Mining numerical data with formal concept analysis and pattern structures

Mehdi Kaytoue

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR Sciences & Technologies

Traitement de données numériques par analyse formelle de concepts et structures de patrons

THÈSE

présentée et soutenue publiquement le 22 Avril 2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Mehdi Kaytoue

Composition du jury

Rapporteurs : Pr. Jean-François BOULICAUT – Université de Lyon
Pr. Bernhard GANTER – Technische Universität Dresden (Allemagne)

Examinateurs : Dr. Sébastien DUPLESSIS – Chargé de recherche INRA
Pr. Bernard GIRAU – Université Henri Poincaré (Nancy 1)
Pr. Sergei O. KUZNETSOV – Higher School of Economics (Moscou – Russie)
Dr. Amedeo NAPOLI – Directeur de recherche CNRS
Pr. Céline ROUVEIROL – Université Paris-Nord

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

Loria

Mis en page avec la classe thloria.

Remerciements

Je remercie en premier lieu Jean-François Boulicaut, Professeur à l'Université de Lyon, et Bernhard Ganter, Professeur à la Technische Universität de Dresden, d'avoir tous deux accepté d'être rapporteurs de ce mémoire. Leur confiance et expertise ont été une aide précieuse et un honneur. Je les remercie particulièrement d'avoir porté un regard profond et attentionné sur ce manuscrit, malgré les délais serrés et leurs tâches professionnelles respectives.

Je remercie aussi Céline Rouveirol, Professeur à l'Université Paris-Nord, et Bernard Girau, Professeur à l'Université Nancy 1 pour leur participation au jury de thèse, l'intérêt qu'ils ont porté à ce travail et les diverses discussions qui ont suivies.

Sergei Olegovich Kuznetsov, mathématicien à Moscou, a été un collaborateur de choix de nombreux travaux présentés dans ce manuscrit. Sa vision mathématique, son esprit critique sur les problèmes posés et la manière formelle de les résoudre en ont donné une toute autre dimension que je me suis efforcé de suivre. Pour toutes nos réunions de travail (et autres moments), à Moscou, Nancy, Darmstadt, Olomouc, Nicosia, etc., j'adresse mon profond respect et mes remerciements les plus sincères. *Spacibo.*

L'étude du vivant, particulièrement de ces petits champignons qui jouent un rôle si important dans nos forêts, est une science passionnante. Merci Sébastien d'avoir partagé ta passion, merci pour ces longues discussions (bien souvent devant le LORIA – souvent aussi sur bien d'autres sujets). De ton encadrement, de ta confiance, resteront de merveilleux souvenirs.

Il est des gens dont la rencontre change une vie. Tu fais parti de cette courte et précieuse liste. Je ne peux, ni ne veux, expliquer pourquoi ou comment. *Grazie mille Amedeo.*

La préparation de cette thèse s'est accompagnée d'une vie de laboratoire riche. Je remercie ainsi tous les membres du LORIA/INRIA, de l'INRA, et d'autres laboratoires, que j'ai rencontré. Cette richesse de cultures et d'opinions différentes est le pilier central de l'épanouissement. Merci à tous, et plus particulièrement aux membres de l'équipe Orpailleur. À la brigade internationale de la science. Je n'ai pas assez d'encre à mon stylo pour les nommer tous, ils se reconnaîtront j'en suis sûr.

Je remercie aussi les membres des UFR de mathématiques et informatique des Universités de Nancy où j'ai enseigné. Je remercie également les étudiants qui m'ont paradoxalement beaucoup appris sur la science de l'enseignement (même –surtout !– ceux du dernier rang).

Merci à ma famille. Votre réconfort, votre encouragement, votre présence m'ont toujours aidé et guidé dans l'accomplissement en tant qu'individu.

Je suis un piètre marin,
au gré de vagues que ma fantaisie discerne mal.
Telle la proue brisant ces vagues,
tel le vent poussant ma voile,
tel un phare qui me guide,
telle la sirène qui me réconforte,
Bénédicte, merci du fond de mon cœur.

Les ordinateurs sont inutiles.
Ils ne savent que donner des réponses.
(Pablo Picasso)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 Conceptual knowledge discovery in databases

We are living in a world of data. Huge volumes of data –web documents, user information– are available without any intended usage. Large volumes of biological data are now available –genome, transcriptome, proteome, etc.– from which biological knowledge is expected to be discovered. Storing commercial data is also common practice for firms –user preferences, visited webpages history, bought products history, etc.–. In this three (non exhaustive) cases, data hide several useful information that can make life of users easier, genes responsible of a disease discovered, or promising sale sectors of a firm highlighted. However, these useful information are generally buried in the very large amount of data. Accordingly, a challenging question arose in the 90's: "Can we make (very large) data speak?".

Knowledge discovery in databases (KDD) is the process of finding non-trivial, potentially usefull, significant and reusable information in data [44, 43]. Starting from rough data, it consists in three major steps: (i) rough data are prepared, (ii) data are mined and (iii) extracted units are interpreted and may be finally considered as derived knowledge. The objective of this process may be unclear, inexact, or not known *a priori*. KDD is accordingly an iterative and interactive process: to ensure usefulness and accuracy of the results both domain experts and technical experts are generally needed to guide the KDD process.

More precisely, the KDD process can be divided in five steps [43, 44, 40].

Selection. The data needed for the data-mining process may be obtained from many different and heterogeneous data sources. A first step consists in collecting the data from various databases, files, non electronic sources (interviews, books, experts,etc.)

Preprocessing. The selected data may suffer from errors and missing values. Some data values may contradict each other since possibly coming from different sources of data. Errors can be corrected, while missing values can be predicted (often with data-mining tools).

Transformation. Some data-mining algorithms operate on certain types of data only. Accordingly, data should be sometimes transformed, e.g. from quantitative to qualitative data. Data reduction is a kind of transformation that reduces the number of data values being considered, sometimes simply for making the computation with a data-mining algorithm possible.

Data-mining. Data-mining is the use of algorithms to extract the information and patterns (regularities, classes, etc.) derived by the KDD process. Actually, data-mining consists in

pattern discovery or deriving/designing a model from the data.

Interpretation. Information units and/or models discovered with data-mining need to be validated by a domain expert. The way they are presented to the expert is very important. Visualization tools and graphical user interfaces (GUI) are considered at this step.

The KDD process may be also understood as a process turning data to information and then to knowledge, considering the following interpretations [113, 129, 93]:

Data. Data are the uninterpreted signals that reach our senses every minutes. A red, a green, or yellow light at an intersection is an example. Computers are full of data: signals consisting of strings, numbers, characters, and other symbols that are blindly and mechanically handled in large quantities.

Information. Information is data equipped with meaning. For a car driver, a red traffic light is not just a signal of some coloured object, rather, it is interpreted as an indication to stop.

Knowledge. Knowledge is the whole body of data and information that people bring to bear to practical use in action[1], in order to carry out tasks and create new information. Knowledge adds two distinct aspects: first, a sense of purpose, since knowledge is the "intellectual machinery" used to achieve a goal; second, a generative capability, because one of the major functions of knowledge is to produce new information.

Finally, knowledge units should be represented in an adequate representation formalism [24] and may be integrated in ontologies to be re-used for solving problems in application domains such as agronomy, biology, medecine, chemistry, etc. KDD methods and principles are widely considered in the literature [40, 44, 43, 89].

Concepts are necessary for expressing human knowledge, hence the KDD process should benefit from a comprehensive formalization of concepts [129]. Formal Concept Analysis (FCA) [47] offers such formalization of concepts by mathematizing concepts that are understood as units of thought constituted by their extent (the instances of the concept) and intent (their common description). To mathematically define concepts, FCA starts with a binary relation, called formal context, between some (formal) objects and (formal) attributes. Concepts are accordingly defined as pairs constituted of an extent (a set of objects) and an intent (a set of attributes shared by these objects). Concepts form a mathematical structure called concept lattice that expresses a generalization/specialization relation of concepts. The concept lattice is a support for so called conceptual knowledge discovery in databases, but revealed itself to be helpful for applications in information and knowledge processing including visualization, data analysis (mining) and knowledge management. FCA emerged in the 1980's [128] from attempts to restructure lattice theory in order to promote better communication between lattice theorists and potential users of lattice theory and is now a field of applied mathematics on its own. We now make precise the notions of formal context, formal concept, and concept lattice.

Formal context and formal concepts. In FCA, data are represented by a formal context (G, M, I) where G denotes a set of objects, M a set of attributes, and $I \subseteq G \times M$ a binary relation between G and M. The statement $(g, m) \in I$ is interpreted as "the object g has attribute m". A concept is a pair (A, B) composed of a set of objects A and a set of attributes B such that objects in A have all the attributes from B, and vice-versa. In (A, B), the set A is called the *extent* and the set B the *intent* of the concept (A, B).

[1] The term actionability is also used.

Concept lattice. Once all concepts are extracted, they are ordered by inclusion of their extent: a concept is greater than another if it contains more objects in its extent (dually less attributes in its intent). With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I). The concept lattice provides an interesting classification of objects in a domain. It entails both notions of maximality and generalization/specialization: a concept corresponds to a maximal set of objects (extent) sharing a common maximal set of attributes (intent) ; the generalization/specialization is given by the partial ordering of concepts. Furthermore, implications between attributes can be read from the concept lattice.

FCA basically applies to formal contexts, i.e. binary data. The main topic of this thesis concerns the analysis of numerical data with Formal Concept Analysis. Gene expression data is a kind of numerical data that brought a lot of interest in the last decade. We now introduce the problem of gene expression data analysis and show that FCA is a natural way to conciliate its objectives.

2 Gene expression data analysis

Biologists at the UMR IAM (INRA) study interactions between fungi and trees. They published the complete genome sequence of the fungus *Laccaria bicolor* [83]. This fungus lives in symbiosis with many trees of boreal and temperate forests. The fungus forms a mixed organ on tree roots and is able to exchange nutrients with its host in a specific symbiotic structure called ectomycorrhiza, contributing to a better tree growth and enhancing forest productivity, see Figure 1. On the other hand, the plant repays its symbiotic partner by providing carbohydrates, allowing the fungus to complete its biological cycle by producing fruit-bodies (e.g. mushrooms). It is thus of major interest to understand how the symbiosis performs at the cellular level. The genome sequence of *Laccaria bicolor* contains more than 20,000 genes [83]. The study of their expression, or they behaviour, in various biological situations helps to understand their roles and functions in the biology of the fungus.

Figure 1: An example of ectomycorrhiza in nature (Credits: INRA).

2.1 Gene expression data

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

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

A *gene expression data* (GED) is generally described as a gene $\times$ situation table with rows corresponding to genes and columns corresponding to situations, see e.g. Table 1. A table entry is called an *expression value*. A row in the table denotes an *expression profile* associated to a gene (GEP). We consider the NimbleGen Systems Oligonucleotide Arrays technology[2]: expression values range from 0 (not expressed) to 65535 (highly expressed).

Gene Id	a	b	c
Gene 1	11050	11950	1503
Gene 2	13025	14100	1708
Gene 3	6257	5057	6500
Gene 4	5392	6020	7300
Gene 5	13070	12021	15548

Table 1: A gene expression dataset.

2.2 Mining gene expression data

Thanks to powerful and scalable biotechnolgies, a major problem in biology is to derive knowledge from very large gene expression data. A first step is to extract groups of co-expressed genes, i.e. groups of similar gene expression profiles. Indeed, co-expressed genes may interact together within the same biological process [117]. Gene expression data analysis involves all the steps of knowledge discovery in databases. First some genes/situations may be selected for a given study. Second numerical data may be binarized in order to apply data-mining algorithms whose input are binary tables. Finally, extracted patterns with data-mining algorithms must be interpreted and validated (generally after in vitro biological experiments).

A crucial step is data-mining since expression patterns have to consider several properties as listed hereafter.

- The data-mining algorithm should depend as little as possible on prior knowledge.
- GED are obtained from complex procedures involving biological experiments, image acquisition and processing, etc. Accordingly, GED contain a huge amount of noise.
- A gene may participate to different biological processes simultaneously: groups of genes should overlap. In other words, a gene may belong to several groups.
- Biologists are interested in groups of co-expressed genes, but also in the relationships between these groups.

[2] Details on this biotechnology can be found at http://www.nimblegen.com/.

Numerous data-mining methods have been designed since the end of 90's allowing the discovery and description of biological processes in living cells [55, 81, 117]. Data-mining methods extracting groups of co-expressed genes can be divided into three categories:

Clustering. A first data-mining family of methods applied to gene expression data is clustering. Clustering aims at grouping gene expression profiles (GEP) into a disjoint set of classes, called clusters, so that GEP within a class have high similarity, while GEP in separate classes are more dissimilar. Accordingly, clustering allows to group genes into clusters with respect to some similarity criteria between their expression profile. The similarity is defined according to an adequate distance, following given characteristics [55]. The most applied clustering methods in biological works are K-means method [48] and hierarchical clustering [41]. However, clusters are *global* patterns since similarity between GEP is computed w.r.t. all situations simultaneously (possibly weighted). Then, clustering may fail to detect biological processes activated in some situations only [81].

Biclustering. In many applications, and especially in gene expression data analysis, *local* patterns are preferred [81, 18] and consist in pairs (A, B) where A is a subset of objects (here genes) related to a subset of attributes B (here biological situations). Indeed, it is known that a set of genes is activated (e.g. translated into proteins for enabling a biological process) under some conditions only, i.e. only for some attributes. Moreover, most of the genes are involved in several processes [117], i.e. biclusters should overlap.

The type of the relation between the subset of objects A and the subset of attributes B can vary, leading to the definition of several types of biclusters. For example, every value taken by attributes in B for objects in A should be identical, leading to the definition of biclusters of constant values. Another possibility is to consider that those values should be similar w.r.t. a given similarity relation between them, leading to biclusters of similar values. Many other types of bicluters exist, e.g. biclusters of coherent evolution of values, etc. as fully described in [81].

The complexity of the problem of mining biclusters is known to be at least NP-complete [81]. Accordingly, almost all biclustering algorithms use heuristic approaches to identify biclusters. Some algorithms avoid heuristics but exhibit an exponential worst case runtime. Then, it becomes difficult to extract homogeneous biclusters based for example on subtables of a GED and respecting given constraints as their number grows exponentially. If constraints are more "heuristic-like", then interesting patterns may be missed [18]. This is also one of the reasons why only a few biclustering algorithms allow overlapping of biclusters.

Biclustering binary GED. Gene expression data can be binarized into binary tables, see e.g. Table 2, allowing to lower computational difficulties when mining biclusters. In this table, a cross corresponds to gene over expression, i.e. above a specified threshold. It follows that a bicluster can be viewed a set of objects having the same, or almost the same, set of attributes. In [104], authors proposed the *Bi-Max* bi-clustering algorithm, which extracts *inclusion-maximal biclusters*, and showed how it outperforms other biclustering algorithms. Such patterns are described as maximal subtables of "1" values, modulo line and columns permutations. Although the authors do not mention it, the definition of inclusion-maximal biclusters exactly correspond to the definition of a formal concept in FCA.

Gene Id	a	b	c
Gene 1	×	×	
Gene 2	×	×	
Gene 3			
Gene 4			
Gene 5	×	×	×

Table 2: An example binary GED encoding over-expression.

2.3 Towards numerical data mining with formal concept analysis

The history on gene expression data-mining started with clustering. Biclustering was introduced for taking into account the fact that genes are activated in some situations only not necessarily all. Due to problem complexity, many researchers have envisaged to consider binary gene expression data, naturally leading to formal concept extraction. Indeed, FCA can be viewed as a kind of binary biclustering method. It provides means for extracting patterns from a binary relation, namely formal concepts. In application to GED analysis, concept extents are maximal sets of genes related to a common maximal set of situations (not necessarily all). The ordering of concepts among a complete lattice makes overlapping of concepts natural. Then a complete enumeration of patterns respecting some families of constraints is natural.

However, binarizing numerical data comes with loss of information that should be measured and minimized. When information loss is avoided, this may come with very large data whose mining is even worst problem. In this thesis, we investigate how to mine numerical data such as GED with FCA, while avoiding discretization (called scaling in FCA terms). Indeed, researchers in FCA have considered the problem of building concept lattices directly from complex data: Instead of scaling, one may work directly with initial data, i.e. complex object descriptions, defining so-called *similarity operators* which induce a semi-lattice on data descriptions. Several attempts were made for defining such semi-lattices on sets of graphs [46, 69, 70, 79] and logical formulas [31, 45]. Indeed, if one is able to order object descriptions in complex data, e.g. with graph morphism when objects are described by labelled graphs, one may attempt to directly build a concept lattice from such data. In [46], a general approach called *pattern structures* was proposed, which allows one to apply standard FCA to any partially ordered data descriptions. Pattern structures will be our main tool for considering numerical data from an FCA point of view.

3 Contributions and structure of the thesis

In this section, we introduce our main contributions and how they are structured in the present document. The work is divided in several chapters whose ordering follows our research study in time. This makes the reading easier since each chapter follows ideas of the preceding one, trying to answer its questions or extending the ability of the method that it presents. In this way, some definitions and notions are recalled from a chapter to another, to make the reading easier.

Our main contribution concerns the mining of numerical data with Formal Concept Analysis. Chapter 2 accordingly introduces FCA. After recalling elementary notions from order theory, the framework is detailed in classical settings, i.e. considering a binary relation between a set of objects and a set of attributes. Then, we introduce pattern structures that will be our main tool in Chapters 4 to 8 to consider numerical data.

In Chapter 3 we present a naive FCA-based approach for mining gene expression data. An interval based discretization transforms the data into binary on which FCA can be applied. Concepts corresponds to groups of genes (extent) having expression values lying in a same interval for some biological situations (intent). Whereas the originality of such approach is to easily reduce the set of concepts to those highlighting strong expression variations (interesting for biologists), intervals for discretization remain to be chosen *a priori*, a hard task especially in unsupervised settings. This work has appeared in [60, 62, 61].

In Chapter 4 we propose to avoid to choosing those intervals *a priori*, but rather to consider every possible intervals of values. This leads to the definition of a new type of numerical pattern called interval patterns. Intuitively, each object of a numerical dataset is a vector of numbers, where each dimension corresponds to an attribute. Accordingly, an interval pattern is a vector of intervals, where each dimension describes the range of possible values for a given numerical attributes associated with some objects. An interval pattern can represented by a hyperrectangle in Euclidean space, whose sides are parallel to the coordinate axes. To efficiently mine these patterns, we adapt the framework of pattern structures for numerical data, with so called *interval pattern structures*. Accordingly, we explore the ability of FCA to deal directly with numerical data. We experiment this method with gene expression data. This work has appeared in [59, 66].

In Chapter 5 we introduce a similarity relation between numerical values in interval pattern structures. Indeed, the major drawback of interval pattern structures is the very large amount of concepts –the prize to pay when avoiding loss of information linked to scaling–. We show how pattern structures can be modified to lead to concepts defined as maximal sets of objects having similar values for a maximal set of attributes by formalizing similarity as a tolerance relation. We experiment this adaptation of interval pattern structures to information fusion problems in agronomy. This work has appeared in [56, 57, 58].

In Chapter 6 we argue that formal concept analysis can enhance a decision problem when facing information fusion problems, following ideas introduced in Chapter 5. Information fusion consists of merging, or exploiting conjointly, several sources of information for answering questions of interest and make proper decisions. A fusion operator is an operation summarizing all information given by sources into an interpretable information. It happens that the fusion of information of all sources is not exploitable for making a decision. We show that several information fusion operators can be directly embedded in pattern structures. Consequently, instead of providing a unique fusion result which can be problematic, resulting pattern concept lattice yields a structured view of partial results labelled by subsets of sources. These partial results are better candidates for decision making. An experiment on agronomic data is carried out and really justifies this work. This contribution has appeared in [5, 7] and extended in [6].

In Chapter 7 we are interested in defining condensed representations of interval patterns. Indeed, the number of possible interval patterns is generally is too large for enabling their interpretation. A deeply investigated solution in the field of itemset-mining involves condensed representations of patterns. A condensed representation aims at removing all redundant information in the pattern collection. Generally, this new representation is much smaller than the original one. For that matter, we adapt the notions of closed itemsets and generators from itemset-mining to interval patterns with the following semantics: a closed interval pattern is the smallest hyper-rectangle containing a given set of objects while generators are the largest hyper-rectangles containing the same set of objects. We show that closed patterns and generators are very compact representations of interval patterns. This preliminary work takes root in pattern-mining. We provide several algorithms for mining those kinds of patterns and show their usefulness in data-mining. This work was detailed in [63, 65].

In Chapter 8 we show how FCA with either particular dizcretization or pattern structures can

handle the problem of biclustering numerical data. Actually, FCA provides many interesting tools for data-mining: a notion of maximality within concepts, a notion of generalization/specialization of concepts, but also tools for considering noise inherent in real-life datasets. How these tools can be shifted to consider the problem of numerical biclustering is an interesting question. We answer this question by showing how two kinds of biclusters (namely biclusters of constant values and biclusters of similar value) can be extracted with FCA-based methods without using heuristic and reasonable practical complexity. This preliminary work can be also found in [64] and is planned to be extended to other types of biclusters.

In Chapter 9 we present a summary of our work and result. We finish with future directions of research and extensions of our work.

