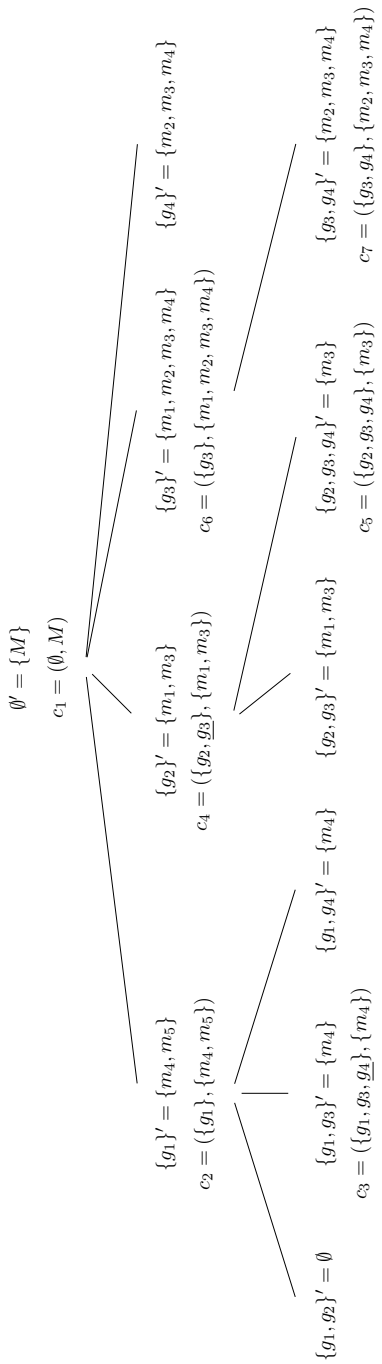


Figure 1.4: Generations of concepts of the formal context in Table 1.1 using In-Close algorithm by object additions. The nodes are generated depth-first. Each time a concept is generated, it is symbolized by $c_n = (A, B)$. The underlined object in an extent means that the object is added after the calculation from the nodes below.

previous intent ($\{m_4\}$), corresponding to O_3. Therefore, the object g_4 is added to the previous extent, modifying the concept c_3 such that its extent is $\{g_1, g_3, g_4\}$.

Moreover, before the acceptation of the new concept from O_2, the algorithm checks whether the new intent is already generated. This is the case with the addition of g_4 to the concept c_1. Although the new intent is a proper subset of the intent of c_1, the pair $(\{g_1, g_4\}, \{m_4\})$ is not accepted as a concept since the intent $\{m_4\}$ is already generated in c_3.

If we compare the exploration of In-Close in Fig. 1.4 with the exploration of CbO in Fig. 1.2, we see some differences. Although both algorithms add objects (or attributes) one by one using lexicographical order, they differ on how the concepts are generated. In CbO, a concept is generated by calculating the closure $(.)''$ of a set of objects. In In-Close, the intent of a concept is generated first, then its extent is calculated recursively. On the other hand, in In-Close2 –an improvement of In-Close– combines depth-first and breadth-first approach when adding objects/attributes.

1.4 Pattern structures

As explained in the previous section, a formal context $\mathcal{K}$ is a triple (G, M, I) where I is a binary relation between G and M. This means that we can not have multi-valued relation as I. This limitation is relaxed in pattern structures [35], generalizations of FCA. In FCA, we can say that each object is described by a set of attributes. On the other hand, in pattern structures, each object can be described by a more complex structure, e.g. a list of interval [60], a sequence [15, 19], a partition [20], an RDF triple [3], etc.

Instead of (G, M, I), a pattern structure is determined by the triple $(G, (D, \sqcap), \delta)$, where $(D, \sqcap)$ is a meet-semilattice of descriptions D having $\sqcap$ as similarity operator, and $\delta : G \to D$ maps an object to its description. Following Eq. 1.1, the derivation operators are defined for any subset of objects and any description:

$$\begin{aligned} {}^{\square} &: \wp(G) \to D, \\ {}^{\square} &: D \to \wp(G). \end{aligned}$$

Given $A \subseteq G$ and $d \in D$, these operators –comparable with Eq. 1.2– are defined by:

$$\begin{aligned} A^{\square} &= \prod_{g \in A} \delta(g), \\ d^{\square} &= \{g \in G | d \sqsubseteq \delta(g)\}. \end{aligned}$$

In other words, $A^{\square}$ is the similarity of descriptions of each object in A, while $d^{\square}$ is the set of objects whose description subsumes d. The subsumption order over these descriptions follows that:

$$d_1 \sqcap d_2 = d_1 \Leftrightarrow d_1 \sqsubseteq d_2. \tag{1.4}$$

A pattern concept is a pair (A, d) where $A^{\square} = d$ and $d^{\square} = A$.

Consider a standard FCA and the formal context in Table 1.1. In pattern structures, each object in this context has a set of attributes as description. For example, $\delta(g_1) = \{m_4, m_5\}$. The $\sqcap$ in this case corresponds to the set intersection ($\cap$). Therefore:

$$\begin{aligned} \delta(g_2) \sqcap \delta(g_3) &= \delta(g_2) \cap \delta(g_3) \\ &= \{m_1, m_3\} \cap \{m_1, m_2, m_3, m_4\} \\ &= \{m_1, m_3\}. \end{aligned}$$

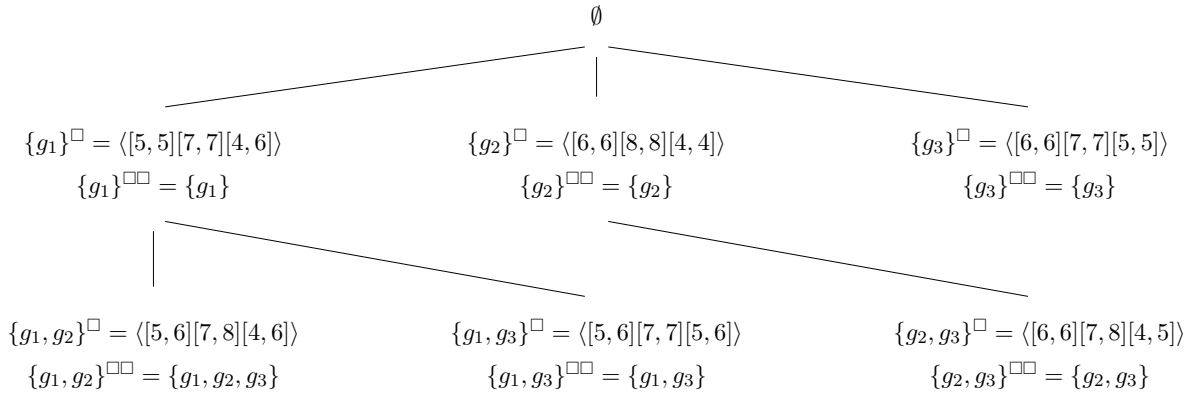


Figure 1.5: Generations of interval pattern concepts of the context in Table 1.2 using CbO algorithm. It is similar to Fig. 1.2, with differences on derivation operators.

If $A = \{g_2, g_3, g_4\}$, then $A^\square = \delta(g_2) \sqcap \delta(g_3) \sqcap \delta(g_4) = \{m_3\}$, which is consistent with A' in FCA.

The relation "is subsumed by" ($\sqsubseteq$) corresponds to subset relation ($\subseteq$) in FCA. In this case $d^\square$ is the set of objects whose description is a superset of d. Still in Table 1.1, with $d = \{m_3\}$, $d^\square = \{m_3\}^\square = \{g_2, g_3, g_4\}$, consistent with $\{m_3\}'$ in FCA. Furthermore, since $\{g_2, g_3, g_4\}^\square = \{m_3\}$ and $\{m_3\}^\square = \{g_2, g_3, g_4\}$, the pair $(\{g_2, g_3, g_4\}, \{m_3\})$ is a pattern concept, which corresponds to a formal concept in FCA.

Using pattern structures, not only a set of attributes, we can have a more complex description by defining the operator $\sqcap$, as long as the set of description is a meet-semilattice. In other words, the set of descriptions must be a partially ordered set having an infimum for any two descriptions. To enumerate all pattern concepts, CbO algorithm can be used by redefining the operator and relation $(.)', \cap, \subseteq$ to $(.)^\square, \sqcap, \sqsubseteq$ respectively.

Other examples of pattern structures are interval pattern structures (IPS) [60]. In IPS, a description of an object is a list of attribute intervals. Consider a numerical context in Table 1.2. The description of g_1 is $\delta(g_1) = \langle[5,5][7,7][6,6]\rangle$. The similarity operator between two descriptions in IPS is defined as the list of smallest interval covering all the corresponding interval in the descriptions. For example, $\delta(g_1) \sqcap \delta(g_2)$ is $\langle[5,6][7,8][4,6]\rangle$. Following Eq. 1.4, larger interval is subsumed by smaller interval, e.g. $\langle[5,6][7,8][4,6]\rangle \sqsubseteq \langle[5,5][7,8][5,6]\rangle$. Using CbO, the process of interval pattern concepts is illustrated in Fig. 1.5.

More detail on IPS is provided in Chapter 4 of this thesis. Other complex structures, like partition and sequence, are explained in Chapter 3 and Chapter 5 respectively.

```
1  AddIntent(intent, GeneratorConcept, L):
2  GeneratorConcept = GetMaximalConcept(intent, GeneratorConcept, L)
3  if GeneratorConcept.Intent = intent then
4  |  return GeneratorConcept
5  end
6  GeneratorParents := GetParents(GeneratorConcept, L)
7  NewParents = ∅
8  foreach Candidate ∈ GeneratorParents do
9  |  if Candidate.Intent ⊄ intent then
10 |  |  Candidate := AddIntent(Candidate.Intent ∩ intent, Candidate, L)
11 |  end
12 |  addParent := true
13 |  foreach Parent ∈ NewParents do
14 |  |  if Candidate.Intent ⊆ Parent.Intent then
15 |  |  |  addParent := false
16 |  |  |  break
17 |  |  end
18 |  |  else if Parent.Intent ⊆ Candidate.Intent then
19 |  |  |  Remove Parent from NewParents
20 |  |  end
21 |  end
22 |  if addParent then
23 |  |  Add Candidate to NewParents
24 |  end
25 end
26 NewConcept := (GeneratorConcept.Extent, intent)
27 L := L ∪ {NewConcept}
28 foreach Parent ∈ NewParents do
29 |  RemoveLink(Parent, GeneratorConcept, L)
30 |  SetLink(Parent, NewConcept, L)
31 end
32 SetLink(NewConcept, GeneratorConcept, L)
33 return NewConcept
```

Algorithm 3: AddIntent algorithm.

	m_1	m_2	m_3
g_1	5	7	6
g_2	6	8	4
g_3	6	7	5

Table 1.2: A numerical context.

Part I

Biclustering in FCA

Chapter 2

Introduction to biclustering

Contents

Biclustering is a simultaneous grouping of rows and columns in a matrix. It is related to FCA, in the sense that both methods discover submatrices having some regularities among their elements. In this chapter, we introduce the definitions of biclustering and some previous works. These definitions will be used in the next chapters, i.e. biclustering with partition and interval pattern structures.

2.1 Clustering and biclustering

Before describing the definitions of biclustering, we first revisit the clustering task. Clustering is a part of the data mining step in KDD, where we discover some patterns that tell us the possible existence of similarity among the objects. In general, this task groups a set of objects based on the attributes of each object. However, the precise definition of "cluster" depends on the algorithms [31]. For example, in k-means clustering, a cluster can be represented as a single vector that is called a centroid, where an object belongs to the nearest centroid. In hierarchical clustering [56], a cluster is a recursive partition of a larger set of objects into some smaller subsets. In DBSCAN [30], a cluster is a group of objects that are closely packed together.

In spite of various definitions of cluster, the majority of clustering algorithms group the objects based on all of the attributes together. On the other hand, biclustering is a unique type of clustering where an object can be grouped with another object even if they share a similarity across a subset of attributes only. In other words, biclustering is clustering of objects and attributes simultaneously [40]. From this point of view, biclustering is naturally related to FCA and pattern structures, since a bicluster can be regarded as a concept.

Table 2.1: Examples of some bicluster types: (a) constant-value, (b) constant-column, (c) similar-column, (d) additive, (e) multiplicative, (f) order-preserving, (g) coherent-sign-changes.

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

(a)

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

(b)

4.0	2.5	5.7	3.1
3.9	2.4	5.6	3.0
4.1	2.4	5.9	2.9
4.1	2.6	5.8	2.9

(c)

2	3	5	1
5	6	8	4
1	2	4	0
4	5	7	3

(d)

2	6	12	4
1	3	6	2
4	12	24	8
3	9	18	6

(e)

2	1	3	4
5	1	8	9
6	5	7	11
1	-9	2	9

(f)

+	+	−	+
+	+	−	+
−	−	+	−
+	+	−	+

(g)

2.2 Basic definitions

We consider that a dataset is a matrix (G, M) where G is a set of rows/objects and M is a set of columns/attributes. The value of $m \in M$ for object $g \in G$ is written as $m(g)$. A submatrix constructed from a subset of rows $A \subseteq G$ and a subset of columns $B \subseteq M$ is denoted as (A, B).

The first type of biclusters is *constant-value*, which is a submatrix having the same value for all of its elements. In a binary matrix, a formal concept corresponds to a constant-value bicluster. This bicluster is described in Def. 1.

Definition 1 (Constant-value bicluster)**.** Given a dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is a constant-value bicluster iff $\forall g \in A, \forall m \in B, m(g) = c$ where c is a constant.

Other types of bicluster are formalized in the following definitions.

Definition 2 (Constant-column bicluster)**.** Given a dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is a constant-column (CC) bicluster iff $\forall m \in B, \forall g, h \in A, m(g) = m(h)$.

An example of CC bicluster is illustrated in Table 2.1b. CC biclustering has more relaxed variation, namely similar-column (SC) biclustering. With these relaxations, instead of finding biclusters with exactly constant columns, we can obtain biclusters whose columns have similar values as shown in Table 2.1c.

An additive bicluster is illustrated in Table 2.1d. Here we see that there is a constant difference between any two columns. For example, each value in the second column is two more than the corresponding value in the fourth row. This type and the related multiplicative biclusters (example in Table 2.1e) are formalized as follows.

Definition 3 (Coherent-values bicluster)**.** Given a dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is an additive bicluster iff $\forall g, h \in A, \forall m, n \in B, m(g) - n(g) = m(h) - n(h)$; or a multiplicative bicluster iff $\forall g, h \in A, \forall m, n \in B, m(g)/n(g) = m(h)/n(h)$.

Order-preserving biclustering is another type of biclustering where the expected submatrix has a common permutation of columns for all rows. In the example (Table 2.1f), there is a sequence of column: second $<$ first $<$ third $<$ fourth column, that is true for all rows. This type of bicluster is formalized as follows.

Definition 4 (Order-preserving bicluster)**.** Given a dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is an order-preserving bicluster iff there exists a sequence of columns $(m_1 \cdots m_{|B|})$ for all $m_i \in B$, such that $m_i(g) < m_{i+1}(g)$ for all $g \in A$.

Lastly, another type of biclustering that is studied in this thesis is coherent-sign-changes biclustering, illustrated in Table 2.1g. In this bicluster, each column is "equal" ($\simeq$) to each other: the first, second, and fourth columns are identical, while the third column is opposite of the others. This bicluster is formalized in the following definitions.

Definition 5 (Column submatrix)**.** In a binary dataset (G, M) where G is a set of objects and M is a set of attributes, (A, m_j) is the column submatrix with $A \subseteq G$ and $m_j \in M$.

Definition 6 (Column submatrix equality)**.** Given a set of objects $A \subseteq G$ and two attributes $m_j, m_k \in M$. The submatrix (A, m_j) is equal to (A, m_k), denoted as $(A, m_j) \simeq (A, m_k)$, iff all rows in (A, m_j) are either entirely identical or entirely opposite to the corresponding rows in (A, m_k). This can be formally written as:

$$(A, m_j) \simeq (A, m_k) \Longleftrightarrow \forall g_i \in A : m_j(g_i) = m_k(g_i) \text{ or } \forall g_i \in A : m_j(g_i) = \neg m_k(g_i).$$

Definition 7 (Coherent-sign-changes bicluster)**.** Given a binary dataset (G, M), a pair (A, B) (where $A \subseteq G$, $B \subseteq M$) is a coherent-sign-changes bicluster iff $\forall m_j, m_k \in B : (A, m_j) \simeq (A, m_k)$.

2.3 Previous works

Various authors have proposed comparative studies of biclustering in gene expression data [53, 72, 74]. In this section, we briefly present certain existing methods in the literature that work with some/all the types of biclusters described in the previous section.

2.3.1 Probabilistic and numerical methods

In the numerical biclustering, a bicluster is a submatrix having minimal deviation of their elements. On the other hand, in the probabilistic approaches, biclustering is regarded as a task of finding a submatrix having low possibility to be observed, hence significant. Some biclustering methods in this category are described below.

Cheng and Church

Cheng and Church [18] was among the first algorithms about biclustering. They work on gene-condition expression matrix, where the value of each cell represents the expression level of the corresponding gene in the corresponding condition. Here a bicluster (A, B) is defined as a submatrix having *mean squared residue* less than a certain threshold δ. A mean squared residue is:

$$H(A, B) = \frac{1}{|A||B|} \sum_{g \in A, m \in B} (m(g) - \mu(g, B) - \mu(A, m) + \mu(A, B))^2, \tag{2.1}$$

where $\mu(A, B)$ is the mean of a submatrix (A, B). The lowest mean square residue, where $H(A, B) = 0$ is obtained from a submatrix that corresponds to a constant-value, constant-column, constant-row, or additive bicluster. Consequently, if the mean square residue is not zero but is low enough, the submatrix can be regarded as a similar-value, similar-column, similar-row, or "relaxed" additive bicluster. This bicluster represents a group of genes having similar regulation under a set of conditions.

The authors describe several algorithms that discover a bicluster (A, B) having $H(A, B) \leq \delta$. Generally, these algorithms find biclusters by performing deletions and additions of rows and columns from the whole matrix (G, M), until the condition $H(A, B) \leq \delta$ is met. The basic method is by performing brute-force approach for each possible row/column addition/deletion, which is not efficient for a large data matrix. The first algorithm proposed by the author is single node deletion, shown in Algo. 4. It starts from the original matrix (G, M), and iteratively removes a row or a column that reduces the mean square residue the most. It stops when a submatrix with mean square residue $\leq \delta$ is obtained.

1 **Single Node Deletion**$((G, M), \delta)$:
2 $A = G$, $B = M$
3 **while** True **do**
4 Compute $\mu(g, B)$ for all $g \in A$
5 Compute $\mu(A, m)$ for all $m \in B$
6 Compute $\mu(A, B)$
7 Compute $H(A, B)$
8 **if** $H(A, B) \leq \delta$ **then**
9 **return** (A, B)
10 **end**
11 Find the row $g \in A$ with the largest
$d(g) = \frac{1}{|B|} \sum_{m \in B} (m(g) - \mu(g, B) - \mu(A, m) + \mu(A, B))^2$
12 and the column $m \in B$ with the largest
$d(m) = \frac{1}{|A|} \sum_{g \in A} (m(g) - \mu(g, B) - \mu(A, m) + \mu(A, B))^2$
13 **if** the largest $d(g) >$ the largest $d(m)$ **then**
14 remove g from A
15 **end**
16 **else**
17 remove m from B
18 **end**
19 **end**

Algorithm 4: Single node deletion algorithm. From the input (G, M) as the matrix and δ as a threshold, this algorithm outputs a bicluster (A, B) having $H(A, B) \leq \delta$.

As the single node deletion removes a row/column one-by-one, it becomes expensive for a larger matrix. The authors also propose the multiple node deletion, where the matrix is significantly reduced, before applying single node deletion.

Furthermore, given a bicluster with $H(A, B) \leq \delta$, the node addition algorithm was also presented to add rows/columns to obtain a larger bicluster. This algorithm is shown in Algo. 5. It starts from a bicluster (A, B), then adds a column and a row such that the new mean square residue is not increasing. It also provides a way to add inverted rows (line 10–12), which can be regarded as co-regulated genes receiving opposite regulations.

Double k-means

K-means is a method for clustering a set of objects based on their attributes into k cluster. It works by finding a centroid for each cluster, such that the sum of distances between every object and its nearest centroid is minimized. In a given data matrix, k-means can be considered as partitioning the set of rows.

Node Addition$((G, M), (A, B))$:
while True **do**
  Compute $\mu(g, B)$ for all $g \in A$
  Compute $\mu(A, m)$ for all $m \in B$
  Compute $\mu(A, B)$
  Compute $H(A, B)$
  Add the column $m \notin B$ to B with
    $\frac{1}{|A|} \sum_{g \in A} (m(g) - \mu(g, B) - \mu(A, m) + \mu(A, B))^2 \leq H(A, B)$
  Recompute $\mu(g, B)$, $\mu(A, B)$, and $H(A, B)$
  Add the row $g \notin A$ to A with
    $\frac{1}{|B|} \sum_{m \in B} (m(g) - \mu(g, B) - \mu(A, m) + \mu(A, B))^2 \leq H(A, B)$
  foreach $g \notin A$ **do**
    add the inverse of g to A if
      $\frac{1}{|B|} \sum_{m \in B} (-m(g) + \mu(g, B) - \mu(A, m) + \mu(A, B))^2 \leq H(A, B)$
  end
  If nothing is added to A or B, then **return** (A, B)
end

Algorithm 5: Node addition algorithm. From the original matrix (G, M) and a bicluster (A, B), it generates a larger bicluster with non-increasing mean square residue, which is (A, B) with some rows/columns added.

The adaptation of k-means modeling into the task of biclustering is studied in double k-means [95], where the partitioning is performed on both rows and columns, creating disjoint blocks in the matrix. It implies that for any two rows, their column partitions is the same, although these rows may not be in the same row partition. This limitation is then relaxed in generalized double k-means [85], such that two row partitions may have different column partitions. Furthermore, the double k-means principle is also studied in creating diagonal co-clustering (illustrated in Fig. 2.2) in a sparse matrix [66].

SAMBA

SAMBA [90] is a numerical method to discover statistically significant biclusters. It is also applied to gene expression data, by first constructing a bipartite graph where the vertices on one side correspond to the conditions and the vertices on the other side correspond to the genes. An edge connects a gene g to a condition m if the expression level of g significantly changes in m. The task of biclustering is then performed as the task of finding heavy bicliques in the bipartite graph.

The weight of a subgraph H of a bipartite graph B is statistically defined by the probability of the occurrence of a subgraph as dense as H in B. Higher probability means low weight, since a heavy subgraph has a low probability to occur. This weighting scheme is used to discover maximum bounded biclique, where the algorithm supposes that the degree of every gene vertex is bounded.

OPSM

A method for order-preserving biclustering is studied using probabilistic model [11] where the authors use the term OPSM (order-preserving submatrix). In a numerical dataset (G, M), an

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 4 & 8 & 12 & 16 \\ 5 & 10 & 15 & 20 \end{bmatrix}$$

Figure 2.1: A multiplicative bicluster as matrix multiplication.

OPSM is modeled as a pair (T, π), where $T \subseteq M$, $|T| = s$, and $\pi = (t_1, t_2, \cdots, t_s)$ is a linear ordering of T. A pair (T, π) is supported by an object g if the permutation of T in g follows π, i.e. $t_1(g) < t_2(g) < \cdots < t_s(g)$. The algorithm tries to find some (T, π) that are statistically significant. Similar to SAMBA, here statistically significant means that a pair (T, π) has a low probability to be supported by high number of objects. This probability of observing a (T, π) with $|T| = s$ supported by k objects is:

$$U(s,k) = |M| \cdots (|M| - s + 1) \sum_{i=k}^{|G|} \binom{n}{i} \left(\frac{1}{s!}\right)^i \left(1 - \frac{1}{s!}\right)^{(n-i)}.$$

Non-negative matrix factorization

Non-negative matrix factorization (NMF) of a matrix X is the task of finding two factors of X, called F and G, where X, F, and G contain no negative element. It is formulated as $X \approx FG^T$, where $X \in \mathbb{R}_+^{p \times n}$, $F \in \mathbb{R}_+^{p \times k}$, and $G \in \mathbb{R}_+^{n \times k}$. Ding et al. [26] showed that if the factor G is orthogonal, the NMF:

$$\min_{F \geq 0, G \geq 0} ||X - FG^T||^2, \text{ s.t. } G^T G = I$$

is equivalent to K-means clustering. Furthermore, when both factors are orthogonal, the NMF:

$$\min_{F \geq 0, G \geq 0} ||X - FG^T||^2, \text{ s.t. } F^T F = I, G^T G = I$$

corresponds to simultaneous K-means clustering of rows and columns of X, also known as co-clustering in analyzing a dataset of textual documents.

The constraint that the factors should be orthogonal is also studied in non-negative matrix tri-factorization (NMTF) [27]. This bi-orthogonal NMTF is able to simultaneuosly cluster documents and terms in various document-term binary matrices. The applications on text datasets are further studied by incorporating the semantic relationship among words [87], or the geometric structures of rows and columns [6].

FABIA

FABIA, stands for Factor Analysis for Bicluster Acquisition [48], is a numerical method focused in finding multiplicative biclusters in a gene expression dataset. Here a bicluster is defined as a submatrix where the rows are similar to each other on the columns and vice versa. A multiplicative bicluster can be regarded as the result of matrix multiplication between a column matrix λ and a row matrix z^T, as illustrated in Fig. 2.1. Therefore, biclustering is formulated as sparse matrix factorization.

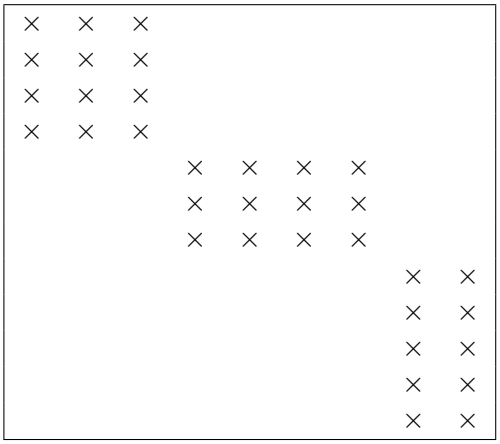


Figure 2.2: A binary block diagonal matrix.

HOCCLUS

HOCCLUS [79] is an algorithm that discover hierarchical and overlapping biclusters/co-clusters on mRNA:miRNA interaction datasets. Similar to SAMBA, HOCCLUS represents the interaction as a bipartite graph, with the vertices for mRNAs on one side and vertices for miRNAs on the other side. An edge connects an mRNA to an miRNA, with the interaction value as the edge's weight. HOCCLUS uses METIS [57] to mine initial non-hierarchical and non-overlapping biclusters from the bipartite graph. From these initial biclusters, HOCCLUS generates hierarchical organization among them. Then the algorithm detects possible overlapping between two biclusters in a same hierarchical level by identifying objects in one bicluster that can be added to another bicluster. These overlaps are used to decide whether two biclusters should be merged.

The METIS approach used in HOCCLUS can mine bicluster from a subgraph that is not biclique, giving the possibility that some biclusters from bicliques are not extracted. It also needs a number of biclusters as a user-defined parameter. HOCCLUS2 [78] is an improvement of HOCCLUS, where instead of METIS, a new approach is proposed to mine biclusters from overlapping bicliques without the needs of supplying the number of biclusters. In addition, a ranking of extracted biclusters is generated, based on their significance. The biclusters from HOCCLUS2 (and some data from several other classification algorithms) are stored in a web-based database called ComiRNet [80].

CoClus

CoClus [2] is an algorithm that performs block diagonal clustering in text data. Block diagonal clustering simultaneously groups rows and columns by applying permutations on them, and as a result a block diagonal matrix (illustrated in Fig. 2.2) is produced. CoClus perform this block diagonal clustering by adapting and maximizing modularity criterion, which is originally a measure in graph clustering.

SBB

An algorithm called Similarity Based Biclustering (SBB) [50] was proposed to discover similar-value biclusters in human cancer microarray data. This method is based on the concept of co-similarity among genes and conditions. In a matrix A, where a_{xy} is the element of A in row

x and column y, SBB uses χ-Sim that defines the similarity between two genes i and j as:

$$sr_{ij} = \sum_k \sum_l a_{ik} \cdot a_{jl} \cdot sc_{kl},$$

where sc_{kl} is the similarity between two conditions k and l. This condition similarity, in turn, is formulated as:

$$sc_{kl} = \sum_i \sum_j a_{ik} \cdot a_{jl} \cdot sr_{ij}.$$

The matrices SR and SC are iteratively computed by updating SC (or SR respectively). These computations is related to biclustering since two genes i and j are similar if sr_{ij} is high, meaning that they exhibits same kind of behavior under similar conditions (given by SC).

After the computation of SR and SC, these matrices are used as similarity measures to cluster rows and columns separately using hierarchical clustering. Then, a bicluster can be extracted from each pair of row cluster and column cluster. The utilization of hierarchical clustering in SBB means that the generated biclusters are not overlapping.

2.3.2 Pattern-based methods

In pattern-based biclusterings, a bicluster is regarded as samples sharing a common pattern. These approaches generally compute neither similarity measure to group rows or columns nor significance score of a bicluster. Some methods included in this approach are described below.

Object-attribute biclustering

Standard FCA –where the context is a binary matrix– discovers formal concepts which can be regarded as a constant-value bicluster. The problem that the number of concepts is large is studied in dense object-attribute (oa) biclustering [51], applied to a dataset of internet advertisement. From this dataset, a formal context (G, M, I) is created, with G is the set of firms, M is the set of advertising terms, and $(g, m) \in I$ means that the firm g bought the term m. An oa-biclusters is defined as a submatrix (m', g') where $(g, m) \in I$. This methods works on the notion of dense oa-bicluster, which is a submatrix (m', g') whose density $\rho = |I \cap (m' \times g')|/(|m'| \cdot |g'|)$ satisfies a user-defined threshold. Therefore, oa-bicluster can be considered as a relaxation of formal concept, based on the fact that an oa-bicluster is a rectangle densely composed of 1s, while a formal concept is a rectangle fully composed of 1s.

BicPAM

BicPAM is a family of pattern-based biclustering, was first proposed in [43]. Working in a gene-condition expression matrix, it is able to discover constant, additive, multiplicative, or symmetrical biclusters. This approach can detect overlapping biclusters, and also handle missing values and noises.