Chapter 2

Formal Concept Analysis

This chapter introduces the framework of Formal Concept Analysis (FCA). Firstly, Section 2.1 introduces basic notions from order theory. Then Section 2.2 presents the important notions of formal contexts, formal concepts and concept lattices, along with mathematical definitions and algorithmic issues. How to handle numerical data with many-valued contexts and conceptual scaling (discretization) is addressed in Section 2.3. Section 2.4 introduces a framework called pattern structures, an extension of FCA to complex data avoiding scaling, that we will use in many of our contributions in the following chapters. Finally, Section 2.5 establishes links between FCA and itemset-mining. These links will be useful for algorithm design and comparison in the next chapters as well.

1 Preliminaries on order theory

In the rest of the dissertation, the following order-theoric notions will be used, and are defined following the first chapter of the seminal book on FCA [47].

Definition 2.1 (Binary relation) *A binary relation R between two arbitrary sets M and N is defined on the Cartesian product $M \times N$ and consists of pairs (m, n) with $m \in M$ and $n \in N$. When $(m, n) \in R$, we usually write mRn. If $M = N$, R is a a binary relation on the set M (or dually on the set N).*

Definition 2.2 (Order relation) *A binary relation R on a set M is called an order relation (or shortly order) if it satisfies the following conditions for all elements $x, y, z \in M$:*

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

For an order relation on a set M, we often use the symbol $\leq$ and write $x < y$ when $x \leq y$ and $x \neq y$. $x \leq y$ is read as usual : "x is less or equal to y". A trivial example of ordered set is the set of real numbers $\mathbb{R}$ with usual relation $\leq$ on numbers. Taking a subset of real numbers $\{1, 6.4, 2, 3.4\}$ one has $1 \leq 2 \leq 3.4 \leq 6.4$. In this example, $\leq$ is a total order, meaning that any two elements can be compared. In many cases, all elements are not comparable, and we have a partial order.

Definition 2.3 (Ordered set) *Given an order relation $\leq$ on a set M, an ordered set is a pair $(M, \leq)$. When $\leq$ is a partial order, $(M, \leq)$ is called partially ordered set, or poset for short.*

Definition 2.4 (Infimum, supremum) *Let $(M, \leq)$ be an ordered set and A a subset of M. A lower bound of A is an element s of M with $s \leq a$ for all $a \in A$. An upper bound of A is defined dually. If it exists a largest element in the set of all lower bounds of A, it is called the infimum of A and is denoted by "inf A" or $\bigwedge A$; dually, a least upper bound is called supremum and denoted by "sup A" or $\bigvee A$. Infimum and supremum are frequently called respectively meet and join, also denoted respectively by the symbols $\sqcap$ and $\sqcup$.*

Definition 2.5 (Lattice, complete lattice) *A poset $\mathcal{V} = (V, \leq)$ is a lattice, if for any two elements $x, y \in V$ the supremum $x \vee y$ and the infimum $x \wedge y$ always exist. $\mathcal{V}$ is called a complete lattice if for any subset $X \subseteq V$, the supremum $\bigvee X$ and the infimum $\bigwedge X$ exist. Every complete lattice $\mathcal{V}$ has a largest element $\bigvee$ called the unit element denoted by 1_V. Dually, the smallest element 0_V is called the zero element. We will rather use the symbol bottom $\perp$ for 0_V and top $\top$ for the unit element in the following.*

The definition of a complete lattice presupposes that both supremum and infimum exist for every subset X. In particular, for $X = \emptyset$, we have $\bigwedge \emptyset = 1_V$, and $\bigvee \emptyset = 0_V$. It follows that $V \neq \emptyset$ for every complete lattice. Every non-empty finite lattice is a complete lattice.

We can reconstruct the order relation from the lattice operations infimum and supremum by

$$x \leq y \iff x = x \wedge y \iff x \vee y = y$$

Definition 2.6 (Join-semi-lattice and meet-semi-lattice) *A poset $\mathcal{V} = (V, \leq)$ is a join-semi-lattice if for any two elements $x, y \in V$ the supremum $x \vee y$ always exists. Dually, it is a meet-semi-lattice if the infimum $x \wedge y$ always exists. A lattice is a poset that is both a meet- and join-semil-attice with respect to the same partial order.*

Finally, one more important notion on which FCA is based concerns closure operators.

Definition 2.7 (Closure operator) *Let S be a set and ψ a mapping from the power set[3] of S into the power set of S, i.e. $\psi : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$. ψ is called a closure operator on S if, for any $A, B \subseteq S$, it is:*

1. *extensive: $A \subseteq \psi(A)$,*
2. *monotone: $A \subseteq B$ implies that $\psi(A) \subseteq \psi(B)$, and*
3. *idempotent: $\psi(\psi(A)) = \psi(A)$.*

A subset $A \subseteq S$ is ψ-closed if $A = \psi(A)$. The set of all ψ-closed $\{A \subseteq S \mid A = \psi(A)\}$ is called a closure system.

[3] The power set of any set S, written $\mathcal{P}(S)$, or 2^S, is the set of all subsets of S, including the empty set and S itself, hence composed of $2^{|M|}$ elements.

	m_1	m_2	m_3	m_4	m_5	m_6
g_1	×	×				×
g_2	×	×		×		×
g_3	×	×		×	×	×
g_4	×		×		×	
g_5	×				×	
g_6	×				×	×
g_7	×		×		×	×

Table 1: An example of formal context $\mathbb{K} = (G, M, I)$

2 Formal concept analysis

Formal Concept Analysis emerged in the 1980's from attempts to restructure lattice theory in order to promote better communication between lattice theorists and potential users of lattice theory [128]. It rapidly growths into a research field leading to a seminal book [47] and FCA dedicated conferences such as the international conferences on concept lattices (ICFCA), on concept lattices and its applications (CLA) and in some extent the international conference on conceptual structures (ICCS). Accordingly, FCA revealed itself to be a simple and well formalized framework useful for several applications in information and knowledge processing including visualization, data analysis (mining) and knowledge management [129, 125, 100]. A website dedicated to FCA is maintenend by Uta Priss[4].

2.1 From a formal context to a concept lattice

In FCA, data are represented by a formal context from which formal concepts are characterized and ordered in a lattice structure.

Definition 2.8 (Formal context) *A formal context $\mathbb{K} = (G, M, I)$ consists of two sets G and M and a binary relation I between G and M. Elements of G are called objects[5] while elements of M are called attributes[6] of the context. The fact $(g, m) \in I$ is interpreted as "the object g has attribute m".*

A formal context is usually represented by a cross table, or binary table. Each line corresponds to an object, while each column to an attribute. A cross in row g and column m means that the object g has the attribute m. A empty table entry means that object in line has not the attribute in column.

Example. Consider the set of objects $G = \{g_1, ..., g_7\}$ where each letter denotes an animal, respectively, "ostrich", "canary", "duck", "shark", "salmon", "frog", and "crocodile". Consider the set of attributes $M = \{m_1, .., m_6\}$ that are properties that animals may have or not, i.e. "borned from an egg", "has feather", "has tooth", "fly", "swim", "lives in air" . Table 1 gives an example of formal context (G, M, I) where I is defined by observing the given animals.

Definition 2.9 (Derivation operators) *For a set of objects $A \subseteq G$ we define the set of attributes that all objects in A have in common as follows:*

$$A' = \{m \in M \mid gIm \ \forall g \in A\}$$

[4] http://www.upriss.org.uk/fca/fca.html

[5] Gegenstande in German.

[6] Merkmal in German.

Dually, for a set of attributes $B \subseteq M$, we define the set of objects that have all attributes from B as follows:

$$B' = \{g \in G \mid gIm \ \forall m \in B\}$$

Example. Consider the formal context in Table 1. We have $\{g_1, g_2\}' = \{m_1, m_2, m_6\}$ and $\{m_1, m_2, m_6\}' = \{g_1, g_2, g_3\}$

Definition 2.10 (Formal concept) *A formal concept of a context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called the extent of the concept (A, B) while B is called its intent. The set of all formal concepts of a context (G, M, I) is written $\mathfrak{B}(G, M, I)$. Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ ($\Leftrightarrow B_2 \subseteq B_1$). The former is called sub-concept of the latter, dually the latter is a super-concept of the former.*

Example. From previous example, it directly follows that the pair $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_6\})$ is a formal concept. Intuitively, a concept corresponds to a maximal rectangle of crosses in its corresponding tabular representation with possible row and column permutations. An example of $\leq$-relation between two concepts is given by:

$$(\{g_1, g_2, g_3\}, \{m_1, m_2, m_6\}) \leq (\{g_1, g_2, g_3, g_6, g_7\}, \{m_1, m_6\})$$

It can be shown that operator $(.)''$, applied either to a set of objects or a set of attributes, is a closure operator. Hence we have two closure systems on G and on M. It follows that the pair $\{(.)', (.)'\}$ is a Galois connection[7] between the power set of objects and the power set of attributes. These mappings put in 1-1-correspondence closed sets of objects and closed sets of attributes, i.e. concept extents and concept intents. In our example, $\{g_1, g_2\}$ is not a closed set of objects, since $\{g_1, g_2\}'' = \{g_1, g_2, g_3\}$. Accordingly, $\{g_1, g_2, g_3\}$ is a closed set of objects hence a concept extent.

The set of all formal concepts from a context $\mathbb{K} = (G, M, I)$ ordered with the relation $\leq$ form a complete lattice called concept lattice of (G, M, I) and denoted by $\underline{\mathfrak{B}}(G, M, I)$. The Basic Theorem on Concept Lattices shows that a concept lattice is complete and defines its infimum and supremum.

Theorem 2.1 (The Basic Theorem on Concept Lattices) *The concept lattice $\underline{\mathfrak{B}}(G, M, I)$ is a complete lattice in which infimum and supremum are given by:*

$$\bigwedge_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcup_{t \in T} B_t \right)'' \right)$$

$$\bigvee_{t \in T} (A_t, B_t) = \left(\left(\bigcup_{t \in T} A_t \right)'', \bigcap_{t \in T} B_t \right)$$

Figure 1 shows the concept lattice associated with Table 1. On this line diagram, each node denotes a concept while a line denotes an order relation between two concepts. Due to *reduced labeling*, the extent of a concept has to be considered as composed of all objects lying in the extents of its sub-concepts. Dually, the intent of a concept is composed of all attributes in the intents of its super-concepts. The top (resp. bottom) concept is the highest (resp. lowest) w.r.t. $\leq$. Along this manuscript, several concept lattice line diagrams will be given. Most of time, we use the software ConExp[8] to draw them.

[7] The definition of Galois connection is not crucial for the understanding of this dissertation. A definition lies in [47], pages 11 and 19.

[8] http://conexp.sourceforge.net/

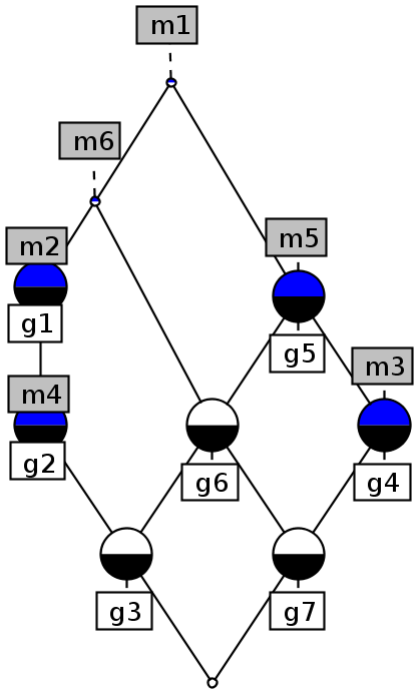


Figure 1: Concept lattice raised from Table 1

2.2 Algorithms

The main algorithmic issue in FCA lies in building concept lattices, or simply the concept set, from formal context that may be very large in real world applications. We first give here a naïve algorithm, before detailing the algorithm *CloseByOne* that will use and adapt in the following chapters.

A naïve algorithm. Consider first the following proposition [47].

Proposition 2.1 *Each concept of a formal context* (G, M, I) *has the form* (A'', A') *for some subset* $A \subseteq G$ *and the form* (B', B'') *for some subset* $B \subseteq M$.

It follows that the set of all formal concepts can be obtained in a naïve way by applying the closure operator $(.)''$ on all possible subsets of G (dually all subsets of M), and removing all redundant concepts. However, this basic algorithm turns to be very inefficient. Several algorithms have been proposed to extract the set of all formal concepts, possibly with their covering relation (actually the concept lattice itself, i.e. concepts ordered with $\leq$). For a detailed analysis and comparison of these algorithms, we refer to [74]. However, the fact that formal concepts can be obtained by "closing" some subsets of objects is interesting and is the basis of several algorithms, e.g. *Ganter's algorithm* known also as *NextClosure* but also *CloseByOne*. In the following, we detail *CloseByOne*, since we will use it and adapt it later in this manuscript.

The algorithm CloseByOne. This algorithm generates all concepts in a bottom-up way (from minimal to maximal extents). It considers objects one by one starting from the minimal one w.r.t. a linear order $<$ on G, e.g. a lexical order on object labels. Given a generated concept (A, B) at a current step, the algorithm adds the next object g w.r.t $<$ in A such as $g \notin A$. Then it applies the closure operator $(\cdot)''$ for generating the next concept (C, D): intent B is intersected

with the description of g, i.e. $D = B \cap g'$, and $C = D'$. Recursiveness of the algorithm induces a tree structure on the set of all concepts, called CbO-tree. To avoid generating several times same concepts, one may use an auxiliary data-structure storing already extracted concepts. To avoid these memory look-ups, the algorithm uses a *canonicity test*. Consider a concept (C, D) obtained from a concept (A, B) by adding object g in A and applying closure. C is said to be canonically generated iff $\{h|h \in C \backslash A \text{ and } h < g\} = \emptyset$, i.e. no object before g has been added in A to obtain C. Moreover, if the canonicity test fails for a given concept, the concept is not stored and the algorithm backtracks. The original pseudo-code for processing a formal context is given in Algorithms 1 and 2. The time complexity of CloseByOne is $O(|G|^2|M||L|)$. More details on this algorithm can be found in [74, 68]. Figure 2 gives an example of formal context and the resulting CbO-tree storing extracted formal concepts. In this figure, each node denotes a concept, and gives successive $(A \cup g)'$ on the first line and $(A \cup g)''$ on the second line, making each $((A \cup g)'', (A \cup g))'$ a formal concept. When crossed-off, the concept is not generated canonically.

Alg. 1 Close By One.

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

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

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

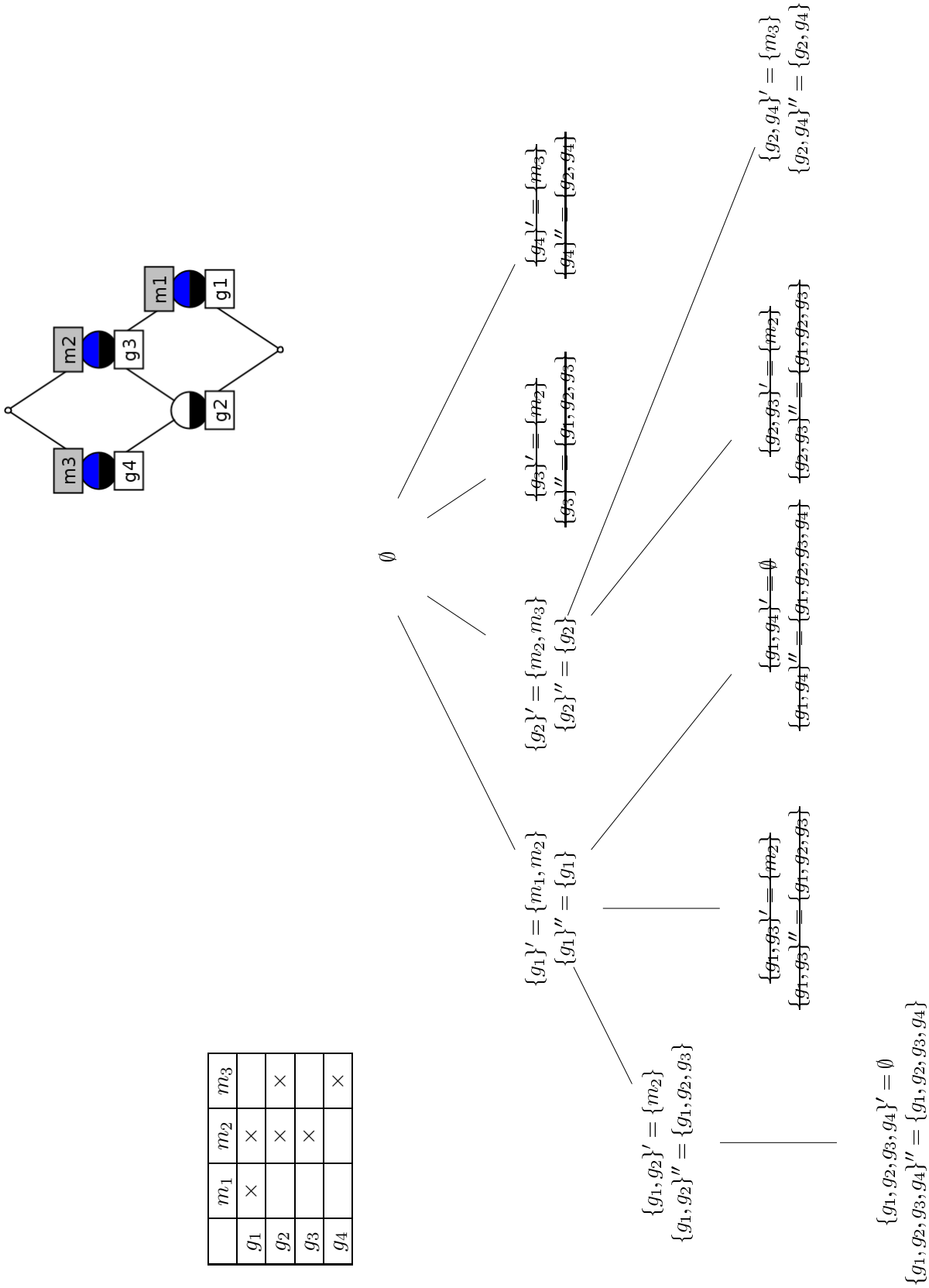
3 Conceptual scaling

Basic formal contexts only consider objects and the attributes they have or not. Such one-valued attributes (or simply binary attributes) contrast with many-valued attributes: an animal can be described also with quantitative attributes such as weight, age, etc. To handle such data in FCA, many-valued contexts are introduced.

Definition 2.11 (Many-valued context) *A many-valued context (G, M, W, I) consits of sets G, M and W and a ternary relation I between those three sets, i.e. $I \subseteq G \times M \times W$, for which it holds that*

$$(g, m, w) \in I \text{ and } (g, m, v) \in I \quad \text{always imply} \quad w = v$$

Elements of G are still called objects. Elements of M are called (many-valued) attributes. Elements of W are called attribute values. Accordingly, the fact $(g, m, w) \in I$ means "the attribute m takes value w for object g", simply written as $m(g) = w$.



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

Figure 2: A formal context with resulting CbO-tree and concept lattice.

Example. Table 2 gives an example of many-valued context (G, M, W, I) with $G = \{g_1, ..., g_5\}$, $M = \{m_1, m_2, m_3\}$ and $W = \{4, 5, 7, 8, 9\}$. Each table entry gives $m(g)$ for attribute m in column and object g in line, e.g. $m_1(g_1) = 5$. This example will support many of our contributions in the next chapters, hence W is here a set of numerical values.

	m_1	m_2	m_3
g_1	5	7	6
g_2	6	8	4
g_3	4	8	5
g_4	4	9	8
g_5	5	8	5

Table 2: A many-valued context (G, M, W, I) also called numerical dataset when $W \subset \mathbb{R}$.

For applying the FCA machinery, a many-valued context needs to be transformed into a formal context with so-called conceptual scaling. Concepts of the resulting concept lattice are interpreted as concepts of the initial many-valued context. Accordingly, the choice of a scale should be wisely done w.r.t. data and goals since affecting the size, the interpretation, and the computation of the resulting concept lattice.

Definition 2.12 *A (conceptual) scale for the attribute m of a many-valued context is a (one-valued) context $\mathcal{S}_m = (G_m, M_m, I_m)$ with $m(G) = \{m(g), \forall g \in G\} \subseteq G_m$. The objects of a scale are called scale values, the attributes are called scale attributes.*

Starting from a many-valued context (G, M, W, I) and the scale context $\mathcal{S}_m$ for all attribute $m \in M$, the derived one-valued context is obtained as follows. The set of objects remains the same. Every many-valued attributes m is replaced by the scale attributes of the scale $\mathcal{S}_m$. Intuitively, each one-valued attribute denotes a "rule" or "constraint" the attribute value of a given object should respect.

We give here three scales taken from [47], page 42. Consider the many-valued context (G, M, W, I) from Table 2. We introduce $W_m \subseteq W$ as the range of each attribute $m \in M$, i.e. $W_m = \{w \in W \mid m(g) = w, \forall g \in G\}$.