The whole process of BicPAM can be summarized in three steps: mapping, mining, and closing. In *mapping* step, normalization and discretization were carried out. Normalization, which means having zero-mean value, allows for symmetries in biclusters. Discretization, although may cause loss of information, could address the noise problem, and to reduce the complexity since an exhaustive biclustering is expected. Three discretization options are studied: fixed range, equal bin, or by statistically calculating cutoff points.

Table 2.2: An example of the transformation from (a) matrix after mapping step to (b) transaction dataset.

G	m_1	m_2	m_3	m_4	m_5
g_1	-1	-3	-6	5	7
g_2	-2	-4	10	4	6
g_3	4	1	-6	-3	1
g_4	8	2	-6	-6	1
g_5	8	3	-6	1	1

(a)

G	itemset
g_1	$\{m_1^{-1}, m_2^{-3}, m_3^{-6}, m_4^{5}, m_5^{7}\}$
g_2	$\{m_1^{-2}, m_2^{-4}, m_3^{10}, m_4^{4}, m_5^{6}\}$
g_3	$\{m_1^{4}, m_2^{1}, m_3^{-6}, m_4^{-3}, m_5^{1}\}$
g_4	$\{m_1^{8}, m_2^{2}, m_3^{-6}, m_4^{-6}, m_5^{1}\}$
g_5	$\{m_1^{8}, m_2^{3}, m_3^{-6}, m_4^{1}, m_5^{1}\}$

(b)

Table 2.3: Examples of additive alignment of Table 2.2a: the alignment of (a) m_1 and (b) m_2.

G	m_1	m_2	m_3	m_4	m_5
g_1	8	6	3	14	16
g_2	8	6	20	14	16
g_3	8	5	-2	1	5
g_4	8	2	-6	-6	1
g_5	8	3	-6	1	1

(a)

G	m_1	m_2	m_3	m_4	m_5
g_1	5	3	0	11	13
g_2	5	3	17	11	13
g_3	6	3	-4	-1	3
g_4	9	3	-5	-5	2
g_5	8	3	-6	1	1

(b)

The *mining* step is the main part, where biclusters are discovered. Among other pattern-based mining, BicPAM uses frequent itemsets as default. First, the algorithm decides whether alignments are needed, based on the type of bicluster. For constant-column biclustering, the alignment is not executed. Then, the mapped matrix is transformed into a kind of transaction database, where each object is represented by an itemset. This transformation is illustrated in Table 2.2. A constant-column bicluster can be found in a frequent itemset. From Table 2.2b, we can find a frequent itemset $\{m_3^{-6}, m_5^{1}\}$ for example, supported by g_3, g_4, and g_5. This itemset corresponds to bicluster $(\{g_3, g_4, g_5\}, \{m_3, m_5\})$ in Table 2.2a. The itemset mining is executed using existing algorithms: F2G [47], Charm [104], or AprioriTID [1] among others.

On the other hand, for additive, multiplicative, and symmetrical biclustering, BicPAM applies column alignments to the mapped matrix before transforming it into a transaction dataset (the alignment is the default option, BicPAM also proposes local normalization). The column alignments for additive biclustering are illustrated in Table 2.3. For a given column m_j, the maximum value x is calculated. Then, for each row g_i, the value $x - m_j(g_i)$ is added to all of its values.

Consider Table 2.3a, the result of alignment of m_1 from Table 2.2a. The maximum value of m_1 is 8. Therefore, for the whole row of g_1, the value $8 - m_1(g_1) = 8 - (-1) = 9$ is added. This additive alignment of m_1 results in m_1 having a single value. Then, similar to constant-column biclustering, this aligned matrix is transformed into a transaction dataset, and frequent itemsets are mined. From Table 2.3a, an itemset $\{m_1^{8}, m_2^{6}, m_4^{14}, m_5^{16}\}$ is found, supported by rows g_1 and g_2. This itemset corresponds to the additive bicluster $(\{g_1, g_2\}, \{m_1, m_2, m_4, m_5\})$. The alignment, transformation to transaction dataset, and itemset mining are done for every column.

The column alignments for multiplicative biclustering, illustrated in Table 2.4, are similar to the additive alignments. Here the objective is having a column with constancy by multiplying each row. Consider the multiplicative alignment of m_1 in Table 2.2a. The least common multiple of all values in m_1 is 8, so each row is multiplied such that the values of m_1 are constant. This

Table 2.4: Examples of multiplicative alignment of Table 2.2a: the alignment of (a) m_1 and (b) m_2.

G	m_1	m_2	m_3	m_4	m_5
g_1	8	24	48	−40	−56
g_2	8	16	−40	−16	−24
g_3	8	2	−12	−6	2
g_4	8	2	−6	−6	1
g_5	8	3	−6	1	1

(a)

G	m_1	m_2	m_3	m_4	m_5
g_1	4	12	24	−20	−28
g_2	6	12	−30	−12	−18
g_3	48	12	−72	−36	12
g_4	48	12	−36	−36	6
g_5	32	12	−24	4	4

(b)

Table 2.5: The sequence dataset, result of transformation of the matrix in Table 2.2a.

G	Sequence
g_1	$\langle\{m_3\},\{m_2\},\{m_1\},\{m_4\},\{m_5\}\rangle$
g_2	$\langle\{m_2\},\{m_1\},\{m_4\},\{m_5\},\{m_3\}\rangle$
g_3	$\langle\{m_3\},\{m_4\},\{m_2,m_5\},\{m_1\}\rangle$
g_4	$\langle\{m_3,m_4\},\{m_5\},\{m_2\},\{m_1\}\rangle$
g_5	$\langle\{m_3\},\{m_4,m_5\},\{m_2\},\{m_1\}\rangle$

results in Table 2.4a.

The *closing* step is a post-processing step in BicPAM. The first part of this step is extension of biclusters. It allows the enlargement of discovered biclusters under some homogeneity criteria. The second part is bicluster merging, which combines some biclusters having many shared rows and columns. The last part is the filtering, where some biclusters that are contained in another larger biclusters are discarded. In this part, it is also possible to remove some rows or columns of a bicluster such that the resulting smaller bicluster is more homogeneous.

The authors of BicPAM also propose a pattern-based method for order-preserving biclustering, namely BicSPAM [44], based on the fact that order-preserving biclustering is related to sequential pattern mining. Similar to BicPAM, in BicSPAM a numerical matrix is transformed into transaction dataset. Here an object corresponds to a transaction which is a sequence of itemsets, illustrated in Table 2.5. A mining step in this table results in a sequence $\langle\{m_2\},\{m_1\},\{m_4\},\{m_5\}\rangle$ for example, supported by rows g_1 and g_2. This is then regarded as an order-preserving bicluster $(\{g_1,g_2\},\{m_1,m_2,m_4,m_5\})$.

Furthermore, several other methods related to BicPAM were also proposed. BicNET [46] is a biclustering approach for a biological networks, where the dataset can be represented as a bipartite graph. Bic2PAM [45] is a constrained biclustering, where background knowledge can be incorporated. In addition, these biclustering approaches are implemented in BicPAMS [42].

Contiguous OPSM

Xue et al. [100, 101] presented an algorithm to discover order-preserving biclusters (or OPSM: order-preserving submatrix) from a gene expression data. This algorithm defines an OPSM as slightly different to Def. 4. Instead of any subset of columns, contiguous OPSM means that the expected bicluster should have columns that are contiguous in the original matrix. Then, similar to BicSPAM [44], the problem of finding contiguous OPSM is translated as sequential pattern mining.

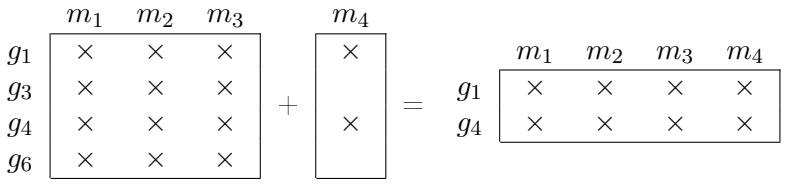


Figure 2.3: Generation of one successor by In-Close2.

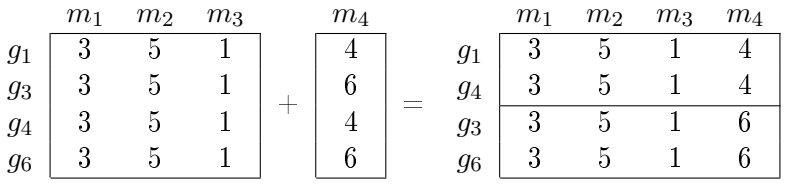


Figure 2.4: Generation of two successors by RIn-Close_CVC_P.

RIn-Close

RIn-Close [94] is a family of algorithms that enumerates all maximal biclusters based on FCA. A formal concept in FCA corresponds to a constant-value bicluster of ones in a binary matrix. Therefore, In-Close2 algorithm [8], that enumerates all concepts in a formal context, can be regarded as an algorithm to discover constant-value biclusters. RIn-Close is an adaptation of this algorithm such that it can also discover constant-column, additive, and multiplicative biclusters.

RIn-Close_CVC_P and RIn-Close_CVC are the algorithms to mine constant-column and similar-column biclusters respectively. In the exploration tree of In-Close2, a child node is generated by intersecting the set of objects of the added attribute with the current extent. Consider Fig. 2.3, where the current node is a concept $(\{g_1, g_3, g_4, g_6\}, \{m_1, m_2, m_3\})$. InClose2 generates at most one successor for every added attribute. By adding m_4, InClose2 obtains the concept $(\{g_1, g_4\}, \{m_1, m_2, m_3, m_4\})$ as the successor.

InClose2 works for binary matrix, and a successor is generated by intersecting the extent of the current concept with the extent of the new attribute. On the other hand, RIn-Close_CVC_P works for numerical matrices, and it can generate many successors from the addition of a single attribute. It is designed to obtain constant-column biclusters. Consider Fig. 2.4 where the current node is a constant-column bicluster $(\{g_1, g_3, g_4, g_6\}, \{m_1, m_2, m_3\})$. Then, the attribute m_4 is added, creating two new constant-column biclusters $(\{g_1, g_4\}, \{m_1, m_2, m_3, m_4\})$ and $(\{g_3, g_6\}, \{m_1, m_2, m_3, m_4\})$. Furthermore, the RIn-Close_CVC_P can be relaxed such that the similar-column biclusters can be obtained. This relaxation is called RIn-Close_CVC. Similar to RIn-Close_CVC_P, this algorithm generates multiple successors from the addition of a single attribute. The difference is that the successors generated by RIn-Close_CVC can have object overlaps among them, due to the nature of similar-column biclusters.

To discover additive and multiplicative biclusters, the algorithms RIn-Close_CHV_P and RIn-Close_CHV were proposed to mine perfect and non-perfect bicluster respectively. RIn-Close_CHV_P is similar to RIn-Close_CVC_P in the sense that they both generate multiple successors from the addition of a single attribute. The difference lies in how an extent is constructed. Consider again Fig. 2.4. In RIn-Close_CVC_P, g_1 and g_4 are grouped together since they have the same value in m_4. On the other hand, in RIn-Close_CHV_P, the algorithms looks at the difference (additive or multiplicative) between the new attribute and the existing attributes. For example, in Fig. 2.5 where m_4 is added, g_1 and g_2 are grouped together

	m_1	m_2	m_3
g_1	1	2	0
g_2	3	4	2
g_3	2	3	1
g_4	4	5	3

$+$

	m_4
g_1	4
g_2	6
g_3	4
g_4	6

$=$

	m_1	m_2	m_3	m_4
g_1	1	2	0	4
g_2	3	4	2	6
g_3	2	3	1	4
g_4	4	5	3	6

Figure 2.5: Generation of two successors by RIn-Close_CHV_P for additive biclustering.

Table 2.6: A numerical matrix having additive biclusters.

	m_1	m_2	m_3	m_4
g_1	2	3	4	5
g_2	5	6	2	8
g_3	1	2	5	4
g_4	4	5	4	7

since their values in m_4 have the same difference with their corresponding values in m_1, i.e. $m_4(g_1) - m_1(g_1) = m_4(g_2) - m_1(g_2)$.

RIn-Close_CHV was proposed to discover non-perfect additive or multiplicative biclusters. From a numerical matrix, this algorithm creates an "augmentation" matrix and applies similar-column biclustering in the augmented matrix. The augmentation process was designed from the observation that in an additive bicluster, every pair of columns has a constant difference. Therefore, by calculating the difference of every column pair, we can find some constant differences that can be regarded as an additive bicluster.

Consider Table 2.6 as the numerical matrix where we want to discover additive biclusters. RIn-Close_CHV first generates the augmentation of this matrix, shown in Table 2.7, where the difference for every pair of attributes in Table 2.6 is calculated. In this augmented matrix, a similar-column bicluster $(\{g_1, g_2, g_3, g_4\}, \{m_1 - m_2, m_1 - m_4, m_2 - m_4\})$ can be obtained using RIn-Close_CVC. This corresponds to the additive bicluster $(\{g_1, g_2, g_3, g_4\}, \{m_1, m_2, m_4\})$ in the original numerical matrix.

2.4 Conclusion

Currently, the literature about biclustering is significantly rich. Biclustering is mainly used in biological domain, while there are also significant number of applications in text mining and recommendation systems. However, in biological domain, many authors are interested to find overlapping biclusters which reflect, for example, that a condition can participate in regulating multiple sets of genes. Meanwhile in text mining domain –where the term co-clustering is more frequent– the biclustering is considered as simultaneous partitioning of rows and columns,

Table 2.7: The augmentation of the matrix in Table 2.6.

	$m_1 - m_2$	$m_1 - m_3$	$m_1 - m_4$	$m_2 - m_3$	$m_2 - m_4$	$m_3 - m_4$
g_1	-1	-2	-3	-1	-2	-1
g_2	-1	3	-3	4	-2	-6
g_3	-1	-4	-3	-3	-2	1
g_4	-1	0	-3	1	-2	-3

indicating the absence of overlaps.

In the biclustering with overlaps, the number of biclusters that exist in a matrix can be enormous. Many approaches then study the problem of finding significant biclusters, which are submatrices fulfilling certain thresholds about their size, probability, density, cohesiveness, etc.

In Chapter 3 and 4 of this thesis, we examine the problem of bicluster enumeration, i.e. retrieving all biclusters in a data matrix, using FCA. FCA is related to biclustering, in the sense that both of them discover a submatrix showing a regularity among its cells. Moreover, there exist several algorithms that deal with the task of concept enumeration in FCA, which can be adapted to enumerate biclusters. We consider this approach as pattern based, since it translates the biclustering task into pattern mining.

FCA is previously used in enumerating biclusters [94] by modifying a concept enumeration algorithm. In this thesis, we extend FCA into pattern structures that deals with numerical matrix. In this way, we also study the potential of building a unified framework of discovering various types of biclusters in a binary and/or numerical matrix.

Chapter 3

Biclustering with partition pattern structures

Contents

The task of constant-column biclustering was studied using partition pattern structures [20], an extension of FCA. In this chapter, we explore the possibility of extending this approach for other types of biclusters. First we revisit the basic definitions of partition pattern structures in Sect. 3.1 and its application to constant-column biclustering in Sect. 3.2. Then, we explain the usability of partition pattern structures to additive and multiplicative biclustering in Sect. 3.3, order-preserving biclustering in Sect. 3.4, and coherent-sign-changes biclustering in Sect. 3.5. Lastly, we give the conclusion and some insights for future research in Sect. 3.6.

3.1 The pattern structures of partitions

3.1.1 Partition

In a given set S, a partition d is a splitting of S into components $\{c_1, \cdots, c_n\}$, such that all components cover all elements in S and there is no overlap between any two components, or:

$$\bigcup_{c_i \in d} c_i = S \quad \text{and} \quad c_i \cap c_j = \emptyset \quad \text{whenever} \quad i \neq j.$$

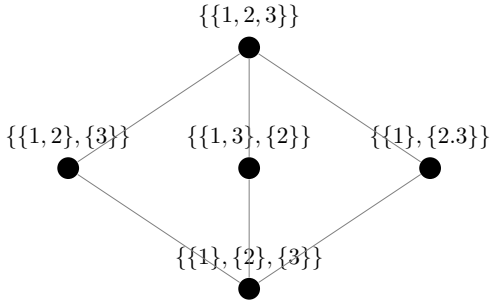


Figure 3.1: The lattice of all possible partition of a set $\{1, 2, 3\}$.

$c_i \in d_1$	$c_j \in d_2$	$c_i \cap c_j$
$\{1, 2\}$	$\{1, 3\}$	$\{1\}$
$\{1, 2\}$	$\{2\}$	$\{2\}$
$\{3\}$	$\{1, 3\}$	$\{3\}$
$\{3\}$	$\{2\}$	$\emptyset$
$\bigcup c_i \cap c_j$		$\{\{1\}, \{2\}, \{3\}\}$

Figure 3.2: The calculation of $d_1 \sqcap d_2$ where $d_1 = \{\{1, 2\}, \{3\}\}$ and $d_2 = \{\{1, 3\}, \{2\}\}$.

Given a set $S_1 = \{1, 2, 3, 4\}$ for example. A partition d_1 over S_1 is written as $d_1 = \{\{1, 2, 3\}, \{4\}\}$. This partition is composed by two components $c_1 = \{1, 2, 3\}$ and $c_2 = \{4\}$. There is no overlap among the components of d_1 and the union of all components results in S_1.

A partition d_1 is a *coarsening* of d_2 if one (or more) component in d_2 is a union of some components in d_1. In other words, for every component c_j in d_2 there exists a component c_i in d_1 where $c_j \subseteq c_i$, or: $\forall c_j \in d_2 \exists c_i \in d_1, c_j \subseteq c_i$. For example, $d_1 = \{\{1, 2, 3\}, \{4\}\}$ is a coarsening of $d_2 = \{\{1, 2\}, \{3\}, \{4\}\}$ since the component $\{1, 2, 3\}$ in d_1 is the union of $\{1, 2\}$ and $\{3\}$ in d_2. Consequently, the partition d_2 is called a *refinement* of d_1.

The coarsening and refinement among partitions give an order among them. A finer partition is subsumed by ($\sqsubseteq$) its coarser partition(s). In the previous example, $d_2 \sqsubseteq d_1$. This makes D, the set of all possible partitions of a set S, is partially ordered, where its lattice can be depicted in a line diagram. For example, the lattice of all possible partitions of a set $\{1, 2, 3\}$ is illustrated as the Hasse diagram in Fig. 3.1.

Furthermore, the set D is a complete lattice, where the meet and join exist for all subsets of D. The meet ($\sqcap$) of two partitions is their coarsest common refinement. It is defined as the set of intersections of every pair $c_i \in d_1$ and $c_j \in d_2$, formulated as:

$$d_1 \sqcap d_2 = \bigcup c_i \cap c_j. \tag{3.1}$$

Consider the partitions $d_1 = \{\{1, 2\}, \{3\}\}$ and $d_2 = \{\{1, 3\}, \{2\}\}$. The meet of these two partitions is $\{\{1\}, \{2\}, \{3\}\}$, illustrated in Fig. 3.2.

The join ($\sqcup$) of two partitions is their finest common coarsening, defined as:

$$d_1 \sqcup d_2 = \left(\bigcup_{p_i \cap p_j \neq \emptyset} p_i \cup p_j \right)^+,$$

where $(.)^+$ is an operator that conserves only the maximal components in d, i.e. the components

$c_i \in d_1$	$c_j \in d_2$	$c_i \cup c_j$
$\{1,2\}$	$\{1,3\}$	$\{1,2,3\}$
$\{1,2\}$	$\{2\}$	$\{1,2\}$
$\{3\}$	$\{1,3\}$	$\{1,3\}$
$\{3\}$	$\{2\}$	$-$
$\bigcup c_i \cup c_j$		$\{\{1,2,3\},\{1,2\},\{1,3\}\}$
$(\bigcup c_i \cup c_j)^+$		$\{\{1,2,3\}\}$

Figure 3.3: The calculation of $d_1 \sqcup d_2$ where $d_1 = \{\{1,2\},\{3\}\}$ and $d_2 = \{\{1,3\},\{2\}\}$. The $\cup$ between c_i and c_j is calculated only if $c_i \cap c_j \neq \emptyset$.

Table 3.1: A numerical table [20].

	m_1	m_2	m_3	m_4	m_5
g_1	1	1	1	1	6
g_2	1	1	1	1	6
g_3	1	1	1	6	6
g_4	8	8	1	6	6

that are not proper subsets of another component. It is defined as:

$$d^+ = \{c_i \in d | \nexists c \in d, c_i \subseteq c\}.$$

Using the previous example with d_1 and d_2, the join of these two partitions is $d_1 \sqcup d_2 = \{\{1,2,3\}\}$, whose computation is illustrated in Fig. 3.3.

The subsumption relation –where coarser partition subsumes finer partition– can also be defined using the meet operator as:

$$d_2 \sqsubseteq d_1 \iff d_2 \sqcap d_1 = d_2, \tag{3.2}$$

meaning that if d_2 is the meet between d_1 and d_2, then d_2 is a refinement of d_1.

3.1.2 Partition of objects

In a numerical matrix (G, M), we can find a partition of objects/rows based on their values across all attributes/columns or vice versa. For example, in Table 3.1, the set of objects are partitioned as $\{\{g_1, g_2\}, \{g_3, g_4\}\}$ according to m_4.

The partition of objects according to an attribute can be obtained using equivalence relation. An equivalence relation for every $g \in G$ w.r.t. an attribute is defined as follows:

$$[g_i]_{m_j} = \{g_k \in G | m_j(g_i) = m_j(g_k)\}. \tag{3.3}$$

The $[g_i]_{m_j}$ relation returns all objects in G whose value in m_j is the same as g_i's value in m_j. For example, in Table 3.1 $[g_1]_{m_4} = \{g_1, g_2\}$.

A partition of objects according to an attribute can be obtained from the equivalence relation. A function $\delta : M \to D$ maps an attribute to an object partition as follows:

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

Therefore, in Table 3.1, $\delta(m_4) = \{[g_1]_{m_4}, [g_2]_{m_4}, [g_3]_{m_4}, [g_4]_{m_4}\}$, which is equal to $\{\{g_1, g_2\}, \{g_3, g_4\}\}$.

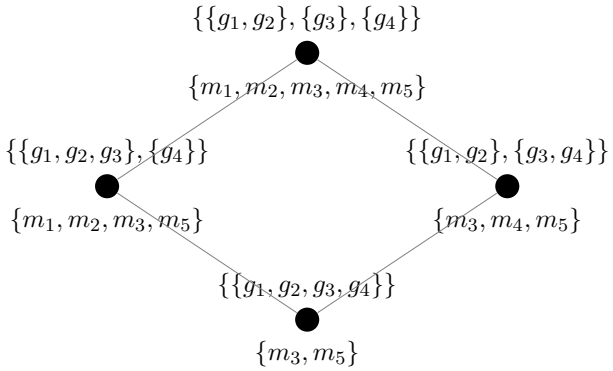


Figure 3.4: The lattice of all partition pattern concepts of Table 3.1.

3.1.3 Partition pattern structures

As explained in Sect. 1.4, pattern structures are generalizations of FCA, where each object is described by a complex structure. A pattern structure is defined by the triple $(G, (D, \sqcap), \delta)$, where $(D, \sqcap)$ is a lattice of descriptions D with $\sqcap$ as the similarity operator or the meet between two partitions, and $\delta : G \to D$ maps an object to its description.

Previously in Sect. 3.1.2, the function $\delta : M \to D$ means that a partition of objects is the description of an attribute. Therefore, in (object) partition pattern structures, the triple is $(M, (D, \sqcap), \delta)$. Following Eq. 1.1, the derivation operators are defined for any subset of attributes and any description:

$$^{\square} : \wp(M) \to D,$$
$$^{\square} : D \to \wp(M).$$

Given $B \subseteq M$ and $d \in D$, these operators are defined by:

$$B^{\square} = \bigsqcap_{m \in B} \delta(m),$$
$$d^{\square} = \{m \in M | d \sqsubseteq \delta(m)\}. \tag{3.4}$$

In other words, given subset of attributes B, $B^{\square}$ is the coarsest common refinement of all object partitions of $m \in B$. Given an object partition d, $d^{\square}$ is the set of all attributes whose partition is a coarsening of d.

Similar to FCA and other pattern structures, a partition pattern concept (or pp-concept) is a pair (B, d) where $B^{\square} = d$ and $d^{\square} = B$. An example of pp-concept from Table 3.1 is $(\{m_3, m_4, m_5\}, \{\{g_1, g_2\}, \{g_3, g_4\}\})$. The set of all pp-concepts in a given matrix is a complete lattice. Following the ordering of formal concepts in Eq. 1.3, the order among pp-concepts is given by relation:

$$(B_1, d_1) \leq (B_2, d_2) \iff B_1 \subseteq B_2 \text{ (dually } d_2 \sqsubseteq d_1).$$

The lattice of pp-concepts from Table 3.1 is illustrated as the Hasse diagram in Fig. 3.4.

3.2 Constant-column biclustering

In a matrix, a constant-column bicluster is a submatrix $(A \subseteq G, B \subseteq M)$ where each column has a constant value across all rows. An example of a constant-column bicluster in Table 3.1 is the submatrix $(\{g_3, g_4\}, \{m_3, m_4, m_5\})$.

Table 3.2: Partition pattern concepts from Table 3.1 and their corresponding constant-column biclusters.

Concept		Bicluster	Maximal
Extent	Intent		
$\{m_3, m_5\}$	$\{\{g_1, g_2, g_3, g_4\}\}$	$(\{g_1, g_2, g_3, g_4\}, \{m_3, m_5\})$	yes
$\{m_1, m_2, m_3, m_5\}$	$\{\{g_1, g_2, g_3\}, \{g_4\}\}$	$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3, m_5\})$	yes
		$(\{g_4\}, \{m_1, m_2, m_3, m_5\})$	no
$\{m_3, m_4, m_5\}$	$\{\{g_1, g_2\}, \{g_3, g_4\}\}$	$(\{g_1, g_2\}, \{m_3, m_4, m_5\})$	no
		$(\{g_3, g_4\}, \{m_3, m_4, m_5\})$	yes
$\{m_1, m_2, m_3, m_4, m_5\}$	$\{\{g_1, g_2\}, \{g_3\}, \{g_4\}\}$	$(\{g_1, g_2\}, \{m_1, m_2, m_3, m_4, m_5\})$	yes
		$(\{g_3\}, \{m_1, m_2, m_3, m_4, m_5\})$	yes
		$(\{g_4\}, \{m_1, m_2, m_3, m_4, m_5\})$	yes

Partition pattern structures was proposed in [20] to discover constant-column biclusters in a numerical matrix. This is possible since the equivalence relation in Eq. 3.3 groups objects based on an attribute. Consider the pp-concept $(\{m_3, m_4, m_5\}, \{\{g_1, g_2\}, \{g_3, g_4\}\})$ in Table 3.1. Any pair of a partition component and the extent in this concept corresponds to a constant-column bicluster:

- $(\{g_1, g_2\}, \{m_3, m_4, m_5\})$, and
- $(\{g_3, g_4\}, \{m_3, m_4, m_5\})$.

Therefore, in any pp-concept (B, d), any pair $(c \in d, B)$ is a constant-column bicluster.

It should be noted that these biclusters are not necessarily maximal, means that they can be found inside another larger bicluster. The constant-column bicluster $b_1 = (\{g_1, g_2\}, \{m_3, m_4, m_5\})$ for example, is not maximal since it is part of the bicluster $b_2 = (\{g_1, g_2\}, \{m_1, m_2, m_3, m_4, m_5\})$. However, this maximality can be verified by inspecting the concept lattice. Consider the concept lattice in Fig. 3.4. We can see that b_1 is found in the concept $(\{m_3, m_4, m_5\}, \{\{g_1, g_2\}, \{g_3, g_4\}\})$, which is a descendant of the concept $(\{m_1, m_2, m_3, m_4, m_5\}, \{\{g_1, g_2\}, \{g_3, g_4\}\})$. Since this higher concept also contains a bicluster with $\{g_1, g_2\}$, we can conclude that b_1 is not maximal. More formally, given two pp-concepts (B_1, d_1) and (B_2, d_2) such that $(B_1, d_1) \leq (B_2, d_2)$, the bicluster $(c \in d_1, B_1)$ is maximal if $c \notin d_2$. All pp-concepts and their corresponding biclusters are listed in Table 3.2.

In the subsequent sections in this chapter, we show that partition pattern structures can be adapted to discover other bicluster types.

3.3 Additive and multiplicative biclustering

For additive and multiplicative biclusters, we introduce the notion of *marked* objects. A marked object g^* is an object g associated to a mark $*$, where $* \in \mathbb{R}$.

Slightly different from partition pattern structures in Sect. 3.1.3, here the function $\delta : M \to D$ maps an attribute to an object partition, which is composed by only one component with a mark for each object. For example, from Table 3.3:

$$\delta(m_1) = \{\{g_1^{-1}, g_2^{-2}, g_3^{4}, g_4^{8}, g_5^{8}\}\}.$$

Table 3.3: Running example for additive and multiplicative biclustering.

	m_1	m_2	m_3	m_4	m_5
g_1	-1	-3	-6	5	7
g_2	-2	-4	10	4	6
g_3	4	1	-6	-3	1
g_4	8	2	-6	-6	1
g_5	8	3	-6	1	1

Any two components are regarded as equal iff the marks of their objects are *coherent*. For additive and multiplicative biclustering, two components are equal iff the marks of one component can be obtained by adding (or multiplying, resp.) a unique value to all marks of the other components. For example:

$$\{g_1^3, g_2^7\} = \{g_1^6, g_2^{10}\} \text{ for additive type}$$

because the marks of the second component can be obtained by adding value 3 to the marks of the first component, and

$$\{g_1^3, g_2^7\} = \{g_1^6, g_2^{14}\} \text{ for multiplicative type}$$

because the marks of the second component can be obtained by multiplying the marks of the first component by 2.