- **Nominal scale** is defined by the context $(W_m, W_m, =)$. We obtain the following scales, respectively for attribute m_1, m_2 and m_3:

$=$	4	5	6
4	×		
5		×	
6			×

$=$	7	8	9
7	×		
8		×	
9			×

$=$	4	5	6	8
4	×			
5		×		
6			×	
8				×

- **Ordinal scale** is given by the context $(W_m, W_m, \leq)$ where $\leq$ denotes classical real number order. We obtain for each attribute the following scales:

$\leq$	4	5	6
4	×	×	×
5		×	×
6			×

$\leq$	7	8	9
7	×	×	×
8		×	×
9			×

$\leq$	4	5	6	8
4	×	×	×	×
5		×	×	×
6			×	×
8				×

- **Interordinal scale** is given by $(W_m, W_m \leq) \mid (W_m, W_m \geq)$ where $\mid$ denotes the apposition of two contexts[9]. We obtain for attribute m_1 the following scale[10]:

	≤ 4	≤ 5	≤ 6	≥ 4	≥ 5	≥ 6
4	×	×	×	×		
5		×	×	×	×	
6			×	×	×	×

Now we apply nominal scale to Table 2 to derive the formal context from which a concept lattice representing in some extent the original many-valued context. First, the scale is applied to each attribute separately, then apposition of resulting contexts is operated. One-valued attributes are renamed to be interpretable, e.g. for nominal scaling we have "$m_1 = 4$" as a derived one-valued attribute. Table 3 gives the derived context, while Figure 3 gives its concept lattice representation.

	$m_1 = 4$	$m_1 = 5$	$m_1 = 6$	$m_2 = 7$	$m_2 = 8$	$m_2 = 9$	$m_3 = 4$	$m_3 = 5$	$m_3 = 6$	$m_3 = 8$
g_1		×		×					×	
g_2			×		×		×			
g_3	×				×			×		
g_4	×					×				×
g_5		×			×			×		

Table 3: Derived context from Table 2 with respect to nominal scaling

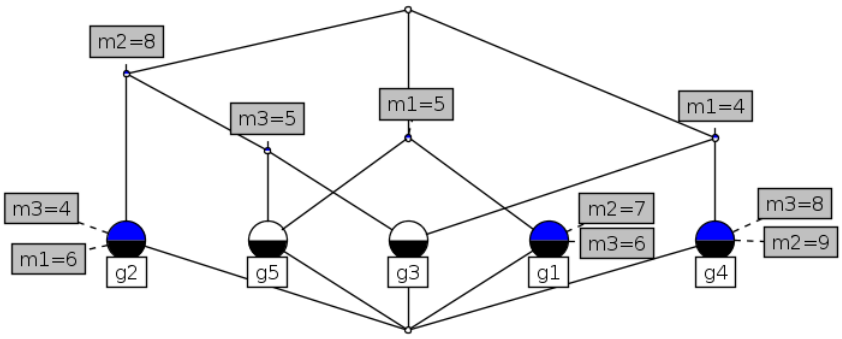


Figure 3: Concept lattice raised from Table 3

We can interpret formal concepts of the obtained concept lattice. Take for example concept $(\{g_3, g_4\}, \{m_1 = 4\})$: m_1 is the only attribute taking the same value for both objects g_3 and g_4, namely the value 4. Accordinly, each concept denotes a maximal set of objects taking the same values for a maximal set of attributes. Choosing either ordinal scale or interordinal scale, we would have a different interpretation. It follows the important choice of the scale, depending on the concept lattice usage. Interordinal scaling will be widely investigated in Chapters 4 and 7.

[9] The *apposition of two contexts* with identical sets of objects, denoted by $\mid$, returns the context with the same set of objects and the set of attributes being the disjoint union of attribute sets of the original contexts.

[10] The double-line column separator intuitively corresponds to context apposition.

Finally, scaling numerical data is closely related to discretization methods transforming quantitative data into binary data. Such methods can also be used to obtain a formal context from a many-valued context when W is a set of real numbers. Many methods are presented in [130] but always involve a loss information that should be measured and minimized. Most of the methods are defined in supervised settings: we known a class membership of the objects, and each attribute range $W_m \subseteq W$ is split into several intervals maximizing some interest functions [42, 116, 97, 130]. The main core of our work is to investigate possibilities to build concept lattices from numerical data without discretization in unsupervised settings using so called *pattern structures* that we introduce now.

4 Pattern structures

Instead of scaling, one may work directly with initial data, i.e. complex object descriptions, defining so-called *similarity operators* which induce a semi-lattice on data descriptions. Several attempts were made for defining such semi-lattices on sets of graphs [46, 69, 70, 79] and logical formulas [31, 45] (see also [49, 126] for FCA extensions and [25, 101, 26, 103] for concept lattices in symbolic data analysis). Indeed, if one is able to order object descriptions in complex data, e.g. with graph morphism when objects are described by labelled graphs, one may attempt to directly build a concept lattice from such data.

In [46], a general approach called *pattern structures* was proposed, which allows one to apply standard FCA to any partially ordered data descriptions from which a concept lattice can be built without *a priori* scaling. In FCA, the operators of the Galois connection put in correspondence elements of the lattices $(2^G, \subseteq)$ of objects and $(2^M, \subseteq)$ of attributes and vice-versa. These lattices are partially ordered sets. This means that if one needs to build concept lattices where objects are not described by binary attributes but by complex descriptions (graphs, intervals, ...), one has to define a partial ordering of object descriptions, see an illustration in Figure 4 taken from [79]. This is the main idea of *pattern structures* formalizing objects from G and their descriptions called *patterns* from a set D where patterns are ordered in a meet-semi-lattice $(D, \sqcap)$ [46]. Indeed in classical FCA, if we consider the lattice of attributes $(2^M, \subseteq)$, it is straightforward that $\forall N, O \subseteq M$, then $N \subseteq O \Leftrightarrow N \cap O = N$, e.g. with $M = \{a, b, c\}$, $\{a, b\} \subseteq \{a, b, c\} \Leftrightarrow \{a, b\} \cap \{a, b, c\} = \{a, b\}$. The set-intersection operator $\cap$ has the properties of a meet operator in a semi-lattice, i.e. commutative, idempotent and associative. This is the underlying idea for ordering patterns with a subsumption relation $\sqsubseteq$: given two patterns $c, d \in D$, $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$. Then, how to build the concept lattice is in full compliance with FCA theory.

Formally, let G be a set (interpreted as a set of objects), let $(D, \sqcap)$ be a meet-semilattice (of potential object descriptions) and let $\delta : G \longrightarrow D$ be a mapping. Then $(G, \underline{D}, \delta)$ with $\underline{D} = (D, \sqcap)$ is called a *pattern structure*. Elements of D are called *patterns* and are ordered by subsumption relation $\sqsubseteq$: given $c, d \in D$ one has $c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$. $\sqcap$ is called a similarity operation, since, given two descriptions, it gives a description representing their similarity. This is natural with set intersection, e.g. $\{a, b\} \cap \{b, c\} = \{b\}$.

A pattern structure $(G, \underline{D}, \delta)$ gives rise to the following derivation operators $(\cdot)^\square$:

$$A^\square = \prod_{g \in A} \delta(g) \qquad for\ A \subseteq G,$$

$$d^\square = \{g \in G | d \in \delta(g)\} \qquad for\ d \subseteq D.$$

These operators form a Galois connection between the powerset of G and $(D, \sqsubseteq)$. *Pattern concepts* of $(G, \underline{D}, \delta)$ are pairs of the form (A, d), $A \subseteq G$, $d \in D$, such that $A^\square = d$ and $A = d^\square$.

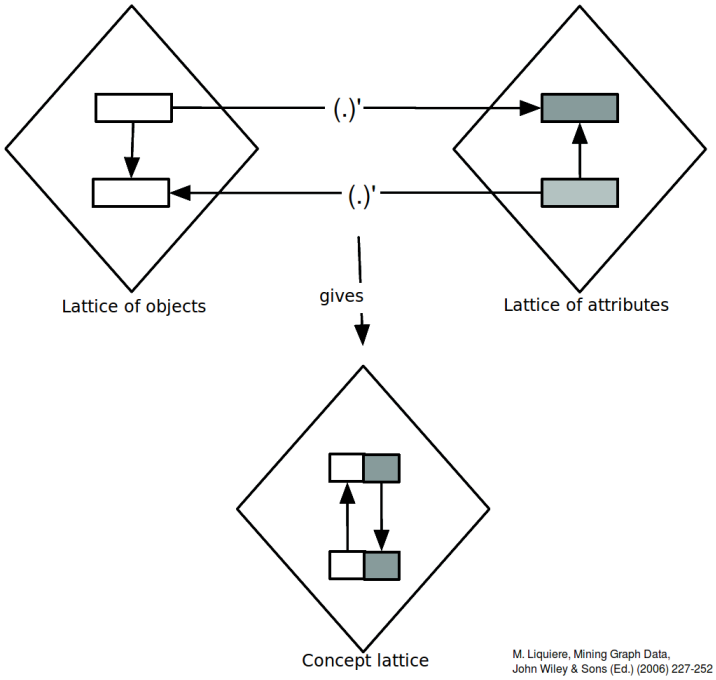


Figure 4: Illustration of the Galois connection

For a pattern concept (A, d) the component d is called a *pattern intent* and is a description of all objects in A, called *pattern extent*. Intuitively, (A, d) is a pattern concept if adding any element to A changes d through $(\cdot)^\square$ operator and equivalently taking $e \supset d$ changes A. Like in case of formal contexts, for a pattern structure $(G, \underline{D}, \delta)$ a pattern $d \in D$ is called *closed* if $d^{\square\square} = d$ and a set of objects $A \subseteq G$ is called *closed* if $A^{\square\square} = A$. Obviously, pattern extents and intents are closed.

5 Links with itemset mining

The problem of frequent itemset mining was introduced in [2]. It takes roots in the application of *market basket analysis*. Firstly, consider a set of customers, and a set of products called items. Each customer has bought some products. Then, it is interesting to search for sets of items, or *itemsets*, that frequently co-occurs together for different customers. For example, it may happens that both products "cereals" and "milk" are often simultaneously bought by customers. We say that the set $\{milk, cereals\}$ is a frequent itemset.

Once frequent itemsets are found, it allows to generate *association rules* among the itemsets. Such rules denotes dependencies between itemsets. For example, a rule could be "Customers that simultaneously bought milk and cereals also tend to buy jam". From this rule, a grocery store with decent practices will ensure to co-locate those three items in the same place. A dishonest practice, more common in supermarkets, would consist at locating the jam in a different place, forcing the customer to walk, and potentially buy other products on his way. It seems indeed to be a fact that "the more time one spend in a supermarket, the more products one will buy".

Leaving aside those ethical considerations, we now formally define frequent itemsets using notations of FCA. Indeed, the basic data in itemset mining is a formal context, i.e. in our example (G, M, I) where G is a set of customers, M a set of products, and $(g, m) \in I$ means

that customer g bought product m.

Definition 2.13 (Itemset, support and frequent itemset) *Given a formal context (G, M, I), an itemset $B \subseteq M$ is a subset of attributes or items. The cardinality of B is also called its length. The image of an itemset consists in the the of objects owning simultaneously all elements in B, i.e. the set B', while its support is the number of these objects, i.e. $|B'|$. Given a so called minimal support $\sigma_s \in [1, |G|]$, an itemset B is said to be frequent if $|B'| \geq \sigma_s$.*

Given a set M of items, there is $2^{|M|}$ possible itemsets, which are clearly not analysable. Mining frequent itemsets allows to reduce this number. However, this is not sufficient enough with low minimal supports. A deeply investigated solution involves condensed representations of itemsets. A condensed representation aims to remove all redundant information in the frequent itemsets and the representation may be much smaller than the original frequent itemsets. Two well known condensed representations are the set of closed itemsets, and the set of generators, also called key-itemsets. Those patterns are defined upon equivalence classes of itemsets [96, 11, 94, 95].

Definition 2.14 (Equivalence class) *Two itemsets B_1 and B_2 are said equivalent iff they have the same image: $B_1' = B_2'$, and we write $B_1 \cong B_2$. The set of itemsets that are equivalent to an itemset B_1 is denoted by $[B_1] = \{B_2 | B_1 \cong B_2\}$ and is called the equivalence class of B_1.*

Definition 2.15 (Closed itemset) *An itemset B_1 is closed if there does not exist any pattern B_2 such as $B_1 \subseteq B_2$ with $B_1 \cong B_2$.*

Definition 2.16 ((Itemset) generator) *An itemset B_2 is a generator if there does not exist a pattern B_1 such as $B_1 \subseteq B_2$ with $B_1 \cong B_2$.*

Accordingly, an equivalence class is a set of itemsets with same image and same closure in FCA terms. In an equivalence class, there is one unique closed itemset with maximal length, and one or several generators with minimal length. We say that both collections of frequent closed itemsets and generators are condensed representations since each one forms a compact and lossless representation of frequent itemsets, from which any frequent itemset can be retrieved.

An intense effort has lead to several algorithms for mining frequent closed itemsets and/or generators, the later being used for generating association rules. Among many others, we should cite here Charm [132] and LCMv2 [124] for closed itemset mining, Gr-Growth [80] for itemset generator mining and Zart for mining both pattern types simultaneously [122].

Concerning our work in the next chapters, the most important fact we will use is the following. Closed itemsets exactly corresponds to concept intents from the same formal context. This means that if one need to compute formal concepts from a formal context (without their covering relation), one may use either FCA algorithms (e.g. CloseByOne), or closed itemset-mining algorithms, whose efficiency depends on the input data size and distribution.

Chapter 3

Extracting gene expression patterns with significant variations

In this chapter, we present a first and simple KDD approach for mining gene expression patterns in gene expression data. This method involves all the steps of a KDD process. First, data are prepared and transformed into binary data, allowing to apply FCA. Then, concept intents are filtered with syntactic constraints to retain those highlighting strong variations of expression. Finally, with real world data, the expert interprets some of the extracted patterns, and establishes biological hypothesis to be validated experimentally. Most importantly, this chapter sets the basis of our main motivation in the next chapters, i.e. building concept lattices from numerical data without binarization.

1 Introduction

A microarray experiment considers a large number of genes, eventually the complete coding space of a genome in multiple situations. These situations can be a time-series during a particular biological process (e.g. cell cycle), a collection of different tissues (e.g. normal and cancerous tissues) or both, sometimes responding to particular environmental stresses.

By measuring the expression value of a gene in m situations, a gene expression profile can be written as a m-dimensional numerical vector $e = (e_1, ..., e_m)$ where e_j is the expression value of the gene in the j^{th} situation ($j \in [1, m]$). A gene expression dataset (GED) is formalized by a matrix $E = (e_{ij})_{1 \leq i \leq n, 1 \leq j \leq m}$ is a collection of n profiles: it is composed of n lines which correspond to genes and m columns which corresponds to situations. e_{ij} is the expression value of the i^{th} gene in the j^{th} situation. For example, in Table 1, $(11050, 11950, 1503)$ is the expression profile for the Gene 1. $e_{11} = 11050$ is the expression value of the Gene 1 in the situation a. Clustering methods groups similar profiles together into a cluster, leading, when interpreted by a domain expert, to the understanding of biological processes and of function of genes [76, 118].

The goal here is to extract groups of genes having similar expression values for some, maybe all, biological situations. Moreover, we wish that expression values between two situations highlight a significant change. Indeed, these changes may characterize particular biological processes. For example, consider the family of genes involved in the growth of the fruit-body of a mushroom. It is supposed that those genes have significant rise of expression between early stage and later stage of the fungus development.

Gene Id	a	b	c
Gene 1	11050	11950	1503
Gene 2	13025	14100	1708
Gene 3	6257	5057	6500
Gene 4	5392	6020	7300
Gene 5	13070	12021	15548

Table 1: An example of GED composed of 5 genes in lines and 3 situations in columns.

2 An FCA-based approach

This section proposes to use FCA to extract from a GED groups of co-expressed genes represented by concepts. Firstly, a GED is mathematically defined as a many-valued context, then turned into a formal context using a particular conceptual scaling. The concepts of the formal context are searched for and structured into a concept lattice. Finally, concepts are filtered using a particular representation of concept intents to retain from the large collection of concepts only those with most significant variations of expression.

2.1 A GED as a many-valued context

A GED is considered as a many valued context $\mathbb{K}_1 = (G, S, W, I_1)$ where G is a set of genes, S a set of situations, and $g(s) = w$ means that the expression value of gene g is w in situation s. In the example used in this section (Table 1), $G = \{g_1, g_2, g_3, g_4, g_5\}$, $S = \{a, b, c\}$, and I_1 is illustrated, for example, by $g_1(a) = 11050$, i.e. $(g_1, a, 11050) \in I_1$. The objectives are to use FCA to extract concepts (A, B), where $A \subseteq G$ is a subset of genes that shares similar values of W in the situations of $B \subseteq S$. As concept lattice construction needs a formal context, $\mathbb{K}_1$ is now scaled.

2.2 Conceptual scaling

Given an attribute value space of the form $[0, u]$, the scale is given by a set of intervals $T = \{[0, u_1],]u_1, u_2], ...,]u_{p-1}, u_p]\}$. p is the number of intervals of T and $u_p = 65535$ for the NimbleGen System. In the present application, the interval bounds u_i ($i \in [1, p]$) are dependent on expert knowledge. The scaling procedure consists in replacing each many-valued attribute of $\mathbb{K}_1 = (G, S, W, I_1)$ with p one-valued attributes to create the formal context $\mathbb{K}_2 = (G, S \times T, I_2)$. $S \times T$ is then a set of pairs: the first value is a situation while the second represents an interval. $(g, (s, t)) \in I_2$ means that the gene g has an expression value in the interval t in the situation s.

This procedure is illustrated in the Table 3 with $T = \{[0, 5000[, [5000, 10000[, [10000, 65535]\}$. The many-valued attribute a is replaced by the three one-valued attributes (a, t_1), (a, t_2) and (a, t_3), i.e $(a, [0, 5000[)$, $(a, [5000, 10000[)$ and $(a, [10000, 65535])$. Then $(g_1, (a, t_3)) \in I_2$ means that gene g_1 has an expression value in t_3, i.e. in $[10000, 65535]$, for the situation a and represented as the first cross in Table 3.

Classical discretization problems appear with conceptual scaling: introduction of biases, loss of information and may strongly influence the size of the resulting lattice. Moreover, a major challenge in microarray data analysis is to effectively dissociate actual gene expression values from experimental noise. To limit biases of scaling involving values close to interval bounds, and to partially manage microarray noise, we follow the idea given in [34, 91]: a threshold $l \in [0, 1]$

is used to define the scale T as follows: $T = \{[0, u_1 + u_1 \times l], \ldots, [u_{p-1} - u_{p-1} \times l, u_p]\}$, meaning that intervals of T can overlap.

	(a,t_1)	(a,t_2)	(a,t_3)	(b,t_1)	(b,t_2)	(b,t_3)	(c,t_1)	(c,t_2)	(c,t_3)
g_1			×			×	×		
g_2			×			×	×		
g_3		×			×			×	
g_4		×			×			×	
g_5			×			×			×

Table 2: A formal context derived from the many-valued context of Table 1.

2.3 Lattice construction and interpretation

In our settings, a concept (A, B) represents a subset of genes A that share similar expression values in the situations defined by the elements of B. The intent B is the common gene expression description of the genes in the extent A.

For example Table 3 contains four concepts (A, B):

- $C1 = (\{g_3, g_4\}, \{(a, t_2), (b, t_2), (c, t_2)\})$: it means that the genes g_3 and g_4 are co-expressed, by sharing expression values in the same interval t_2 in situations a, b and c.
- $C2 = (\{g_5\}, \{(a, t_3), (b, t_3), (c, t_1)\})$
- $C3 = (\{g_1, g_2\}, \{(a, t_3), (b, t_3), (c, t_3)\})$
- $C4 = (\{g_1, g_2, g_5\}, \{(a, t_3), (b, t_3)\})$

Figure 1 represents the concept lattice of context given in Table 3. It provides interesting insights of relation between genes for the biologists and thus may lead to knowledge discovery. First to consider a single concept is intersting because it represents a group of genes having similar quantitative expression values, and thus that may belong to a same biological process or share a close function. Another approach may consist to consider several concepts at the same time. For example, biologists may look at several linked concepts. If we consider concepts $C2, C3, C4$, we note that $C4$ is a super concept of $C2$ and $C3$. Genes of these two lasts concepts share thus a common description that is the intent of $C4$. Intents of $C3$ and $C2$ differ in situation c only. Biologists know that the expression of a gene is controlled by molecules called *transcription factors*. They may infer for example that g_5 expression is controlled by another transcription factor which is over-expressed in the situation c. Another advantage of the concept ordering relation is to take natively noise into account. On the same example, if the numerical value derived into (c, t_3) is an error, then grouping g_1, g_2 and g_5 is possible.

2.4 Concept filtering

A GED can contain thousands of genes and dozens of situations. For these reasons, the resulting lattice may contain a large number of concepts (up to a million). The biologist focuses on small and homogeneous gene groups presenting the most important variations simultaneously. Interpretation of variations leads after experimental validations to the discovery of gene functions and biological processes. Large variations are important to discriminate genes responsible of a particular cellular process [76]. Concepts are groups of genes co-expressed in a certain number

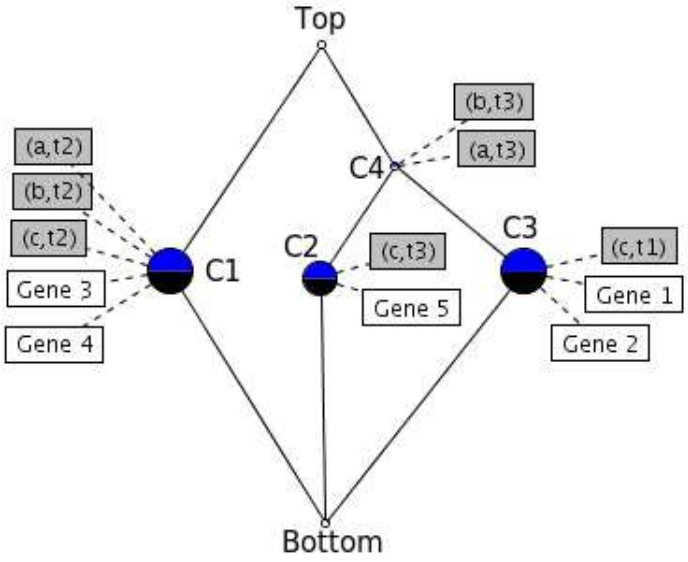


Figure 1: The concept lattice raised from Table 3.

of situations and a gene (or a situation) may belong to multiple concepts. To focus on patterns with most significant variations of expression, we introduce the following filterings.