The similarity operator of any two components also depends on the bicluster that we want to discover. In general, this operator groups objects according to the coherency explained before. For additive biclustering, the similarity operator is written as $\cap^A$. From two components c_1 and c_2, $\cap^A$ groups together all objects according to the difference from both components. For example, if $c_1 = \{g_1^2, g_2^3, g_3^4, g_4^5\}$ and $c_2 = \{g_1^5, g_2^7, g_3^7, g_4^9\}$, then $c_1 \cap^A c_2 = \{\{g_1^2, g_3^4\}, \{g_2^3, g_4^5\}\}$. This is because the marks of g_1 and g_3 in c_2 are their mark in c_1 plus 3, while the marks of g_2 and g_4 in c_2 are their mark in c_1 plus 4. The similarity operator $\sqcap$ for this bicluster is similar to Eq. 3.1, with the adjustment in the similarity between components:

$$d_1 \sqcap d_2 = \bigcup_{c_i \in d_1, c_j \in d_2} c_i \cap^A c_j. \tag{3.5}$$

The component similarity operator for multiplicative biclustering (written as $\cap^M$) is related to $\cap^A$. Here the objects are grouped according to the multiplicative difference between two components. For example, if $c_1 = \{g_1^2, g_2^3, g_3^4, g_4^5\}$ and $c_2 = \{g_1^4, g_2^9, g_3^8, g_4^{15}\}$, then $c_1 \cap^A c_2 = \{\{g_1^2, g_3^4\}, \{g_2^3, g_4^5\}\}$. This is because the mark of g_1 and g_3 in c_2 is twice of their mark in c_1, while the mark of g_2 and g_4 in c_2 is three times of their mark in c_1. The $\sqcap$ for this biclustering is:

$$d_1 \sqcap d_2 = \bigcup_{c_i \in d_1, c_j \in d_2} c_i \cap^M c_j. \tag{3.6}$$

We can see in Eq. 3.5 and 3.6 that the operator $\sqcap$ for these three biclustering are similar, the difference lies in the similarity between two components ($\cap^A$ and $\cap^M$). For those types of biclustering, the order between any two partitions is also the same as Eq. 3.2 for constant-column biclustering:

$$d_1 \sqsubseteq d_2 \iff d_1 \sqcap d_2 = d_1. \tag{3.7}$$

Table 3.4: Running example for order-preserving biclustering.

	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 3.5: Some examples of pairs and their partitions over Table 3.4.

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 corresponding definition of $\sqcap$ for each type, we can follow Eq. 3.4 to obtain object partition concepts. In a concept (B, d) any pair $(c \in d, B)$ is a bicluster, where we can omit the mark of each object.

For example, in Table 3.3, $(\{m_1, m_2, m_4, m_5\}, \{\{g_1^{-1}, g_2^{-2}\}, \{g_3^4\}, \{g_4^8\}, \{g_5^8\}\})$ is a concept for additive biclustering, and the pair $(\{g_1, g_2\}, \{m_1, m_2, m_4, m_5\})$ is an additive bicluster. Furthermore, $(\{m_1, m_2, m_4\}, \{\{g_1^{-1}\}, \{g_2^{-2}\}, \{g_3^4, g_4^8\}, \{g_5^8\}\})$ is a concept for multiplicative biclustering, and the pair $(\{g_3, g_4\}, \{m_1, m_2, m_4\})$ is a multiplicative bicluster.

3.4 Order-preserving biclustering

Order-preserving (OP) biclustering was studied in gene expression data [11, 44, 100, 101]. This type of bicluster may represent a set of conditions which are different stages in the progress of a disease, and the expression levels of the genes show the same tendency (raising or falling) across these stages.

OP biclusters are also useful in the domain of recommendation systems. Given a user-item rating matrix, with U as the set of previous users and I as the set of items, one approach for producing a suggestion for a new user is by retrieving a set of similar users in U. 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 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 $rating(u_2, i_1) > rating(u_2, i_2) > \cdots > rating(u_2, i_{10})$.

3.4.1 Pattern structures for OP biclustering

Consider the dataset given by Table 3.4, 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 $\binom{n}{2}$ pairs.

First, we introduce $[g_i]_{r_x^y}$, an equivalence relation of an object w.r.t. a pair of attribute (instead of w.r.t a single attribute as in Eq. 3.3). It groups objects whose behavior in m_x and m_y is similar (either $m_x(g) > m_y(g)$ or $m_x(g) < m_y(g)$), defined as:

$$[g_i]_{r_y^x} = \{g_k \in G | \arg\max_j(m_j(g_i)) = \arg\max_j(m_j(g_k)), j \in \{x, y\}\}. \tag{3.8}$$

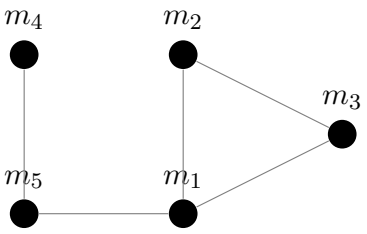


Figure 3.5: A graphical representation of the extent of $(\{r_2^1, r_3^1, r_5^1, r_3^2, r_5^4\}, \{\{g_1, g_3\}, \{g_2, g_4\}, \{g_5\}\})$.

Example from Table 3.4:

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

Using this equivalence relation, we define the function $\delta : R \to D$ as:

$$\delta(r_y^x) = \{[g_i]_{r_y^x} | g_i \in G\}. \tag{3.9}$$

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 3.5.

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:

$$B^\square = \bigsqcap_{r \in B} \delta(r) \qquad B \subseteq R,$$
$$d^\square = \{r \in R | d \sqsubseteq \delta(r)\} \qquad d \in D.$$

The meet and subsumption relation between two partitions follow Eq. 3.1 and Eq. 3.2 respectively.

Here, the extent of a concept is a set of attribute pairs. Consider the concept pc_1 with extent $\{r_2^1, r_3^1, r_5^1, r_3^2, r_5^4\}$ and intent $\{\{g_1, g_3\}, \{g_5\}, \{g_2, g_4\}\}$. Its extent can be depicted as a graph, illustrated in Fig. 3.5, where a vertex represents an attribute in a pair r, and an edge connects two attributes m_j and m_k if there is r_k^j in the extent. For example, r_2^1 makes the connection between m_1 and m_2.

An OP bicluster can be obtained from a clique of this extent graph. The clique is important to ensure the order preservation among attributes. This is because r_y^x and r_z^x only may mean that $x < y$ and $x < z$, but we can not say the relation among y and z unless we have r_z^y. The set $\{m_1, m_2, m_3\}$, $\{m_1, m_5\}$, or $\{m_4, m_5\}$ is an example of clique in the extent graph in Fig. 3.5.

In a concept (B, d), any pair (c, q) is an OP bicluster where $c \in d$ is a set of objects and q is a set of attributes that forms a clique in the graph of B. For example, from pc_1 which has 3 partition components and 3 cliques, we can obtain 9 OP biclusters. The biclusters having more than two columns are:

- $(\{g_1, g_3\}, \{m_1, m_2, m_3\})$, 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 $m_2 < m_1 < m_3$.

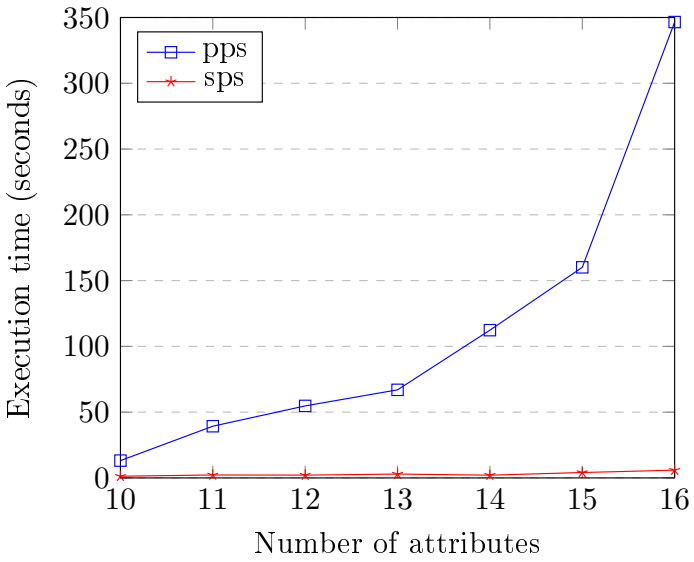


Figure 3.6: Comparison of partition pattern structure (pps) and sequence pattern structure (sps) in the task of finding OP biclusters in matrices with 10 rows and varying number of columns.

3.4.2 Experiments

OP biclustering is related to sequential pattern mining, and some approaches have studied sequential pattern mining algorithms for this biclustering [44, 100, 101]. FCA has been generalized to have a sequence of itemset as description of objects [15, 19] (will be explained in this thesis later in Chapter 5).

In this thesis, 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).

The comparison of runtime are shown in Figure 3.6. 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 $m \times \binom{n}{2}$ 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 [41]. This dataset has 3226 rows (genes) and 21 columns (tissues). As shown in Table 3.6, 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 $3226 \times \binom{21}{2} = 3226 \times 210$ 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.

Table 3.6: The columns in the breast cancer dataset in [41].

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

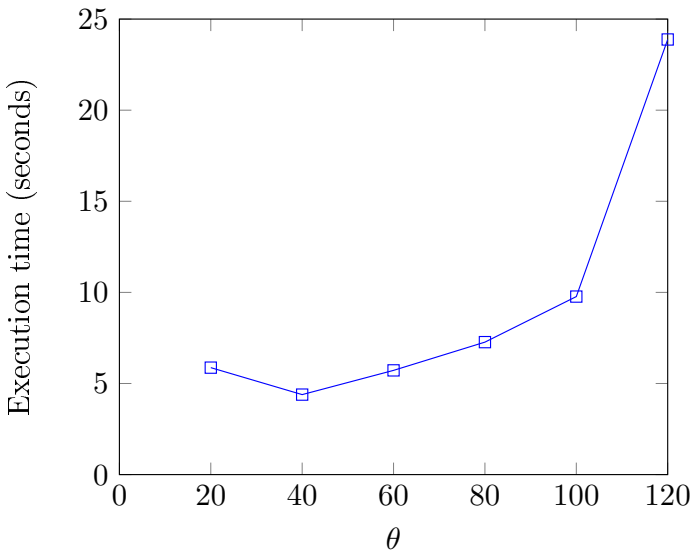


Figure 3.7: Runtime of pps-based approach with AddIntent and varying θ in obtaining 10K concepts, applied to the breast cancer dataset. Any partition component having elements less than θ is discarded.

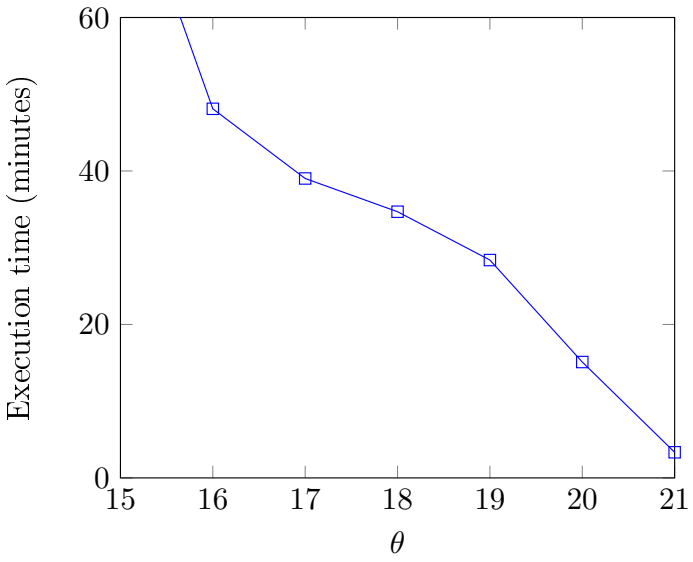


Figure 3.8: Runtime of sps-based approach with AddIntent and varying ℓ, applied to the breast cancer dataset. Any sequence having length less than ℓ is discarded.

The result of experiment with varying θ is shown in Fig. 3.7. 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 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.

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. 3.8. 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.

3.5 Biclustering with coherent sign changes

Gene expression data can be represented as a matrix, where rows and columns represent genes and experiments respectively. Each cell contains the numeric expression level of a given gene under a given experiment. In such data, we can say that an experiment affect a gene by either lowering or raising its expression, according to the gene's normal level. One may be interested in finding a subset of genes and a subset of experiments, such that the experiments affect the genes in a consistent way. In other words, any two experiments in the subset have always either the same effect or the opposite effect on every gene in the subset. In this thesis, this task corresponds to the mining of coherent-sign-changes (CSC) biclusters as defined in Def. 7. First, in Sect. 3.5.1 below we present some approaches for CSC biclustering using standard partition pattern structures. Then in Sect. 3.5.2 we discuss CSC biclustering using a new adaptation of partition pattern structures.

3.5.1 Using constant-column biclustering

In this subsection, we present two approaches of CSC biclustering using standard partition pattern structures (PPS) in Sect. 3.2. First, we describe how the problem can be solved via PPS-based constant-column biclustering. Second, we show that CSC biclusters can also be retrieved directly from PPS.

In the first procedure, we scale a binary matrix into a new matrix where each column corresponds to a pair of attributes. Then, we perform PPS in this new matrix to obtain constant-column biclusters. We call this approach *scaling-based*, and we present it in Algorithm 6.

Input: A binary sign matrix S, with a set of columns C and a set of rows R
Output: A set of CSC biclusters from S

```
1  csc := ∅
2  Scaling = new matrix    /* a scaled matrix of S */
3  foreach c_a ∈ C do
4  |  foreach c_b ∈ C, b > a do
5  |  |  new_column.elements ← XOR(c_a, c_b)
6  |  |  new_column.title ← {c_a, c_b}
7  |  |  Scaling.add(new_column)
8  |  end
9  |  set_of_cc ← constant-column biclusters from Scaling
10 |  foreach cc_cluster ∈ set_of_cc do
11 |  |  /* cc_col is the columns of cc_cluster*/
12 |  |  /* cc_row is the rows of cc_cluster */
13 |  |  csc_col ← ⋃ cc_col.title
14 |  |  csc_row ← cc_row
15 |  |  csc.add((csc_row, csc_col))
16 |  end
17 |  Scaling.clear()
18 end
19 return csc
```

Algorithm 6: *scaling-based*

Consider the binary sign matrix given in Table 3.7a. This matrix can be scaled into three

binary matrices in Table 3.7b, where each pair of columns (c_a, c_b) constitutes a new column. The new columns have value 1 if the pair's sign is different, and 0 if it is the same (a XOR operation). For example, the value in row r_2 column c_1c_3 is 0 because in r_2, both c_1 and c_3 are '$-$'. Then, we perform constant-column biclustering in each new binary matrix. A constant-column bicluster found in this new table corresponds with a CSC bicluster on the original table, as shown in Table 3.8. The set of columns in a CSC bicluster is a union of attribute pairs in constant-column bicluster's columns.

	c_1	c_2	c_3	c_4
r_1	$-$	$-$	$+$	$+$
r_2	$+$	$-$	$+$	$+$
r_3	$+$	$+$	$-$	$+$
r_4	$-$	$+$	$-$	$-$

(a)

c_1c_2	c_1c_3	c_1c_4	c_2c_3	c_2c_4	c_3c_4
0	1	1	1	1	0
1	0	0	1	1	0
0	1	0	1	0	1
1	0	0	1	1	0

(b)

Table 3.7: (a) A binary sign matrix and (b) its three scaling matrices.

constant-column biclusters	CSC biclusters
$(\{r_2, r_4\}\{c_1c_2, c_1c_3, c_1c_4\})$	$(\{r_2, r_4\}\{c_1, c_2, c_3, c_4\})$
$(\{r_1, r_3\}\{c_1c_2, c_1c_3\})$	$(\{r_1, r_3\}\{c_1, c_2, c_3\})$
$(\{r_1, r_2, r_3, r_4\}\{c_2c_3\})$	$(\{r_1, r_2, r_3, r_4\}\{c_2, c_3\})$
$(\{r_1, r_2, r_4\}\{c_2c_3, c_2c_4\})$	$(\{r_1, r_2, r_4\}\{c_2, c_3, c_4\})$

Table 3.8: Some constant-column biclusters from Table 3.7b and their corresponding CSC biclusters in Table 3.7a.

To avoid the combination of columns from a binary sign matrix, we may apply PPS directly into it. This is our second procedure (called *PPS-based*), where we retrieve CSC biclusters by examining the generated pp-concepts.

As explained in Section 3.1.3, a pp-concept is symbolized as $(\mathtt{A}, \mathtt{d})$ where $\mathtt{A}$ is a set of attributes and $\mathtt{d}$ is a set of partition component p. Moreover, any pair $(p, \mathtt{A})$ is a constant-column bicluster. Then, a CSC bicluster is either a single constant-column bicluster or any pair of constant-column biclusters that are contrasting to each other.

For example, consider again the matrix given in Table 3.7a and the pp-concept $(\{c_2, c_3, c_4\}, \{\{r_1, r_2\}, \{r_3\}, \{r_4\}\})$. From this pp-concept, we have three constant-column biclusters:

- $b_x = (\{r_1, r_2\}, \{c_2, c_3, c_4\})$
- $b_y = (\{r_3\}, \{c_2, c_3, c_4\})$
- $b_z = (\{r_4\}, \{c_2, c_3, c_4\})$

Here b_x is contrasting to b_z, because the sign in each column of b_x ('$-\quad +\quad +$') is the opposite of those in b_z ('$+\quad -\quad -$'). Therefore, $(\{r_1, r_2, r_4\}, \{c_2, c_3, c_4\})$ is a CSC bicluster. Meanwhile, b_y is not contrasting to any other constant-column bicluster, thus b_y itself is a CSC bicluster.

Experiment

We described two approaches of CSC biclustering: by constructing scaling matrices and performing constant-column biclustering and PPS on them (*scaling-based*), or by applying PPS

Input: A binary sign matrix S, with a set of columns C and a set of rows R
Output: A set of CSC biclusters from S

```
1  csc := ∅
2  pp_concepts ← PPS in S
3  foreach pp ∈ pp_concepts do
4  |  /* a pp is a pp-concept (A,d)
5  |  where A is the set of attributes
6  |  and d is partition of objects */
7  |  foreach p_a ∈ d do
8  |  |  foreach p_b ∈ d, p_a ≠ p_b do
9  |  |  |  if (p_a, A) is contrasting with (p_b, A) then
10 |  |  |  |  csc.add((p_a ∪ p_b, A))
11 |  |  |  |  remove {p_a, p_b} from d
12 |  |  |  end
13 |  |  end
14 |  end
15 |  foreach remaining_p ∈ d do
16 |  |  csc.add((remaining_p,A))
17 |  end
18 end
19 return csc
```

Algorithm 7: *PPS-based*

directly to the original matrix and searching CSC biclusters in each pp-concept (*PPS-based*). In our experiment, we compare the execution time between the two approaches in some randomly generated numerical datasets. Therefore, given an $m \times n$ dataset, both methods preprocess it into a $\binom{m}{2} \times n$ binary sign matrix.

First, we inspect the effect of number of attributes in both approaches. The result is shown in Figure 3.9. With more attributes (>6), the execution time of *PPS-based* is higher than *scaling-based* although both approaches follow similar growth. As illustrated in Table 3.7b, *scaling-based* produce some matrices and executing PPS in each of them. Therefore, given 10 attributes, *scaling-based* applies PPS in matrices of 9, 8, 7,... and 1 attributes separately, while *PPS-based* applies PPS in a single matrix of 10 attributes. Even without the computation of contrasting constant-column biclusters, the runtime of *PPS-based* is still longer than *scaling-based* (not shown here). This result implies the exponential nature of PPS regarding the number of attributes.

Also, Figure 3.10 shows that the execution time of *scaling-based* is better than *PPS-based* with larger number of objects. Similar with the first experiment, this also suggests that executing PPS in one $m \times n$ matrix needs more execution time than executing PPS in $m \times (n-1)$, $m \times (n-2)$, $\ldots m \times 1$ matrices separately. Therefore, *scaling-based* should be preferred over *PPS-based*.

3.5.2 Using pattern structures of signed partition

In the task of CSC bicluster discovery in a formal context (G, M, I), here we explore an approach based on an extension of partition pattern structures. Instead of partition of objects in G as described in Sect. 3.1.2, here we use *partition of attributes* in M. It is still similar to an object partition since an attribute partition covers every attribute in M and there is no overlapping between any two partition components.

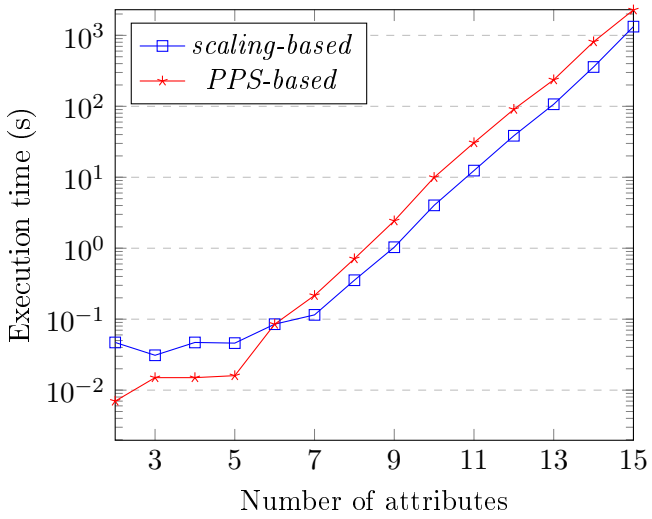


Figure 3.9: Comparison of *scaling-based* and *PPS-based* with number of objects = 20 and varying number of attributes.

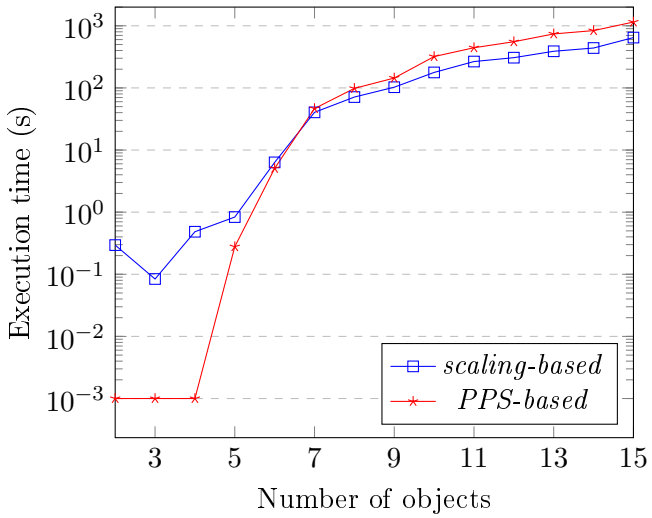


Figure 3.10: Comparison of *scaling-based* and *PPS-based* with number of attributes = 15 and varying number of objects.

Table 3.9: Running example for coherent-sign-changes biclustering.

$\mathcal{M}$	m_1	m_2	m_3	m_4
g_1	+	+	−	−
g_2	+	+	−	−
g_3	−	−	+	−
g_4	+	+	+	+
g_5	−	−	−	−

To formally define our signed partition, first we define the notion of signed attribute (comparable to marked objects in Sect. 3.3) and signed partition component as follows.

Definition 8 (Signed attribute)**.** Let M be a set of attributes, $m \in M$ be an attribute, and $* \in \{-, +\}$ be a sign. A *signed attribute* m^* is an attribute m having a sign $*$.

Definition 9 (Signed partition component)**.** A *signed partition component* (or *sp-component*) c is a subset of M, where each attribute in c is associated to their corresponding sign $*$. Therefore, $c = \{m_1^*, \cdots, m_n^*\}$.

For example, m_1^+ is a signed attribute where the sign $+$ is given to m_1, and $\{m_1^+, m_2^-, m_4^+\}$ is a signed partition component. Since an sp-component contains not only attributes but also their associated sign, we define the equality of two sp-components according to these two aspects as follows.

Definition 10 (SP-component equality)**.** Any two sp-components are equal iff both contain the same set of attributes, and they have either entirely same sign or entirely opposite sign.

Therefore, if we have $c_1 = \{m_1^+, m_2^-, m_4^+\}$, $c_2 = \{m_1^+, m_2^-, m_4^+\}$, and $c_3 = \{m_1^-, m_2^+, m_4^-\}$, then $c_1 = c_2 = c_3$.

Definition 11 (Signed partition)**.** A *signed partition* (or *s-partition*) d is a collection of sp-components, written as $d = \{c_1, \cdots, c_n\}$, such that every attribute in M is present in exactly one sp-component.

For example, given $M = \{m_1, \cdots, m_4\}$, then $\{\{m_1^+, m_2^-, m_4^+\}, \{m_3^+\}\}$ is a valid signed partition of M. The set of all possible s-partitions is denoted as D. This allows us to create an s-partition mapping $\delta : G \to D$ which assigns an object to an s-partition over M. For an object m, $\delta(m)$ is an s-partition containing only one sp-component. This sp-component contains all attributes in M with the corresponding sign according to the object g. Example from Table 3.9:

$$\begin{aligned}
\delta(g_1) = \delta(g_2) &= \{\{m_1^+, m_2^+, m_3^-, m_4^-\}\} \\
\delta(g_3) &= \{\{m_1^-, m_2^-, m_3^+, m_4^-\}\} \\
\delta(g_4) &= \{\{m_1^+, m_2^+, m_3^+, m_4^+\}\} \\
\delta(g_5) &= \{\{m_1^-, m_2^-, m_3^-, m_4^-\}\}.
\end{aligned}$$

Notice that since the sp-components in $\delta(g_4)$ and $\delta(g_5)$ contain the same attributes with entirely opposite sign, according to Def. 10 we have $\delta(g_4) = \delta(g_5)$. This mapping is formulated as follows:

$$\begin{aligned}
\delta(g) &= \{\{m_j^{*_j} | m_j \in M\}\} \\
&\text{where } *_j = m_j(g).
\end{aligned} \tag{3.10}$$

For the task of CSC bicluster discovery, here we define relations between any two s-partitions. The set of all possible s-partitions D is a meet-semilattice where we can define the meet of any two s-partitions.

First, we define the notation $m(c)$ as the sign of an attribute m in an sp-component c. For example, if $c = \{m_1^+, m_2^-, m_3^-\}$, then $m_1(c) = +$. With this notation, we define the similarity ($\cap^{\pm}$) between any two sp-components as:

$$\begin{aligned}
c_1 \cap^{\pm} c_2 = \{&\{m_j^* \in c_1 | m_j(c_1) = m_j(c_2)\}, \\
&\{m_j^* \in c_1 | m_j(c_1) = \neg m_j(c_2)\}\},
\end{aligned} \tag{3.11}$$

where $*$ corresponds to the sign of m_j in c_1, i.e. $m_j(c_1)$.

In other words, the operator $\cap^{\pm}$ between c_1 and c_2 gives $\{c_{12}, c_{1|2}\}$. The c_{12} represents all attributes who are present in c_1 and c_2 with the same sign, while $c_{1|2}$ represents all attributes who are present in c_1 and c_2, but with opposite sign. The signs in the resulting sp-component are the same as those in the first sp-component. Example:

$$\begin{aligned} \text{if } c_x &= \{m_1^+, m_2^-, m_3^-, m_4^-\} \\ \text{and } c_y &= \{m_1^+, m_2^-, m_3^+, m_4^+, m_5^-\}, \\ \text{then } c_x \cap^{\pm} c_y &= \{\{m_1^+, m_2^-\}, \{m_3^-, m_4^-\}\}. \end{aligned}$$

Since the signs in $c_{1|2}$ follow the first sp-component, the result of $c_1 \cap^{\pm} c_2$ could be different to $c_2 \cap^{\pm} c_1$. This can be resolved by Def. 10 that ensures the commutativity of $\cap^{\pm}$. For example:

$$\begin{aligned} c_x \cap^{\pm} c_y &= \{\{m_1^+, m_2^-\}, \{m_3^-, m_4^-\}\}, \\ c_y \cap^{\pm} c_x &= \{\{m_1^+, m_2^-\}, \{m_3^+, m_4^+\}\}, \\ c_x \cap^{\pm} c_y &= c_y \cap^{\pm} c_x. \end{aligned}$$

Having defined the similarity of any two sp-components, we can now define the similarity of any two s-partitions. The similarity (or the meet) of two s-partitions $d_1 = \{c_1 \cdots c_k\}$ and $d_2 = \{c_1 \cdots c_n\}$, with $k = |d_1|$ and $n = |d_2|$, is defined as:

$$d_1 \sqcap d_2 = \{c_i \cap^{\pm} c_j | \forall c_i \in d_1, c_j \in d_2\}, \tag{3.12}$$

and the order between two s-partitions is given by:

$$d_1 \sqsubseteq d_2 \iff d_1 \sqcap d_2 = d_1. \tag{3.13}$$

Let C the set of all sp-components in M, and D is the set of all s-partitions in M. We have $\cap^{\pm} : C^2 \to D$ and $\sqcap : D^2 \to D$. Example from Table 3.9:

$$\begin{aligned} \delta(g_1) \sqcap \delta(g_3) &= \{\{m_1^+, m_2^+, m_3^-, m_4^-\}\} \sqcap \{\{m_1^-, m_2^-, m_3^+, m_4^-\}\} \\ &= \{\{m_4^-\}, \{m_1^+, m_2^+, m_3^-\}\}. \end{aligned}$$

Suppose that $d_1 = \{\{m_4^-\}, \{m_1^+, m_2^+, m_3^-\}\}$. Then $d_1 \sqsubseteq \delta(g_1)$, $d_1 \sqsubseteq \delta(g_2)$, and $d_1 \sqsubseteq \delta(g_3)$.

In order to define a partial order among $d \in D$, the $\sqcap$ operator has to be commutative, idempotent, and associative. These properties are shown in the following propositions.

Proposition 1. The operator $\sqcap$ is commutative, i.e. $d_1 \sqcap d_2 = d_2 \sqcap d_1$.

Proof. Consider $d_1 = \{c_1 \cdots c_n\}$ with $n = |d_1|$ and $d_2 = \{c_1 \cdots c_k\}$ with $k = |d_2|$.

$$\begin{aligned} d_1 \sqcap d_2 &= d_2 \sqcap d_1 \\ \{c_i \cap^{\pm} c_j | \forall c_i \in d_1, c_j \in d_2\} &= \{c_j \cap^{\pm} c_i | \forall c_i \in d_1, c_j \in d_2\} \end{aligned}$$

It is previously stated that $\cap^{\pm}$ is also commutative. Therefore, both sides of the equation above are equal. □

Proposition 2. The operator $\sqcap$ is idempotent, i.e. $d_1 \sqcap d_1 = d_1$.

Proof. Consider $d_1 = \{c_1 \cdots c_n\}$ with $n = |d_1|$.

$$d_1 \sqcap d_1 = \{c_i \cap^{\pm} c_j | \forall c_i \in d_1, c_j \in d_1\}$$

Since there is no overlap among $c_i \in d_1$, then $c_i \cap^{\pm} c_j$ is an empty set for $i \neq j$. Therefore :

$$\begin{aligned} d_1 \sqcap d_1 &= \{c_i \cap^{\pm} c_j | \forall c_i, c_j \in d_1 \text{ and } i = j\} \\ &= \{c_i \cap^{\pm} c_i | \forall c_i \in d_1\} \\ &= \{c_i | c_i \in d_1\} \\ &= d_1 \end{aligned}$$

□

Proposition 3. The operator $\sqcap$ is associative, i.e. $(d_1 \sqcap d_2) \sqcap d_3 = d_1 \sqcap (d_2 \sqcap d_3)$.

Proof. Consider $d_1 = \{c_1 \cdots c_n\}$ with $n = |d_1|$, $d_2 = \{c_1 \cdots c_p\}$ with $p = |d_2|$, and $d_3 = \{c_1 \cdots c_q\}$ with $q = |d_3|$.

$$\begin{aligned} &(d_1 \sqcap d_2) \sqcap d_3 \\ =&\{c_i \cap^{\pm} c_j | \forall c_i \in d_1, c_j \in d_2\} \sqcap d_3 \\ =&\{c_i \cap^{\pm} c_j \cap^{\pm} c_k | \forall c_i \in d_1, c_j \in d_2, c_k \in d_3\} \\ =&d_1 \sqcap \{c_j \cap^{\pm} c_k | \forall c_j \in d_2, c_k \in d_3\} \\ =&d_1 \sqcap (d_2 \sqcap d_3) \end{aligned}$$

□

With the definition of similarity ($\sqcap$) and the associated partial ordering ($\sqsubseteq$) between two s-partitions, we then define the notion of signed partition pattern concept in the following subsection.

Let G a set of objects, M a set of attributes. The lattice of s-partitions of M is $(D, \sqcap)$, where $\delta : G \to D$ maps an object to an s-partition as defined in Eq. 3.10. A *signed partition pattern structure* is determined by the triple $(G, (D, \sqcap), \delta)$, where the derivation operators for $A \subseteq G$ and $d \in D$ are defined as:

$$A^{\square} = \prod_{g \in A} \delta(g), \tag{3.14}$$

$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\}. \tag{3.15}$$

(A, d) is a *signed partition pattern concept* (or *spp-concept*) when $A^{\square} = d$ and $d^{\square} = A$. From an spp-concept (A, d), a CSC bicluster is any pair (A, c) where $c \in d$ (we can ignore the attribute signs in c). All spp-concepts from Table 3.9 are listed in Table 3.10. In the concept $(\{g_1, g_2, g_3\}, \{\{m_1^+, m_2^+, m_3^-\}, \{m_4^-\}\})$ for example, we can find the CSC bicluster $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$. Looking back to the original table, this CSC bicluster means that in $A = \{g_1, g_2, g_3\}$, we have $m_1(A) \simeq m_2(A) \simeq m_3(A)$ (recall the definition of $\simeq$ in Def. 6).

The order between any two spp-concepts is given by $(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2$ or $d_2 \sqsubseteq d_1$. Using this order, the lattice of all spp-concepts from Table 3.9 can be constructed and is shown in Figure 3.11. It should be noticed that the lattice is readable and interpretable only if its size is small. This lattice is useful not only for understanding the hierarchical structure among all biclusters, but also for detecting maximal biclusters.

Table 3.10: All sign partition pattern concepts from Table 3.9 and their corresponding CSC biclusters.

Concept		CSC bicluster	
Extent	Intent	Objects	Attributes
$\{g_1, g_2\}$	$\{\{m_1^+, m_2^+, m_3^-, m_4^-\}\}$	$\{g_1, g_2\}$	$\{m_1, m_2, m_3, m_4\}$
$\{g_3\}$	$\{\{m_1^-, m_2^-, m_3^+, m_4^-\}\}$	$\{g_3\}$	$\{m_1, m_2, m_3, m_4\}$
$\{g_4, g_5\}$	$\{\{m_1^+, m_2^+, m_3^+, m_4^+\}\}$	$\{g_4, g_5\}$	$\{m_1, m_2, m_3, m_4\}$
$\{g_1, g_2, g_3\}$	$\{\{m_1^+, m_2^+, m_3^-\}, \{m_4^-\}\}$	$\{g_1, g_2, g_3\}$	$\{m_1, m_2, m_3\}$
		$\{g_1, g_2, g_3\}$	$\{m_4\}$
$\{g_1, g_2, g_4, g_5\}$	$\{\{m_1^+, m_2^+\}, \{m_3^+, m_4^+\}\}$	$\{g_1, g_2, g_4, g_5\}$	$\{m_1, m_2\}$
		$\{g_1, g_2, g_4, g_5\}$	$\{m_3, m_4\}$
$\{g_3, g_4, g_5\}$	$\{\{m_1^+, m_2^+, m_4^+\}, \{m_3^+\}\}$	$\{g_3, g_4, g_5\}$	$\{m_1, m_2, m_4\}$
		$\{g_3, g_4, g_5\}$	$\{m_3\}$
$\{g_1, g_2, g_3, g_4, g_5\}$	$\{\{m_1^+, m_2^+\}, \{m_3^+\}, \{m_4^+\}\}$	$\{g_1, g_2, g_3, g_4, g_5\}$	$\{m_1, m_2\}$
		$\{g_1, g_2, g_3, g_4, g_5\}$	$\{m_3\}$
		$\{g_1, g_2, g_3, g_4, g_5\}$	$\{m_4\}$

The bicluster $(\{g_1, g_2, g_4, g_5\}, \{m_1, m_2\})$ from Table 3.9 is not maximal, since we can add g_3 that constructs another bicluster $(\{g_1, g_2, g_3, g_4, g_5\}, \{m_1, m_2\})$. The non-maximal biclusters can be detected from the concept lattice [59]. Consider two concepts (A_1, d_1) and (A_2, d_2) such that $(A_1, d_1) \leq (A_2, d_2)$, and an sp-component c. The bicluster (A_1, c) is maximal iff $c \in d_1$ and $c \notin d_2$.

For example, consider the concept $p_1 = (\{g_1, g_2, g_4, g_5\}, \{\{m_1^+, m_2^+\}, \{m_3^+, m_4^+\}\})$ and $p_2 = (\{g_1, g_2, g_3, g_4, g_5\}, \{\{m_1^+, m_2^+\}, \{m_3^+\}, \{m_4^+\}\})$, where $p_1 \leq p_2$. We see that the sp-component $\{m_1^+, m_2^+\}$ is in the intent of both p_1 and p_2. Therefore, the bicluster $(\{g_1, g_2, g_4, g_5\}, \{m_1, m_2\})$ from p_1 is not maximal.

CSC bicluster is a submatrix in a binary matrix. Therefore, given a numerical matrix, it is required to transform it into binary matrix. This can be done by scaling, for example by introducing a threshold, and each numerical value can be transformed to $+$ or $-$ based on whether it is above or below the threshold. In a gene expression data for example, a threshold can be the normal expression level for each gene. An expression that is above (or below) this normal level should be transformed to $+$ (or $-$ respectively).

In the task of mining formal concepts, the AddIntent can be used for any pattern structures by defining the meet ($\sqcap$) and the order ($\sqsubseteq$) between any two descriptions. Having defined the meet in Eq. 3.12 and the order in Eq. 3.13, we then use AddIntent to mine spp-concepts in a binary matrix. Furthermore, this algorithm is also effective for building a concept lattice, which is needed in our case to detect the maximality of any CSC bicluster.

Experiments

As previously explained in Section 3.5.2, CSC biclusters can be found in any spp-concept. Therefore, from a binary matrix, we should retrieve all spp-concepts. To do that, we reuse the AddIntent source code in [19] by modifying the definition $\sqcap$ and $\sqsubseteq$ operators. This algorithm also allows us to build the lattice of all concepts. We can reduce the lattice by choosing a threshold θ that applies to the intent of a concept. This threshold defines the minimal size of an sp-component that an intent should have. Since the lattice construction is performed by a bottom-up ap-

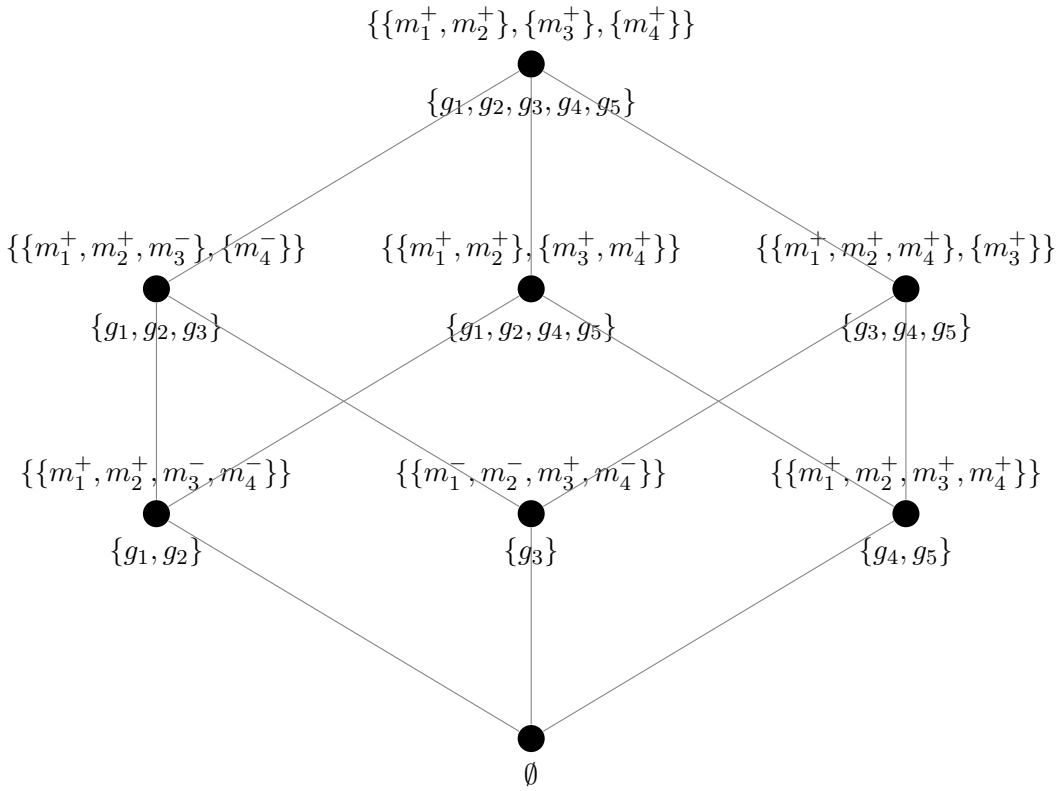


Figure 3.11: Sign partition pattern lattice for pattern structure in Table 3.9.

proach, this threshold allows to "prune" the lattice. For example, with $\theta = 3$, the lattice for Table 3.9 does not contain the concept $(\{g_1, g_2, g_4, g_5\}, \{\{m_1^+, m_2^+\}, \{m_3^+, m_4^+\}\})$–since none of its sp-components has ≥ 3 attributes–as shown in Figure 3.12.

We tested our method to lymphoma dataset provided in [5] and BicAT yeast dataset[1]. The lymphoma dataset contains the numerical expression levels of 4026 genes over 96 tissues, while the yeast contains 419 probesets and 70 conditions. The objective of CSC bicluster discovery in these datasets is to find a subset of genes that behave in a consistent way over a subset of tissues. For this task, we convert them to binary by assigning $-$ and $+$ for the values < 0 and ≥ 0 respectively.

For lymphoma dataset, the number of concepts and runtime for different thresholds are listed in Table 3.11. We tested three thresholds: 70, 80, and 90. As shown here, higher θ can reduce the number of concepts, and consequently reduce the runtime.

For $\theta = 70$, around 157K concepts are obtained. Among them, only 153K have extent size larger than 1. This means that there are 153K CSC biclusters having at least 70 columns and at least 2 rows. Furthermore, still with $\theta = 70$, the largest extent size is 8, meaning that among the biclusters with ≥ 70 columns, there are no bicluster with > 8 rows.

Higher θ corresponds to higher number of columns in the biclusters and thus lower number of rows. With $\theta = 90$, we see that among the biclusters with ≥ 90 rows, there are no bicluster with > 3 rows.

Furthermore, the effect of θ can also be seen for both datasets in Fig. 3.13 and 3.14. For each iteration of AddIntent algorithm, we calculate the number of concepts generated so far. With

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

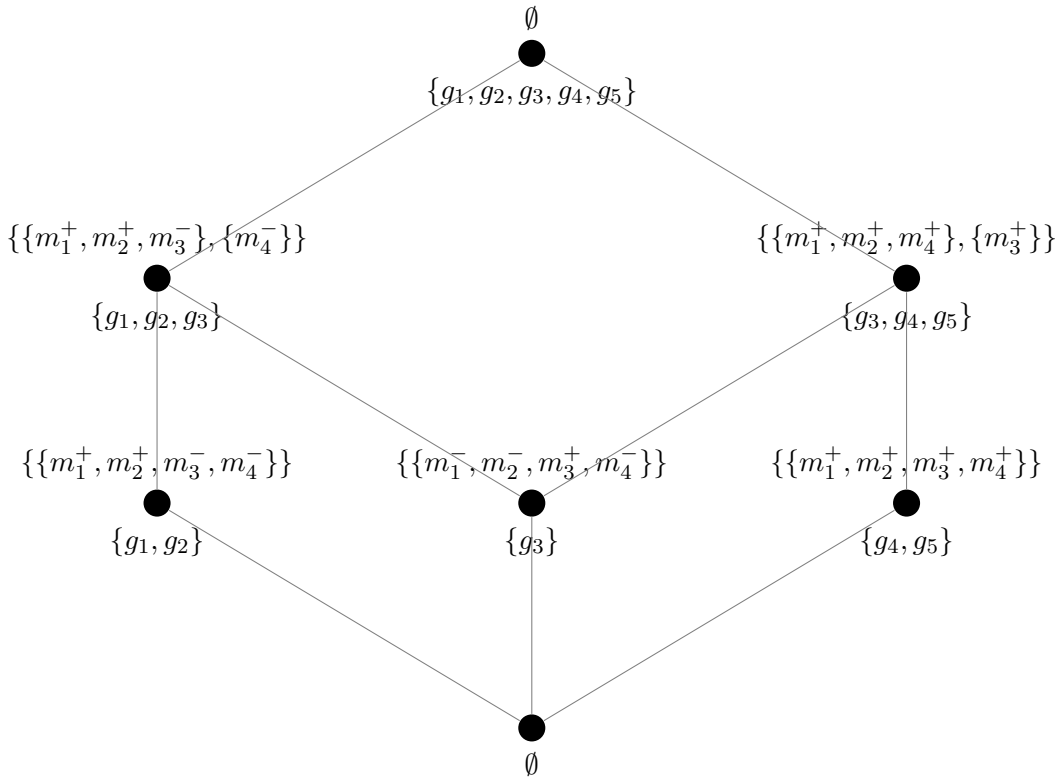


Figure 3.12: Sign partition pattern lattice for pattern structure in Table 3.9 with $\theta = 3$.

Table 3.11: Experiments on lymphoma dataset.

θ	Runtime (minutes)	Number of concepts All	Number of concepts Extent size > 1	Largest extent size
70	229.4	157K	153K	8
80	62.9	7K	2K	7
90	62.1	4K	83	3

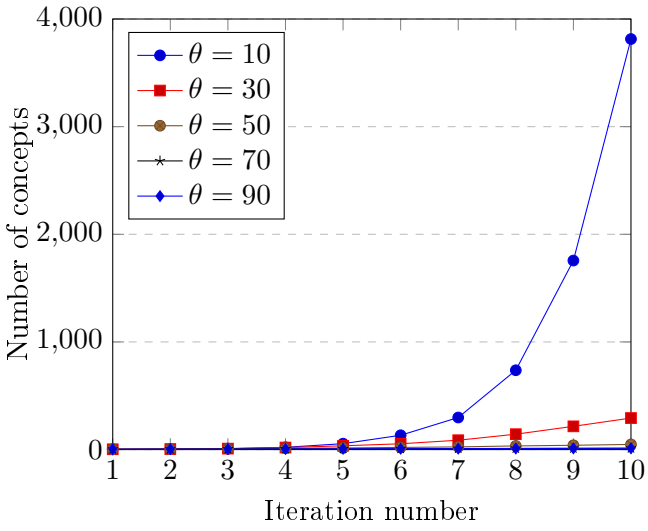


Figure 3.13: Number of concepts generated until each iteration of AddIntent over lymphoma dataset.

smaller θ, we discard smaller number of concepts, thus higher number of remaining concepts. This high number of concepts in turn contributes to the higher runtime as previously seen in Table 3.11.

3.6 Conclusion

In this chapter we presented a unified approach based on partition pattern structure for discovering different types of biclusters in a given numerical matrix. Partition pattern structures are an extension of FCA, based on complex object descriptions and adapted similarity operators. We showed that by designing a suitable object descriptions and associated similarity operators, we can extract the main types of biclusters from pattern concepts.

However, for OP biclustering in this thesis, we do not consider yet the equality (the case with $m_i(g) = m_j(g)$). Another aspect that should be studied is the possibility of a matrix that has another sign in addition to $+$ and $-$ for CSC biclustering. This new sign can represent a missing value, or in the case of threshold-based transformation, a value that is equal to the threshold. It can be resolved using tolerance relation introduced in [58], such that a value equal to the threshold should be regarded as similar to both $+$ and $-$. In the case of missing value, it can be resolved by modifying the definition of attribute partition which permits an attribute to be not present in any sp-component. This modification may consequently require modifications on the definition of meet and order between s-partitions.

Eventually, the CSC bicluster discovery can be applied in a domain besides gene expression data. Frequent gradual itemset mining was studied in [25] to extract gradual rules from a numerical table, e.g. a hotel price table with 3 attributes: m_p for city population, m_d for distance from city center, and m_r for room price. We may find an sp-component $\{m_p^+, m_d^-, m_r^+\}$. It is related to the rule saying "the more/less m_p, the less/more m_d, then the more/less m_r".

Moreover, some studies ([21, 52]) show the benefits of biclustering in the recommendation systems. In a user–movie rating matrix for example, a constant-column bicluster represents a set of users having the same interest across a set of movies. On the other hand, a CSC bicluster

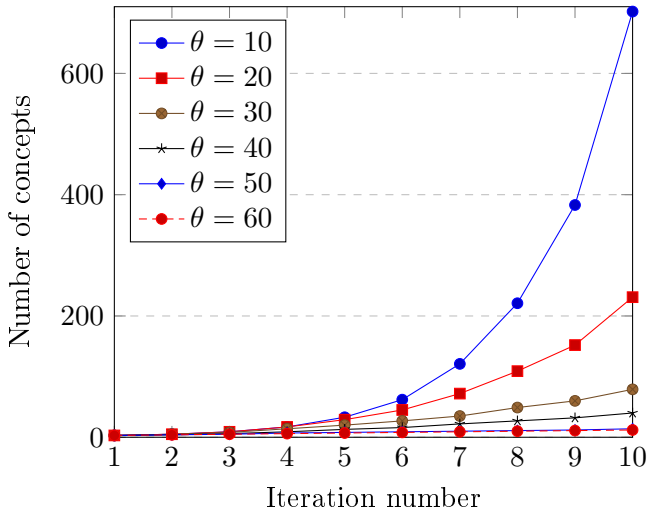


Figure 3.14: Number of concepts generated until each iteration of AddIntent over BicAT yeast dataset.

in this matrix represents a set of users having either the same or the opposite interest. This is useful for a new user u: we can recommend movies liked by users similar to u and movies disliked by users opposite to u.

Chapter 4

Biclustering with interval pattern structures

Contents

Previously in Chapter 3, we explore the adaptation of partition pattern structures to discover many types of biclusters. One limitation of this approach is that we need some discretization to obtain non-perfect biclusters. In this chapter, we show how to mitigate this limitation using interval pattern structures, which was proposed to discover similar-column biclusters in a numerical matrix. Here we extend this method such that it is also able to discover additive and multiplicative biclusters.

Interval pattern structures are a solution in the problem of scaling many-valued contexts, presented in Sect. 4.1. We revisit some definitions of interval pattern structures in Sect. 4.2 and its application in similar-column biclustering in Sect. 4.3. Then, we describe how it can be extended to additive and multiplicative biclustering in Sect. 4.4. We explain some modifications of Close by One in order to enumerate interval pattern concepts in Sect. 4.5 and some experiments in Sect. 4.6. Lastly we conclude this chapter in Sect. 4.7.

4.1 Scaling of many-valued contexts

In standard FCA, the attributes M are binary, giving us the binary context (G, M, I). It can be considered that if $m(g) = 1$, then the object g has the attribute m. These attributes are called one-valued attributes, and the context is called one-valued context.

	m_1	m_2
g_1	a	b
g_2	b	b
g_3	a	a
g_4	a	a

$\Longrightarrow$

	$m_1 = \text{a}$	$m_1 = \text{b}$	$m_2 = \text{a}$	$m_2 = \text{b}$
g_1	×			×
g_2		×		×
g_3	×		×	
g_4	×		×	

Figure 4.1: An example of nominal scaling.

	m_1	m_2
g_1	good	very good
g_2	good	good
g_3	very good	very good
g_4	very good	good

$\Longrightarrow$

	m_1		m_2	
	good	very good	good	very good
g_1	×		×	×
g_2	×		×	
g_3	×	×	×	×
g_4	×	×	×	

Figure 4.2: An example of ordinal scaling.

A many-valued context (G, M, W, I) is composed by a set of objects G, a set of attributes M, a set of values W, and a ternary relation $I \subseteq G \times M \times W$. $(g, m, w) \in I$ means that an attribute m has a value w for object g. This can also be written as $m(g) = w$. To find concepts in a many-valued context, it has to be transformed into a one-valued context, by the process called *scaling*.

In nominal scaling, the values of an attribute mutually exclude each other. An example of this scaling is illustrated in Fig. 4.1. An example of a concept on this scaled context is $(\{g_3, g_4\}, \{m_1 = a, m_2 = a\})$, which can be regarded as a constant-column bicluster $(\{g_3, g_4\}, \{m_1, m_2\})$ on Fig. 4.1 left.

In ordinal scaling, illustrated in Fig. 4.2, the values are ordered, and there is implication among values. For example in Fig. 4.2 left, a value "very good" is stronger than "good", therefore "very good" implies "good". A concept $(\{g_1, g_2, g_3, g_4\}, \{m_1 = \text{ good}, m_2 = \text{ good}\})$ in Fig. 4.2 right means that all objects have the value "good" in m_1 and m_2, although this is not explicit in the multi-valued context in Fig. 4.2 left.

Furthermore, if we have bipolar ordering where there are two extremities, an interordinal scaling should be applied to obtain a one-valued context. An example of this scaling is depicted in Fig. 4.3. An interordinal scaling can be considered as interval of values. In the one-valued context in Fig. 4.3 right, there is a concept:

$$(\{g_1, g_2\}, \{m_1 \leq 1, m_1 \leq 2, m_1 \geq 1, m_2 \leq 2, m_2 \geq 1\}), \tag{4.1}$$

which reflects an interval for each attribute. For g_1 and g_2, the values of m_1 and m_2 are in the interval $[1, 1]$ and $[1, 2]$ respectively.

4.2 Interval pattern structures

Interval pattern structures (IPS) was introduced by Kaytoue et al. [60] to overcome some limitations of enumerating concepts in many-valued context using interordinal scaling. As depicted in Fig. 4.3, the interordinal attribute m_1 is scaled to four binary attributes. With larger range of values, the number of attributes in the scaled context can be significantly increased. Furthermore, the representation of a concept from interordinal scaling is not efficient, since there are

	m_1	m_2
g_1	1	1
g_2	1	2
g_3	2	1
g_4	2	2

$\Longrightarrow$

	m_1				m_2			
	≤ 1	≤ 2	≥ 1	≥ 2	≤ 1	≤ 2	≥ 1	≥ 2
g_1	×	×	×		×	×	×	
g_2	×	×	×			×	×	×
g_3		×	×	×	×	×	×	
g_4		×	×	×		×	×	×

Figure 4.3: An example of interordinal scaling.

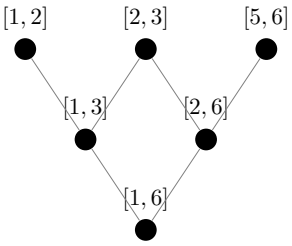


Figure 4.4: The meet-semilattice of a set of intervals.

redundancies. The concept in Eq. 4.1 for example, has the intent with $m_1 \leq 1$ and $m_1 \leq 2$. The presence of $m_1 \leq 2$ is redundant since it is implied from $m_1 \leq 1$.

An interval of values between a and b is written as $[a, b]$. Moreover, vector of intervals d is a list of intervals, written as $d = \langle [a_1, b_1], [a_2, b_2] \cdots \rangle$. In IPS, an interval $[a_1, b_1]$ is subsumed by another interval $[a_2, b_2]$ if $[a_2, b_2]$ is found within $[a_1, b_1]$. Accordingly, a vector d_1 is subsumed by another vector d_2 if every interval in d_1 is subsumed by the corresponding interval in d_2.

The ordering among intervals makes the set of intervals I a meet-semilattice, i.e. there exists one greatest lower bound for any subset of I. The meet-semilattice of an example set of intervals is illustrated in Fig. 4.4. The greatest lower bound –or the meet ($\sqcap$)– of two intervals $[a_1, b_1]$ and $[a_2, b_2]$ is the smallest interval containing both $[a_1, b_1]$ and $[a_2, b_2]$. This corresponds to the convex hull of the intervals, and defined as:

$$[a_1, b_1] \sqcap [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]. \tag{4.2}$$

For example, $[1, 2] \sqcap [5, 6]$ is $[1, 6]$. The subsumption order among intervals can be defined with the meet operator. If an interval $[a_2, b_2]$ is found within $[a_1, b_1]$, then following Eq. 4.2 the meet between both of them is $[a_1, b_1]$. Therefore:

$$[a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \Longleftrightarrow [a_1, b_1] \sqsubseteq [a_2, b_2]. \tag{4.3}$$

Consequently, the set of vectors of intervals is also a meet-semilattice. The meet of two vectors is the convex hull of each corresponding interval. Therefore, given $d_1 = \langle [a_1, b_1], \cdots, [a_n, b_n] \rangle$ and $d_2 = \langle [c_1, d_1], \cdots, [c_n, d_n] \rangle$:

$$d_1 \sqcap d_2 = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,n]}. \tag{4.4}$$

For example, $\langle [1, 2][4, 6] \rangle \sqcap \langle [3, 4][8, 9] \rangle$ is $\langle [1, 4][4, 9] \rangle$. Furthermore, the subsumption order among vectors follows the subsumption order among intervals in Eq. 4.3, i.e.:

$$d_1 \sqcap d_2 = d_1 \Longleftrightarrow d_1 \sqsubseteq d_2, \tag{4.5}$$

Table 4.1: A numerical matrix, and a similar-column bicluster in gray.

G	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	2	1	1	0	6
g_3	2	2	1	7	6
g_4	8	9	2	6	7

reflecting that d_1 is subsumed by d_2 if each interval in d_1 is subsumed by the corresponding interval in d_2.

Interval pattern structures was originally proposed to obtain similar-column (SC) bicluster in gene expression dataset (GED). A GED is typically represented as a 2-D numerical matrix with genes as rows and conditions as columns, as shown in Table 4.1. In this matrix, the submatrix $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3, m_5\})$ is an SC bicluster, defined by the parameter $\theta = 1$. It means that the range of values of each column in the submatrix has the length of at most 1.

An interval pattern structure is determined by the triple $(G, (D, \sqcap), \delta)$, where G is the set of objects, $(D, \sqcap)$ is the lattice of all interval vectors, and $\delta : G \to D$ maps an object to an interval vector. In a numerical matrix, an interval vector describes the values of every column. For example, the description of g_1 –denoted by $\delta(g_1)$– in Table 4.1 is $\langle[1,1][2,2][2,2][1,1][6,6]\rangle$.

Given $A \subseteq G$ and $d \in D$, $A^\diamond$ is the meet of all objects' interval vectors (defined in Eq. 4.4), while $d^\diamond$ is the set of objects whose interval vector subsumes d (defined in Eq. 4.5) or:

$$A^\diamond = \prod_{g \in A} \delta(g),$$
$$d^\diamond = \{g \in G | d \sqsubseteq \delta(g)\}.$$

Following the definition of a concept of any pattern structure, an interval pattern concept is a pair (A, d), for $A \subseteq G$ and $d \in D$, where $A^\diamond = d$ and $d^\diamond = A$. Furthermore, the set of interval pattern concepts is partially ordered, and is a complete lattice. An interval pattern concept (A_1, d_1) is a subconcept of (A_2, d_2) if $A_1 \subseteq A_2$ (dually $d_2 \sqsubseteq d_1$). All interval pattern concepts from Table 4.1 are listed in Table 4.2.

4.3 Similar-column biclustering with IPS

A similar-column (SC) bicluster can be found in an interval pattern concept by introducing a parameter θ. This parameter acts as the maximum difference between any two values to be considered as similar. For example, with $\theta = 1$, the value 1 is similar to 2, but not similar to 3.

In calculating the similarity between any two descriptions, if the length of an interval is larger than θ, then the star sign $(*)$ is put as the interval. From Table 4.1, $\delta(g_2) \sqcap \delta(g_4)$ without θ is $\langle[2,8][1,9][1,2][0,6][6,7]\rangle$, and with $\theta = 1$ is $\langle * * [1,2] * [6,7]\rangle$.