Filter to control both intent and extent sizes. A concept is relevant if the extent is not composed of "too many" genes, and if the intent contains a least "a few" situations [81]. A first filtering step keeps only concepts (A, B), with $|A| \leq a$ and $|B| \geq b$. a and b are chosen by the biologist and materialize the modalities "too many" and "a few".

Filter to retain concepts showing variations of expression. A concept describes a group of co-expressed genes, i.e. having expression values in the same interval in each the situations. However biological knowledge implies that these expression values may often be similar between the situations, i.e. presenting no high variation of expression. The key idea is the following: the concept $(A, B) = (\{g_3, g_4\}, \{(a, t_2), (b, t_2), (c, t_2)\})$ presents no high variation of expression because each $t \in T$ such as $(s, t) \in B$ is the same (in this case, $t = t_2$), i.e. the expression values are always in the same interval. To identify and remove such type of concepts, we introduce the following formalism.

We consider an index set K on T and replace in an intent all elements of T by corresponding element of K (indexes begin at position 1). Previous concept example intent becomes $\{(a, 2), (b, 2), (c, 2)\})$. Now, for each concept, the intent B is a set of pairs (s, k) where $k \in K$ is an integer valuation providing a control on expression values: $B = \{(a_1, k_1), \ldots, (a_p, k_p)\}$. In the current and next paragraphs, we consider (a_i, k_i) and (a_j, k_j) as two distinct elements of B $(i \neq j)$. A *variation* is defined as a non null difference between k_i and k_j. This definition naturaly relies on the number of intervals of the scaling and their size. Then retaining *variant concepts*, i.e. having variations, consists to keep those having intents B respecting the predicate (1), i.e. $hasVariation(B) = true$. Others, called *constant concepts*, are removed.

$$hasVariation(B) = \exists (a_i, k_i) \in B \text{ and } \exists (a_j, k_j) \in B \text{ such as } k_i \neq k_j \quad (1)$$

Filter to control gene expression variation amplitude. One may notice that $\{(a,15),(b,2),(c,2)\}$ has unformally higher variations than $\{(a,3),(b,2),(c,2)\}$, because $15-2 > 3-2$. Thus to have more control on variations, we define the *α-variation* as a difference between k_i and k_j of at least α, i.e. $|k_i - k_j| \geq \alpha$. Then a concept is *α-variant* if its intent B respects (1), i.e. $hasVariation(B,\alpha) = true$, with $\alpha \geq 0$.

$$hasVariation(B,\alpha) = \exists (a_i,k_i) \in B \; and \; \exists (a_j,k_j) \in B \; such \; as \; |k_i - k_j| \geq \alpha \tag{2}$$

Filter to control occurrences of an α-variation. Finally, yet another may notice that $\{(a,15),(b,2),(c,12)\}$ has more variations than $\{(a,15),(b,2),(c,2)\}$. Then a concept is $(\alpha,\beta)-$*variant* if its intent B respects (1), i.e. $hasVariation(B,\alpha,\beta) = true$, with $\alpha \geq 0$ *and* $\beta \geq 1$. Intuitivly an $(\alpha,\beta) - variant$ concept presents in B at least a number β of α-variations.

$$hasVariation(B,\alpha,\beta) = (|\{((a_i,k_i),(a_j,k_j)) \; with \; |k_i - k_j| \geq \alpha\}| \geq \beta) \tag{3}$$

Examples:

- the concept $(A,B) = (A,\{(a,6),(b,6),(c,6)\})$ is constant,
- the concept $(A,B) = (A,\{(a,2),(b,6),(c,6)\})$ is variant,
- the concept $(A,B) = (A,\{(a,2),(b,6),(c,6)\})$ is α-variant with $\alpha \leq 4$,
- the concept $(A,B) = (A,\{(a,2),(b,4),(c,4)\})$ is α-variant with $\alpha \leq 2$,
- the concept $(A,B) = (A,\{(a,2),(b,6),(c,11)\})$ is $(4,3)$-variant.
- the concept $(A,B) = (A,\{(a,2),(b,6),(c,8)\})$ is not $(4,3)$-variant.

α and β are two parameters allowing the biologist to focus on the most important variations. The choice of these parameters strongly depends on the choice T

3 Experiments

In this section, we apply our methodology on a real dataset implying a fungus species *Laccaria bicolor* for its symbiosis capacity with trees. We show that our methodology is able to extract groups of co-expressed genes in some or all situations. As the genome of *Laccaria bicolor* has been recently published [84], it is hard for now to check the hypothesis we formulate in the following (a few knowledge on specific processes of the symbiosis is available), experimental validation by biologist is required. Indeed, it is the first genome of a fungus with this lifestyle (symbiosis) that has been sequenced. However, we show that the parameters α and β are meaningful to reduce the number of concepts, and that this discrimination allows a number of hypothesis.

3.1 Data and material

Biologists at the UMR IAM (INRA) study interactions between fungi and trees. They recently published the complete sequencing of the genome of the fungus *Laccaria bicolor* [84]. This fungus live in symbiosis with many trees of the temperate forest: the fungus grabs mineral nutrients in surrounding soil, improves the nutrition of the tree by allocating a part of its nutrients, and receives carbon in return through association to the root tissue. This fungus has a beneficial

impact on tree growth and positively influences forest productivity. It is thus a major interest to understand how the symbiosis performs at the cellular level.

The sequencing of *Laccaria bicolor* genome has allowed the prediction of more than 20,000 genes [84]. It remains now to study expression of those genes to understand their functions in the fungal lifestyle. Microarray measurements in several situations is a critical solution. For example, it enables to compare the expression values of genes between different situations like free-living cells of the fungus (i.e. mycelium), cells engaged in the symbiotic association (i.e. ectomycorrhiza), and cells of specialized fruiting-body structures (i.e. mushroom).

A GED is available at the Gene Expression Omnibus at National Center for Biotechnology Information (NCBI)[11]. It is composed of 22,294 genes in lines and 7 various biological situations in columns, i.e. free-living cells (M81306 and MS238), young (FBe) and mature (FBl) fruiting body cells and fungal cells in association with roots of different trees (Poplar, MPgh, Mpiv, Douglas Fir, MD).

We mainly use the *Coron System* [121] composed of several modules corresponding to the different steps of the methodology. First the module *Transformer* has been added to scale the data. Then closed itemsets have been extracted with the *Charm* algorithm [33]. Indeed, the intension of a concept is a closed itemset.

3.2 Method and results

Applying the methodology consists in selecting genes G and situations S to study, to scale the resulting many-valued context $K_1 = (G, S, W, I_1)$ into the formal context $K_2 = (G, M, I_2)$ and to extract and filter concepts from K_2.

For first experiment, we work with the whole set of genes (i.e. $|G| = 22,294$) and a subset of the situations $S = \{MP, MD, Fbe, FBI, Myc\}$ such as MP represents in-symbiosis cells (the mean of the columns $MPgh$ and $Mpiv$), and Myc represents mycelium cells (mean of $M81306$ and $MS238$). The expert biologist choose a simple scale T whose interval borders are $u_1 = 20000$ and $u_2 = 40000$ (3 intervals) and an overlapping threshold of 0.05. Extraction returns 893 concepts. We apply filters: a concepts (A, B) of this set is retained if $|A| \leq 50$, $|B| \geq 4$ and if it is $(2, 3) - variant$. We finally obtain 35 concepts that are analysable by the expert. Two of these concepts are presented in Figure 2 (a) et (b). In these line-plots, Y-axis contains situations y such as $(y, t) \in B$. X-axis is the expression value axis. A point (x, y) is the expression value y of a gene in the situation x. All expression values of one single gene are linked by a line. Thus, each line represents the expression profile of a gene like in Table 1. The intent of the concept (a) is $B = \{(MD, t_1), (FBe, t_3), (FBi, t_3), (Myc, t_1)\}$ while the intent of (b) is $B = \{(MP, t_3), (MD, t_3), (FBe, t_3), (FBi, t_3), (Myc, t_1)\}$. By observing the graphical representation, we are able to say that these concepts represents groups of genes sharing the same behavior. Most of the genes of 35 concepts remains today of unknow function. However, some hypothesis can be made. Genes of group (a) may be involved processes of the fruit body structure. Indeed their expression values are high only in Fbe and FBi. Genes of group (b) may play a major role in the symbiosis: their expression is high in in-symbiosis and fruit cells and low in free-living mycelium cells. Biologists know that symbiosis is favoured when the fruit is well established.

One may notice the capacity of the method to take partially noise into account. Concept (a) $= (A, B)$ is such that $B = \{(MD, t_1), (FBe, t_3), (FBi, t_3), (Myc, t_1)\}$. It describes no condition on the interval for the situation MP, but the behaviour of the genes remains coherent, except

[11] http://www.ncbi.nlm.nih.gov/geo/ as series GSE9784

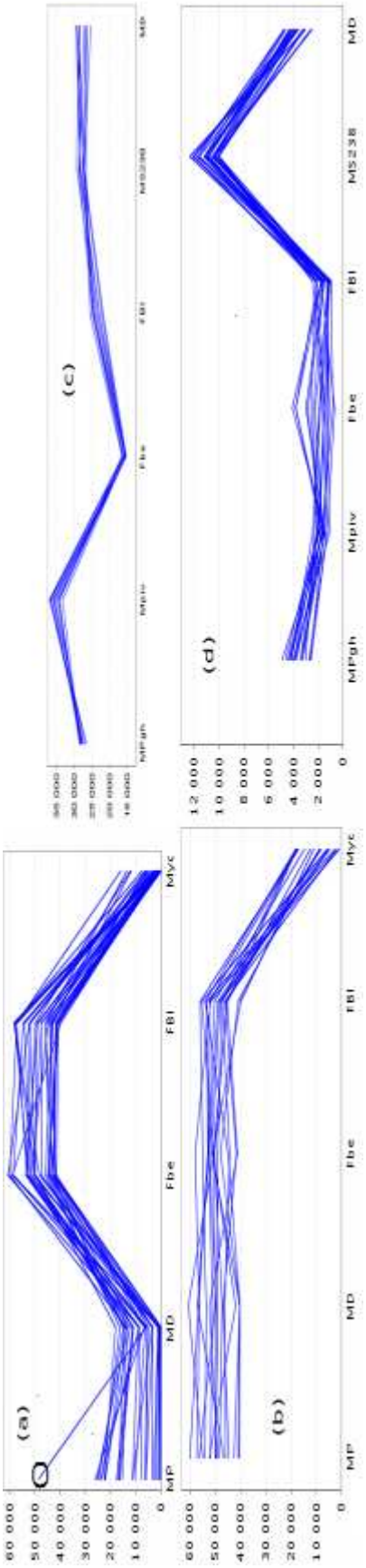


Figure 2: Graphical representation of gene expression concepts.

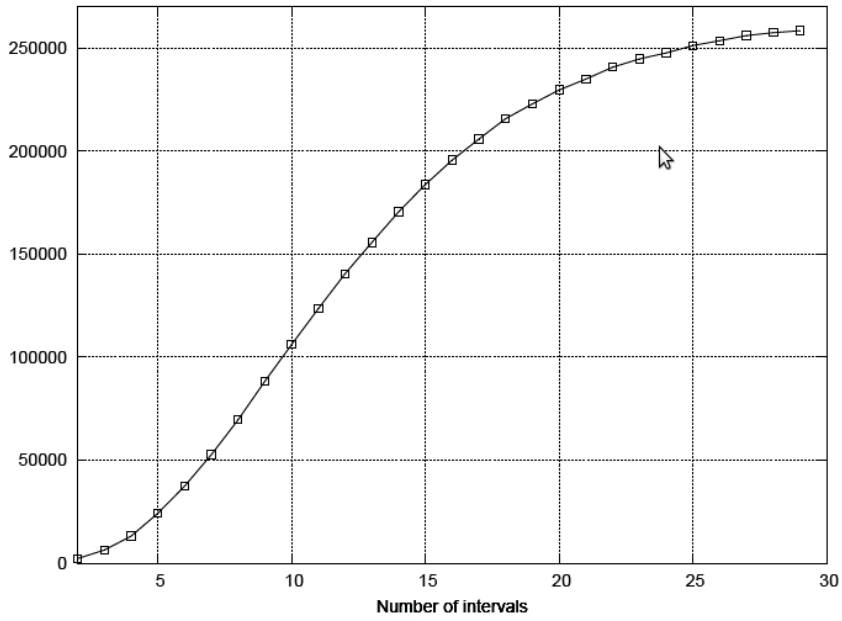


Figure 3: Number of concepts w.r.t. different scales.

for one gene: the incoherent value is indicated by a circle. Despite of this artifact the grouping is possible.

Second experiment starts with S composed of every situation of the dataset except $M81306$ for its bad quality (a priori knowledge) and all the genes. The scale is composed of 15 intervals and $l = 0.05$. We extract $71,391$ concepts and retain those respecting the following properties: $|A| \leq 50$, $|B| \geq 4$ and the concept is $(4,2) - variant$. $9,324$ remains and are not analysable. However, we remark that many concepts contain only a few genes (due to the high cardinality of T). We also add the following constraint, concepts must verify: $|A| \geq 10$. Now, 54 concepts remains. Two of these concepts are presented in 2 (c) and (d). Genes of concept (c) are strongly co-expressed but their function is here again unknown. However, they have been identified as potential proteins of the same type in the yeast species *Candida albicans* by comparing DNA sequences. Genes of group (d) may be involved in growth of mycelium (highly expressed in $MS238$ only).

3.3 Variation constraint evaluation

We have shown two experiments, the first with a low $|T|$ (i.e. a few intervals) and the second with a high $|T|$ (i.e. several intervals). The choice of the number of intervals and their size is difficult and directly influence the quality (not studied here) and the cardinality of the result as shown in Figure 3. This figure gives the number of concepts w.r.t. a scale of a $|T|$ intervals, obtained by the quantile discretization method of data of the second experiment. If $|T|$ is low, the number of concepts and their quality is generally low w.r.t. a higher $|T|$. If $|T|$ is high, the number of concepts explodes, but the quality is better, and the filters allows to reduce it (see Figure 4). Concepts of 2 (c) and (d) would have not been found with $|T| = 3$. The right scale for a given data and a given goal is done via iterative application of the method, in interaction with the experts (both computer scientists and biologists) like most of methods of KDD.

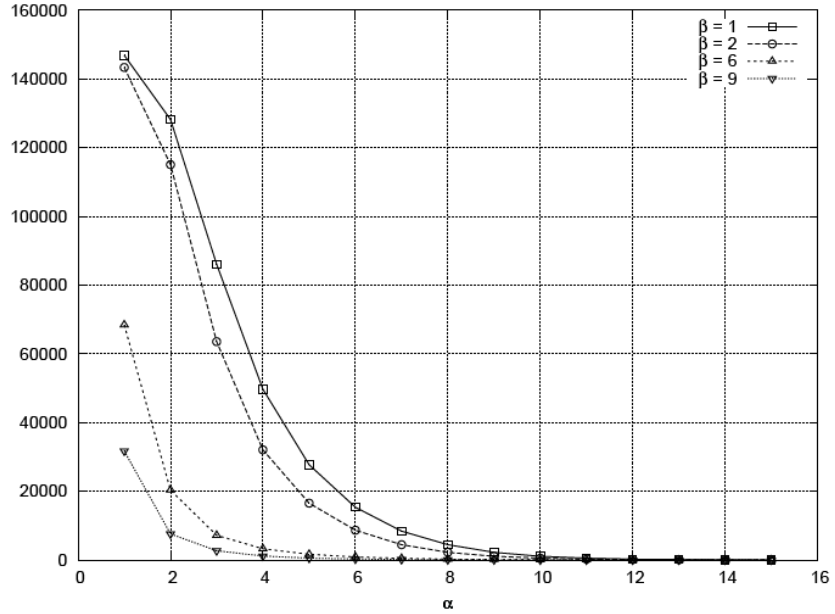


Figure 4: Number of concepts w.r.t. parameters α and β .

4 Towards interval patterns

In this chapter, we have shown an example of how formal concept analysis can be used to mine gene expression data. A simple and fully customizable conceptual scaling allows the expert to use knowledge to filter the resulting formal concepts. However, from a qualitative point of view, there is no universal scaling. The impact of a scaling on the quality of the extracted formal concepts must be studied in each different case [107]. One may use our methodology for any numerical data (sometimes a normalization procedure is required) of whom one wants to extract sets of objects sharing a similar behaviour and presenting important variations, where minimal frequency is not sufficient to extract relevant patterns (e.g. financial or demographic analysis).

As stated earlier, the major drawback of our approach is the choice of the thresholds required to scale the numerical dataset. Whereas a lot of effort has been done in this area, see e.g. [130], an appropriate discretization splits attribute ranges into intervals maximizing some interest functions, e.g. support, confidence. In a lot of cases, this requires to know the class of each object (i.e. supervised settings, see e.g. [42]).

From a knowledge discovery point of view, one should not choose those thresholds to define the intervals, but rather consider all possible intervals and then consider the best patterns w.r.t. some interest functions, constraints, condensed representations of patterns etc. In this way, a so called interval pattern can be written as a vector $\langle [a_i, b_i] \rangle$ where each i correspond to a unique attribute of the dataset. For making the searchspace of such interval patterns finite and thus explorable, a_i and b_i should belong to the attribute range of the i^{th} attribute. However, this will lead to a huge amount of interval patterns. The questions that arise are the following: Can we design efficient algorithms to extract such patterns? Can we reduce the set of patterns to only those of interest w.r.t. a particular need? Is it possible to define condensed representations of such patterns? The next chapter brings first answer elements to these questions by introducing interval pattern structures, from which a concept lattice can be raised efficiently without scaling. The rest of the thesis will focus on those structures by extending their capacity especially in

knowledge discovery.

Chapter 4

Mining interval patterns with FCA

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

1 Introduction

In real-world applications, e.g. in biology or chemistry, one rarely obtains binary data directly, *complex* and *heterogeneous* data involving numbers, graphs, intervals, etc., are more typical. To apply FCA-based methods to such data, the latter have to be binarized, i.e. *scaled*. Many types of scaling are known in FCA literature [47]. Although scaling allows one to apply FCA tools, it faces a trade-off. On one hand, it can come with loss of information (e.g. cutting attribute value domains into several ranges in previous chapter). On the other hand, in the case of complex data such as graph data, they do not always suggest the most efficient implementation right away, and there are situations where one would choose original data representation rather than scaled data [46]. It may accordingly dramatically increase the complexity of computation and representation, and make worse the visualization of results.

Instead of scaling, one may work directly with initial data, i.e. complex object descriptions, defining so-called *similarity operators* which induce a semi-lattice on data descriptions. Several attempts were made for defining such semi-lattices on sets of graphs [46, 69, 70, 79] and logical formulas [31, 45] (see also [49, 126] for FCA extensions). Indeed, if one is able to order object descriptions in complex data, e.g. with graph morphism when objects are described by labelled graphs, one may attempt to directly build a concept lattice from such data. In [46], a general approach called *pattern structures* was proposed, which allows one to apply standard FCA to any partially ordered data descriptions.

This chapter addresses the problem of FCA-based classification of numerical data, where object descriptions are vectors of numbers, with pattern structures and a particular similarity operator. We focus on gene expression data (GED), where *gene expression profiles* represent the

"behaviour" of genes in biological situations, and a situation corresponds to tissues at different time points or cellular loci (different organs, healthy or cancerous tissues, etc.). The example of gene expression data we consider in this chapter is given in Table 1. Let us recall that genes with similar expression profiles are said to be co-expressed. It is now widely accepted that co-expressed genes interact together within the same biological process [117]. GED analysis is an important task and an active area of research involving mainly data-mining methods: clustering [55], biclustering [81, 104]. FCA-based methods have been recently designed and applied in this domain [18, 60, 92].

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

Table 1: Gene expression data

For analysing GEDs by means of FCA, one needs to build a formal context from a GED, attribute values have to be discretized and intervals of entry values have to be considered as binary attributes, implying possible loss of actual data values [60]. In [47], *interordinal scaling* is defined and allows one to build a formal context that encodes all possible intervals of attributes values, without loss of information. However this scaling produces large and dense binary data, which are hard to process with existing FCA algorithms [74]. This is probably one of the reasons why this scaling has never been used for GED analysis. By contrast, the formalism of pattern structures, defined in full compliance with the FCA framework in [46], allows one to build a concept lattice without *a priori* scaling procedure. Accordingly, in this chapter, we introduce an interval convexification as a similarity operator for ordering intervals within a semi-lattice, i.e. by taking the convex hull of any arbitrary set of intervals. However, this operation between complex descriptions of objects may be harder to process than classical set intersection and inclusion test after a scaling. Then, a challenging question arises for numerical data like GEDs: *should one scale numerical attributes*?