The similarity $\sqcap$ between $*$ and any other interval is $*$. For example, suppose that we have two descriptions $d_x = \langle[1,1][2,3]\rangle$ and $d_y = \langle[2,2]*\rangle$. Then, $d_x \sqcap d_y = \langle[1,2]*\rangle$. This also means that $*$ is subsumed by any other interval. Therefore, the description of each object in Table 4.1 subsumes $\langle * * [1,2] * [6,7]\rangle$. With $\theta = 1$, $(\{g_1, g_2, g_3, g_4\}, \langle * * [1,2] * [6,7]\rangle)$ is an interval pattern concept. All interval pattern concepts with $\theta = 1$ from Table 4.1 are listed in Table 4.3.

From an interval pattern concept, an SC bicluster can be formed by the concept's extent and the set of columns where the interval is not $*$ in the concept's intent. For example, from the

Table 4.2: All interval pattern concepts from Table 4.1.

A	d
$\{g_1\}$	$\langle[1,1][2,2][2,2][1,1][6,6]\rangle$
$\{g_2\}$	$\langle[2,2][1,1][1,1][0,0][6,6]\rangle$
$\{g_3\}$	$\langle[2,2][2,2][1,1][7,7][6,6]\rangle$
$\{g_4\}$	$\langle[8,8][9,9][2,2][6,6][7,7]\rangle$
$\{g_1, g_2\}$	$\langle[1,2][1,2][1,2][0,1][6,6]\rangle$
$\{g_1, g_3\}$	$\langle[1,2][2,2][1,2][1,7][6,6]\rangle$
$\{g_1, g_4\}$	$\langle[1,8][2,9][2,2][1,6][6,7]\rangle$
$\{g_2, g_3\}$	$\langle[2,2][1,2][1,1][7,7][6,6]\rangle$
$\{g_2, g_4\}$	$\langle[2,8][1,9][1,2][0,6][6,7]\rangle$
$\{g_3, g_4\}$	$\langle[2,8][2,9][1,2][6,7][6,7]\rangle$
$\{g_1, g_2, g_3\}$	$\langle[1,2][1,2][1,2][0,7][6,6]\rangle$
$\{g_1, g_2, g_4\}$	$\langle[1,8][1,9][1,2][0,6][6,7]\rangle$
$\{g_1, g_3, g_4\}$	$\langle[1,8][2,9][1,2][1,7][6,7]\rangle$
$\{g_2, g_3, g_4\}$	$\langle[2,8][1,9][1,2][0,7][6,7]\rangle$
$\{g_1, g_2, g_3, g_4\}$	$\langle[1,8][1,9][1,2][0,7][6,7]\rangle$

Table 4.3: All interval pattern concepts with $\theta = 1$ from Table 4.1.

Extent	Intent
$\{g_1\}$	$\langle[1,1][2,2][2,2][1,1][6,6]\rangle$
$\{g_2\}$	$\langle[2,2][1,1][1,1][0,0][6,6]\rangle$
$\{g_3\}$	$\langle[2,2][2,2][1,1][7,7][6,6]\rangle$
$\{g_4\}$	$\langle[8,8][9,9][2,2][6,6][7,7]\rangle$
$\{g_1, g_2\}$	$\langle[1,2][1,2][1,2][0,1][6,6]\rangle$
$\{g_1, g_3\}$	$\langle[1,2][2,2][1,2] * [6,6]\rangle$
$\{g_1, g_4\}$	$\langle * * [2,2] * [6,7]\rangle$
$\{g_2, g_3\}$	$\langle[2,2][1,2][1,1] * [6,6]\rangle$
$\{g_3, g_4\}$	$\langle * * [1,2][6,7][6,7]\rangle$
$\{g_1, g_2, g_3\}$	$\langle[1,2][1,2][1,2] * [6,6]\rangle$
$\{g_1, g_2, g_3, g_4\}$	$\langle * * [1,2] * [6,7]\rangle$

Table 4.4: Example of additive column alignments. (a) Original table and the additive bicluster in gray, (b) alignment on m_1, (c) alignment on m_2.

	m_1	m_2	m_3	m_4
g_1	4	1	3	0
g_2	6	4	6	3
g_3	2	3	5	2
g_4	1	6	1	7

(a)

	m_1	m_2	m_3	m_4
g_1	4	1	3	0
g_2	4	2	4	1
g_3	4	5	7	4
g_4	4	9	4	10

(b)

	m_1	m_2	m_3	m_4
g_1	4	1	3	0
g_2	3	1	3	0
g_3	0	1	3	0
g_4	-4	1	-4	2

(c)

Table 4.5: Example of multiplicative column alignments. (a) Original table and the multiplicative bicluster in gray, (b) alignment on m_2.

	m_1	m_2	m_3	m_4
g_1	3	1	2	3
g_2	1	3	6	9
g_3	2	2	4	6
g_4	1	2	6	8

(a)

	m_1	m_2	m_3	m_4
g_1	3	1	2	3
g_2	0.3	1	2	3
g_3	1	1	2	3
g_4	0.5	1	3	4

(b)

concept $(\{g_1, g_2, g_3\}, \langle [1,2][1,2][1,2] * [6,6] \rangle)$, $(\{g_1, g_2, g_3\}, \{m_2, m_2, m_3, m_5\})$ is an SC bicluster with $\theta = 1$.

By using IPS with parameter θ, constant-column biclustering is a specific case of SC biclustering. It can be noticed that with $\theta = 0$, we obtain intervals with length 0, and that corresponds to constant-column biclusters.

4.4 Additive and multiplicative biclustering

An additive bicluster is a submatrix where there is a constant (or similar) difference between any two columns across all of its rows. Constant (or similar) column biclustering is a specific case of additive biclustering. Using this fact, we can obtain additive biclusters by aligning (similar to [43]) each column, and then find interval pattern concepts on the alignments.

Table 4.4 provides an example of column alignment for additive biclustering. The original matrix is shown in Table 4.4a, having 4 rows and 4 columns. The submatrix $(\{g_1, g_2, g_3\}, \{m_2, m_3, m_4\})$ is an additive bicluster in the original matrix. This bicluster can be found by applying constant-column or similar-column biclustering to the column alignments. Table 4.4b shows the first column alignment, can be seen by the consistency of the first column (m_1). In this example, each object value is converted such that its m_1 value is equal to the value of m_1 in g_1. This means that the values 0, -2, 2, and 3 are added to g_1, g_2, g_3, and g_4 respectively. This alignment is repeated for every column. Table 4.4c is the alignment of m_2, by adding 0, -3, -2, and -5 to g_1, g_2, g_3, and g_4 respectively.

Constant-column (or similar-column) biclustering is applied to every column alignment to find additive biclusters. In the second column alignment (Table 4.4c), we obtain $(\{g_1, g_2, g_3\}$, $\{m_2, m_3, m_4\})$ as a constant-column bicluster. This corresponds to the additive bicluster $(\{g_1$, $g_2, g_3\}, \{m_2, m_3, m_4\})$ in the original matrix (Table 4.4a).

Multiplicative biclusters can also be obtained using similar column alignment. In multiplicative column alignment, instead of adding values to each row, we multiply each row such that

a column has a constant value. Table 4.5b shows the second column alignment of the original matrix in Table 4.5a. Here, a constant value is achieved for m_2 by multiplying g_1, g_2, g_3, and g_4 by 1, $\frac{1}{3}$, $\frac{1}{2}$, and $\frac{1}{2}$ respectively. Then, by applying IPS to each alignment, we can obtain the multiplicative biclusters. For example, constant-column biclustering using IPS in Table 4.5b returns $(\{g_1, g_2, g_3\}, \{m_2, m_3, m_4\})$, which is the corresponding multiplicative bicluster in Table 4.5a.

4.5 Concept mining

Being a generalization of FCA, the mining of interval pattern concepts can be performed using some existing algorithms that generate a complete list of formal concepts. In this chapter, we use CloseByOne (CbO) [65] since it requires us to only define the similarity ($\sqcap$) and subsumption relation ($\sqsubseteq$) of any two descriptions.

In a given numerical matrix, we may obtain an exponential number of interval pattern concepts. To reduce the number of concepts, we should introduce some parameters that can filter out some uninteresting concepts.

The first parameter, θ, is previously mentioned in Sect. 4.3. It limits the length of intervals, and later in Sect. 4.6 we demonstrate the effect of θ on the runtime and number of concepts.

The second parameter *min_col* is the minimum number of columns in the retrieved biclusters. The number of columns in a bicluster corresponds to the number of non-star intervals in the concept's intent. For example, the concept with intent $\langle * * [2,2] * [6,7] \rangle$ gives us a bicluster with two columns (the third and the fifth). To take into account the *min_col* parameter, it is necessary to modify the definition of similarity between any two descriptions. In addition to the definition of $\sqcap$ in Eq. 4.4, we verify if the number of non-star intervals in the description is less than *min_col*. If yes, then every interval is converted to $*$. In Table 4.1 with $\theta = 1$, $g_1 \sqcap g_4$ is $\langle * * [2,2] * [6,7] \rangle$. Using *min_col* $= 3$ for example, $g_1 \sqcap g_4$ becomes $\langle * * * * * \rangle$.

Related to *min_col* is *min_row*, a parameter that put a constraint on the number of rows in a bicluster. It corresponds to the number of objects in a concept's extent. The inclusion of *min_row* to CbO is given in Algorithm 8. Here we see that the calculation of $Y^{\diamond}$ (all objects whose description subsumes Y) in line 21 is performed only if the number of objects in Z is at least *min_col*.

4.6 Experiments

In this section, we report some experimental results to show the scalability of IPS in the task of biclustering. First, in Sect. 4.6.1 we study the effects of our parameters (θ, *min_row*, and *min_col*) to the runtime. Then, we compare our IPS-based biclustering method to other pattern-based method (Sect. 4.6.2) and numerical method (Sec. 4.6.3).

4.6.1 Effects of parameters

We use the synthetic datasets provided by Henriques and Madeira [43]. First, we investigate the effect of θ on the runtime and the number of concepts. The results are illustrated in Fig. 4.5. The left figure confirms that the larger θ generates more interval pattern concepts, and generally longer runtime as it can be seen in the right figure. The $\theta = 0.4$ requires longer runtime than $\theta = 0.5$ to 0.9. This is normal since for similar number of concepts, the probability of smaller θ obtaining a concept is smaller than the larger θ. Using CbO with smaller θ, a candidate concept will have shorter intervals in its intent, hence smaller number of objects whose description

```
1  CloseByOne:
2  L := ∅
3  foreach g ∈ G do
4  |  Process({g}, g, (g^◇◇, g^◇))
5  end
6  L is the concept set
7
8
9  Process(A, g, (C, D)):
10 if {h|h ∈ C\A and h < g} = ∅ then
11 |  if size(C) ≥ min_row then
12 |  |  L := h ∪ {(C, D)}
13 |  end
14 |  foreach f ∈ {h|h ∈ G\C and g < h} do
15 |  |  Z := C ∪ {f}
16 |  |  Y := D ∩ {f}^◇
17 |  |  if size(Z) < min_row then
18 |  |  |  X := Z
19 |  |  end
20 |  |  else
21 |  |  |  X := Y^◇
22 |  |  end
23 |  |  Process(Z, f, (X, Y))
24 |  end
25 end
```

Algorithm 8: The introduction of *min_row* to CbO algorithm (line 11–13 and 17–22).

subsumes this interval. For example, with smaller θ, algorithm may perform:

$$\cdots \rightarrow \boxed{\{g_1, g_2\}^{\diamond\diamond} = \{g_1, g_2\}} \rightarrow \boxed{\{g_1, g_2, g_3\}^{\diamond\diamond} = \{g_1, g_2, g_3\}} \rightarrow \cdots,$$

while larger θ may "jump" from $\{g_1, g_2\}^{\diamond\diamond}$ to larger set of objects:

$$\cdots \rightarrow \boxed{\{g_1, g_2\}^{\diamond\diamond} = \{g_1, \cdots, g_4\}} \rightarrow \boxed{\{g_1, \cdots, g_5\}^{\diamond\diamond} = \{g_1, \cdots g_8\}} \rightarrow \cdots.$$

The effect of *min_col* is shown in Fig. 4.6. Lesser *min_col* produces more concepts, and therefore longer runtime. Similarly, Fig. 4.7 shows that larger *min_row* generates more concepts, but not necessarily longer runtime. This is due to our approach of including *min_row* to CbO (see Algorithm 8) that prevents the calculation of $Y^{\diamond}$ until Z has at least *min_row* objects. For example, a concept with extent $\{g_1, g_2, \cdots g_{10}\}$ may be generated when Z has 10 objects. Using smaller *min_row*, this concept may also be generated when Z has only 2 objects (if for example $\{g_1, g_2\}^{\diamond\diamond} = \{g_1, g_2, \cdots g_{10}\}$).

4.6.2 Comparison with pattern-based method

In the previous experiments, the CbO algorithm was terminated until all interval pattern concepts were retrieved. In the following experiment, CbO is terminated until 500 concepts are found. We compare them to BicPAM [43] that uses a discretization parameter (as a number

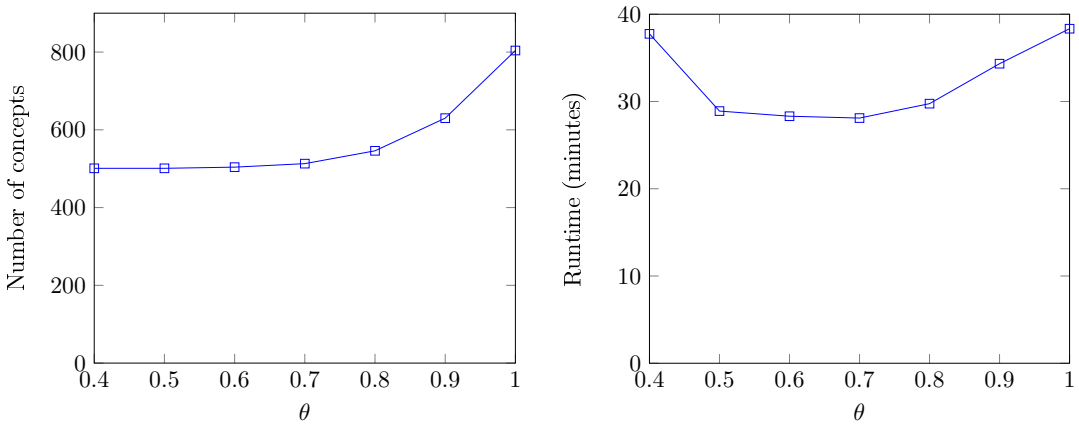


Figure 4.5: Effect of θ on a 500×60 dataset with $min\_col = 20$ and $min\_row = 1$.

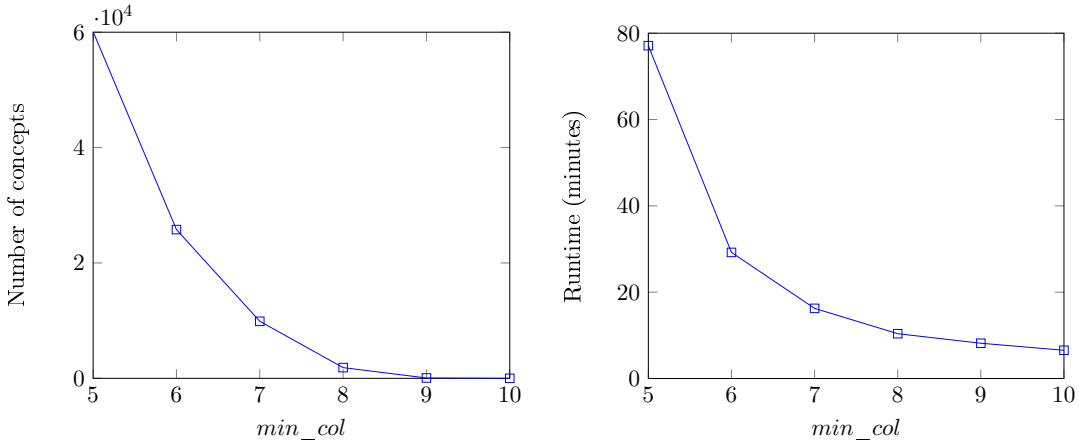


Figure 4.6: Effect of min_col on a 500×60 dataset with $\theta = 1$ and $min\_row = 5$.

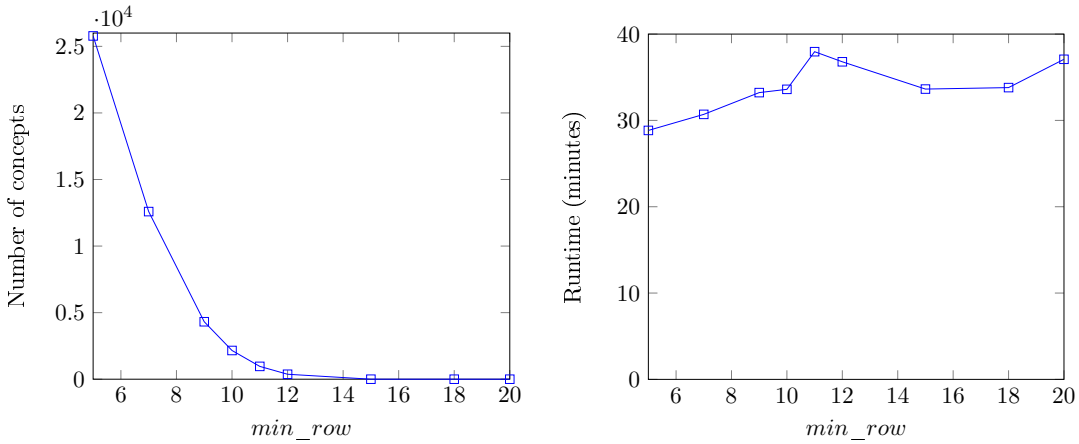


Figure 4.7: Effect of min_row on a 500×60 dataset with $\theta = 1$ and $min\_col = 6$.

Table 4.6: Comparison with BicPAM on 1000×100 dataset. For the IPS, the parameters $min\_row = 10$ and $min\_col = 5$ are used, with varying θ.

Method	Parameter	Runtime (s)	Number of biclusters
BicPAM	alphabet = 20	<15	∼100
	alphabet = 10	<15	<200
	alphabet = 7	<15	<200
	alphabet = 5	<30	∼200
IPS	$\theta = 1$	37	500
	$\theta = 2$	>500	500
	$\theta = 4$	47	500
	$\theta = 8$	39	500

of alphabet/items), while IPS uses the length of intervals as θ. After the mapping step (normalization, discretization, and missing values and noise handling), BicPAM applies a pattern mining method (F2G [47] as default), and the closing step (extension, merging, and filtering) is performed. Results in Table 4.6 show a similar performance of both methods. It should be noted that the number of biclusters from BicPAM is lower due to the closing step.

Furthermore, still from Table 4.6, the runtime of IPS is not exactly correlated with θ (especially with $\theta = 2$), similar to our previous experiment shown in Fig. 4.5. Overall, with similar runtime, biclustering with IPS can return similar number of biclusters without discretization.

4.6.3 Comparison with numerical method

In this subsection we are interested to compare IPS as a pattern-based method with a Cheng and Church (CC) algorithm [18] as a numerical method in additive biclustering. CC algorithm finds a set of biclusters having MSR (mean squared residue) less than a user-defined threshold. These biclusters correspond to constant, similar-row, similar-column, and additive biclusters.

We use BicAT Yeast dataset[2], a data matrix containing the expression levels of 419 probesets over 70 conditions (hence a 419×70 numerical matrix). The BicAT Yeast dataset is also available in the `biclust`[3] package of R, along with an implementation of CC algorithm which we use in our experiments.

The comparison is given in Table 4.7. The first column of this table provides the largest MSR among all retrieved biclusters. Smaller MSR means that the additive bicluster has less noise. In CC, the largest MSR is given by user-defined parameter, while in IPS, it is the consequence of the parameter θ. Larger θ means that the intervals are wider, which are more prone to noise. Consequently, larger θ results in biclusters with larger MSR.

In the fourth column, the number of biclusters is given for each entry. For CC, this is the final number of all biclusters that can be found, while for IPS, this is the number of biclusters found within the corresponding runtime. For example, in the first entry, after finding 97 biclusters, CC can not find any other possible biclusters, so it stops at 13.37 seconds. On the other hand, in the second entry, in 13.02 seconds IPS can find 500 bicluster, but it could find other biclusters with more runtime.

In general, with similar MSR, IPS can retrieve more biclusters in similar runtime. Furthermore, with similar MSR and runtime, CC can find larger biclusters. This is because CC generally

[2] http://www.tik.ee.ethz.ch/sop/bicat/
[3] https://CRAN.R-project.org/package=biclust

Table 4.7: Comparison of additive biclustering using Cheng and Church's algorithm (CC) and IPS-based algorithm. The size of a bicluster is defined as the number of its cells.

MSR max.	Method & parameters	Runtime (s)	Number of biclusters	Size of biclusters max	Size of biclusters min
0.001	CC	13.37	97	56	6
0.001	IPS ($\theta = 0.1, r = 2, c = 10$)	13.02	500	58	20
0.01	CC	3.80	35	630	18
0.009	IPS ($\theta = 0.3, r = 2, c = 10$)	3.02	100	52	20
0.02	CC	2.21	20	1275	30
0.02	IPS ($\theta = 0.5, r = 4, c = 10$)	2.52	75	110	40
0.03	CC	1.43	16	2070	33
0.03	IPS ($\theta = 1, r = 7, c = 10$)	1.56	50	240	70
0.04	CC	1.24	13	2821	39
0.043	IPS ($\theta = 1.4, r = 10, c = 10$)	2.14	50	430	310

MSR = mean squared residue; θ = maximum width of an interval;

r = minimum number of rows in a bicluster = minimum extent size of a concept;

c = minimum number of columns in a bicluster = minimum number of non-* intervals in a concept's intent.

finds the largest bicluster at the first iteration. Then it "masks" the submatrix corresponding to this bicluster, so in the next iteration, another smaller bicluster is found. The iteration stops when the algorithm can not find any more bicluster. Consequently, this also means that CC retrieves many small biclusters. These small biclusters can be discarded in IPS via the parameters *min_row* and *min_col* (in Table 4.7, r and c respectively). Therefore, in the table, we can see that with similar MSR and runtime, the smallest bicluster found by IPS is still larger than the smallest bicluster found by CC.

4.7 Conclusion

In this chapter, we propose an alternative method of biclustering in numerical datasets. Discretization is a general preprocessing step while working with numerical values. Here we explore the possibility of working directly on numerical datasets without discretization. This can be achieved using interval pattern structures, where a bicluster can be found from any interval pattern concept. To filter the number of concepts (which can be very large) it is necessary to provide some parameters, like the length of intervals, minimum number of rows and columns, or even minimum number of biclusters. Our experiments show that these parameters can reduce the computation to a reasonable runtime.

We use the CbO algorithm, a formal concept generator that can be generalized to interval pattern structures. In-Close 2 [8] in particular is faster than CbO in formal concept mining, but its efficiency in interval pattern concept mining should be studied. Another future research is to extend our FCA-based approach to other types of biclusters, e.g. coherent-evolution, coherent-sign-changes, etc. Furthermore, the existence of missing values and/or outliers should be considered in improving the proposed biclustering method.

Part II

Experiments in mining complex data

Chapter 5

Sequence mining within FCA for analyzing visitor trajectories

Contents

In this chapter we explore the application of FCA and pattern structures in analyzing visitor trajectories for CrossCult project. First, we describe the project in Sect. 5.1. Then, we explain some definitions of general sequence mining in Sect. 5.2 and sequence mining in FCA in Sect. 5.3. The approach for analyzing visitor trajectories using FCA is explained in Sect. 5.5–5.7. Lastly, we end this chapter with some discussions and conclusions in Sect. 5.8.

5.1 CrossCult project

This chapter is focused on KDD in the framework of CrossCult (http://www.crosscult.eu/), a European Project about cultural heritage. The general idea of CrossCult is to support the emergence of a European cultural heritage by allowing visitors in different locations (e.g. museum, city, archaeological site) to attentively think of their visit at a European level by using adapted computer-based devices.

The objectives of this project can be summarized as two groups: humanities and innovation. In the first group, the researchers investigate how historical facts are retained by an individual, such that an insight of how the same facts may be differently interpreted by different person can be gained.

This thesis is a part of the second objective, which is the innovation. In this objective, a semantic knowledge base is created, which contains multi-level, cross-repository interconnection of venues and digital cultural heritage resources. Furthermore, some new technologies are developed for smart venues and cities. Particularly, we assess the impact of state-of-the-art technologies of geolocalization, content adaptation, and personalization in mobile applications.

In concretizing the history reflection framework, the CrossCult technology platform is demonstrated to stimulate reflection and reinterpretation of history. This is performed by using mobile tools and applications on real sites across Europe. These demonstrations are classified into four flagship pilots:

- Pilot 1: Large multi-thematic venue. This pilot works in a specific venue having broad theme of collection, particularly the National Gallery in London, where the application provides new visitor experiences associated with exploring and searching gallery collections. Some features on this pilot include a personalized guide through paintings and search for similar paintings.

- Pilot 2: Multiple small venues, which works on various related venues distributed in multiple cities. This pilot was applied in Chaves (Portugal), Ludo (Spain), Montegrotto Terme (Italy), and Epidaurus (Greece), where each of them has various Roman historic sites. The visitors were expected to collectively engage in the discovery of unknown connections between these related sites.

- Pilot 3: One venue, non-typical transversal connections. Presented in Archaeological Museum of Tripolis (Greece), this flagship pilot focuses on the life of women in ancient Greece by providing the hidden narratives between physical and digital objects in the museum.

- Pilot 4: Multiple cities, "past & present" interplay. As the name suggests, this pilot connects physical cultural heritages that shape the everyday spheres of life. It was presented in Tandil (Argentina), Luxembourg City, and Valletta by means of outdoor exhibition, treasure hunt, etc.

For each of these flagship pilots, a mobile application was developed as the interaction tool between visitors as users and CrossCult team as knowledge provider. This application registers the trajectory of each visitor while interacting with them by various means. A map is provided in the application and visitors can explore the venue and read in their mobile phone the description of any point of interest. In certain venues, a maze is given in the application such that visitors solve and unlock paths by providing information about certain items. In addition to the simplicity for visitors to discover point of interests, the application also allows them to view objects on map and 360 degrees view of the field location, share contents among their friends, rate contents, contribute personal stories and react on other's stories, etc. This leads the visitors to understand the multi-faceted perspective on the study of history and to reckon various possible interpretations of any historical items.

5.2 The mining of sequences

Pattern mining is the task of finding repeated occurrences in a dataset. For example, in a data about customer transactions, an objective can be to find a set of items that are frequently ordered in a single transaction. This specific task in pattern mining is related to sequential pattern mining. We recall below the basic definitions that we will need.

Definition 12. A sequence is an ordered list $\langle s_1 s_2 \dots s_m \rangle$, where s_i is an itemset $\{i_1, \dots, i_n\}$, and m is the *size* of a sequence. The *length* of a sequence is the total number of items, i.e. $\sum |s_i|$.

Definition 13. A sequence $s = \langle s_1 s_2 \dots s_m \rangle$ is a subsequence of sequence $s' = \langle s'_1 s'_2 \dots s'_n \rangle$, denoted by $s \preceq s'$, if there exist indices $1 \leq i_1 < i_2 < \dots < i_m \leq n$ such that $s_j \subseteq s'_{i_j}$ for all $j = 1 \dots m$ and $m \leq n$.

Therefore, the sequence $\langle \{a\}\{d\} \rangle$ is a subsequence of $\langle \{a,b\}\{a,c,d\} \rangle$, while sequence $\langle \{c\}\{d\} \rangle$ is not.

One way of evaluating the quality of a subsequence is to compute the support of the subsequence. Given a user-defined threshold, the subsequence can be "frequent", i.e. the support is above the threshold.

Definition 14. Let $\mathcal{S}$ be a database of sequences. The *support* of a sequence s in $\mathcal{S}$ is: $support(s, \mathcal{S}) = |\{s_i \in \mathcal{S}; s \preceq s_i\}|$

There exist algorithms that can retrieve all frequent sequences [39, 10]. Beside mining frequent sequences, another complex task is finding homogeneous sequence groups (clustering). To achieve such a task, a distance or a similarity measure between two sequences has to be defined. The similarity measure sim_{ACS} was proposed in [29], which counts the number of all common subsequences (ACS), formulated as:

$$sim_{ACS}(S_i, S_j) = \frac{\phi_C(S_i, S_j)}{\max\{\phi_D(S_i), \phi_D(S_j)\}}$$

where $\phi_C(S_i, S_j)$ is the number of all common distinct subsequences between S_i and S_j, while $\phi_D(S_i)$ is the number of all distinct subsequences of S_i.

5.3 Sequence mining in FCA

In this section we briefly present the two algorithms that are adapted for mining the trajectories of visitors in a museum, namely `MFCS` [15] and `MRGS` [19]. The names of the algorithms are not used as such in the papers but here we use them by commodity. Both algorithms are original and very efficient, and among the few algorithms performing sequence mining in the framework of FCA.

Using the definition of subsequence Def. 13 as an order between two sequences, the set of sequences is partially ordered where a sequence subsumes their subsequences. Meanwhile the set of sequences is not a meet-semilattice, since there is no infimum (single largest element) for a certain subset. This is because two sequences can have two common subsequences which are not comparable to each other. Consider the set of sequences formed by the sequence $\langle \{a\}\{b\}\{c\} \rangle$, $\langle \{a\}\{c\}\{b\} \rangle$, and their subsequences. This set is partially ordered, whose Hasse diagram is depicted in Fig. 5.1. For $\langle \{a\}\{b\}\{c\} \rangle$ and $\langle \{a\}\{c\}\{b\} \rangle$, $\langle \{a\}\{b\} \rangle$ and $\langle \{a\}\{c\} \rangle$ are their highest lower bounds, but there is no single largest element among them.

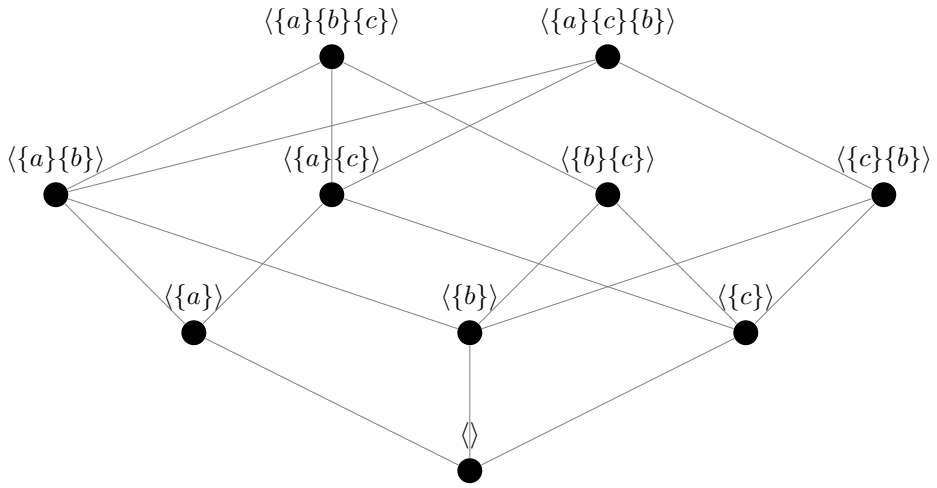


Figure 5.1: Hasse diagram of $\langle\{a\}\{b\}\{c\}\rangle$, $\langle\{a\}\{c\}\{b\}\rangle$, and their subsequences.

Therefore, the notion of a set of closed sequences is used in sequence pattern structures. Given a set of sequence S, its set of closed sequences is S^+, and defined as:

$$S^+ = \{s_i \in S | \nexists s_j \in S : s_i \prec s_j\},$$

meaning that in S^+ there is no two sequences where one is a proper subsequence of another. The set S^+ is the description d of an object in pattern structures. The d_1 is subsumed by d_2 if for every sequence in d_1 there exists its supersequence in d_2, or:

$$d_1 \sqsubseteq d_2 \iff \forall s_i \in d_1 \exists s_j \in d_2 : s_i \preceq s_j.$$

For example, $\{\langle\{a\}\rangle, \langle\{b\}\rangle\}$ is subsumed by $\{\langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle\}$

The set of d, called D, is a meet-semilattice since there exists an infimum for every subset of D. The meet of $d_1 \in D$ and $d_2 \in D$, written $d_1 \sqcap d_2$, is their set of common closed subsequences (SCCS), defined as:

$$d_1 \sqcap d_2 = \{s_i \cap s_j | \forall s_i \in d_1, \forall s_j \in d_2\}^+,$$

where $s_i \cap s_j$ is the set of all common subsequences between s_i and s_j. For example, given $d_1 = \{\langle\{a\}\{b\}\{c\}\rangle\}$ and $d_2 = \{\langle\{a\}\{c\}\{b\}\rangle\}$, then $d_1 \sqcap d_2 = \{\langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle\}$.

As explained previously, there are two approaches of sequence mining using pattern structures: `MFCS` and `MRGS`. One difference between them is their definition of $\cap$, the set of common subsequences. `MFCS` only considers "contiguous" subsequences, while `MRGS` considers all subsequences, contiguous or not.

`MFCS` was originally introduced for mining trajectories of patients in hospitals. The algorithm is based on pattern structures and projections, and stability as well. One important characteristic of `MFCS` is that it mines contiguous subsequences, or stated differently, subsequences without any gap between items. This is due to the fact that physicians are mainly interested in consecutive events when analyzing healthcare trajectories. In addition, but this is not needed in our framework, `MFCS` is able to take into account a partial ordering – given by domain knowledge for example – defined on the items composing the sequences.

`MRGS` is also a sequence miner based on pattern structures but with a different purpose. The objective of `MRGS` is to mine rare rather than frequent subsequences, and in particular long subsequences with special characteristics. The algorithm is based on a specific pattern structure

Table 5.1: An example of one visitor trajectory.

Start time	End time	Item name
12:55:39	12:58:05	Crafts and Arts
12:58:06	12:58:22	Religion and Cult
12:58:22	12:58:27	Building Methods and Facilities
12:58:29	13:05:09	Wooden Tools

of subsequences, where the similarity operation is based on the discovery of SCCS. The SCCS operation is based on a directed graph of alignments (DAG of alignments) which guides the mining of common subsequences. The algorithm shows very good performances and is most probably one of the few algorithms whose objective is the mining of rare subsequences. In our framework, we adapted MRGS and the support threshold for comparison purposes with frequent subsequences. However, in our context we will use MRGS as a standard sequence miner and we will be interested in frequent subsequences.

5.4 The dataset of museum visitors

5.4.1 The museum

In the framework of the CrossCult project, we are working on a specific dataset about the trajectories of 254 visitors in Hecht Museum in Haifa, Israel [67]. In the raw dataset, a visitor trajectory contains a list of visited items, where each visit is composed of three elements: "start time", "end time", and "item name". An example is presented in Table 5.1. When modeling trajectories into sequences, in this chapter we consider only the "item name", such that every itemset contains only one item. For simplicity, we omit the curly brackets to describe an itemset. Therefore for the remaining part of this chapter we will write $\langle\{a\}\{d\}\{e\}\rangle$ as $\langle a, d, e\rangle$.

A visitor can have visits with various time lengths. In order to obtain more meaningful results and to reduce the complexity, we only consider visits lasting at least 90 seconds, but this is a parameter than can be relaxed or more constrained. Thirty-eight trajectories have no visit more than this threshold, so they are ignored, leaving us with 216 trajectories. Moreover, we model each trajectory as a sequence of visited items. Therefore, for trajectory in Table 5.1, the corresponding sequence is $\langle$Crafts and Arts, Wooden Tools$\rangle$. This preprocessing results in sequences of various size. Forty-five sequences have only one itemset, while three sequences have more than 15 itemsets.

We group the museum items according to their location, so that we obtain 8 categories of items. To illustrate the numbering of items, the first two categories and their items are listed in Table 5.2. We convert the raw dataset into sequences of items, where each item is represented by its ID. We define the IDs such that we can infer the category of an item by its first digit. Therefore, we obtain a dataset of 216 sequences of visitor trajectories (named V_1–V_{216}) where each sequence is composed by a list of IDs, as illustrated in Table 5.3.

MFCS and MRGS algorithms are suitable for our dataset, since we want to include the hierarchical information in the mining process. For example, given two sequences $\langle 102, 203\rangle$ and $\langle 103, 204\rangle$, it is interesting to mine subsequence $\langle 1, 2\rangle$.

Table 5.2: Grouping of museum items.

Category	Items and their ID
1	Entrance Reuben Hecht (101), Symbols Jewish Menorah (102), Persian Cult (103), Jerusalem Photo (104)
2	Religion and Cult (201), Everyday Pottery (202), Phoenician Writing (203), Burial Tradition (204), Building Methods and Facilities (205), Maritime Commerce (206), Imported Pottery (207), Crafts and Arts (208)

Table 5.3: Examples of visitor trajectories.

Visitor	Trajectory
V_1	$\langle$101, 101, 401, 704$\rangle$
V_2	$\langle$102, 402, 808, 206, 808$\rangle$
V_3	$\langle$302, 102, 201, 302, 705, 402, 802$\rangle$
V_4	$\langle$104, 704, 602, 302, 402, 103$\rangle$

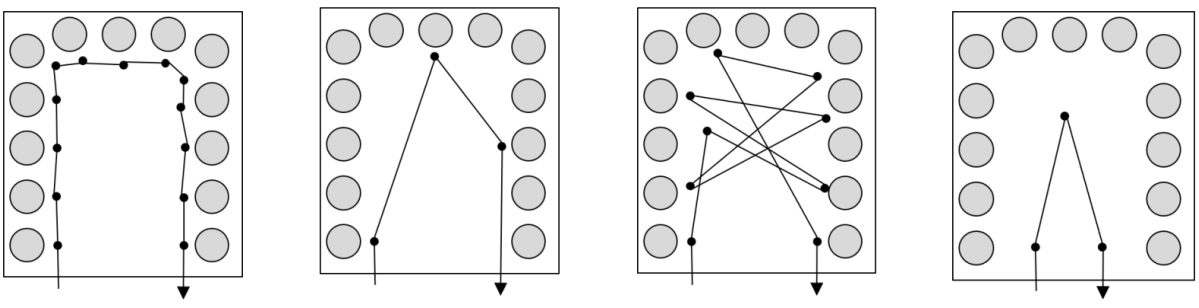


Figure 5.2: An illustration of four hypothetical visiting styles [62, 105], from left to right: *ant*, *grasshopper*, *butterfly*, and *fish*. Each image represents a room, with gray circles as the items, and connected black dots as a visitor path.

5.4.2 The four visiting styles

In a seminal work about the typing of visitor styles in a museum [96], four main behaviors have been detected and described, leading to different recommendations all along a visit [62, 105]. These four styles are illustrated in Fig. 5.2 and summarized below:

- The *ant* is a visitor who will surely see all the works following their location order in the museum. Then the recommendation can be the following item, but depending also on some environmental factors such as the crowd in the museum, the accessibility of the item and the fatigue of the visitor.
- The *grasshopper* is a visitor who will see only certain artworks, jumping from one to another. Then, to encourage such a person to visit more items, the recommendation can be to visit items having a content similar to items already visited.
- The *butterfly* is a visitor wanting to discover some and not all artworks, without having any exact preferences. Then, the recommendation is open and can be based on surprise (items which are very different one from the other).
- The *fish* is a visitor who does not feel that much interested in the artworks and stays most of the time in the center of the rooms without any precise objective. Then the recommendation can be to visit the most famous items in the museum which are the closer to the current visitor location, for encouraging the visitor to continue the visit and gain more interest.

Indeed, a visitor can change his/her style during a visit and other elements may be of importance, e.g. crowd or fatigue of the visitor.

5.5 Workflow for analyzing the trajectories

In the following, one objective is to map specific subsequences included in the visitor trajectories to each visiting style for characterizing more precisely the style and then making smarter recommendations. To identify the behavior of each visitor, we propose the following workflow:

1. Cluster the visitor trajectories and assign a label for each visitor (Section 5.6).
2. Create two concept lattices using `MFCS` and `MRGS` over the whole dataset (Section 5.7.1).
3. From the two lattices, find jumping emerging patterns (JEPs) for each label (Section 5.8.1).
4. Based on their JEPs, these labels are then mapped into four visiting styles as explained in Section 5.4.2.

5.6 Clustering of trajectories

In this first experiment, we reuse the sim_{ACS} similarity measure for clustering the visitor trajectories. The idea is to check whether it is possible to distinguish the four visiting styles introduced above. We apply hierarchical clustering[4] based on sim_{ACS} to build a distance matrix between individuals. From the resulting dendrogram, we retained 5 clusters denoted by "A", "B", "C", "D",

[4]We use the *hclust* method from the R software [83].

and "E". Four of them are expected to match the four visiting patterns, namely *ant*, *butterfly*, *fish*, and *grasshopper*. The last cluster will gather all non-classified trajectories. These five clusters have various sizes. Cluster "A", "B", "C", "D", and "E" have 11, 11, 59, 102, and 33 visitors respectively.

Actually, it is not easy to directly match the five clusters to corresponding visiting styles. For doing so, we will analyze the subsequences that can be attached to each cluster of trajectories. The benefit of the clustering is actually to provide a label among "A", "B", "C", "D", and "E" to the visitors. Thanks to these labels, we can search the so-called "jumping emerging patterns" and attach a characterization to the clusters based on the mined subsequences.

5.7 The mining of trajectories considered as sequences

5.7.1 Mining subsequences with MFCS and MRGS

Below, we explain the application of the MFCS and MRGS algorithms to the museum dataset and the building of an associated concept lattice. Moreover, as discussed in Section 5.7.2, the mining of jumping sequential patterns will help us to characterize the visitor trajectories.

In MFCS and MRGS, pattern structures are used for mining sequences. The similarity operator ($\sqcap$) between any two sets of sequences is defined as the set of closed common subsequences (SCCS) in the two input sequences. Then, given two sequences, say $S_1 = \langle$401,502,503$\rangle$ and $S_2 = \langle$401,503,502$\rangle$, the similarity between these descriptions is:

$$\begin{aligned}\delta(S_1) \sqcap \delta(S_2) &= \{\langle 401,502,503\rangle\} \sqcap \{\langle 401,503,502\rangle\} \\ &= \{\langle 401,502\rangle, \langle 401,503\rangle\}\end{aligned}$$

In the dataset, the items are grouped into categories (indicated by their first digit) and the SCCS calculation is performed, checking whether two items belong to the same category. Using the MFCS algorithm it becomes:

$$\begin{aligned}\delta(S_1) \sqcap \delta(S_2) &= \{\langle 401,502,503\rangle\} \sqcap \{\langle 401,503,502\rangle\} \\ &= \{\langle 502\rangle, \langle 503\rangle, \langle 401,5,5\rangle\}\end{aligned}$$

It should be noticed that MFCS mines contiguous subsequences, i.e. in Definition 13, $i_k = i_{k-1}+1$ for all $k \in \{2,3,\ldots,m\}$. Furthermore, subsequence $\langle$401,5,5$\rangle$ can be regarded as a generalization, meaning that after item 401, the next two visited items are something in category 5.

In parallel, the default similarity operator of MRGS algorithm can be modified to accommodate our needs, such that non-contiguous common subsequences can be mined:

$$\begin{aligned}\delta(S_1) \sqcap \delta(S_2) &= \{\langle 401,502,503\rangle\} \sqcap \{\langle 401,503,502\rangle\} \\ &= \{\langle 401,502\rangle, \langle 401,503\rangle, \langle 401,5,5\rangle\}\end{aligned}$$

Then, based either on MFCS or MRGS, a concept has an extent including a set of trajectories and an intent including a set of common subsequences. Again, it should be noticed that, based on whether a subsequence is contiguous or not, the obtained concepts are different.

For example, the concepts corresponding to Table 5.3 are shown in Table 5.4. Notice that both algorithms obtain a concept whose extent is V_2, V_3, V_4, albeit with different intent. Based on MRGS, the common subsequence of V_2, V_3, V_4 is $\langle 1, 402\rangle$, while according to MFCS, their common subsequences are $\langle 1\rangle$ and $\langle 402\rangle$. This is because items 1 and 402 are not contiguous in V_3 and V_4.

Table 5.4: The concepts that are computed by of MFCS and MRGS from four visitors in Table 5.3.

Extent	Intent (MFCS)	Intent (MRGS)
V_1	⟨101,101,401,704⟩	
V_2	⟨102,402,808,206,808⟩	
V_3	⟨302,102,201,302,705,402,802⟩	
V_4	⟨104,704,602,302,402,103⟩	
$V_{1,2}$	⟨1,4⟩	*not present*
$V_{1,4}$	⟨1⟩, ⟨4⟩, ⟨704⟩	⟨1,1⟩, ⟨1,4⟩, ⟨1,704⟩
$V_{2,3}$	⟨2⟩, ⟨102⟩, ⟨402,8⟩	⟨102,402,8⟩, ⟨102,2,8⟩
$V_{3,4}$	⟨1⟩, ⟨302⟩, ⟨402⟩, ⟨7⟩	⟨1,302,402⟩, ⟨302,1⟩, ⟨1,7,402⟩
$V_{1,3,4}$	⟨1⟩, ⟨4⟩, ⟨7⟩	⟨1,4⟩, ⟨1,7⟩
$V_{2,3,4}$	⟨1⟩, ⟨402⟩	⟨1,402⟩
$V_{1,2,3,4}$	⟨1⟩, ⟨4⟩	⟨1,4⟩

5.7.2 Jumping emerging patterns

FCA is a non supervised classification process that can be turned into a supervised process thanks to the adding of a target attribute in the context, generally corresponding to a target class. Then the idea is to search for the so-called "Jumping Emerging Patterns" (JEPs) [28]. This approach is already applied in [9] for analyzing and characterizing clusters of biological inhibitors. Here we adapt the same idea for characterizing this time the clusters of visitors discovered with the similarity measure sim_{ACS}.

More precisely, five clusters are discovered by classifying visitor trajectories with sim_{ACS}. These same trajectories are then considered as sequences composed of subsequences. Then a set of characteristic subsequences is extracted and these subsequences are used as "attributes" in a formal context where objects are visitor trajectories. The resulting formal context is completed with an extra attribute corresponding to the "cluster information", i.e. the cluster in which the trajectory is classified according to sim_{ACS}. A concept lattice can then be built from this completed context.

More interestingly, the cluster information is used for characterizing the concepts whose extents include trajectories of a single cluster. The intents – made of subsequences – of these particular concepts are JEPs, and as such they can be used to characterize the corresponding clusters. For example, if the extent of the concept $(\{V_{103}, V_{165}, V_{188}\}, \{\langle 4\rangle, \langle 1\rangle, \langle 306\rangle, \langle 701,707\rangle\})$ includes visitors from cluster B only, then its intent is a JEP for that cluster.

5.8 Discussion

5.8.1 Cluster characterization

Now we are interested in characterizing the five clusters that were introduced in the previous section. For doing so, JEPs are searched in the two concept lattices obtained with MFCS and MRGS algorithms. Some of these concepts are listed in Table 5.5 and Table 5.6.

First, from both MFCS and MRGS, we cannot find any satisfying concept for JEP of cluster "E". This is because among all the concepts whose extent is exclusively from cluster "E", none of them has more than one visitor. If we consider the dataset, among 33 members of cluster "E", 32 of them visit less than 2 items during their whole visit. We can assume that these visitors are not really interested in visiting the museum. Therefore, we can "safely" label this cluster as *fish*.

Table 5.5: Interesting concepts discovered by the `MFCS` algorithm.

Concept ID	Extent	Intent	Support	Cluster
`FA1`	$\{V_{70}, V_{107}, V_{121}, V_{133}, V_{201}, V_{202}\}$	$\{\langle 1,1,402\rangle, \langle 103\rangle, \langle 2\rangle\}$	6	A
`FA2`	$\{V_{70}, V_{93}, V_{107}, V_{121}\}$	$\{\langle 402\rangle, \langle 103,104\rangle\}$	4	A
`FB1`	$\{V_{103}, V_{165}, V_{188}\}$	$\{\langle 4\rangle, \langle 1\rangle, \langle 306\rangle, \langle 701,707\rangle\}$	3	B
`FC1`	$\{V_4, V_8, V_{28}, V_{32}, V_{84}, V_{152}\}$	$\{\langle 102\rangle, \langle 101,1,101\rangle\}$	6	C
`FC2`	$\{V_{53}, V_{152}, V_{169}, V_{189}, V_{190}, V_{203}\}$	$\{\langle 7\rangle, \langle 102,4\rangle\}$	6	C
`FC3`	$\{V_4, V_8, V_{32}\}$	$\{\langle 101,102,101\rangle\}$	3	C
`FD1`	$\{V_{54}, V_{105}, V_{139}, V_{168}\}$	$\{\langle 202,4\rangle\}$	4	D
`FD2`	$\{V_{139}, V_{168}\}$	$\{\langle 202,405,701\rangle\}$	2	D
`FD3`	$\{V_{46}, V_{47}\}$	$\{\langle 101,602\rangle\}$	2	D
`FD4`	$\{V_{89}, V_{163}\}$	$\{\langle 602,203\rangle\}$	2	D

Cluster "D" is more easily distinguishable. Based on subsequences of concept `FD2`–`FD4`, many visitors in this class skip some items. Also, in concept `RD1` and `RD2`, some of them visit other items after item `701`. This is not a natural direction, because items in category `7` are located farther from the entrance than items in category `4` or `5`. We can interpret the visitors of this cluster as *grasshopper*, since they "jump" from one item to another.

Clusters "A", "B", and "C" are relatively similar to each other. The visitors associated to these clusters follow an *ant* behavior: a natural flow (based on `RA1`–`RC1`) and no "jump" (based on `FA1`–`FC2`). However, in `FC3`, three visitors visit `101`, then `102`, then back again to `101`, indicating rather a *butterfly* behavior.

5.8.2 Conclusion

In this chapter, we have presented our experiments in mining visitor trajectories that are modeled as sequences of items. First, we clustered the trajectories according to their common subsequences. Then, we tried to validate the clusters as behaviors (ant, butterfly, fish, and grasshopper). This is done by applying two sequence miners based on FCA to the visitor trajectories, namely `MFCS` and `MRGS`, to discover characterizing contiguous and general subsequences in each cluster.

Our result highlights some interesting patterns that may define visitor behaviors. This can help museum researchers to analyze and evaluate the placement of items and the visiting styles. Moreover, this analysis is useful to build a recommendation system for future visitors, but we did not yet study this aspect.

In this thesis, we only included partial information about the museum in the sequences. More interesting results are expected if other elements are taken into account, such as more general knowledge about history and geography, as well as the duration and time of the visit. Furthermore, the selection of interesting concepts can be also guided by the "stability" of concepts [64]. Finally, from a more dynamic point of view, ongoing information such as comments and state of the visitors during the visit could be also considered for analysis and on-line recommendation.

Table 5.6: Interesting concepts discovered by the MRGS algorithm.

Concept ID	Extent	Intent	Support	Cluster
RA1	$\{V_{70}, V_{107}, V_{121}, V_{133}, V_{201}, V_{202}\}$	$\{\langle 1,1,402,2\rangle, \langle 1,1,4\rangle, \langle 103,402,2\rangle, \langle 103,4\rangle\}$	6	A
RB1	$\{V_{142}, V_{183}, V_{192}\}$	$\{\langle 102,1,1,1,1\rangle, \langle 102,103,1,1\rangle, \langle 1,1,1,1,1\rangle, \langle 1,103,1,1\rangle\}$	3	B
RC1	$\{V_4, V_8, V_{28}, V_{84}, V_{152}\}$	$\{\langle 1,1,1,101\rangle, \langle 1,101,1,101\rangle, \langle 1,1,1,1\rangle, \langle 1,101,1,1\rangle,$ $\langle 101,1,1,1\rangle, \langle 101,101,1,1\rangle, \langle 101,101,101\rangle, \langle 102,101\rangle, \langle 102,1\rangle\}$	5	C
RD1	$\{V_{71}, V_{79}\}$	$\{\langle 701,504\rangle\}$	2	D
RD2	$\{V_{97}, V_{98}\}$	$\{\langle 701,406\rangle\}$	2	D

Chapter 6

Guiding antibacterial discovery based on log-linear analysis

Contents

Feature selection is an important step in KDD, since it reduces the complexity of a dataset. This complexity reduction is an ongoing research in antibacterial drug discovery, where the data is a set of molecules with thousand attributes. In this chapter we discuss the interest of log-linear analysis, especially Chordalysis, applied to the task of feature selection. We begin this chapter by presenting the domain of antibacterial drug discovery in Sect. 6.1 and feature selection in Sect. 6.2. The log-linear analysis and Chordalysis as feature selection are explained in Sect. 6.3–6.4. Lastly, we present the detail and result of experiment in Sect. 6.5–6.7.

6.1 Antibacterial drug discovery

Bacterial and parasitic diseases are the second leading cause of death worldwide[5]. Being safe and effective in treating infectious diseases, antibiotics have been used without diagnostic culture and

[5]According to a recent report on antibiotic research released Sept. 17 by the London School of Economics and Political Science (LSE), 175,000 deaths are attributed to hospital-acquired infections each year in Europe alone.

susceptibility testing. With its overuse, many bacteria undergo evolution, mutation and selection, so they become resistant to those drugs [33]. And because of the emergence of drug-resistant "superbugs", like methicillin-resistant *Staphylococcus aureus* (MRSA), traditional antibiotics such as penicillin and its derivatives are becoming obsolete.

New antibiotics are desperately needed, but the amount of money being spent on the research and development of these drugs is woefully inadequate as major pharmaceutical companies have abandoned or cut back antibiotic research and development. Their decision is also supported by the fact that the investments required for developping new drugs keep increasing while the expected returns drop as the usability span of the newly developed drugs become shorter.

The unwillingness of many pharmaceutical companies leads the effective drug-discovery to be developed by the academic community, including the computer scientists. The convergence of computer science and computational chemistry has given rise to the new field of chemoinformatics. Chemoinformatics combines various research areas and techniques, including machine learning, pattern recognition, molecular dynamics, quantum mechanics, and statistics. It is attracting growing attention and recognition as available computing power and data volumes keep increasing. By harnessing knowledge discovery techniques it aims to offer more efficient drug development methods so as to tackle the pressing challenge posed by drug resistant bacterial diseases.

Considering the millions of available chemicals spread among chemical providers, companies' collections and academic laboratories, the huge number of indexed chemical compounds is both a blessing and a curse. Indeed, the volume of these databases means that they certainly contain molecules that would provide efficient weapons against bacterial diseases but this is also what makes is so difficult to identify the promising candidates among scores of unsuitable chemical compounds. For that purpose, a learning procedure using a limited subset of well identified antibiotics and a large collection of chemical attributes can be used.

Structure and properties of molecules are a gold mine of information to characterize antibiotics. Not only physicochemical and chemical graph properties can be calculated from their structures, but also a large amount of information is obtained from dynamic structure changes and from the interaction between compounds and their targets. For effective use of a flood of information obtained from a large number of compounds, chemical information needs to be converted into meaningful and useful data. Any molecule can be represented by a set of numerical or categorical values, called "descriptors", that characterize its physicochemical and topological properties. Such data can be calculated by a large variety of methods and vary in complexity of the encoded information and in the computer time for calculated them. Among such descriptors, some are related to the chemical graph while others are linked to a 3D representation. Consequently, for a given set of chemicals, we have to handle matrices with rows as the molecules IDs, and columns as the descriptors values.

In computer-aided drug design, considering the huge variety (several thousands) of molecular descriptors that are presently available for describing physico-chemical properties, the problem is how to achieve an optimal selection of appropriate sets in order to analyze, with highest accuracy, chemical diversity/similarity among large chemical libraries (several million of objects). Such "optimally selected descriptors" should provide the most efficient rules which can be used next to describe molecules with a common behavior. Such knowledge extraction phase would help to extract compounds presenting all the same required action. The difficulty is therefore to achieve properly the reduction in size of the descriptor matrix (from several thousands to several hundreds) without losing important information.

The usual way for that is to look for possible correlations between the descriptors to avoid redundancies. But it has been proved that such basic statistical analysis is not really efficient –

new approaches are desired. Consequently, there is a need to design a method that can help us to understand the data better. This is the subject of Knowledge Discovery in Database (KDD), where computational theories and tools can help us in discovering knowledge from large datasets [32], in this case the molecular descriptors.

The present chapter is concerned with the task of dimensionality reduction, or feature selection, on a difficult problem for medicinal chemistry. Dimensionality reduction of the data space refers to the process of converting a set of data having large dimensions into data with lesser dimensions ensuring that it conveys similar information concisely. These techniques are typically used while solving machine learning problems to obtain better features for a classification or regression task and concern many research fields of life sciences (see for example very recent papers assessing the strong interest for this field) [24, 68, 82, 102, 107]. Such reduction techniques, also related to feature selection methods, are of crucial importance in several aspects of drug design [54, 71, 84, 91, 98, 103], especially in drug/non-drug classification [61] adverse drug effects [70] and QSAR.

6.2 Feature selection

For discovering new antibiotics, we inspect a number of chemical molecules, to know which ones can characterize an antibiotic. To do that, it is necessary to look at the properties of each molecules, for example the number of atoms, the presence of certain atoms, the presence of rings, polarity, eccentricity, etc. All those properties sum up to thousands of attribute which means that we need a significant amount of time and space.

In order to minimize the required time and space, before we mine the data, we should reduce the dimension of the data set, as a *transformation* step in Fig. 1. It is performed by combining some attributes into one, or by completely eliminating some of them.

Within the thousands attributes for each molecule, we can identify some redundancies. Percentage of H atoms correlates with number of H atoms and number of atoms, number of C atoms correlates with number of CR3X structure, and many more hidden correlations. Therefore, we can ignore an attribute if they can be represented with another attributes. The selected attributes should be able to define the molecules without significant loss of information.

Beside the redundancies, we should also ignore the attributes which is not descriptive. For example, if we have an attribute describing the number of phospanes of a molecule, but all molecules have no phospanes, this attribute clearly can't distinguish between antibacterial and non-antibacterial molecules.

Many algorithms have been proposed to do this feature selection task. As illustrated in Fig. 6.1, they can be classified into two categories [55]:

1. *Filter model*, where feature selection is executed independently before the data mining process. This model doesn't take into account the classification algorithm that will be used. One common method is χ^2, who assigns a score to each feature based on the dependencies between them and the class. The higher scores belong the more important features. We then do a feature selection by removing the features whose score is less than a threshold.

2. *Wrapper model*, means that feature selection process exists as a wrapper around the classification algorithm. This algorithm will become a part of an evaluation function. An example of this model is studied by Li and Yang [69]. They examined two wrapper models, recursive and non-recursive. At each iteration in recursive approach, a classifier is trained, and feature weights are obtained. The feature with the smallest weight is then removed.

(a) (b)

Figure 6.1: Two approaches of feature selection: filter model (a) and wrapper model (b). [55]

The iteration stops when the desired number of features t is attained. On the other hand, in non-recursive approach, after all feature weights are calculated, t features are selected, and the rests are immediately discarded. They examined both approaches using three classifiers: SVM, ridge regression, and Rocchio. Eventually, it is shown that ridge regression was the best choice, since it penalizes many redundant features effectively.

6.3 Log-linear analysis

This section describes log-linear analysis (LLA) and its graphical representation. LLA can find any associations among attributes, allowing feature selection according to those associations.

Suppose that we have a data set WAR of certain molecules with three variables: molecular weight (W), number of atoms (A) and ring perimeter (R). Relationship between two variables can be studied with two-way χ^2 test of association. However, if we have more than two variables, we need to do a multiway frequency analysis to study the two- and three-way associations. LLA is an extension of multiway frequency analysis, and tries to discover any statistical relationships between three or more non-continuous variables. It will create a model (like the one in Eq. 6.4) to find the log of expected frequencies.

For the WAR data set with contingency table in Table 6.1, LLA tries to answer some relationship questions. Is a molecule's weight related to its number of atoms? Is a molecule's number of atoms related to its ring perimeter? Is there a three-way relationship among molecular weight, number of atoms, and ring perimeter? By knowing ring perimeter of a molecule, can we predict its weight?

To do a multiway frequency analysis with LLA, we develop a linear model of the logarithm of expected cell frequencies. An example of such model is shown in Eq. 6.4, with each term representing an association. As the number of variables increases, the number of associations also increases. With three variables in data set WAR, we have seven possible associations: one three-way associations, three two-way associations, and three one-way associations. The model in Eq. 6.4 contains all possible associations. To keep the simplicity of a model, LLA tries to find which association will be kept or removed. To do that, we should determine the significance of an association by examining the goodness-of-fit of the model containing it.