To discuss this question, we have experimented with both approaches, comparing their computational efficiency, the respective results and their representations. We show that both methods have equivalent outputs, but the method based on pattern structures is more computationally efficient than that based on interordinal scaling, and provides better readable and interpretable results. Finally, a real world experiment with gene expression data shows data-mining ability of pattern structures for numerical data.

2 Interval patterns in scaled formal contexts

This section starts with the definition of a particular scaling for representing value intervals from numerical datasets called interordinal scaling. The concept lattice is accordingly built from resulting formal context. Each concept represents a set of objects associated to interval of values they take for the different attributes.

	$s_1 \leq 4$	$s_1 \leq 5$	$s_1 \leq 6$	$s_1 \geq 4$	$s_1 \geq 5$	$s_1 \geq 6$
4	×	×	×	×		
5		×	×	×	×	
6			×	×	×	×

Table 2: The interordinal scale $(W_{s_1}, W_{s_1}, \leq)|(W_{s_1}, W_{s_1}, \geq)$.

2.1 Interordinal scaling

Interordinal scaling defined in [47] can help describing all value intervals without loss of information. Let G be a set of genes, S a set of situations, $W \subset \mathbb{R}$ a set of expression values and I_1 a ternary relation defined on the Cartesian product $G \times S \times W$. The fact $(g, s, w) \in I_1$ or simply $g(s) = w$ means that gene g has expression value w for situation s (see for example Table 1). $\mathbb{K}_1 = (G, S, W, I_1)$ is called a many-valued context representing a GED. The objective is to extract formal concepts (A, B) from $\mathbb{K}_1$, where $A \subseteq G$ is a subset of genes sharing "similar values" of W, i.e. lying in a same interval. An appropriate binarization (scaling) technique is used to build a formal context $\mathbb{K}_2 = (G, S_2, I_2)$ called derived context of $\mathbb{K}_1$.

A scale is a formal context (cross-table), objects being the attributes of $\mathbb{K}_1$ and attributes being the derived ones of $\mathbb{K}_2$. As attributes do not take necessarily the same values, each of them is scaled separately. Let $W_s \subseteq W$ be the set of all values of the attribute s. The following interordinal scale (see pp. 42 in [47]) can be used to represent all possible intervals of attribute values:

$$\mathbb{I}_{W_s} = (W_s, W_s, \leq)|(W_s, W_s, \geq).$$

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

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

2.2 Concept lattice construction

The choice of an algorithm to build the concept lattice depends on the size and density of the formal context to process. Density of a formal context (G, M, I) is defined as the proportion of elements of I w.r.t. the size of the Cartesian product $G \times M$, i.e. density $d = |I|/(|G|.|M|)$. In the case of interordinal scaling, density of derived context $\mathbb{K}_2$ is

$$\frac{\sum_{i=1}^{i \leq p}(|W_i| + 1)}{2 \cdot \sum_{i=1}^{i \leq p} |W_i|},$$

where p is the number of attributes in $\mathbb{K}_1$. When $|W|$ grows, d tends towards 50%. Moreover, the number of derived attributes is $2 \cdot \sum_{i=1}^{i \leq p} |W_i|$ and $|g'| = |W| + 1$ for all $g \in G$. This makes the derived contexts dense, large and difficult to process. For comparison, density of binary data in [104] does not exceed 6% and the number of derived attributes remains the same after scaling.

2.3 Interpretation and limits

Consider a concept of the lattice given in Figure 3, e.g.

$$(\{g_1, g_3, g_4, g_5\}, \{s_1 \leq 5, s_1 \leq 6, s_1 \geq 4, s_2 \leq 9, s_2 \geq 7, s_3 \geq 4, s_3 \geq 5, s_3 \leq 8\})$$

The intent of this concept can be interpreted as a so-called interval pattern: it is composed on constraints on a set of values. This means that objects in the extent all have their values for attribute s_1 in the interval $[4, 5]$, for attribute m_2 in interval $[7, 9]$ and for attribute m_3 in interval $[5, 8]$.

A first drawback of interordinal scaling is the form of such intents. One can notice that many constraints are redundant, e.g. the attribute $s_1 \leq 6$ is redundant w.r.t attribute $s_1 \leq 5$. Therefore, the intent should have the following form:

$$\{s_1 \leq 5, s_1 \geq 4, s_2 \leq 9, s_2 \geq 7, s_3 \geq 5, s_3 \leq 8\}$$

It can also simply be represented by a vector of intervals where dimension i corresponds to attribute s_i:

$$\langle [4, 5], [7, 9], [5, 8] \rangle$$

which is more comprehensive.

But beyond hard interpretation, the form of such object description is such that the mining of the context is hard. Indeed, one needs a huge number of binary attributes to described all possible intervals for each attribute. We show in the next section how to extract mathematically equivalent concept without scaling with efficient algorithms.

Let us now consider the whole concept lattice of $\mathbb{K}_2$ given in Figure 3[12]. Concept extents near the Bottom concept contain a few genes, since the corresponding intents are related to the smallest intervals. The extent of the Top concept contains all genes and its intent corresponds to intervals of maximal size. The higher a concept lies in the diagram, the larger is the interval corresponding to its intent. Concepts near the Top are not interesting: they allow almost all possible values of attributes. The problem of selecting the best concepts in GED analysis is addressed latter in biological experiments.

3 Interval patterns in pattern structures

3.1 Intuitions

In this section, we present an alternative to scaling when a context includes many-valued attributes. This alternative is based on the idea of *pattern structures* [46] which was motivated by research on learning with labelled graphs and other complex descriptions [69, 70].

Intuitively, the *similarity* of two sets of labelled graphs X and Y, denoted by $X \sqcap Y$, is given by the maximal common subgraphs of graphs from X and Y. Then a *graph pattern* may be defined as a set of graphs X such that $X \sqcap X = X$, i.e. X is "maximal" w.r.t. the similarity

[12] Drawn with the Concept Explorer software (http://conexp.sourceforge.net/)

$s_3 \geq 8$				×	
$s_3 \geq 6$	×			×	
$s_3 \geq 5$	×		×	×	×
$s_3 \geq 4$	×	×	×	×	×
$s_3 \leq 8$	×	×	×	×	×
$s_3 \leq 6$	×	×	×		×
$s_3 \leq 5$		×	×		×
$s_3 \leq 4$		×			
$s_2 \geq 9$				×	
$s_2 \geq 8$		×	×	×	×
$s_2 \geq 7$	×	×	×	×	×
$s_2 \leq 9$	×	×	×	×	×
$s_2 \leq 8$	×	×	×		×
$s_2 \leq 7$	×				
$s_1 \geq 6$		×			
$s_1 \geq 5$	×	×			×
$s_1 \geq 4$	×	×	×	×	×
$s_1 \leq 6$	×	×	×	×	×
$s_1 \leq 5$	×		×	×	×
$s_1 \leq 4$			×	×	
	g_1	g_2	g_3	g_4	g_5

Table 3: Interordinally scaled context $\mathbb{K}_2 = (G, S, I_2)$.

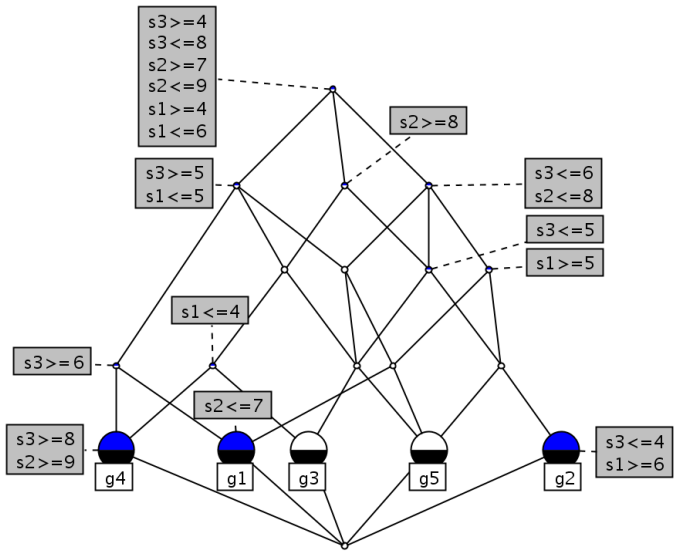


Figure 1: Concept lattice of formal context $\mathbb{K}_2 = (G, S, I_2)$.

operation. It is easily seen that the operation $\sqcap$ is idempotent, associative and commutative. The similarity operation $\sqcap$ on sets of graphs is a sort of "attribute sharing", as in the binary case, where objects in extent share the maximal set of attributes in the corresponding intent. Denote by D the set of all graph patterns, then $(D, \sqcap)$ is a semi-lattice with infimum (meet) operator $\sqcap$. A natural subsumption order on graph patterns is given by $X \sqsubseteq Y \Leftrightarrow X \sqcap Y = X$.

More generally, a *pattern structure* is a triple $(G, (D, \sqcap), \delta)$ where G is a set of objects, $(D, \sqcap)$ is a meet-semi-lattice of object descriptions or *patterns*, and $\delta : G \longrightarrow D$ is a mapping providing any object $g \in G$ with a description $d \in (D, \sqcap)$. As $(D, \sqcap)$ or equivalently $(D, \sqsubseteq)$ are semi-lattices, the following Galois connection, denoted by $\{(.)^{\square}, (.)^{\square}\}$, between $(2^G, \subseteq)$ and $(D, \sqsubseteq)$ gives rise to a complete lattice called the *pattern concept lattice* of $(G, (D, \sqcap), \delta)$ [46].

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

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

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

3.2 Similarity between intervals

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

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

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

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

$$\begin{aligned} & [a_1, b_1] \sqsubseteq [a_2, b_2] \\ \Leftrightarrow\ & [a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \\ \Leftrightarrow\ & [min(a_1, a_2), max(b_1, b_2)] = [a_1, b_1] \\ \Leftrightarrow\ & a_1 \leq a_2 \quad and \quad b_1 \geq b_2 \\ \Leftrightarrow\ & [a_1, b_1] \supseteq [a_2, b_2]. \end{aligned}$$

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

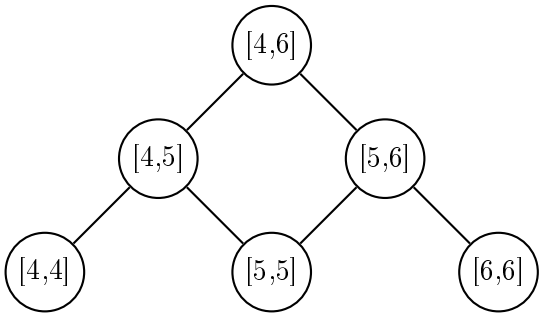


Figure 2: Diagram of $(D_{m_1}, \sqcap)$ or equivalently$(D_{m_1}, \sqsubseteq)$.

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

3.3 Similarity between interval vectors

We call an *interval vector* a p-dimensional vector of intervals. When e and f are vectors of p intervals, we write $e = \langle [a_i, b_i] \rangle_{i \in [1,p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$. The similarity operation $\sqcap$ is defined

by the meet of corresponding components for vector of the same size (knowing that the order of the components is canonical):

$$\begin{aligned} & e \sqcap f = \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & \Leftrightarrow e \sqcap f = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]}. \end{aligned}$$

Therefore, interval vectors are partially ordered by:

$$\begin{aligned} & e \sqsubseteq f \\ & \Leftrightarrow \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & \Leftrightarrow [a_i, b_i] \sqsubseteq [c_i, d_i], \ \forall i \in [1, p].i \in [1, p], \end{aligned}$$

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

3.4 Concept lattice construction

GED in Table 1 can be formalized as a pattern structure $(G, (D, \sqcap), \delta)$ where $G = \{g_1, \ldots, g_5\}$ and D is a set of interval vectors or 3-dimensional vectors, where each component corresponds to an attribute of the table. For example, $\delta(g_1) = \langle [5,5], [7,7], [6,6] \rangle$, where $[a, a]$ stands for any $a \in \mathbb{R}$. When $A \subseteq G$ is a set of objects and $d \in (D, \sqcap)$ is an interval vector, $A^{\square}$ returns an interval vector composed, for each dimension, of the smallest interval containing all intervals in the description of each object in A, i.e. their convex hull. On the other hand, $d^{\square}$ returns the set of objects being described for each dimension by an interval included in the corresponding interval of d.

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

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

$$\begin{aligned} \langle [5,6], [7,8], [4,6] \rangle^{\square} &= \{g \in G | \langle [5,6], [7,8], [4,6] \rangle \sqsubseteq \delta(g)\} \\ &= \{g_1, g_2, g_5\} \end{aligned}$$

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

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

3.5 Algorithms for computing interval patterns

Many algorithms for generating formal concepts from a formal context are compared in [74]. Experimental results highlight *Norris*, *CloseByOne* and *NextClosure* algorithms as the best algorithms when the context is dense and large, which is the case of interordinally derived formal

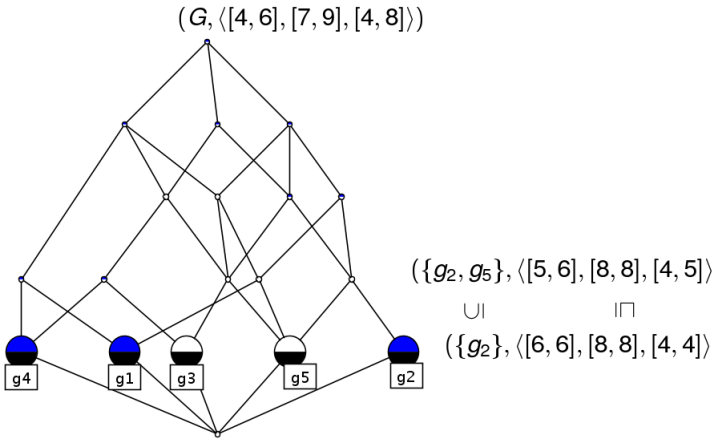


Figure 3: Pattern concept lattice of pattern structure from Table 1.

contexts. Worst-case upper bound time complexity of the three algorithms for computing a set of formal concepts from a formal context (G, M, I) is $O(|G|^2 \cdot |M| \cdot |L|)$ with G the set of genes, M the set of attributes (here the set of attributes of the scaled context), and L the set of generated concepts.

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

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

We now propose an adaptation of the *CloseByOne* algorithm for processing pattern structures such as vectors of intervals. This algorithm detailed in Chapter 2 is the most efficient in our case (see Subsection 4.4.2). To adapt this algorithm for pattern structures, one has to replace each call to a $(.)'$ operator by a call to the corresponding $(.)^{\square}$ operator. Then, computing $A^{\square}$ for a set $A \subseteq G$ is realized by taking min (respectively max) of all left (respectively right) limits of the intervals of each object description. For a pattern $d \in (D, \sqcap)$, $d^{\square}$ is computed by testing for each object $g \in G$ if each interval of its description is included in the corresponding interval of d.

4 Comparing both approaches

4.1 Theoretical comparison

The following proposition establishes an isomorphism between the concept lattice of $\mathbb{K}_I$ with the relation $\mathbb{I}_{W_s} = (W_s, W_s, \leq) | (W_s, W_s, \geq)$, resulting from the interordinal scaling, and the pattern concept lattice of $(G, (D, \sqcap), \delta)$.

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

1. A is an extent of the pattern structure $(G, (D, \sqcap), \delta)$ and $A^{\square} = \langle [\underline{m}_i, \overline{m}_i] \rangle_{i \in [1,p]}$, where $\underline{m}_i$ and $\overline{m}_i$ respectively denote the minimum and maximum of values of the objects in A for the i^{th} attribute.

2. A is a concept extent of the context $\mathbb{K}_I$ so that for all $i \in [1,p]$ $\underline{m}_i$ is the largest number n such that the attribute $s_i \geq n$ is in A' and $\overline{m}_i$ is the smallest number n such that the attribute $s_i \leq n$ is in A'.

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

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

where

$$n_{\min} \prec \ldots \prec n_1 \leq n_2 \prec \ldots \prec n_{\max}$$

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

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

4.2 Practical comparison

Here we compare time performance of three algorithms for mining pattern structures of interval vectors and equivalent interordinally scaled contexts. We have implemented the *Norris*, *NextClosure*, and *CloseByOne* algorithms, for both processing formal contexts and pattern structures. We have added the Charm algorithm [53] that extracts closed itemsets, i.e. concept intents in a formal context. FCA algorithms have been implemented in original versions as described in [74]. These algorithms are run within the Coron System [120].[13] All implementations are in Java: sets of objects and binary attributes are described with the BitSet class and interval descriptions with standard double arrays. The experiments were carried out on an Intel Core2 Quad CPU 2.40 Ghz machine with 4 GB RAM running under Ubuntu 8.10.

We began to compare algorithms on the data presented in biological experiments, i.e. from a many-valued context (G, S, W, I_1) where $|G| = 10,225$ and $|S| = 5$ (see next Section for more biological details). Even by reducing the number of attribute values, computation is infeasible. Indeed we do not consider here constraints like the maximal interval size. Then we randomly selected samples of the data, by increasing the number of objects. As attribute values are real numbers with about five digits after the comma, the size of W is large. In the worst case, $|W| = |G| \times |S|$, i.e. each attribute value is different in the dataset. This implies very large formal contexts to process and a large number of concepts. The execution times for this case are shown in Table 4. The *Norris* algorithm shows the best results in formal contexts, meeting conclusions of [74] for large and dense contexts. However, *CloseByOne* performs better for pattern structures, and most importantly is the only one able to compute a very large collection of concepts.

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

Datasets							
$\|G\|$	10	20	30	40	50	75	100
$\|W\|$	50	100	150	199	249	374	252
density	51.00%	50.50%	50.33%	50.25%	50.20%	50.13%	50.20%
Generation time in formal contexts (in milliseconds)							
Charm	60	916	16,469	N/A	N/A	N/A	N/A
Next Closure	5	145	1,299	12,569	68,969	N/A	N/A
Norris	**2**	**90**	**609**	**5,180**	**28,831**	N/A	N/A
Close By One	3	106	906	7944	41,238	N/A	N/A
Generation time in pattern structures (in milliseconds)							
Next Closure	6	100	763	5,821	35,197	N/A	N/A
Norris	6	172	1982	15,522	83,837	N/A	N/A
Close By One	**2**	**85**	**585**	**3,094**	**18,320**	**1,004,073**	**2,288,200**
Concept set L							
$\|L\|$	280	9,587	78,173	455,008	1,857,725	40,325,176	64,571,385

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

When strongly reducing the size of W by rounding attribute values to the integer, i.e. $|W| \ll |G| \times |S|$, the *Charm* algorithm outperforms the others. The *Norris* algorithm is still the best FCA-algorithm in formal contexts and *CloseByOne* is the best in pattern structures (see Table 5).

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

5 Biological experiments

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

5.1 Data

Biologists at the UMR IAM (INRA) study interactions between fungi and trees. They published the complete genome sequence of the fungus *Laccaria bicolor* [83]. This fungus lives in symbiosis with many trees of boreal and temperate forests. The fungus forms a mixed organ on tree roots and is able to exchange nutrients with its host in a specific symbiotic structure called ectomycorrhiza, contributing to a better tree growth and enhancing forest productivity. On the other hand, the plant repays its symbiotic partner by providing carbohydrates, allowing the fungus to complete its biological cycle by producing fruit-bodies (e.g. mushrooms). It is thus of major interest to understand how the symbiosis performs at the cellular level. The genome sequence of *Laccaria bicolor* contains more than 20,000 genes [83]. The study of their expression in various biological situations helps to understand their roles and functions in the

Datasets							
$\|G\|$	25	50	75	100	125	150	200
$\|W\|$	34	37	44	53	58	62	66
Generation time for formal contexts (in milliseconds)							
density	51.47%	51.35%	51.14%	50.94%	50.86%	50.81%	50.76%
Charm	55	**154**	**184**	**243**	**394**	**936**	**1856**
Next Closure	100	933	3,333	22,973	30,854	78,790	593,416
Norris	**38**	320	861	2,697	5,954	15,359	46,719
Close By One	84	483	2,424	8,452	22,173	59,070	227,432
Generation time for pattern structures (in milliseconds)							
Next Closure	59	372	1,924	6,215	15,417	42,209	143,501
Norris	44	479	2,602	7,243	16,257	40,991	109,814
Close By One	**40**	**220**	**1,084**	**3,832**	**9,289**	**23,989**	**89,804**
Concept set L							
$\|L\|$	1,165	5,928	23,962	48,176	73,463	163,316	252,515

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

biology of the fungus. Microarray techniques enable to compare expression values of all the genes between contrasted situations like free-living cells of the fungus (i.e. mycelium), cells engaged in the symbiotic association (i.e. ectomycorrhiza), and specialized cells forming the fruit-body structure (i.e. mushroom). *Laccaria bicolor* gene expression data is available at the Gene Expression Omnibus of the National Center for Biotechnology Information (NCBI)[14]. It is composed of 22,294 genes in lines and 5 various biological situations in columns, reflecting cells of the organism in various stages of its biological cycle, i.e. free living mycelium (situation FLM), symbiotic tissues (situations MP and MD) or fruiting bodies (situations FBe and FBl).