Table 6.1: An example of contingency table of a dataset about molecular weight (W), number of atoms (A) and ring perimeter (R) [88].

Molecular weight	Number of atoms	Ring perimeter		Total
		Narrow	Wide	
Light	Small	15	15	30
	Large	10	15	25
	Total	25	30	55
Medium	Small	10	30	40
	Large	5	10	15
	Total	15	40	55
Heavy	Small	5	5	10
	Large	10	25	35
	Total	15	30	45

Eventually, with thousands of variables, the number of possible associations will be so large that it would be impractical to test each association. This limitation can be solved by Chordalysis.

6.3.1 Goodness-of-fit

The goodness-of-fit of LLA measures the conformity between the expected and observed frequencies. With only two variables, there is one two-way association. To test the goodness-of-fit of a two-way association, we can use χ^2 test:

$$\sum_i \frac{(O_i - E_i)^2}{E_i}, \tag{6.1}$$

where for each cell i of contingency table, O_i is its observed frequency and E_i is its expected frequency. The summation is done over all cells.

However, since there are multiway associations in LLA, we use G^2 test. This test can replace χ^2 test in measuring the conformity of observed and expected frequencies. G^2 statistic is distributed as χ^2, so we can use χ^2 table to evaluate its significance. This statistic has the equation:

$$2\sum_i O_i \ln\left(\frac{O_i}{E_i}\right). \tag{6.2}$$

Moreover, G^2 statistic is used because of its additivity property, which is not present in χ^2 statistic. Therefore, in a two-way analysis of variable A and B, test of overall association G^2_T is the sum of first-order tests, G^2_A and G^2_B, and association test G^2_{AB}:

$$G^2_T = G^2_A + G^2_B + G^2_{AB}. \tag{6.3}$$

This property is useful to test the residual frequency of a model, which will be shown in Section 6.3.2. For the computation detail of G^2 please refer to [88].

6.3.2 Log-linear model

Log-linear model is represented as an equation to find a logarithm of E (expected frequency of a combination of variables' values), e.g. the expected value of the cell *light-small-wide* in Table 6.1.

From data set WAR, we can generate a model that contains all possible associations. This is the saturated model, which can be written as:

$$\ln E_{war} = \theta + \lambda_W(w) + \lambda_A(a) + \lambda_R(r) + \lambda_{WA}(wa) + \lambda_{WR}(wr) + \lambda_{AR}(ar) + \lambda_{WAR}(war) \quad (6.4)$$

where θ is a constant, and λ term represents an effect. Each λ has values as many as the number of levels, and these values sum to zero. For example, since we define three levels of molecular weight: *light*, *medium*, and *heavy*, $\lambda_W(w)$ has three possible non-zero values: for *light* ($\lambda_W(\text{lig})$), *medium* ($\lambda_W(\text{med})$), and *heavy* ($\lambda_W(\text{hvy})$), with $\lambda_W(\text{lig}) + \lambda_W(\text{med}) + \lambda_W(\text{hvy}) = 0$.

A log-linear model can be hierarchical or non-hierarchical. A hierarchical model can be represented by its highest-order association in square brackets. For example, a model $[WA][R]$ contains $\lambda_{WA}(wa)$ and $\lambda_R(r)$, as well as $\lambda_W(w)$ and $\lambda_A(a)$.

The score of a model is obtained by calculating its G^2 and examining its significance. Based on Eq. 6.3, G^2 of a model is obtained by subtracting the G^2 of each association from total G^2, therefore revealing its residual frequency.

For example, let's examine the model $[WA][R]$. This model has the equation:

$$\ln E_{war} = \theta + \lambda_{W_w} + \lambda_{A_a} + \lambda_{R_r} + \lambda_{WA_{wa}}.$$

This model has G^2:

$$\begin{aligned} G^2_{[WA][R]} &= G^2_T - G^2_{WA} - G^2_A - G^2_W - G^2_R \\ &= 48.09 - 27.12 - 0.16 - 1.32 - 13.25 \\ &= 6.24 \end{aligned} \quad (6.5)$$

and $df = 11 - 2 - 1 - 2 - 1 = 5$. This residual is not statistically significant (>0.05 in χ^2 table), so the model is good.

We then evaluate a more complex hierarchical model, $[WA][WR]$ for example. Similar with the example in Eq. 6.5, we have $G^2_{[WA][WR]} = 2.48$ and its $df = 3$. This model is also good. Therefore, we must choose between $[WA][R]$ and $[WA][WR]$.

Notice that both models are hierarchical, and $[PS][R]$ is a submodel of $[WA][WR]$, i.e. we can find all λ terms of $[WA][R]$ in $[WA][WR]$. In this condition, the difference between their G^2s is itself a G^2:

$$\begin{aligned} G^2_{\text{diff}} &= G^2_{[\text{PS}][\text{R}]} - G^2_{[\text{PS}][\text{PR}]} \\ &= 6.24 - 2.48 \\ &= 3.76 \end{aligned}$$

with $df = 5 - 3 = 2$. This difference is not statistically significant, so we can select a less complex model, $[WA][R]$.

6.3.3 Graphical model

Consider the graph shown in Fig. 6.2, a graphical representation of a log-linear model. Its vertices are connected by an edge to each other because there are $\lambda_W(w)$, $\lambda_A(a)$, $\lambda_R(r)$, $\lambda_{WA}(wa)$, $\lambda_{WR}(wr)$, and $\lambda_{AR}(ar)$ in the model. Furthermore, because the graph between the three variables is complete, we can conclude that there is a three-way association between them, $\lambda_{WAR}(war)$. Therefore, the model is a saturated one, as shown in Eq. 6.4.

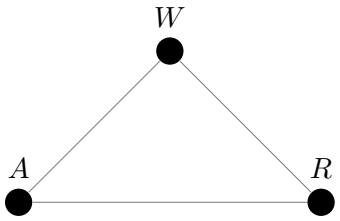


Figure 6.2: Graphical representation of log-linear model in Eq. 6.4.

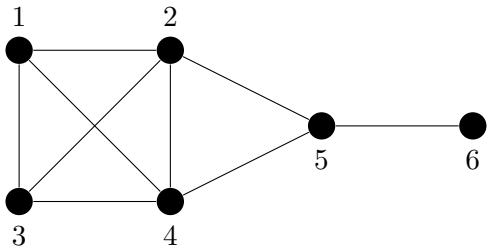


Figure 6.3: A graph with 6 vertices. Its maximal cliques are $\{1,2,3,4\}$, $\{2,4,5\}$, and $\{5,6\}$, with $\{1,2,3,4\}$ as the maximum clique. Its minimal separators are $\{2,4\}$ and $\{5\}$, with $\{5\}$ as the minimum separator. The minimal $(2,6)$-separator is $\{5\}$.

Meanwhile, if we exclude $\lambda_{WAR}(war)$:

$$\ln E_{war} = \theta + \lambda_W(w) + \lambda_A(a) + \lambda_R(r) + \lambda_{WA}(wa) + \lambda_{WR}(wr) + \lambda_{AR}(ar) \tag{6.6}$$

we can't use the graph in Fig. 6.2 as its representation. The graph is complete, so it should have $\lambda_{WAR}(war)$. But the model in Eq. 6.6 doesn't have it. Actually, we can't draw a graph with $\lambda_{WA}(wa)$, $\lambda_{WR}(wr)$, and $\lambda_{AR}(ar)$ without including $\lambda_{WAR}(war)$. Consequently, the model in Eq. 6.6 is not graphical.

Therefore, a log-linear model is *graphical* if, whenever a model contains all two-way associations generated from a higher-order association, the model also contains the higher-order association. The model shown in Fig. 6.2 contains all two-way associations from $\lambda_{WAR}(war)$. Thus, it should also contains $\lambda_{WAR}(war)$.

Associated with each graph, we have *maximal cliques* and *minimal separators*, as illustrated in Fig. 6.3. A clique of graph G is a subset of vertices of G whose induced subgraph is complete, i.e. every two distinct vertices are connected by an edge. A maximal clique is a clique that is not a subset of another clique. A maximum clique is a clique with the largest number of vertices.

A separator of graph G is a subset of vertices of G whose removal will make G disconnected. A minimal separator is a separator which doesn't have another separator in its subsets. A separator with the smallest number of vertices is a minimum separator.

A minimal separator is also a property of a pair of vertices. A minimal (a,b)-separator is a subset of vertices whose removal make a disconnected from b, and such subset does not have another separator in its subsets.

6.3.4 Decomposable models

A graphical log-linear model is decomposable if its graph is decomposable. A decomposition of a graph D is a partition of its vertices V into three disjoint subsets (A,B,S), where:

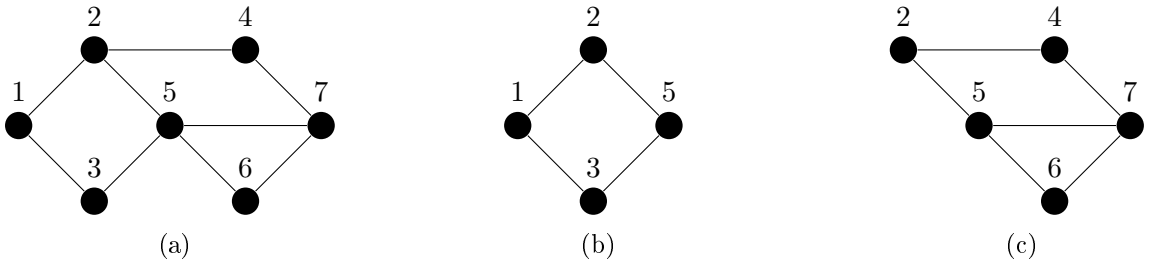


Figure 6.4: Decomposition of a graph D (a), into its two components (b) and (c).

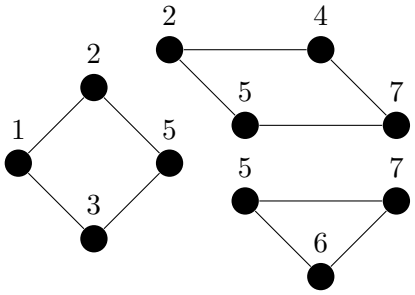


Figure 6.5: Maximal prime subgraphs of graph D in Fig. 6.4a.

- $A \neq \emptyset$ and $B \neq \emptyset$,
- S forms a complete subgraph,
- A and B are not connected in $D - S$.

The results of this decomposition are called *components* of D. They are induced subgraphs from $A \cup S$ and $B \cup S$. An example of a decomposition[6] is illustrated in Fig. 6.4. Vertices in graph D in Fig. 6.4a can be partitioned into three subsets: $A = \{1, 3\}$, $B = \{4, 6, 7\}$, and $S = \{2, 5\}$. This produces the components $A \cup S$ in Fig. 6.4b and $B \cup S$ in Figure Fig. 6.4c.

Any graph can be recursively decomposed into its maximal *prime* subgraphs, i.e. a subgraph with no decomposition. The graph D in Fig. 6.4a, for example, can be decomposed into three subgraphs shown in Fig. 6.5.

A graph is decomposable if it is complete or if it can be decomposed into another decomposable subgraphs. From this recursive definition, this means that all maximal prime subgraphs are cliques. Consequently, the graph in Fig. 6.4a is not decomposable, since two of its components (in Fig. 6.5) aren't decomposable.

Moreover, a graph is decomposable if and only if it is chordal. In a chordal graph, every cycle with length more than three has a chord, i.e. an edge that is not part of the cycle but connects two vertices of the cycle. The graph in Fig. 6.6a is chordal, but the graph in Fig. 6.6b is not, because its (A, B, D, E) cycle does not have any chord.

Decomposable models are the only log-linear models which have a closed-form maximum likelihood estimates [38]. Furthermore, a decomposable model has an advantage concerning its minimal separators and maximal cliques. They can be generated in linear time in a single pass, using Lexicographic Breadth First Search or using Maximum Cardinality Search [12].

[6] taken from http://www.stats.ox.ac.uk/~steffen/teaching/cimpa/decomp.pdf

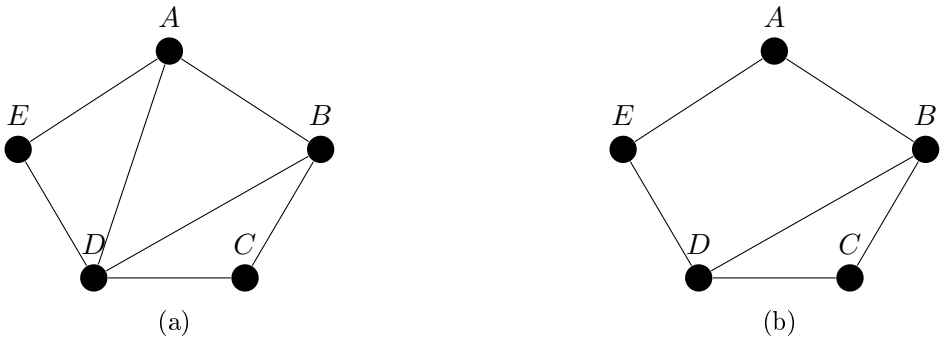


Figure 6.6: Examples of chordal (a) and non-chordal graph (b).

6.4 Chordalysis

There are two ways to select which associations to include in a log-linear model: *backward elimination* and *forward selection*. Backward elimination starts from a saturated model and eliminates non-significant associations one by one. On the other hand, forward selection starts from an empty model and iteratively adds an association until the difference is not significant. The existing LLA considers all possible associations to determine which one to be added or removed. This becomes infeasible when the number of attributes increases, since the number of associations is exponential w.r.t. it.

Chordalysis tries to guide the existing LLA in selecting which associations are significant enough to be included in the model [75, 77]. This method is focusing on decomposable log-linear models, whose G^2 can be calculated by inspecting the maximal cliques and minimal separators of the corresponding graph.

As a forward approach, Chordalysis starts with an empty graph as initial model. It has neither vertex nor edges. Then at the first step of each iteration, the candidate models are generated. Each candidate M^c differs from the current best model M^* by a single edge only. This edge addition must keep the graph chordal. To find such edges, we look at the connectivity between its two vertices. An edge (a, b) is eligible if:

- a and b are not connected (they belong to different connected components), or
- a and b are connected with all of its chordless paths are of length two.

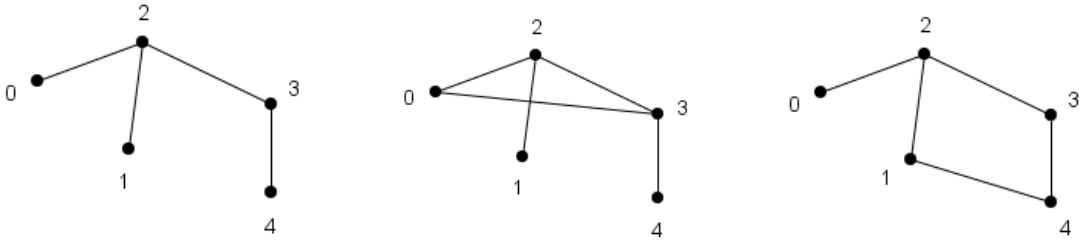
Since an edge depicts an association between two variables, an addition of a single edge corresponds to an inclusion of a two-way (and possibly a higher order) association in the log-linear model. An example is illustrated in Fig. 6.7. Suppose that at a given iteration we have a best model [02][12][23][34] (Fig. 6.7a) with equation:

$$\ln E = \theta + \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_{0,2} + \lambda_{1,2} + \lambda_{2,3} + \lambda_{3,4} \tag{6.7}$$

An eligible candidate must be decomposable. Therefore, instead of generating all models which have a new edge, it only produces a fewer number of them which are chordal, e.g. [023][12][34] in Fig. 6.7b with equation:

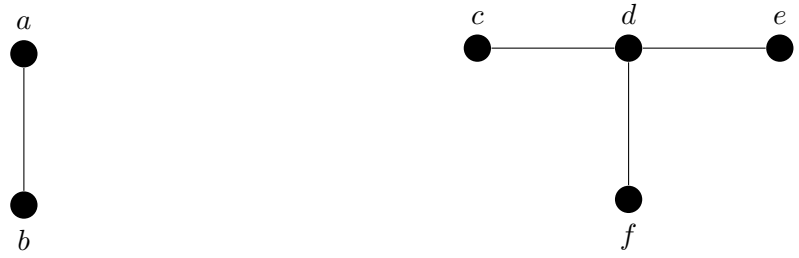
$$\ln E = \theta + \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_{0,2} + \lambda_{0,3} + \lambda_{1,2} + \lambda_{2,3} + \lambda_{3,4} + \lambda_{0,2,3} \tag{6.8}$$

The models which aren't hierarchical are not selected, because their corresponding graph is not chordal (Fig. 6.7c). This greatly reduces the search space of selecting a new model. Based on the



(a) (b) (c)

Figure 6.7: An example of current best model (a). At each iteration, Chordalysis generates all possible candidates which differ only in one edge and are chordal. (b) is an available one, while (c) is not, because the addition of edge (1,4) won't keep the graph chordal.



(a) (b)

Figure 6.8: Examples of connected components with 2 vertices (a) and more than 2 vertices (b).

fact that the graph is chordal and differs by only one edge between iterations, the G^2 computation is scalable to thousand variables [76]. After the score calculations, Chordalysis selects the best M^c. Its score is then compared to the significance threshold. If it is lower, then we replace M^* with M^c and continue to the next iteration. If not, then the current M^* is the final model because the G^2 difference by adding an association is not significant.

The significance threshold α is updated at each iteration so it doesn't accept a candidate too often. This update rule follows the *layered critical values* [97]. At iteration i, where the current best model M^* has L edges, the significance threshold α_i is $\alpha/(2^L.S_i)$, where S_i is the search space, i.e. the number of chordal graphs that can be formed by adding a single edge to M^*, and α is a p-value threshold (usually set to 0.05).

Having a graph representing associations among attributes, we then perform feature selection based on this graph. The attributes which are independent (have no association) are kept, because they can't be represented by another attribute. Then, basically we remove the attributes that have only one association. For example, if we encounter a connected component shown in Fig. 6.8b we keep only attribute d. This means that d can represent c, e, and f. But there is an exception for some of those one-association attributes. If a connected component has only 2 vertices, like the one in Fig. 6.8a, we randomly choose one attribute and discard the other.

6.5 Classification of antibacterials and non-antibacterials

The attributes that are selected using Chordalysis will be used in three machine learning method to classify dataset of antibacterials and non-antibacterials.

The first method is Support Vector Machine (SVM) [22]. Given a dataset of labelled points,

SVM builds a hyperplane that best separates the two labels. To deal with non-linearly separable dataset, SVM use a kernel to map points to higher dimension. The second method is random forest [13]. It constructs a family of decision trees which have different training set between each other. To classify data, each tree gives a classification (or "vote"), and random forest takes the majority vote. The third method is naive Bayes that is based on Bayes' theorem. When classifying a data D, naive Bayes calculates the posterior probability of each class given D. The classifier then chooses the class that has higher posterior probability.

6.6 Evaluation of classifier performance

To measure the goodness of each classifier, we calculate the following classical measures:

- TP (true positive), the amount of antibacterial data that are classified as antibacterial,
- TN (true negative), the amount of non-antibacterial data that are classified as non-antibacterial,
- FP (false positive), the amount of non-antibacterial data that are classified as antibacterial,
- FN (false negative), the amount of antibacterial data that are classified as non-antibacterial,

and we use five metrics:

1. Accuracy (percentage of data that are correctly classified):
$$\frac{TP+FN}{TP+TN+FP+FN}.$$
2. Precision (percentage of antibacterial-classified data that are truly antibacterial):
$$\frac{TP}{TP+FP}.$$
3. Recall (percentage of truly antibacterial data that are classified as antibacterial):
$$\frac{TP}{TP+FN}.$$
4. AUC (Area Under the ROC Curve) [14]. The ROC (Receiving Operating Characteristic) curve is a graphical plot of recall and 1-CNF. CNF (Correct Negative Fraction) is defined by:
$$\frac{TN}{TN+FP}.$$
This curve can be obtained by varying discrimination threshold.
5. $f1_{score}$ (harmonic mean of precision and recall), with formula:
$$\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}}.$$

Table 6.2: Attribute filters from previous work.

Filter	Standard deviation	Pair correlation	# of selected attributes
1	> 0.010	< 0.4	87
2	> 0.010	< 0.8	576
3	> 0.001	< 0.8	588
4	> 0.100	< 0.8	524

6.7 Experiments

We focus on the task of reducing dimensionality, by selecting features among thousands of features. Chordalysis will find some associations among variables, and from that we will remove all variables that can be represented by another.

6.7.1 Previous work

The aim of the previous work [49] was to evaluate the performance of six machine learning techniques for distinguishing between antibacterial and non-antibacterial molecules. Those six classification techniques are: Support Vector Machine (SVM) with linear kernel, random forest, logistic regression, gradient boosted trees, naive Bayes, and decision trees.

The dataset of antibacterial and non-antibacterial molecules were collected from antibacterials already in the market (152 molecules), as well as molecules from the MDDR database (2873 molecules with reported antibacterial properties and 52604 other drugs with no known antibacterial activity) and the Life Chemicals Inc. catalog (38907 molecules labeled as potential antibacterials, 52604 from other categories). The Dragon software [93] was then used to define 4885 attributes for each molecule. The values of those attributes were calculated using Corina [86].

Selection procedures were initially performed to reduce the initial set of attributes. Those with missing values were removed. Furthermore, if there were a group of attributes which were perfectly correlated, only one of them was kept. These two removal procedures resulted in 4532 attributes.

From this reduced data set, four filters were used independently to obtain a fewer number of attributes, considering their standard deviations and pair correlations. The parameters and results of these filters (called Filter 1, Filter 2, Filter 3, and Filter 4) are shown in Table 6.2. Finally, we ended with 5 data sets: one non-filtered data set with 4532 attributes and four filtered data sets.

This previous work shows us that SVM has the highest precision, but its recall is low. The recall of random forest is much better than SVM, so it can be used in sacrificing the high precision of SVM. In the next sections, we detail our work of applying Chordalysis to perform feature selection among hundreds of molecular descriptors. This includes which data set is used, which preprocessing is needed, and the results from our experiment.

6.7.2 Dataset

Our work is focused on 3025 antibacterial molecules, which is composed by 152 molecules from market and 2873 molecules from MDDR antibacterials. From those molecules, we defined 4885 attributes. Besides removing missing-valued attributes and perfectly correlated attributes, we

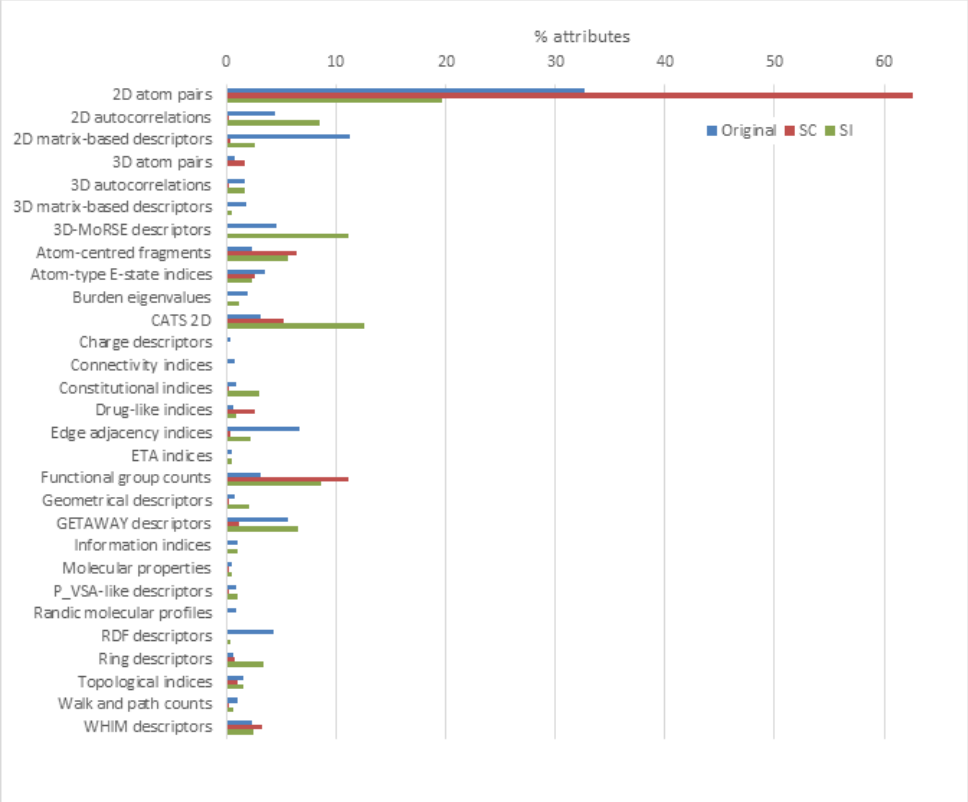


Figure 6.9: The distribution of families in 4885 original attributes, 595 attributes of SC, and around 500 attributes of S2–4.

also removed attributes that have the same value for all molecules. This resulted in 3769 attributes.

Some of the attributes are continuous. Because LLA and Chordalysis work on discrete variables, these numerical attributes are preprocessed so that all of them become discrete. This preprocessing step is applied to attributes which have more than 10 distinct values. Equal-width discretization method is used, with 10 bins as desired output.

6.7.3 Result of Chordalysis

Attribute selection

Chordalysis (available open-source at http://sourceforge.net/p/chordalysis/) was tested on 3769 attributes, using $\alpha = 0.05$ as p-value threshold, and it found 1024 associations. The selection procedure explained in Sect. 6.4 results in 3171 selected attributes. From each filter of the previous work, we have four sets of selected attributes. As seen in Table 6.2, Filters 2, 3, and 4 gave us around 500 attributes each; those will be next referred to as S2, S3, and S4 respectively. In order to compare Chordalysis-based feature selection, we let Chordalysis find more associations without being limited by p-value threshold. After some experiments, we found that after 3613 associations were found, the selection procedure applied on the graph resulted in a set of 595 selected attributes. We call this set SC. Since S2, S3 and S4 are very similar, all three sets are referred to as S2–4.

All attributes are uniquely categorized within the Dragon software into 29 attribute families.

Table 6.3: Intersections of the sets of attributes from three filters and from Chordalysis.

Set	S2	S3	S4
SC (595 attr.)	153	154	146
S2 (576 attr.)		569	497
S3 (588 attr.)			496
S4 (524 attr.)			

Table 6.4: Population of most relevant attribute families. Last column is the number of attributes present in all sets (SC, S2, S3 and S4). Percentages correspond to the number of retained attributes from the original set.

Attribute family	Original set	SC	Overlap
2D atom pairs	1596	373 (23%)	80
2D matrix-based	550	2	1
Edge adjacency ind.	324	2	1
GATEWAY	273	7	1
Functional group cnt.	154	66 (43%)	24
CATS 2D	150	31 (21%)	19
Atom-centered frag.	115	38 (33%)	10
Drug-like indices	27	15 (56%)	1
TOTAL	4885	595 (12%)	144 (3%)

Fig. 6.9 and 6.10 show how those families are distributed within the different sets, with weights in percentages for S2–4 being defined as the average value between S2, S3 and S4. Table 6.4 summarizes the results of Chordalysis-based selection, and also highlights the overlap between SC and S2–4.

2D atom pairs are by far the most prevalent family in all cases, accounting to 33, 63 and 20% of the total attributes population in the original, SC and S2–4 sets respectively. It is very clear that Chordalysis privileged this family, now twice as present in SC compared to the original set, while on the contrary, it is significantly less important in S2–4.

The atom-centred fragments, CATS 2D and functional group counts families were not prevalent in the original set (2, 3, and 3% respectively), and become over-selected in all reduced sets, with CATS 2D being more prevalent in S2–4 compared to SC. These 3 sets account for 23% of SC, so with the addition of 2D atom pairs only 15% is remaining.

The relative diversity of the different sets can be compared by focusing on the number of families accounting for more than 5% of the total number of attributes and on their total weight (see Figure 8). Only the 4 aforementioned families obey these criteria in SC, totaling 85% of the set. There are 7 families with more in S2–4 (73%); 4 in the original set (56%). Therefore, three attribute families are seen significantly more prevalent in S2–4 compared to the original set (26% / 15%), but are almost completely filtered out by Chordalysis: 2D autocorrelations, 3D-MoRSE and GETAWAY descriptors.

Apart from these families, there is a consensus between S2–4 and SC regarding the removal of some attribute families from the original set, with SC appearing more stringent than S2–4. Indeed, the 2D matrix-based descriptors, edge adjacency indices and RDF descriptors account together for 22, <1 and 5% in the original, SC and S2–4 sets respectively. This suggests that those descriptors are not of much value for modeling the probability that a given chemical compound

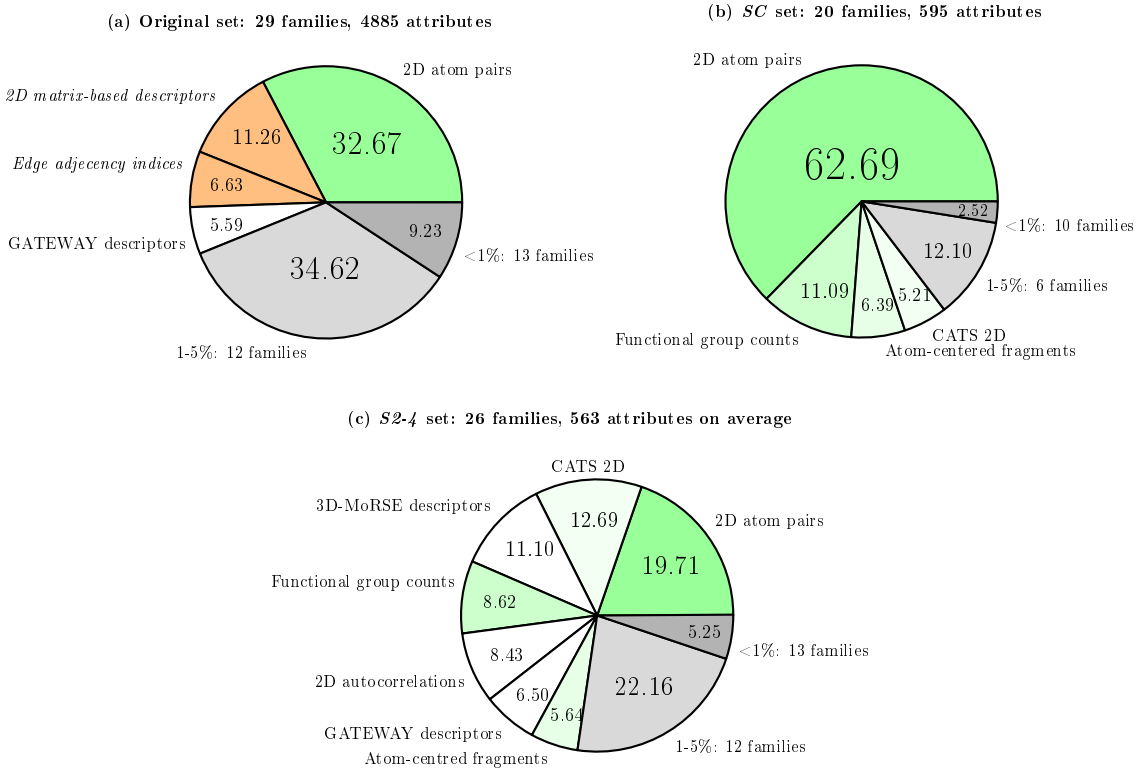


Figure 6.10: Distribution of major attribute families in (a) the original set, (b) set SC, and (c) average of S2, S3, and S4. All values in percentages. Only sets weighting >5% are represented. Categories in green are the 4 most-represented in SC. Those in orange with description in italic are mostly filtered out of both SC and S2–4. Grey ones summarize remaining families (1–5% and <1%.

Table 6.5: Means and standard deviations of classifiers' metrics on the test set. All data in percentages.

Metric	SVM ($C = 0.01$)	Random Forest	Naive Bayes
Accuracy	97.0 ± 2.5	96.6 ± 1.5	55.7 ± 1.4
Recall	98.9 ± 2.0	95.6 ± 1.5	21.2 ± 0.6
Precision	95.9 ± 3.5	98.3 ± 2.2	93.7 ± 9.1
AUC	99.3 ± 1.1	99.5 ± 0.4	65.3 ± 4.5
$f1_{score}$	97.3 ± 2.2	96.9 ± 1.3	34.5 ± 1.0

Table 6.6: Precision and recall of random forest classifier on the test set. All values in percentages.

α	4532 attributes		SC	
	Precision	Recall	Precision	Recall
10	10.0	71.2	11.6	65.4
20	10.7	68.4	11.8	64.5
30	11.8	65.4	12.2	63.5
40	11.8	65.4	12.2	63.5
50	12.3	64.3	12.3	63.0
60	13.3	61.5	12.6	62.4
70	14.1	60.1	12.8	61.8
80	14.1	60.1	12.8	61.8
90	15.1	59.1	13.0	61.2
100	16.6	56.3	13.3	60.1

would possess antibacterial properties.

Eventually, it should be noticed that while SC size compared to the original set is 12%, only a single attribute family has been filtered less than 50%: drug-like indices. This is specific to SC (5 attributes are retained by S2, S3 and S4). There were only 27 residues of this kind in the original set, which is not significant enough to determine that there could be a correlation between drug-likeness and antibacterial potency that would be most efficiently selected by Chordalysis.

Table 6.3 lists the number of overlapping selected attributes from S2, S3 and S4 with SC. Table 6.4 summarizes SC-related data for most-relevant attribute families identified above, and highlights the number of consensus attributes i.e. attributes found in all sets.

Training/testing strategy

Using attributes in SC, we trained SVM, random forest, and naive Bayes on data from market antibacterials, MDDR antibacterials, Life Chemicals non-antibacterials, and MDDR non-antibacterials. After the training process, the three classifiers are tested on Life Chemicals list of predicted antibacterials. The results are summarized in Table 6.5. By tuning the parameter C for SVM, we get the best result for $C = 0.01$.

Based on the values of all metrics used to evaluate classifier performance, it appears that SVM and RF perform significantly better than naive Bayes. The two best performing model –SVM and random forest– were then trained on the market antibacterials and Life Chemical antibacterials to test the MDDR data. We regarded a molecule as an antibacterial if it is classified as such in at least α percent of runs. The precision and recall of the two models are shown in Table 6.6–6.7. SVM has better recall on the majority of alphas, and it has higher precision for $\alpha \geq 20$.

Table 6.7: Precision and recall of SVM classifier on the test set. All values in percentages.

α	4532 attributes		SC	
	Precision	Recall	Precision	Recall
10	21.3	16.4	11.1	82.7
20	24.0	15.2	13.3	78.1
30	25.9	12.8	17.0	71.9
40	25.9	12.8	17.0	71.9
50	26.6	12.5	19.0	69.9
60	27.4	11.8	23.8	65.1
70	27.9	11.4	27.2	62.4
80	27.9	11.4	27.2	62.4
90	27.7	10.0	31.8	58.7
100	32.4	9.2	39.9	53.4

6.8 Conclusions

In this chapter, we describe the application of the Chordalysis technique for molecular attribute set reduction. We show that it leads to improved performance when a machine learning technique is used next to discriminate between antibacterials and compounds with no antibacterial activity. It is suggested that a two-step strategy, with a Chordalysis-refined attribute set being fed to a SVM classifier could be highly efficient for antibacterials identification. An alternate techniques for selecting an optimal attribute set [92], such as recursive feature elimination [37, 99], RF variable importance [16], SVM variable selection [17], tabu search [89, 106], and evolutionary algorithms [34] should be further studied. In the process, precise clues on implementing new attributes that could be more efficient for our purpose (antibacterials selection) than the broad generic reference set of molecular attributes that is available in the Dragon software could be obtained. When we reach a state where no clear methodological improvement could be reached, we will apply the optimized methodology to mine chemical space for possible new antibacterials. A limited number of hits from this virtual screening process will be tested experimentally. Only interesting results backed up by the resulting experimental data will validate the ongoing chemoinformatics approach.

Summary and perspectives

Summary

Knowledge discovery in database (KDD) is still actively studied given that available datasets are more and more complex. In this thesis, we had the opportunity to work on several types of complex data, namely numerical, sequential, and molecular datasets. We separate our contributions into two parts: (1) FCA-based biclustering and (2) mining complex data.

In the task of biclustering, we explore the potential of enumerating biclusters in numerical datasets using FCA. One task of FCA is enumerating formal concepts, which is a set of objects and attributes having regularity among them. FCA is generalized into pattern structures, from which biclustering in numerical matrix is possible. Two previous works of FCA-based biclustering are the starting point for this part of the thesis. Partition pattern structures (PPS) and interval pattern structures (IPS) were proposed to mine constant and similar column biclusters. In this thesis we generalize these approaches to mine other bicluster types. First, we adapt the original PPS by modifying the similarity operator between two partitions, such that other bicluster types: additive, multiplicative, order-preserving, and coherent-sign-changes can also be found in a partition pattern concept. Second, we generalize IPS to perform non-perfect additive and multiplicative biclusterings, using column alignments. We show that PPS and IPS can be adapted to enumerate various types of biclusters, and by providing a parameter of expected bicluster size, we can perform this task in a reasonable runtime.

The second part of this thesis has two major axis. The first is related to CrossCult project, a European Project whose general idea is to support the emergence of a European cultural heritage. One aspect of this project is to build a mobile application that registers, among others, the trajectory of each visitor while they interact with it, providing a kind of sequential dataset. A particular dataset is concerned with the trajectories of around 200 visitors in Hecht Museum in Haifa, Israel. Each trajectory is composed by sequence of visited items. We develop a method to understand these trajectories such that the visitor behaviors can be extracted according to the four theoretical visiting patterns. This approach is composed of two independent steps: the clustering and the mining of trajectories. Given that the trajectory dataset can be regarded as a sequential dataset, a proper sequence clustering method is used where the distance between any two sequences is obtained from the number of their common subsequences. On the other hand, the mining of trajectory patterns is performed by two methods based on FCA. These patterns are then used to find the characteristic behavior of each cluster. Our result highlights some interesting patterns that may define visitor behaviors. This can help museum researchers to analyze and evaluate the placement of items and the visiting styles. Moreover, this analysis is useful to build a recommendation system for future visitors.

The second is related to mining associations among attributes in a numerical molecular dataset. These associations are used to perform attribute selection, which can reduce the complexity of the dataset. Such attribute associations can be found using log-linear analysis, and fur-

ther by Chordalysis to deal with thousand attributes. We examined the application of Chordalysis in attribute selection by analyzing the resulting association graph. Since the graph informs us the correlation among attributes, we are able to determine which attributes can be represented by another, so they can be discarded, leaving only attributes which are roughly not correlated.

Perspectives

In Chapter 3 and 4, we present an approach of biclustering using partition pattern structures and interval pattern structures respectively. Given that pattern structures is a generalization of FCA, there exist various algorithms capable of enumerating formal concepts that can be adapted to enumerate pattern concepts, from which a set of biclusters can be discovered. Here we adapt CbO and AddIntent to perform pattern concept mining, given their flexibility of generalizing similarity operator and subsumption relation needed in pattern structures. Another previous work about FCA-based biclustering [94] uses the adaptation of In-Close to mine formal concepts, and it is interesting to adapt this algorithm to mine pattern concepts, and further to mine biclusters.

In Chapter 5, we analyze a dataset of visitor trajectories, such that we can extract some patterns and detect visitor behaviors. In doing that, we included only partial information about the museum in the sequences, which are the visited items. More interesting results are expected if other elements are taken into account, such as more general knowledge about history and geography, as well as the duration and time of the visit. Finally, this analysis can be incorporated to build recommendation systems, by including ongoing information such as comments and state of the visitors during their visit.

Lastly, in Chapter 6 we provide a thorough examination of feature selection using log-linear analysis. We explored the association graph generated from Chordalysis, which can be regarded as attribute correlation graph. Such graph can be mined with other methods, for example graph triangulation and tree decomposition. From a triangulated graph, we can obtain maximal cliques which groups a set of highly correlated attributes. Furthermore, with a tree decomposition on these cliques, we can decide the representatives of each clique, hence discarding other non-representative attributes.

Bibliography

[1] Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)

[2] Ailem, M., Role, F., Nadif, M.: Graph modularity maximization as an effective method for co-clustering text data. Knowledge-Based Systems **109**, 160–173 (2016)

[3] Alam, M., Buzmakov, A., Napoli, A.: Exploratory knowledge discovery over web of data. Discrete Applied Mathematics **249**, 2–17 (2018)

[4] Alam, M., Le, T.N.N., Napoli, A.: Latviz: A new practical tool for performing interactive exploration over concept lattices (2016)

[5] Alizadeh, A.A., Eisen, M.B., Davis, R.E., Ma, C., Lossos, I.S., Rosenwald, A., Boldrick, J.C., Sabet, H., Tran, T., Yu, X., et al.: Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature **403**(6769), 503 (2000)

[6] Allab, K., Labiod, L., Nadif, M.: Multi-manifold matrix decomposition for data co-clustering. Pattern Recognition **64**, 386–398 (2017)

[7] Andrews, S.: In-Close, a fast algorithm for computing formal concepts. In: International Conference on Conceptual Structures (ICCS) (January 2009), `http://shura.shu.ac.uk/38/`

[8] Andrews, S.: In-Close2, a high performance formal concept miner. In: International Conference on Conceptual Structures. pp. 50–62. Springer (2011)

[9] Asses, Y., Buzmakov, A., Bourquard, T., Kuznetsov, S.O., Napoli, A.: A Hybrid Classification Approach based on FCA and Emerging Patterns - An application for the classification of biological inhibitors. In: Proceedings of CLA. CEUR Workshop Proceedings, vol. 972, pp. 211–222 (2012)

[10] Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 429–435. ACM (2002)

[11] 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)

[12] Berry, A., Pogorelcnik, R.: A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph. Information Processing Letters **111**(11), 508–511 (2011)

[13] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)

[14] Brown, C.D., Davis, H.T.: Receiver operating characteristics curves and related decision measures: A tutorial. Chemometrics and Intelligent Laboratory Systems **80**(1), 24–38 (2006)

[15] Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S.O., Napoli, A., Raïssi, C.: On mining complex sequential data by means of FCA and pattern structures. International Journal of General Systems **45**(2), 135–159 (2016)

[16] Cano, G., Garcia-Rodriguez, J., Garcia-Garcia, A., Perez-Sanchez, H., Benediktsson, J.A., Thapa, A., Barr, A.: Automatic selection of molecular descriptors using random forest: Application to drug discovery. Expert Systems with Applications **72**, 151–159 (2017)

[17] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) **2**(3), 27 (2011)

[18] Cheng, Y., Church, G.M.: Biclustering of expression data. In: ISMB. vol. 8, pp. 93–103 (2000)

[19] Codocedo, V., Bosc, G., Kaytoue, M., Boulicaut, J.F., Napoli, A.: A proposition for sequence mining using pattern structures. In: Bertet, K., Borchmann, D., Cellier, P., Ferré, S. (eds.) Proceedings of ICFCA. pp. 106–121. Springer (2017)

[20] 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)

[21] Codocedo-Henríquez, V.: Contributions à l'indexation et à la récupération d'information utilisant l'analyse formelle de concepts. Ph.D. thesis, Université de Lorraine (2015)

[22] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)

[23] Couceiro, M., Napoli, A.: Elements about exploratory, knowledge-based, hybrid, and explainable knowledge discovery. In: International Conference on Formal Concept Analysis. pp. 3–16. Springer (2019)

[24] Davoudi, A., Ghidary, S.S., Sadatnejad, K.: Dimensionality reduction based on distance preservation to local mean for symmetric positive definite matrices and its application in brain–computer interfaces. Journal of Neural Engineering **14**(3), 036019 (2017)

[25] Di-Jorio, L., Laurent, A., Teisseire, M.: Mining frequent gradual itemsets from large databases. In: International Symposium on Intelligent Data Analysis. pp. 297–308. Springer (2009)

[26] Ding, C., He, X., Simon, H.D.: On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of the 2005 SIAM International Conference on Data Mining. pp. 606–610. SIAM (2005)

[27] Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 126–135. ACM (2006)

[28] Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 43–52. ACM (1999)

[29] Egho, E., Raïssi, C., Calders, T., Jay, N., Napoli, A.: On measuring similarity for sequences of itemsets. Data Mining and Knowledge Discovery **29**(3), 732–764 (May 2015). https://doi.org/10.1007/s10618-014-0362-1

[30] Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. vol. 96, pp. 226–231 (1996)

[31] Estivill-Castro, V.: Why so many clustering algorithms: A position paper. SIGKDD Explorations **4**(1), 65–75 (2002)

[32] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Magazine **17**(3), 37–37 (1996)

[33] Fernandes, P.: The global challenge of new classes of antibacterial agents: an industry perspective. Current Opinion in Pharmacology **24**, 7–11 (2015)

[34] Freitas, A.A.: Advances in evolutionary computing (2003)

[35] Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: International Conference on Conceptual Structures. pp. 129–142. Springer (2001)

[36] Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer Science & Business Media (2012)

[37] Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46**(1-3), 389–422 (2002)

[38] Haberman, S.J.: The analysis of frequency data: Statistical research monographs (1977)

[39] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering. pp. 215–224 (2001)

[40] Hartigan, J.A.: Direct clustering of a data matrix. Journal of the American Statistical Association **67**(337), 123–129 (1972)

[41] 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)

[42] Henriques, R., Ferreira, F.L., Madeira, S.C.: BicPAMS: software for biological data analysis with pattern-based biclustering. BMC Bioinformatics **18**(1), 82 (2017)

[43] Henriques, R., Madeira, S.C.: BicPAM: Pattern-based biclustering for biomedical data analysis. Algorithms for Molecular Biology **9**(1), 27 (2014)

[44] Henriques, R., Madeira, S.C.: BicSPAM: flexible biclustering using sequential patterns. BMC Bioinformatics **15**(1), 130 (2014)

[45] Henriques, R., Madeira, S.C.: BiC2PAM: constraint-guided biclustering for biological data analysis with domain knowledge. Algorithms for Molecular Biology **11**(1), 23 (2016)

[46] Henriques, R., Madeira, S.C.: BicNET: Flexible module discovery in large-scale biological networks using biclustering. Algorithms for Molecular Biology **11**(1), 14 (2016)

[47] Henriques, R., Madeira, S.C., Antunes, C.: F2G: Efficient discovery of full-patterns. ECML/PKDD nfMCP pp. 1–9 (2013)

[48] Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Van Sanden, S., Lin, D., Talloen, W., et al.: FABIA: Factor analysis for bicluster acquisition. Bioinformatics **26**(12), 1520–1527 (2010)

[49] Hung, J.: An experiment about the classification of antibacterial molecules. Tech. rep., Orpailleur team, LORIA/Inria Nancy-Grand Est (2015)

[50] Hussain, S.F., Ramazan, M.: Biclustering of human cancer microarray data using co-similarity based co-clustering. Expert Systems with Applications **55**, 520–531 (2016)

[51] Ignatov, D.I., Kuznetsov, S.O., Poelmans, J.: Concept-based biclustering for internet advertisement. In: Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. pp. 123–130. IEEE (2012)

[52] Ignatov, D.I., Poelmans, J., Zaharchuk, V.: Recommender system based on algorithm of bicluster analysis RecBi. arXiv preprint arXiv:1202.2892 (2012)

[53] Ignatov, D.I., Watson, B.W.: Towards a unified taxonomy of biclustering methods. arXiv preprint arXiv:1702.05376 (2017)

[54] Ivanenkov, Y.A., Savchuk, N.P., Ekins, S., Balakin, K.V.: Computational mapping tools for drug discovery. Drug Discovery Today **14**(15-16), 767–775 (2009)

[55] John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Machine Learning Proceedings 1994, pp. 121–129. Elsevier (1994)

[56] Johnson, S.C.: Hierarchical clustering schemes. Psychometrika **32**(3), 241–254 (1967)

[57] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20**(1), 359–392 (1998)

[58] Kaytoue, M., Assaghir, Z., Napoli, A., Kuznetsov, S.O.: Embedding tolerance relations in formal concept analysis: an application in information fusion. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 1689–1692. ACM (2010)

[59] Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Biclustering numerical data in formal concept analysis. In: International Conference on Formal Concept Analysis. pp. 135–150. Springer (2011)

[60] Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. Information Sciences **181**(10), 1989–2001 (2011)

[61] Korkmaz, S., Zararsiz, G., Goksuluk, D.: Drug/nondrug classification using support vector machines with various feature selection strategies. Computer Methods and Programs in Biomedicine **117**(2), 51–60 (2014)

[62] Kuflik, T., Boger, Z., Zancanaro, M.: Analysis and prediction of museum visitors' behavioral pattern types. In: Ubiquitous Display Environments, pp. 161–176. Springer (2012)

[63] Kuznetsov, S.O.: A fast algorithm for computing all intersections of objects from an arbitrary semilattice. Nauchno-Tekhnicheskaya Informatsiya Seriya 2-Informatsionnye Protsessy i Sistemy (1), 17–20 (1993)

[64] Kuznetsov, S.O., Ignatov, D.I.: Concept stability for constructing taxonomies of web-site users. arXiv preprint arXiv:0905.1424 (2009)

[65] Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence **14**(2-3), 189–216 (2002)

[66] Laclau, C., Nadif, M.: Hard and fuzzy diagonal co-clustering for document-term partitioning. Neurocomputing **193**, 133–147 (2016)

[67] Lanir, J., Kuflik, T., Dim, E., Wecker, A.J., Stock, O.: The influence of a location-aware mobile guide on museum visitors' behavior. Interacting with Computers **25**(6), 443–460 (2013)

[68] Lee, S., Son, D., Yu, W., Park, T.: Gene-gene interaction analysis for the accelerated failure time model using a unified model-based multifactor dimensionality reduction method. Genomics & Informatics **14**(4), 166 (2016)

[69] Li, F., Yang, Y.: Using recursive classification to discover predictive features. In: Proceedings of the 2005 ACM Symposium on Applied Computing. pp. 1054–1058. ACM (2005)

[70] Liu, L., Chen, L., Zhang, Y.H., Wei, L., Cheng, S., Kong, X., Zheng, M., Huang, T., Cai, Y.D.: Analysis and prediction of drug–drug interaction by minimum redundancy maximum relevance and incremental feature selection. Journal of Biomolecular Structure and Dynamics **35**(2), 312–329 (2017)

[71] Liu, Y.: A comparative study on feature selection methods for drug discovery. Journal of Chemical Information and Computer Sciences **44**(5), 1823–1828 (2004)

[72] 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)

[73] van der Merwe, D., Obiedkov, S., Kourie, D.: AddIntent: A new incremental algorithm for constructing concept lattices. In: International Conference on Formal Concept Analysis. pp. 372–385. Springer (2004)

[74] Padilha, V.A., Campello, R.J.: A systematic comparative evaluation of biclustering techniques. BMC bioinformatics **18**(1), 55 (2017)

[75] Petitjean, F., Allison, L., Webb, G.I.: A statistically efficient and scalable method for log-linear analysis of high-dimensional data. In: 2014 IEEE International Conference on Data Mining. pp. 480–489. IEEE (2014)

[76] Petitjean, F., Webb, G.I.: Scaling log-linear analysis to datasets with thousands of variables. In: Proceedings of the 2015 SIAM International Conference on Data Mining. pp. 469–477. SIAM (2015)

[77] Petitjean, F., Webb, G.I., Nicholson, A.E.: Scaling log-linear analysis to high-dimensional data. In: 2013 IEEE International Conference on Data Mining. pp. 597–606. IEEE (2013)

[78] 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)

[79] Pio, G., Ceci, M., Loglisci, C., D'Elia, D., Malerba, D.: Hierarchical and overlapping co-clustering of mrna: mirna interactions. In: ECAI. pp. 654–659. Citeseer (2012)

[80] Pio, G., Ceci, M., Malerba, D., D'Elia, D.: ComiRNet: a web-based system for the analysis of miRNA-gene regulatory networks. BMC Bioinformatics **16**(9), S7 (2015)

[81] Pontes, B., Giráldez, R., Aguilar-Ruiz, J.S.: Biclustering on expression data: A review. Journal of biomedical informatics **57**, 163–180 (2015)

[82] Prescott, A.M., Abel, S.M.: Combining in silico evolution and nonlinear dimensionality reduction to redesign responses of signaling networks. Physical Biology **13**(6), 066015 (2017)

[83] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2014)

[84] Reutlinger, M., Schneider, G.: Nonlinear dimensionality reduction and mapping of compound libraries for drug discovery. Journal of Molecular Graphics and Modelling **34**, 108–117 (2012)

[85] Rocci, R., Vichi, M.: Two-mode multi-partitioning. Computational Statistics & Data Analysis **52**(4), 1984–2003 (2008)

[86] Sadowski, J., Gasteiger, J., Klebe, G.: Comparison of automatic three-dimensional model builders using 639 x-ray structures. Journal of Chemical Information and Computer Sciences **34**(4), 1000–1008 (1994)

[87] Salah, A., Ailem, M., Nadif, M.: Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[88] Tabachnick, B.G., Fidell, L.S., Ullman, J.B.: Using Multivariate Statistics, vol. 5. Pearson Boston, MA (2007)

[89] Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier. Pattern Recognition Letters **28**(4), 438–446 (2007)

[90] Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics **18**(suppl_1), S136–S144 (2002)

[91] Tang, H., Su, Z.D., Wei, H.H., Chen, W., Lin, H.: Prediction of cell-penetrating peptides with feature selection techniques. Biochemical and Biophysical Research Communications **477**(1), 150–154 (2016)

[92] Tang, J., Alelyani, S., Liu, H.: Feature selection for classification: A review. Data Classification: Algorithms and Applications p. 37 (2014)

[93] Todeschini, R., Consonni, V.: Molecular Descriptors for Chemoinformatics, vol. 41. John Wiley & Sons (2009)

[94] Veroneze, R., Banerjee, A., Von Zuben, F.J.: Enumerating all maximal biclusters in numerical datasets. Information Sciences **379**, 288–309 (2017)

[95] Vichi, M.: Double k-means clustering for simultaneous classification of objects and variables. In: Advances in Classification and Data Analysis, pp. 43–52. Springer (2001)

[96] Véron, E., Levasseur, M.: Ethnographie de l'exposition. Bibliothèque Publique d'Information, Centre Georges Pompidou, Paris (1983)

[97] Webb, G.I.: Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. Machine Learning **71**(2-3), 307–323 (2008)

[98] Xu, X., Li, A., Wang, M.: Prediction of human disease-associated phosphorylation sites with combined feature selection approach and support vector machine. IET Systems Biology **9**(4), 155–163 (2015)

[99] Xue, Y., Li, Z.R., Yap, C.W., Sun, L.Z., Chen, X., Chen, Y.Z.: Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents. Journal of Chemical Information and Computer Sciences **44**(5), 1630–1638 (2004)

[100] Xue, Y., Li, M., Liao, Z., Luo, J., Li, T., Xiao, H., Hu, X.: A biclustering algorithm with coherent evolution on the contiguous columns facing time-series gene data. In: 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). pp. 328–333. IEEE (2014)

[101] Xue, Y., Li, T., Liu, Z., Pang, C., Li, M., Liao, Z., Hu, X.: A new approach for the deep order preserving submatrix problem based on sequential pattern mining. International Journal of Machine Learning and Cybernetics **9**(2), 263–279 (2018)

[102] Yang, J., Wang, H., Ding, H., An, N., Alterovitz, G.: Nonlinear dimensionality reduction methods for synthetic biology biobricks' visualization. BMC Bioinformatics **18**(1), 47 (2017)

[103] Yang, M., Chen, J., Shi, X., Xu, L., Xi, Z., You, L., An, R., Wang, X.: Development of in silico models for predicting p-glycoprotein inhibitors based on a two-step approach for feature selection and its application to chinese herbal medicine screening. Molecular Pharmaceutics **12**(10), 3691–3713 (2015)

[104] Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE transactions on knowledge and data engineering **17**(4), 462–478 (2005)

[105] Zancanaro, M., Kuflik, T., Boger, Z., Goren-Bar, D., Goldwasser, D.: Analyzing museum visitors' behavior patterns. In: International Conference on User Modeling. pp. 238–246. Springer (2007)

[106] Zhang, H., Sun, G.: Feature selection using tabu search method. Pattern Recognition **35**(3), 701–711 (2002)

[107] Zhao, X., Nie, F., Wang, S., Guo, J., Xu, P., Chen, X.: Unsupervised 2d dimensionality reduction with adaptive structure learning. Neural Computation **29**(5), 1352–1374 (2017)

Résumé

L'extraction de connaissances dans les bases de données (ECBD) est un processus qui s'applique à de (potentiellement larges) volumes de données pour découvrir des motifs qui peuvent être signifiants et utiles. Dans cette thèse, on s'intéresse à deux étapes du processus d'ECBD, la transformation et la fouille, que nous appliquons à des données complexes. Nous présentons de nombreuses expérimentations s'appuyant sur des approches et des types de données variés.

La première partie de cette thèse s'intéresse à la tâche de biclustering en s'appuyant sur l'analyse formelle de concepts (FCA) et aux pattern structures. FCA est naturellement liées au biclustering, dont l'objectif consiste à grouper simultanément un ensemble de lignes et de colonnes qui vérifient certaines régularités. Les pattern structures sont une généralisation de la FCA qui permet de travailler avec des données plus complexes. Les "partition pattern structures" ont été proposées pour du biclustering à colonnes constantes tandis que les "interval pattern structures" ont été étudiées pour du biclustering à colonnes similaires. Nous proposons ici d'étendre ces approches afin d'énumérer d'autres types de biclusters : additif, multiplicatif, préservant l'ordre, et changement de signes cohérents.

Dans la seconde partie, nous nous intéressons à deux expériences de fouille de données complexes. Premièrement, nous présentons une contribution dans la quelle nous analysons les trajectoires des visiteurs d'un musée dans le cadre du projet CrossCult. Nous utilisons du clustering de séquences et de la fouille de motifs séquentiels basée sur l'analyse formelle de concepts pour découvrir des motifs dans les données et classifier les trajectoires. Cette analyse peut ensuite être exploitée par un système de recommandation pour les futurs visiteurs. Deuxièmement, nous présentons un travail sur la découverte de médicaments antibactériens. Les jeux de données pour cette tâche, généralement des matrices numériques, décrivent des molécules par un certain nombre de variables/attributs. Le grand nombre de variables complexifie la classification des molécules par les classifieurs. Ici, nous étudions une approche de sélection de variables basée sur l'analyse log-linéaire qui découvre des associations entre variables.

En somme, cette thèse présente différentes expériences de fouille de données réelles et complexes.

Mots-clés: analyse de concepts formels, biclustering, extraction de motifs séquentiels, sélection d'attribut.

Abstract

Knowledge discovery in database (KDD) is a process which is applied to possibly large volumes of data for discovering patterns which can be significant and useful. In this thesis, we are interested in data transformation and data mining in knowledge discovery applied to complex data, and we present several experiments related to different approaches and different data types.

The first part of this thesis focuses on the task of biclustering using formal concept analysis (FCA) and pattern structures. FCA is naturally related to biclustering, where the objective is to simultaneously group rows and columns which verify some regularities. Related to FCA, pattern structures are its generalizations which work on more complex data. Partition pattern structures were proposed to discover constant-column biclustering, while interval pattern structures were

studied in similar-column biclustering. Here we extend these approaches to enumerate other types of biclusters: additive, multiplicative, order-preserving, and coherent-sign-changes.

The second part of this thesis focuses on two experiments in mining complex data. First, we present a contribution related to the CrossCult project, where we analyze a dataset of visitor trajectories in a museum. We apply sequence clustering and FCA-based sequential pattern mining to discover patterns in the dataset and to classify these trajectories. This analysis can be used within CrossCult project to build recommendation systems for future visitors. Second, we present our work related to the task of antibacterial drug discovery. The dataset for this task is generally a numerical matrix with molecules as rows and features/attributes as columns. The huge number of features makes it more complex for any classifier to perform molecule classification. Here we study a feature selection approach based on log-linear analysis which discovers associations among features.

As a synthesis, this thesis presents a series of different experiments in the mining of complex real-world data.

Keywords: biclustering, feature selection, formal concept analysis, sequential pattern mining.