5.2 Preprocessing

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

Filtering the genes consists in removing genes having no significant difference of expression across all situations. For each couple of situation, a t-test is performed and a p-value is attributed. If the p-value > 0.05 (cut-off classically applied in biology) for all couples of situations then the current gene is removed from the dataset. The CyberT tool[15] was used to filter the dataset and obtain $11,930$ genes. Another classical pre-processing in GED analysis is to transform expression values using log_2. Indeed, it allows the capture of small expression values into intervals that should be larger for high expression values. Finally, for making computation possible, a last pre-processing consists in rounding log_2 expression values to one digit after the comma, recalling that the more there are different attribute values, the more they are concepts.

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

5.3 Method

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

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

The *CloseByOne* algorithm was run on the resulting pattern structure with $max_{size} = 0.35$. A concept is retained if it describes at least 7 co-expressed genes in at least 5 situations, i.e. the intent has at least 5 intervals whose size do not exceed the max_{size} parameter. Indeed, let us recall that concepts near the Bottom, i.e. in the lowest levels of the concept lattice, are composed of a few genes described by small intervals. Processing time was about 2 minutes and returns 2,120 concepts (hardware details are given in next section).

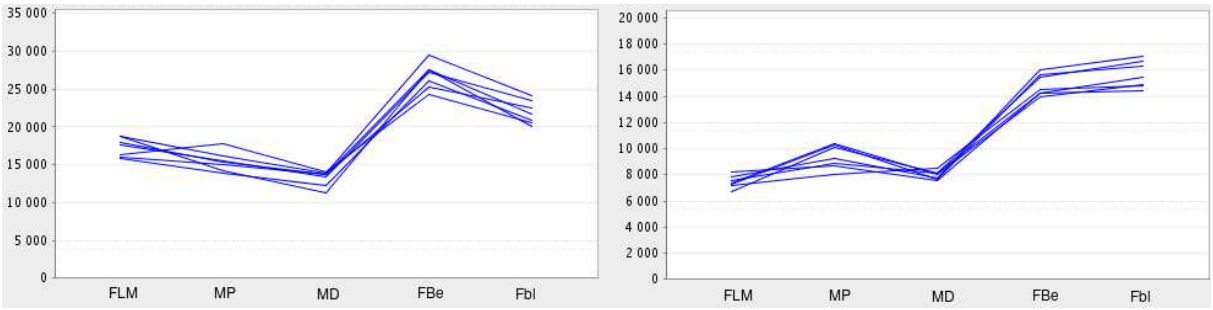


Figure 4: Graphical visualisation of two extracted concepts.

5.4 First results

Here we present two extracted patterns selected as grouping genes with high expression levels in the fruit-bodies situations, whereas their expression remains similar between the mycelium

and symbiosis situations. In Figure 4, X-axis is composed of situations, Y-axis is the expression values axis. Each line denotes the expression profile of a gene in the concept extent. Values are taken before the logarithmic transformation. These patterns have been extracted from the whole list of 2,120 patterns for the following characteristic: in both cases, the expression levels measured are about two times higher in the fruit-body compared to the other situations. It indicates that these genes correspond to biological functions of importance at this stage. The expression measured in the mycelium and symbiosis situations tends to indicate that these genes are also involved in general cellular processes as they are already expressed in all situations.

The pattern in Figure 4 (left) contains 7 genes, of which only 3 possess a putative cellular function assignment based on similarity in international gene databases at NCBI. Interestingly, these genes all encode enzymes involved in distinct metabolic pathways. A gene encodes a 1-pyrroline-5-carboxylate dehydrogenase which is involved in amino-acid metabolism, another corresponds to an acyl-coA dehydrogenase, involved in fatty acid metabolism and a last gene encodes a transketolase, an enzyme involved in the pentose phosphate pathway of carbohydrate metabolism. All these metabolic functions are essential for the fungus and reflect that the fruit-body is a highly active tissue. The fruit-body is a specific fungal organ that differentiate in order to produce spores and that further ensure spore dispersal in nature [108]. Previous gene expression analyses of the fruit-body development conducted in the ectomycorrhizal fungus *Tuber borchii* also reported the strong induction of several genes involved in carbon and nitrogen metabolisms [54] as well as in lipid metabolism [110]. The present results are consistent with these observations and supports an important mobilization of nutrient sources from the mycelium to the fruit-body. It seems obvious that the primary metabolism requires to be adapted to use these sources in order to properly build spores and provide spore-forming cells with nutrients [108].

The pattern on Figure 4 (right) also contains 7 genes, of which only 3 possess a putative biological function. Interestingly, one of these genes encodes one pseudouridylate synthase, an enzyme involved in nucleotide metabolism that might also be involved in remobilization of fungal components from the mycelium to spore-forming cells and spores. The 2 other genes encode a cytoskeleton protein (actin) and a protein related to autophagy (autophagy-related 10 protein), a process that can contribute to the recycling of cellular material in developing tissues. Both functions participate in re-constructive cellular processes [108], which is consistent with the involvement of metabolic enzymes in remobilization of fungal resources towards the new organ in development.

Analysis of these two patterns that present a high expression level in the fruit-body situation is highly informative, confirms existing knowledge in the field and highlights the importance of remobilization in the developing organ. These co-expressed genes share related roles in a particular process. This could indicate that they are under the control of common regulators of gene expression. Interestingly, these patterns also contained a total of 8 genes of unknown functions, i.e. for which no functional assignment was possible in international gene databases. There were 4 genes encoding hypothetical proteins with a homology in databases but no detailed function and 4 genes not previously described in fungi or other organism and which are considered specific to *Laccaria bicolor*. There are about 30% of such genes specific to this fungus and these may play specific roles in the biology of this soil fungus [83]. All these genes show consistent profiles with those encoding metabolic functions. Thus, these genes are interesting investigation leads as they may contain new enzymes not previously described of the pathways or eventual regulator of the cellular process. Altogether, these results contribute to a better understanding of the molecular processes underlying the fruit-body development. As stated earlier, the expression of these genes was not specific to this biological situation. Their expression levels was already high in the mycelium and the symbiotic tissue indicating that these processes are essential not

only to the fruit-body development but also to general cellular processes as previously described in expression studies of the tree-fungus symbiosis development [109].

6 Discussion

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

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

Indeed, our FCA based approach can be viewed as a biclustering method. It provides means for extracting patterns from numerical data, namely *formal concepts*. In application to GED analysis, concept extents are maximal sets of genes related to a common maximal set of situations (not necessarily all, due to our constraints on maximal interval size). The ordering of concepts among a complete lattice makes overlapping of concepts natural. Then a complete enumeration of patterns respecting some constraints like maximal interval size is possible. Indeed, the subsets of patterns satisfying these constraints is an order ideal of the lattice of patterns. Actually, in this chapter, we pay particular attention to scaling problems, such as boundary problems, and we proposed monotone constraints to retain best concepts for a GED analysis.

A similar work to build concept lattices from numerical data was proposed in [102] in the framework of Symbolic Data Analysis (SDA) [17], however no links with interordinal scaling and efficiency comparison was proposed.

Among other directions of further research, one may involve domain knowledge. The semi-lattice of descriptions $(D, \sqcap)$ may be viewed as a hierarchy, where domain knowledge may be encoded, e.g. in some dimensions of a pattern vector. Domain knowledge can be given by text annotations on genes, e.g. [90], for which a similarity operation $\sqcap$ can be defined. Moreover, each dimension of the vector may correspond to a particular data-type for which a similarity operation $\sqcap$ is defined. For example, some dimension may corresponds to numerical attributes, an other to graph-valued attributes, or classical sets, etc.

In this chapter, we do not have considered fuzzy settings. Although FCA has already been extended in [12] where an object is associated to an attribute with a truth degree, it can be interesting to study how fuzzy settings can be considered within pattern structures. A first study we addressed can be found in [6] and is not detailed here.

Most importantly, considering the similarity operation $\sqcap$ as interval convexification generates too many patterns: some patterns and their sub-patterns w.r.t. $\sqsubseteq$ may describe almost the same set of genes, i.e. a few genes differs in their extents. Concept stability was introduced in [72]

for measuring this phenomena. In this chapter, we solved the problem of un-interesting patterns thanks to a monotone constraint. In the next chapter, we extend this proposition and show how to embed a tolerance relation in an interval pattern structure to produce only concepts with similar objects, w.r.t. a distance on their values.

Chapter 5

Introducing a similarity relation between numerical values

1 Introduction

In the framework of formal concept analysis, a concept lattice is derived from a formal context. Thanks to a Galois connection, a concept represents a maximal set of objects associated with their common attributes: the intent of a concept represents the set of attributes the objects in the extent have in common. This statement can be expressed as follows: the intent represents the attributes for which the objects in the extent are similar.

When facing numerical data, valued either with number or intervals, the latter have to be scaled to be in adequate form. However, it follows from previous statement that classifying objects having similar attribute values within same concepts may be thought as a more natural way. In that sense, authors of [86, 87, 88] defined *FCA guided by Similarity* denoted by FCAS in this chapter. They propose to consider a similarity relation between "numerical" objects to directly build the concept lattice, i.e. without scaling. Intuitively, two objects are similar if the difference of their value (either a number or an interval of number) does not exceed a given parameter for each attribute, e.g. $[2, 4] \simeq_\theta [4, 8]$ means that both values are similar with a parameter $\theta = 6$. This leads to the original notion of attribute sharing: two objects share the attributes for which the values they take are similar. Quite naturally, this similarity relation is not transitive and raises a problem for ordering concepts. The authors propose to consider a pairwise similarity of objects instead, and give applications to biological resource retrieval on the web. However, the associated theory provides no efficient algorithm at present.

On another hand, in the previous chapter, pattern structures have been used to build a concept lattice directly from numerical data, also avoiding scaling. So-called *interval pattern structures* (IPS) relies on a theory in full compliance with FCA and thus benefits of its "tool-box" including efficient algorithms. However, the notion of similarity of objects is complex and different from the intuitive one used in FCAS: it relies on a similarity operator $\sqcap$ and associated subsumption relation $\sqsubseteq$ between object descriptions, e.g. $[2, 8] \sqsubseteq [4, 8]$. The so-called similarity operator $\sqcap$ gives the description representing the similarity of some object description.

Whereas those two methods (FCAS and IPS) use different notion of similarity, this chapter holds on a study of the relations between them, extending the classification ability of FCA for dealing with objects with many-valued attributes in an original way. Actually, the parallel study of FCAS and IPS helps to understand how these two methods are interrelated and how they can be applied to complex data for building concept lattices. IPS uses a framework in full compliance

with FCA with efficient and scalable algorithms. In turn, FCAS brings an intuitive notion of similarity and helps understanding the resulting concept lattices.

After showing that FCAS can be expressed in terms of pattern structures, a natural question arises. Can we design a scaling procedure leading to a context whose concept lattice is isomorphic to the pattern concept lattice? In others words, can we define a scaling procedure leading to a formal context whose concepts are maximal sets of pairwise similar objects? We answered this question in the previous chapter showing that IPS outperforms classical FCA on interordinal scaled contexts. However, the notion of similarity relation on numerical values was not taken into account. Accordingly, we show how to defined this scaling. This scaling relies on the formalization of similarity by a tolerance relation, providing concepts with an adequate semantic, namely *tolerance classes*.

Finally, an experiment with real-world agronomic data supports the notions discussed in this chapter and addresses the problem of decision helping in agricultural practices.

2 FCAS: FCA guided by similarity

FCAS is an FCA based method allowing to build a concept lattice from complex data without scaling and considering similarity between objects from a many-valued context [86, 87]. Table 1 shows the kind of contexts we are interested in: contexts (G, M, W, I) such as attribute values in W are intervals of numbers or simply numbers. Firstly we recall an intuitive similarity between intervals and the problem it sets. Then, pairwise similarity is shown to be a interesting solution and is used to define the Galois connection to build a concept lattice.

	m_1	m_2	m_3
g_1	$[2, 4]$	$[25, 29]$	0.3
g_2	$[4, 8]$	19	0.1
g_3	$[10, 15]$	29	0.5
g_4	$[9, 13]$	17	0.5
g_5	$[8, 13]$	$[17, 19]$	0.3
g_6	$[9, 15]$	$[14, 19]$	$[0.5, 0.7]$

Table 1: Interval data

2.1 Similarity between intervals

In FCA, a set of objects A possesses an attribute m iff any single object of A possesses m. When objects are described by numbers or intervals, the sharing is not straightforward and requires scaling procedure to obtain a formal context. By contrast, usual intuition calls for a classical similarity between numbers or intervals: a set of objects possesses an attribute iff all their values are similar for this attribute. In other words, two values are similar if their difference is not significant. Formally, given $[\alpha_i, \beta_i]$ and $[\alpha_j, \beta_j]$ two intervals of real numbers, and θ a similarity threshold, the two intervals are said to be similar iff:

$$[\alpha_i, \beta_i] \simeq_\theta [\alpha_j, \beta_j] \Leftrightarrow max(\beta_i, \beta_j) - min(\alpha_i, \alpha_j) \leq \theta$$

The similarity threshold θ expresses the maximal variation allowed between two similar intervals and reflects the precision requirements to be considered during the analysis of data. For example, with $\theta = 6$, $[2, 4] \simeq_\theta [4, 8]$ but $[2, 4] \not\simeq_\theta [9, 13]$ whereas for $\theta = 11$ the three intervals are

similar. It is important to notice that the similarity operator $\simeq_\theta$ is not transitive: with $\theta = 9$, $[2,4] \simeq_\theta [4,8]$, $[4,8] \simeq_\theta [9,13]$ but $[2,4] \not\simeq_\theta [9,13]$.

2.2 Similarity between objects

FCAS introduces the notion of object similarity as follows.
– Two objects g_1 and g_2 share an attribute m iff $m(g_1) \simeq_\theta m(g_2)$. θ may be different for each attribute, as attributes may have a different domain of values.
– A set of objects $A \subseteq G$ shares an attribute m whenever *any pair* of objects in A shares m. This is why it is called a pairwise similarity of objects.
– A set of objects $A \subseteq G$ shares a set $B \subseteq M$ of attributes whenever any pair of objects in A shares all attributes in B. Then A is said to be *valid* w.r.t. B.

When a set of objects shares a set of attributes, objects are pairwise similar w.r.t. this set of attributes. For example, if we consider $\theta = 6$ for attribute m_1, $\theta = 4$ for attribute m_2, and $\theta = 0.2$ for attribute m_3, then objects in $\{g_3, g_4, g_6\}$ are pairwise similar w.r.t. m_1 and m_3: they share the attributes m_1 and m_3. This means that each pair of objects has similar values for attributes m_1 and m_3. For the attribute m_1 this means that $m_1(g) \simeq_\theta m_1(h)$ for any $g, h \in \{g_3, g_4, g_6\}$, e.g. $m_1(g_3) \simeq_\theta m_1(g_6)$.

From these statements, a Galois connection can be defined. A first operator associates to a set of objects the set of attributes they share and for each of these attributes, the interval of values containing all of them (this is required to order attributes). As a result, this operator gives a set of pairs (attribute,interval). Dually, the second operator associates to a set of pairs, the maximal set of objects that share attributes from pairs in this set. These operators are detailed later.

2.3 Maximal sets of pairwise similar objects

In spirit of FCA, it is important to determine maximal sets of pairwise similar objects. This corresponds to the notion of closed sets (on which relies the definition of a concept). As in classical FCA, one has to characterize maximal sets of objects sharing maximal sets of attributes. For example, $\{g_3, g_6\}$ is valid, as well as $\{g_3, g_4, g_6\}$ for the same attributes m_1 and m_3. This very last set only will determine a formal concept, as being a maximal set of objects similar for both m_1 and m_3.

Starting from a set of objects, the idea to obtain its maximal set of pairwise similar objects is the following. Given a set of objects A, one should (i) search for all objects similar with all objects in A, (ii) remove all pairs of objects that are not pairwise similar, and finally (iii) build the description of remaining objects, i.e. an interval needed for the Galois connection. (i) and (ii) can be seen as a closure in mathematical morphology, consisting in (i) a dilatation and (ii) an erosion by a structuring element characterizing θ [114].

(i) Set of reachable objects. Given an interval context (G, M, W, I), $g_i \in G$ *reaches* $g_j \in G$ w.r.t. $m \in M$ whenever $m(g_i) \simeq_\theta m(g_j)$. The set of all *reachable* objects from a valid set of objects $A \subseteq G$ w.r.t. m is defined as follows:

$$\mathfrak{R}(A, m) = \{g_i \in G \mid m(g_i) \simeq_\theta m(g),\ \forall g \in A\}$$

The set of reachable objects from A w.r.t. $B \subseteq M$ is: $\mathfrak{R}(A, B) = \bigcap_{m \in B} \mathfrak{R}(A, m)$. Considering the interval context in Table 1 and a threshold $\theta = 0.2$ for attribute m_3, then $\mathfrak{R}(\{g_1\}, m_3) = \{g_1, g_2, g_3, g_4, g_5\}$. This set of objects is not valid with respect to m_3 because $m(g_2) \not\simeq_\theta m(g_3)$

and $m(g_2) \not\simeq_\theta m(g_4)$. Actually, this is due to the fact that in the general case, the set of objects $\mathfrak{R}(A,m)$ may not be valid w.r.t. m because of the non transitivity of $\simeq_\theta$.

(ii) Maximal valid set of reachable objects. The maximal valid set of objects containing A is the subset of $\mathfrak{R}(A,m)$ obtained by removing from $\mathfrak{R}(A,m)$ all pairs of objects which do not share m (i.e. g_i, g_j such that $m(g_i) \not\simeq_\theta m(g_j)$). Formally this set is defined as follows:

$$\mathfrak{R}_v(A,m) = \mathfrak{R}(A,m) \setminus \{g_i, g_j \mid m(g_i) \not\simeq_\theta m(g_j)\}.$$

The maximal valid set containing A w.r.t. $B \subseteq M$ is: $\mathfrak{R}_v(A,B) = \bigcap_{m\in B} \mathfrak{R}_v(A,m)$. In the example, $\mathfrak{R}_v(\{g_1\},m_3) = \{g_1,g_5\}$ (i.e. obtained from $\mathfrak{R}(\{g_1\},m_3)$ by removing g_2, g_3, and g_4).

(iii) Description of a maximal valid set of objects. When $A \subseteq G$ shares an attribute $m \in M$ ($\mathfrak{R}(A,m) \neq \varnothing$) then $A \subseteq \mathfrak{R}_v(A,m)$ and $\mathfrak{R}_v(A,m)$ shares m. The interval describing the set $\mathfrak{R}_v(A,m)$ is given by:

$$\gamma(A,m) = [min(\alpha_i), max(\beta_i)] \text{ for } [\alpha_i,\beta_i] = m(g_i), g_i \in \mathfrak{R}_v(A,m)$$

When a set of objects A shares an attribute m for a threshold θ, then we say that A shares $(m,\gamma(A,m))$. For example, $\{g_1,g_2\}$ shares $(m_1,[2,8])$ for a threshold $\theta = 6$. When A is not valid w.r.t. m then $\gamma(A,m) = \varnothing$. Indeed, consider $\theta = 6$ and the attribute m_1. The objects g_1 and g_2 share m_1. The objects g_3 and g_4 share m_1. However g_1, g_2, g_3, and g_4 do not share m_1. This means that an object description, has to be composed of pairs: the first value gives an attribute name while the second provides with its value.

2.4 Building the concept lattice

In [86], it is shown that the two following operators form a Galois connection between 2^G and the partially ordered set $(M \times \mathcal{I}_\Theta, \sqsubseteq)$. $\mathcal{I}_\Theta$ is the set of all intervals possibly returned by the function γ. $\sqsubseteq$ orders pairs (attribute,interval) by inclusion of intervals of same attributes. With $A \subseteq G$ and $B \subseteq M \times \mathcal{I}_\Theta$:

$$A^\uparrow = \{(m,\gamma(A,m)) \in M \times \mathcal{I}_\Theta \mid \gamma(A,m) \neq \varnothing\}$$
$$B^\downarrow = \mathfrak{R}_v(\{g \in G \mid \forall (m,[\alpha,\beta]) \in B,\ m(g) \simeq_\theta [\alpha,\beta]\}, B)$$

$A^\uparrow$ is the set of attributes shared by all the objects in A and $B^\downarrow$ is the set of objects sharing all attributes in B. We illustrate these operators on our example, with resp. $\theta = 6$, $\theta = 4$ and $\theta = 0.2$ for resp. attributes m_1, m_2 and m_3:

$$\{g_3,g_6\}^\uparrow = \{(m_1,[9,15]),(m_3,[0.5,0.7])\}$$
$$\{(m_1,[9,15]),(m_3,[0.5,0.7])\}^\downarrow = \{g_3,g_4,g_6\}.$$

The pair $(A,B) = (\{g_3,g_4,g_6\},\{(m_1,[9,15]),(m_3,[0.5,0.7])\})$ is a concept as $A^\uparrow = B$ and $A = B^\downarrow$. The set of all concepts classically ordered by $(A_1,B_1) \leq (A_2,B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \sqsubseteq B_1)$ generates a complete lattice, e.g. in Figure 1. Reading the extent of a concept remains as stated earlier with reduced labeling. This is not the case for intents, as an attribute can take on several values: each concept intent is given separately.

3 IPS: Interval pattern structures

This section recalls the interval pattern structure approach presented in Chapter 4. Only the most important facts are recalled here for making the comparison with FCAS easier. Firstly, we recall how the similarity operator $\sqcap$ is defined for numerical data, and then how the Galois connection of pattern structure is illustrated.

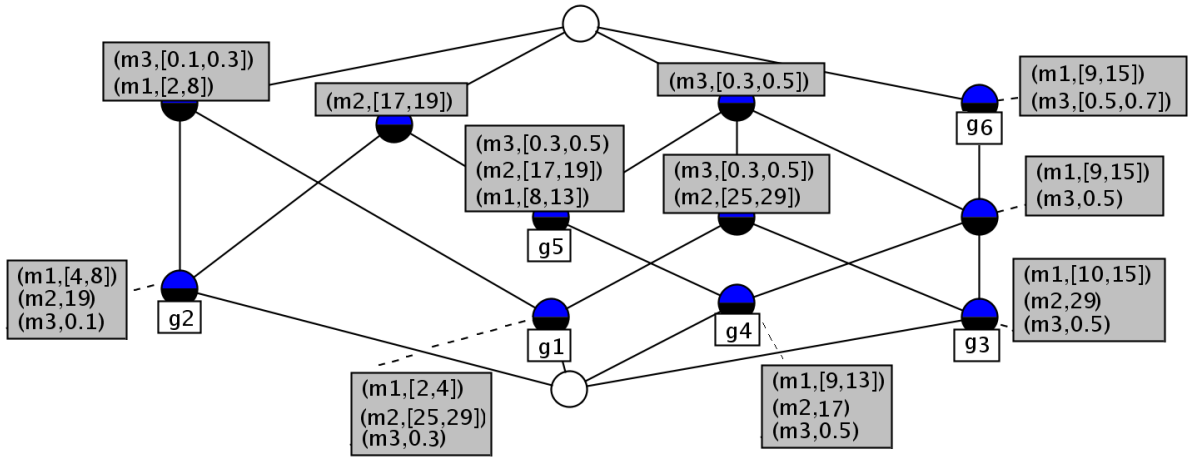


Figure 1: Interval concept lattice raised from Table 1 with FCAS

3.1 Similarity between intervals

Intervals are patterns: they may be ordered within a meet-semi-lattice making them potential object descriptions. The meet $\sqcap$ of two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ is: $[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$, i.e. the largest interval containing them. Indeed, when c and d are intervals, $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ holds:

$$\begin{array}{rcrcl} [a_1, b_1] \sqsubseteq [a_2, b_2] & \Leftrightarrow & [a_1, b_1] \sqcap [a_2, b_2] & = & [a_1, b_1] \\ & \Leftrightarrow & [min(a_1, a_2), max(b_1, b_2)] & = & [a_1, b_1] \\ & \Leftrightarrow & a_1 \leq a_2 & \text{and} & b_1 \geq b_2 \\ & \Leftrightarrow & [a_1, b_1] & \supseteq & [a_2, b_2]. \end{array}$$

This definition means that, contrarily to intuition, smaller intervals subsume larger intervals containing them, and that the meet of n intervals is the smallest interval containing all of them. Figure 2 gives an example of meet-semi-lattice of intervals. The interval labelling a node is the meet of all intervals labelling its ascending nodes, e.g. $[0.1, 0.5] = [0.1, 0.3] \sqcap [0.3, 0.5]$, and is also subsumed by these intervals, e.g. $[0.1, 0.5] \sqsubseteq [0.3, 0.5]$. In other words, if $[a_2, b_2] \subseteq [a_1, b_1]$ then $[a_1, b_1] \sqsubseteq [a_2, b_2]$; but if $[a_2, b_2] \not\subseteq [a_1, b_1]$ then $[a_1, b_1] \sqcap [a_2, b_2]$ returns the largest interval containing both $[a_1, b_1]$ and $[a_2, b_2]$.

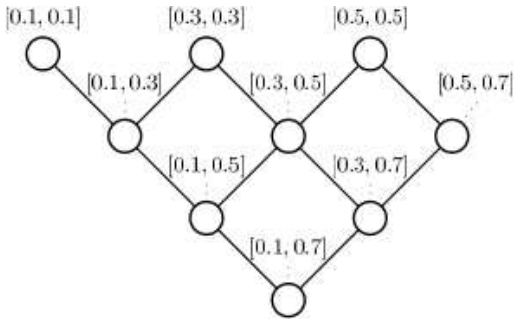


Figure 2: A meet-semi-lattice of intervals.

3.2 Similarity between objects

As objects are generally described by several intervals, each one standing for a given attribute, *interval vectors* have been introduced as p-dimensional vector of intervals. When e and f are interval vectors, we write $e = \langle [a_i, b_i] \rangle_{i \in [1,p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$. Interval vectors are patterns: they may be partially ordered within a meet-semi-lattice. Indeed, the similarity operation $\sqcap$ and consequently subsomption relation $\sqsubseteq$ are given by:

$$\begin{array}{rcl} e \sqcap f & = & \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & = & \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]} \end{array} \qquad \begin{array}{rcl} e \sqsubseteq f & \Leftrightarrow & \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & \Leftrightarrow & [a_i, b_i] \sqsubseteq [c_i, d_i],\ \forall i \in [1, p] \end{array}$$

These definitions state that computing $\sqcap$ (resp. testing $\sqsubseteq$) for interval vectors results in computing $\sqcap$ (resp. testing $\sqsubseteq$) between intervals of each dimension, e.g. $\langle [9, 15], [14, 29] \rangle \sqsubseteq \langle [10, 15], [29, 29] \rangle$ as $[9, 15] \sqsubseteq [10, 15]$ and $[14, 29] \sqsubseteq [29, 29]$. Then, each dimension of a vector corresponds to one and only one attribute or column of a dataset and requires a canonical order of vector dimensions.

3.3 Building the concept lattice

As interval vectors are patterns, Table 1 shows a pattern structure $(G, (D, \sqcap), \delta)$ where $G = \{g_1, \ldots, g_6\}$, D is a set of interval vectors or 3-dimensional vectors, where each component corresponds to an attribute or a column of the table. $(D, \sqcap)$ is composed of five interval vectors, i.e. a description for each object, plus all possible meets: by definition, any pair of elements (d, e) of a meet-semi-lattice admits a meet $d \sqcap e$. Description of g_3 is $\delta(g_3) = \langle [10, 15], [29, 29], [0.5, 0.5] \rangle$. Operators of the general Galois connection given in [46] are applied.

$$\begin{array}{rcl} \{g_3, g_6\}^{\square} & = & \bigsqcap_{g \in \{g_3, g_6\}} \delta(g) \\ & = & \delta(g_3) \sqcap \delta(g_6) \\ & = & \langle [10, 15], [29, 29], [0.5, 0.5] \rangle \sqcap \langle [9, 15], [14, 19], [0.5, 0.7] \rangle \\ & = & \langle [10, 15] \sqcap [9, 15], [29, 29] \sqcap [14, 19], [0.5, 0.5] \sqcap [0.5, 0.7] \rangle \\ & = & \langle [9, 15], [14, 29], [0.5, 0.7] \rangle \end{array}$$

$$\begin{array}{rcl} \langle [9, 15], [14, 29], [0.5, 0.7] \rangle^{\square} & = & \{g \in G \mid \langle [9, 15], [14, 29], [0.5, 0.7] \rangle \sqsubseteq \delta(g)\} \\ & = & \{g_3, g_4, g_6\} \end{array}$$

Obviously, g_3 and g_6 belongs to $\langle [9, 15], [14, 29], [0.5, 0.7] \rangle^{\square}$. g_4 also belongs to this set as its description is composed, for each dimension, of an interval that is included in the corresponding interval in $\langle [9, 15], [14, 29], [0.5, 0.7] \rangle$, i.e. $\langle [9, 15], [14, 29], [0.5, 0.7] \rangle \sqsubseteq \delta(g_4)$. Deriving the set $\{g_3, g_6\}$ with both Galois connection operators forming a closure operator makes the pair $(A, d) = (\{g_3, g_4, g_6\}, \langle [9, 15], [14, 29], [0.5, 0.7] \rangle)$ a pattern concept, i.e. $A^{\square} = d$ and $A = d^{\square}$. Partial ordering of all concepts is in full compliance with FCA and gives rise to a concept lattice.

4 FCAS formalized by means of pattern structures

Previously, we have detailed two methods for building a concept lattice from interval data. This section highlights the links existing between both methods and shows how the general formalism of pattern structures obtains same results as FCAS on interval data. In other words, we show how to handle with patterns structures a similarity and a pairwise similarity like in FCAS, taking advantage of efficient algorithms. Another contribution, useful for real-world experiments, shows how handling missing values with patterns structures. Consequently, this section also shows how both methods benefit from each other.

4.1 First statements

Both methods rely on a Galois connection between two partially ordered sets, i.e. $(2^G, \subseteq)$ and an ordered set of descriptions. For FCAS, descriptions are pairs composed of an attribute name and an interval. For IPS, descriptions are interval vectors with fixed size. In both case, intervals are ordered with inclusion.

The first operator of the Galois connection of FCAS associates to any set of objects the set of attributes they share. Firstly, pairwise similar objects are searched for, then γ returns the maximal shared interval. With IPS, the similarity operator $\sqcap$ accomplishes the same task as it returns a description representing the similarity between its arguments: $\sqcap$ is a *kernel operator* [46, 105]. Thus, this operator may handle other kind of similarities.

The second operator of the Galois connection in FCAS returns for a given description, i.e. set of pairs $(m, [a, b])$ with $m \in M$ et $a, b \in \mathbb{R}$, the maximal set of all objects that share these attributes. IPS performs a similar operation. However, IPS does not consider a pairwise similarity involving θ. In the following, we show how it can be achieved in full compliance with the existing framework of FCA.

4.2 Similarity between patterns

Basically, pattern structures consider the meet operator $\sqcap$ as a similarity operator [46]. Intuitively, given two objects g and h, and their respective descriptions $d = \delta(g)$ and $e = \delta(h)$ from a meet-semi-lattice, $d \sqcap e$ gives a description representing similarity between g and h. As a meet-semi-lattice is defined on the existence of a meet for any pair of elements, it follows that any two objects are similar and that their "level" of similarity depends on the level of their meet in the semi-lattice. Then, how to state that two objects are similar or not in sense of FCAS can be achieved as follows. Given $c, d \in D$ two patterns, then c and d are said to be similar iff $c \sqcap d \neq *$ where $*$ materializes the pattern that is subsumed by any other pattern. This pattern is added in D and can be interpreted as the pattern denoting "no subsumption" or "non similarity" between two patterns.

When considering patterns of type interval and remembering that any interval subsumes largest intervals containing it, the element $*$ can be introduced in association with a parameter θ as follows. Given a,b,c,$d \in \mathbb{R}$ and a parameter $\theta \in \mathbb{R}$,

$$[a, b] \sqcap_\theta [c, d] = \begin{cases} [min(a, c), max(b, d)] \text{ if } max(b, d) - min(a, c) \leq \theta \\ * \text{ otherwise,} \end{cases}$$

and

$$* \sqcap_\theta [a, b] \quad = \quad * \Leftrightarrow \; * \sqsubseteq_\theta [a, b].$$

Then, the meet-semi-lattice of intervals given in Figure 2 becomes the one given in Figure 3 when $\theta = 0.2$. In this way, we have defined a meet operator in a semi-lattice, such as the following links with FCAS hold:

$$[a, b] \sqcap_\theta [c, d] \neq * \Leftrightarrow [a, b] \simeq_\theta [c, d] \text{ and } [a, b] \sqcap_\theta [c, d] = * \Leftrightarrow [a, b] \not\simeq_\theta [c, d].$$

Operators $\sqcap$ and $\sqsubseteq$ for interval vectors use the $\sqcap_\theta$ for "constrained" intervals instead of $\sqcap$ for intervals, and formulas still hold. An example of concept is $(\{g_3, g_4, g_6\}, \langle [9, 15], *, [0.5, 0.7] \rangle)$: objects in the extent are similar for the first and third attributes. In FCAS, equivalent concept is $(\{g_3, g_4, g_6\}, \{(m_1, [9, 15]), (m_3, [0.5, 0.7])\})$: only shared intervals are represented, where attribute labels are inserted.

4.3 Pairwise similarity by means of projections

The use of $\sqcap_\theta$ does not allow the construction of intervals whose length exceeds θ like in FCAS. However, we cannot be sure these intervals describe maximal valid sets of objects in FCAS: definition of $\mathcal{R}_v$ starts with a set of objects A and returns the maximal valid set of objects: this set contains A plus all objects similar with objects in A and pairwise similar. Then γ returns the interval shared by the resulting set of objects for a given attribute. This means that though intervals from a semi-lattice $(D, \sqcap_\theta)$ all describe valid set of objects, some of them may not be "maximal". Below, we show how to replace any interval by its "maximal" interval thanks to a so-called *projection* in a meet-semi-lattice.

"Ball of patterns". Firstly, consider the meet-semi-lattice $(D, \sqcap_\theta)$ of interval values for a given attribute. Then, for any interval $d \in D$, we define the ball $B(d, \theta)$ as the set of intervals in D similar to d as follows.

$$B(d, \theta) = \{e \in D \mid e \simeq_\theta d\} \text{ with } e \simeq_\theta d \iff e \sqcap_\theta d \neq *$$

This ball of center d and diameter θ contains all intervals e whose meet with d is different of *, meaning that d and e are *similar*: $B([0.1, 0.1], 0.2) = \{[0.1, 0.1], [0.3, 0.3]\}$. This set is linked with $\mathcal{R}$ in FCAS, for a given attribute: $B(d, \theta)$ is the set of intervals shared by objects in $\mathcal{R}(A, m)$ when $A = g$ and $m(g) = d$.

Intervals representing maximal pairwise similar sets of objects. Now, among this set of intervals, we should remove any pair of intervals that are not pairwise similar, i.e. computing $\mathcal{R}_v$, and build an interval with left border (resp. right border) as the minimum (resp. maximum) of all intervals, i.e. computing γ. In terms of IPS it can be done by replacing any d of the meet-semi-lattice of intervals by the meet of all intervals e from the ball $B(d, \theta)$ that are not dissimilar with another element e' of this ball, i.e. $e \sqcap_\theta e' \neq *$:

$$\psi(d) = \prod_{\theta\ e \in B(d,\theta)} e \sqcap_\theta d$$
$$\text{such as } \nexists e' \in B(d, \theta) \text{ with } e \sqcap_\theta e' = *$$

In our example, $\psi([0.1, 0.1]) = [0.1, 0.1] \sqcap [0.3, 0.3] = [0.1, 0.3]$, for the third attribute and $\theta = 0.2$. In FCAS, the set returned by $\mathcal{R}_v$ is composed of objects whose attribute values respect the condition $\nexists e' \in B(d, \theta)$ with $e \sqcap e' = *$, i.e. objects are pairwise similar. Then $\prod_\theta$ returns the meet of all remaining intervals. With FCAS, we have $\gamma(g_2, m_3) = [0.1, 0.3]$ as well. In case of A is not valid w.r.t. m, remembering that any interval whose size exceeds θ is replaced by *, the mapping ψ returns * and γ in FCAS returns $\varnothing$.

ψ is a mapping that associates to any $d \in D$ an element $\psi(d) \in (D, \sqcap_\theta)$ such that $\psi(d) \sqsubseteq d$, as $\psi(d)$ is the meet of d and all intervals similar to d and pairwise similar. The fact $\psi(d) \sqsubseteq d$ means that ψ is contractive. In sense of [46], ψ is a *projection* in the semi-lattice $(D, \sqcap_\theta)$ as also monotone and idempotent. Moreover, any projection of a complete semi-lattice $(D, \sqcap)$ is $\sqcap$-preserving, i.e. for any $d, e \in V$, $\psi(d \sqcap e) = \psi(d) \sqcap \psi(e)$ [46].

Thereby, the projection may be computed in advance, replacing each pattern by a "weaker" or "more general" pattern without loss of information. It also naturally implies better computational properties as the number of elements in the semi-lattice is reduced. Indeed, in the previous chapter, we have shown that this parameter mostly influences complexity of adapted FCA algorithms for processing interval pattern structures. However, FCAS does not suggest easily such a preprocessing, and γ needs to be processed each time operators of Galois connection are calculated.

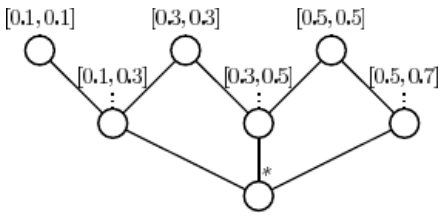


Figure 3: A meet-semi-lattice of intervals with additional element *

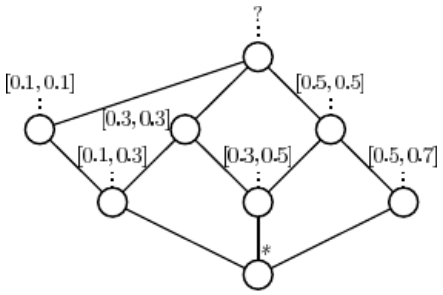


Figure 4: A lattice of intervals with additional elements * and ?

4.4 Handling missing values with pattern structures

Considering missing values requires to order them within a meet-semi-lattice of patterns or more generally within a lattice of patterns. Two possibilities are straightforward: a missing value (i) *subsumes* or (ii) *is subsumed* by any other element. In terms of FCAS, this means that the missing value (i) is *similar* or (ii) *dissimilar* with any other.

A missing value as the join of all elements. This is the most intuitive approach. As we do not know the actual value of a missing value, denoted by "?", it can be any other value: it has to subsume any element. Then we should not restrict D to a meet-semi-lattice $(D, \sqcap)$, but allow a lattice $(D, \sqcap, \sqcup)$ of patterns, such as $? \in D$. This requires some definitions: the meet $\sqcap$ is already defined except for "?", and the join $\sqcup$ has to be defined for any pair of elements. In fact, this is rather easy as we just add one element subsuming all the others in a meet-semi-lattice. Most importantly, for $d \in D$, we have: $d \sqcap ? = d \Leftrightarrow d \sqsubseteq ?$.

An example of a lattice of patterns $(D, \sqcap, \sqcup)$ is given in Figure 4: actually it results from adding "?" in the meet-semi-lattice given by Figure 3. In case of intervals, the join operator is given by

$$[a, b] \sqcup [c, d] \quad = \quad \begin{cases} [max(a, c), min(b, d)] \text{ iff } min(b, d) \leq max(a, c) \\ ? \text{ otherwise} \end{cases}$$

A missing value as the meet of all elements. The fact that a missing value is dissimilar with any other (except itself) is also interesting (see the application with real-world data at the end of this chapter). This underlines the fact that if the value is not given then it should not be considered as unknown: there is simply no information. This kind of missing value can by represented by the element $*$ introduced earlier. Indeed, * represents the dissimilarity between object descriptions and $*$ is subsumed by any other value.

Computation. In the previous chapter we have shown how slight modifications of well-known FCA algorithms enable computation of interval pattern structures. Interval vectors suggested to be implemented as arrays or vectors of intervals. With this implementation, and due to canonical order of vector dimensions, a missing value has to be materialized by * each time it is necessary, e.g. $\langle [15, 18], * \rangle$ where * is a missing value. Some data contain numerous attributes and are very sparse. Then the representation by vectors is not adequate as it leads to pattern intents containing a major proportion of * values. By contrast, FCAS suggest to IPS to consider pairs composed of an attribute name and a value, better for sparse data as representing only non-missing values.

5 A scaling approach based on tolerance relations

In this section, we define a scaling handling the relation $\simeq_\theta$. This allows to obtain a formal context on which classical FCA can be applied. The concepts are, like in FCAS, composed of maximal sets of objects pairwise similar for a maximal set of attributes and their respective range of values.

For that matter, the mathematical formalization of similarity relies on a *tolerance relation* which is reflexive and symmetric. A tolerance relation can be used for building a context in which concepts represents tolerance classes of similar objects for a given attributes. All tolerance classes are then reused to properly define a scaling for initial numerical data allowing FCA to be applied. The running example we consider in this section is given by Table 2.

	m_1	m_2	m_3
g_1	6	0	$[1, 2]$
g_2	8	4	$[2, 5]$
g_3	11	8	$[4, 5]$
g_4	16	8	$[6, 9]$
g_5	17	12	$[7, 10]$

Table 2: Another interval dataset

5.1 Tolerance relation and classes

Similarity has been studied from many points of view in artificial intelligence and pattern recognition [123, 78]. For example, considering documents being described by their attributes, e.g. keywords, similarity of documents x and y can be defined by non-emptiness of the set of their common attributes, $x' \cap y' \neq \emptyset$. The similarity is reflexive and symmetric, but not necessarily transitive. Following this idea, a tolerance relation captures the characteristics of a similarity [71].

Definition 5.1.1 *For a set G, a binary relation $T \subseteq G \times G$ is called* tolerance *if:*
(i) $\forall x \in G\ xTx$ *(reflexivity)*
(ii) $\forall x, y \in G\ xTy \rightarrow yTx$ *(symmetry)*

Let us consider now a set of objects G, a tolerance relation T, and a formal context (G, G, T). First, some objects, say g_1 and g_2, are observed to be pairwise similar, i.e. $g_1 T g_2$. Then pairs of the tolerance relation lead to a class of similar objects or "class of similarity". Moreover, among the classes of similarity, some classes are maximal meaning that the class is not included in any larger class.

Definition 5.2.2 *Given a set G, a subset $K \subseteq G$, and a tolerance relation T on G, K is a* class of tolerance *if:*
(i) $\forall x, y \in K\ xTy$ *(pairwise similarity)*
(ii) $\forall z \notin K, \exists u \in K\ \neg(zTu)$ *(maximality)*
An arbitrary subset of a class of tolerance is a preclass.

Now, let us consider the classes of tolerance associated with the formal context (G, G, T). The class of tolerance of an object g has to be considered along two dimensions: *(i)* the class is defined as the set of all objects which are tolerant with g, *(ii)* the class is maximal in the sense

that objects in the class are pairwise similar, and adding any other object in the class results in some pairs of non tolerant objects. A class of tolerance may be given a name which can be further used as an "attribute name" that describes the object. The result is a formal context (G, M, I) where I associates any object in G with its classes of tolerance $m \in M$.

Based on this observation, we show below how to use tolerance relations for designing scales for complex attributes and for building formal concepts whose extent are made of pairwise similar objects. Indeed, the similarity relation $\simeq_\theta$ defined in FCAS is symmetric and reflexive but not necessarily transitive, i.e. $\simeq_\theta$ is a tolerance relation. For example, with $\theta = 2$, $a = 1$, $b = 3$ and $c = 5$, $a \simeq_\theta b$ and $b \simeq_\theta c$ but $a \not\simeq_\theta c$ ($1 \not\simeq_\theta 5$), recalling that $a = [a, a]$ for any $a \in \mathbb{R}$.

5.2 Tolerance classes in numerical data

Let us consider a numerical many-valued context (G, M, W, I) where the range W_m of an attribute m is such that $W_m \subseteq W \subset \mathbb{R}$. Given an attribute $m \in M$, let us consider the formal context $(W_m, W_m, \simeq_\theta)$. Related objects in W_m are related are similar w.r.t. $\simeq_\theta$. For example, given $\theta = 5$ and m_1 in Table 2, the formal context $(W_{m_1}, W_{m_1}, \simeq_5)$ can be read in Figure 5 (left). As $\simeq_5$ is symmetric and reflexive, so is $(W_{m_1}, W_{m_1}, \simeq_5)$ and it contains a diagonal of crosses. Furthermore, the associated concept lattice (see Figure 5 (right)) is also symmetric.

Proposition 5.5.1 *Given a context $(W_m, W_m, \simeq_\theta)$ and the associated lattice, any concept (A, B) is such that either $A \subset B$, $B \subset A$, or $A = B$. Then, for each concept (A, B), there exists a unique concept (B, A).*

Proof. In the context $(W_m, W_m, \simeq_\theta)$, the set of objects is the same as the set of attributes. Then, for a concept (A, B), either $A \subset B$, $B \subset A$, or $A = B$. Since both $A, B \in W_m$ and for any formal concept (A, B), $A' = B$ and $B' = A$. (B, A) is also a formal concept, as verifying $B' = A$ and $A' = B$.

For example, the upper right concept on Figure 5 (right) can be read as $(\{8, 6, 11, 16\}, \{11\})$ and has a corresponding concept $(\{11\}, \{8, 6, 11, 16\})$ lower still on the right. One consequence of the above proposition is that the concept lattice can be separated in two parts w.r.t. the mapping $(A, B) \mapsto (B, A)$. In [47], such a mapping is called a *polarity*, i.e. an order-reversing bijection inverse of itself, and the resulting concept lattice is a *polarity lattice*. Then, we have the notion of axis of polarity:

Definition 5.5.3 (Axis of polarity) *In a polarity lattice, the set of all concepts (A, B) such that $A = B$ forms an* axis of polarity *of the concept lattice.*

For example, the set of concepts $\{(\{16, 17\}, \{16, 17\}), (\{11, 16\}, \{11, 16\}), (\{6, 8, 11\}, \{6, 8, 11\})\}$ is the axis of polarity of the concept lattice on Figure 5 (right). The set of all concepts (C, D) such that $(A, B) \leq (C, D)$, denoted by U, forms the upper part of the concept lattice. Dually, the set of all concepts (E, F) such that $(E, F) \leq (A, B)$, denoted by L, forms the lower part of the concept lattice. If $(A, B) \in U$ then $(B, A) \in L$ and $B \subset A$. Dually, if $(A, B) \in L$ then $(B, A) \in U$ and $A \subset B$.

Let us now consider the concept $(\{16, 17\}, \{16, 17\})$ of the axis of polarity in the lattice on Figure 5 (right). The values in $\{16, 17\}$ are all similar w.r.t. $\simeq_5$ and $\{16, 17\}$ cannot be extended with any other value without violating the internal similarity, i.e. there does not exist any element that does not belongs to $\{16, 17\}$ and that is similar with all elements in $\{16, 17\}$. This is true for all concepts in the axis of polarity.

m_1	6	8	11	16	17
6	×	×	×		
8	×	×	×		
11	×	×	×	×	
16			×	×	×
17				×	×

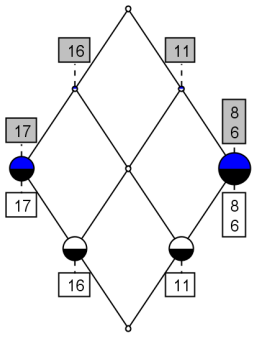


Figure 5: A tolerance relation and its associated concept lattice

This means that the extent or intents of the concepts in the axis of polarity are tolerance classes. Let us now consider the upper left concept $(\{11, 16, 17\}, \{16\})$ in the lattice on Figure 5 (right). This concept is in U and the values in the extent $\{11, 16, 17\}$ are similar to 16. Moreover, the intent $\{16\}$ is contained in the larger intent $\{16, 17\}$ meaning that $\{16\}$ determines a preclass of tolerance. Dually, we have the same interpretation for the symmetric concept $(\{16\}, \{11, 16, 17\}) \in L$.

Proposition 5.5.2 *Let (A, B) be a concept of the axis of polarity, i.e. $A = B$. Then, A (or B) is a set of maximal pairwise similar values, i.e. A determines a class of tolerance. Let (C, D) a concept in U but not in the axis of polarity, i.e. $D \subset C$. D is a preclass of tolerance and C is the set of all values similar to values in D.*

Proof. Both derivation operators $(\cdot)'$ have same domain and range W_m, and $(\cdot)'$ associates with a subset A of values in W_m the maximal subset of similar values in W_m, i.e. related through $\simeq_\theta$. Then, for a concept (A, B) where $A = B$ and $A' = B$ or $A = B'$, then $A = A'$ or $B = B'$ are maximal and define a same tolerance class. Moreover, the set of all extents A or all intents B from concepts of the axis of polarity covers the set W_m. For a concept (C, D) with $D \subset C$, since $C' = D$, all values in C are similar to values in D. Now, relying on the preceding proposition, as the concept (C, D) does not verify $C = D$ but instead $D \subset C$, it exists a class of tolerance say F such as $D \subset F \subset C$ and thus D is a preclass of tolerance.

The intents of the concepts in the upper part of the lattice –or dually the extents in the lower part– are partially ordered and determine sets of similar values. Among these intents, the intents in the axis of polarity are maximal and are classes of tolerance, and the other intents are only preclasses of tolerance. For example, taking $\theta = 5$ and m_1 in Table 1, there are 5 intents, namely $\{16\}$, $\{11\}$, $\{16, 17\}$, $\{11, 16\}$, and $\{6, 8, 11\}$, where the three last intents are tolerance classes. When there is no ambiguity, we use the term of "class of similarity" for a class or a preclass of tolerance.

These classes of similarity are used to define a scale allowing the application of FCA algorithms to a numerical many-valued context. Classical FCA algorithms can be used to compute classes of similarity and require slight modifications for generating the upper (dually lower) part of the concept lattice only (discussed later).

5.3 Scaling and concept lattice construction

At present, we have made precise how a partially ordered set of classes of similarity can be built from attributes valued by numbers or intervals of numbers in a many-valued context. Now,

	$(m_1, 11)$	$(m_1, 16)$	$(m_1, [6, 11])$	$(m_1, [11, 16])$	$(m_1, [16, 17])$	$(m_2, 4)$	$(m_2, 8)$	$(m_2, [0, 4])$	$(m_2, [4, 8])$	$(m_2, [8, 12])$	$(m_3, [1, 5])$	$(m_3, [4, 9])$	$(m_3, [6, 10])$	$(m_3, [4, 5])$	$(m_3, [6, 9])$
g_1			×					×			×				
g_2			×			×		×	×		×				
g_3	×		×	×			×		×	×	×	×		×	
g_4		×		×	×		×		×	×		×	×		×
g_5					×					×			×		

Table 3: A formal context obtained handling classes of tolerance.

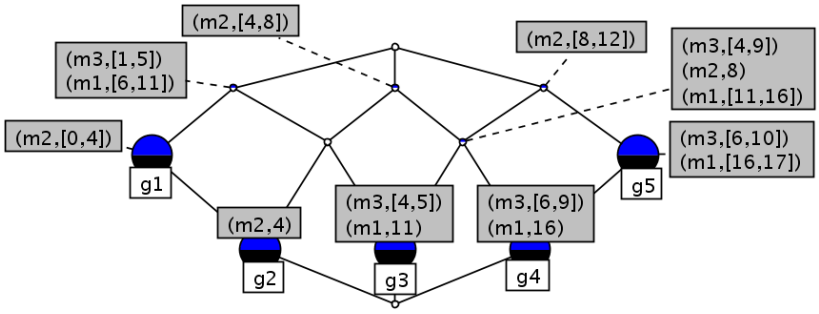


Figure 6: Concept lattice raised from Table 3.

classes of similarity have to be named before being used as attribute names for scaling the original many-valued context and derive a scaled binary context from which the final concept lattice is built. Actually, the name of the elements of the scale can be related to the content of the corresponding class of similarity and to the name of the original attribute that is scaled. In the present case, an element of the scale is named by a pair associating the name of the original attribute and either the content of the class of similarity, e.g. $\{16, 17\}$ for m_1, or the convex hull, e.g. $[16, 17]$.

Let us consider the numerical many-valued context (G, W, M, I) in Table 1. Three sets of classes of similarity, one for each attribute m_1, m_2, and m_3, are computed thanks to three tolerance relations relying on three different similarities $\simeq_\theta$, and extracted from the symmetric concept lattices associated with each tolerance relation. The transformation of the original (G, W, M, I) context into the derived (G, N, J) reads as follows:

- G is the set of original objects,
- $N = \bigcup_{m \in M}(\{m\} \times C_m)$ with C_m is the set of all classes of similarity of attribute m,
- $(g, (m, C_m)) \in J$ means that the value of object g in the many valued context, i.e. $m(g)$, belongs to class C_m,

For example, the derived binary context associated with Table 1 is given in Table 3 where the thresholds are $\theta = 5$ for m_1 and $\theta = 4$ for m_2 and $\theta = 5$ for m_3. Figure 6 shows the resulting concept lattice.

6 An information fusion problem in agronomy

The problem of information fusion is encountered in various fields of application, e.g sensor fusion, merging multiple sources, etc. Information fusion consists of merging several sources of information for answering questions of interest and make proper decisions [39]. Accordingly, a fusion operator is an operation summarizing information given by sources into a consensual and representative information. In this section, we introduce a real-world information fusion problem in agronomy, concerning pesticide application to fields. Then, we show how this fusion information problem can be solved with a concept lattice involving a tolerance relation. The output is an analysis and an evaluation of agricultural practices w.r.t. pesticide application and subsequent ecological problems.

6.1 Problem settings

Agronomists compute indicators for evaluating the impact of agricultural practices on the environment. Questions such as the following are of importance: what are the consequences of the application of a pesticide given the characteristic of this pesticide, the period of application, and the characteristics of the field? The risk level for a pesticide to reach groundwater is computed by the indicator I_{gro} in [21]. Based on the value of I_{gro}, agronomists try to make a diagnosis of agronomic know-how w.r.t. the use of pesticides. Pesticide characteristics depend on the chemical characteristics of the product while pesticide period application and field characteristics depend on domain knowledge. This knowledge lies in information sources among which books, databases, and expert knowledge in agronomy. Moreover, values for some characteristics may vary w.r.t. information sources.

Here, we are interested in the analysis of practices through the use of *glyphosate* in different countries w.r.t. farmers habits. Glyphosate is a widespread product used by farmers in temperate areas, actually one of the mostly used herbicide in USA[16]. In 2006, IFEN, for French Institute for the Environment, observed that glyphosate is the most encountered substance in French waters, possibly leading to long-term adverse effects in the aquatic environment[17].

Below, three characteristics of glyphosate, namely $DT50$, *koc*, and ADI, are given in Table 4 (simplified data), according to 12 different information sources.

- $DT50$ represents "half-life" of the pesticide, i.e. time required for the pesticide concentration to decrease of 50% under some conditions. Pesticides with $DT50$ value lower than 100 days can be considered as having a weak impact on groundwater quality in general temperate conditions.

- *koc* characteristic represents the mobility of the pesticide and depends on pesticide properties and type of soil. Pesticides with high *koc* values typically stay in upper level of soil and do not reach groundwater. By contrast, pesticides with *koc* value less than 2200 have good chances to contaminate groundwater.

- ADI (for "Acceptable daily intake") represents toxicity for humans. Glyphosate is considered as having a low toxicity, i.e. no toxic effects were observed for doses of 400 mg/kg/day according to specialized studies. However, the values 0.3 and 0.05 are separated for expert reasons.

[16] http://www.epa.gov/
[17] http://www.ifen.fr/

Table 4: Characteristics of pesticide *glyphosate*.

	$DT50$ day	koc L/kg	ADI g/kg/day
BUS	47	24000	0.3
PM10	[3,60]	[25,68000]	0.3
INRA	[38,60]	167	0.05
Dabene	[38,60]	167	0.05
ARSf	[2,174]	[500,2640]	[0.05,0.3]
ARSl	[2,174]	[500,2640]	[0.05,0.3]
Com96	[2,174]	[25,68000]	0.3
Com98	[38,60]	[500,2640]	0.3
RIVM	[18,66]	[3566,40420]	[0.05,0.3]
BUK	[3,60]	[25,68000]	0.3
AGXf	[8,30]	[301,59000]	0.3
AGXl	[14,111]	[301,59000]	0.3

In Table 4, information sources are not always in agreement. Then, it can be interesting for experts in agronomy to analyse such a table from the point of view of information fusion: which are the sources being in agreement and at which level are they in agreement? This is done using a concept lattice involving a tolerance relation as explained below.

6.2 Method and first results

Now, we apply one of the three methods presented in this chapter, i.e. FCAS, IPS or tolerance base scaling, to build a concept lattice from Table 4. Three thresholds are defined according to the above observations: $\theta = 100$ for $DT50$, $\theta = 2200$ for koc, and $\theta = 0$ for ADI. Then, for each attribute, classes of similarity and the scale for each attribute are computed and can be read on the lattice in Figure 7.

The lattice shows an interesting classification of information sources w.r.t. information fusion. Each concept in the lattice is composed of an extent with a maximal set of sources in agreement w.r.t. the interval of values in the intent.

The operator used for managing information fusion is convex hull, controlled by a similarity parameter θ, i.e. for two similar intervals the lower bound is the minimum of the two lower bounds and the upper bound is the maximum of the two upper bounds. Let us examine the lattice in detail. The highest concept in the lattice, $\top$, has the intent with the larger intervals (since $*$ is subsumed by any other interval): $[2, 174]$ for $DT50$, $[25, 68000]$ for koc, and $[0.05, 0.3]$ for ADI. The higher a concept is in the lattice, the more information sources in the extent agree on the values to be verified by the attributes. This could be considered as the maximal agreement of all sources but this does not provide any precise information (indeed, the calculation of I_{gro}, which cannot be detailed here, does not allow any recommendation). Moreover, the concepts in the lower levels of the lattice have more restricted intervals. Going further, we can observe that there are four descendants of $\top$ that determine four main parts of the lattice. On the left, there are mainly French and UK information sources, namely AGXf, AGXl, PM10 (French), and BUK and BUS (UK), with com96 denoting an expert committee. In the middle of the lattice, there are mainly French sources, namely RIVM, Dabene, and INRA. Finally, on the right, there are US

information sources, namely ARSl, ARSf, and the committee Com98. Interestingly, there is an agreement between European sources as English or French sources share some upper level values such as $[3, 66]$ for $DT50$ or 0.3 for ADI. By contrast, there is no agreement between European and US sources except for the expert committee com98. This shows that practices are actually different and allowed values for pesticide characteristics are not the same w.r.t. the country. Among the possible explanations, it remains very difficult to harvest agronomic data (in cost and time) in every country. For example, the circumstances in which these data are collected are very different w.r.t. climate, soil type, measure devices, etc. In this sense, according to experts in agronomy, the lattice on Figure 7 is a good witness (confirmation) of this diversity of practices and of the agreement degree between sources as given by the lower level concepts.

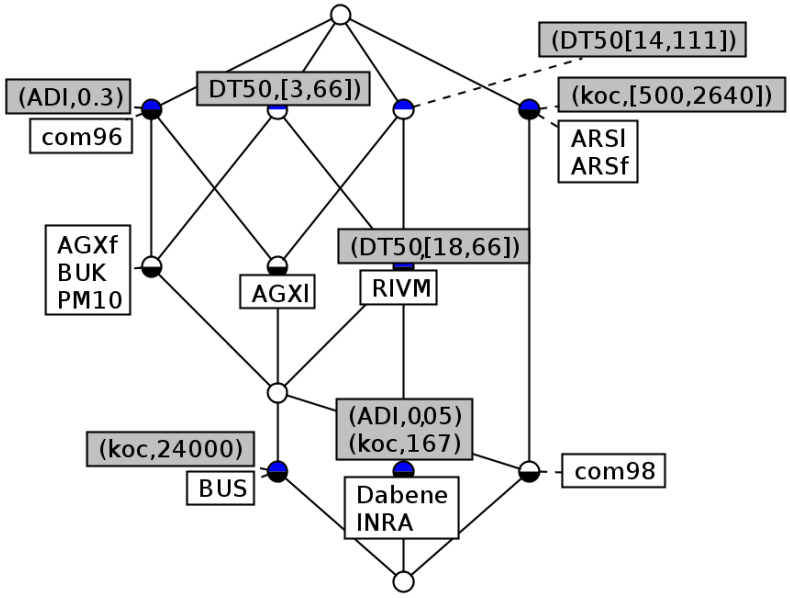


Figure 7: Concept lattice raised from Table 4

7 Discussion

We have presented three different approaches for building a concept lattice from numerical data involving a similarity relation between numerical attribute values. The first one (FCAS) defines a Galois connection for that matter. The second one (IPS) uses an existing framework (and Galois connection) and shows that considering a similarity relation consists in projecting the object description space. Finally, a tolerance based scaling allows classical FCA to be applied. These three methods are conceptually equivalent [58]. However, they all bring interesting clues or elements on the problem of designing concept lattices from numerical data. FCAS brings intuitions to consider similarity and pairwise similarity of objects by means of attribute sharing. IPS allows to consider this similarity within an existing framework provided with efficient algorithms. Finally, the third approach establishes links between projection of partially ordered sets and scaling. Most importantly, it provides a semantic to concepts: objects in the extents share the same classes of tolerance described in the intent. As in the previous chapter, the most efficient methods between IPS and scaling depends on the data distribution, such as the number of different values and their sparsity. Each attribute has a different range and different similarities

and thresholds θ have to be defined. However, data can be normalized (e.g. by subtracting the mean, followed by dividing the standard deviation) and use a single threshold can be used for a given context. The choice of θ and a study of its variation effect can be found in [86].

The discussion is now divided into several parts, each one with a particular topic.

Concept lattices and similarity. Tolerance relations in connection with FCA were studied in several papers [47, 12, 71]. In [71], tolerance relations are introduced from an historical perspective and their role in the formalization of similarity of documents is detailed. In the basic reference [47], it is shown that starting from any complete lattice and a tolerance relation between its elements (from any arbitrary set), there exists a formal context encoding tolerance (pre-)classes. In this work, the statement is used in the opposite way: starting from an arbitrary numerical context, a tolerance relation formalizes the similarity between numerical values and the resulting classes of similarity are then reused for defining scales and a concept lattice encoding the initial numerical context. Other important related work can be found in [12], where fuzzy formal concept analysis introduced. A fuzzy context contains truth values and both attribute and object sets are considered as fuzzy sets. Then a fuzzy concept lattice can be built in the same way as this is done here by grouping pairwise similar objects or attributes with a tolerance-like relation. However, the research work in [12] is much more oriented on the study of mathematical properties of similarity within a concept lattice, contrasting our work on the embedding of constrained tolerance relations in FCA for classifying objects with complex numerical attributes.

Discretization approaches. The scaling procedure proposed in this chapter transforms quantitative data into qualitative data. Following [130], this method is: *unsupervised* since class membership of objects is unknown ; *parametric* since a similarity parameter θ has to be given and relies on domain knowledge ; *univariate* as processing each attribute separately ; *ordinal* since taking advantage of the implicit ordering information in quantitative attributes ; and finally and most importantly, *hierarchical* as it builds a partially ordered set (poset) of similarity classes. This poset is actually given by a concept lattice from a formal context encoding a tolerance relation and by a projected meet-semi-lattice of object descriptions. This poset is finally used to raise a concept lattice giving a conceptual classification of objects of a domain. Thereby, it can be applied to any kind of structured data for which a similarity can be defined (sequences, graphs, etc.). Cluster-based discretization methods are close to our scaling (see [130]). First, some clusters are searched for, then their intents are used to define intervals for the discretization process. In this chapter, we focused on showing how discretization can be automated and controlled (with tolerance relation), with different approaches, while resulting concept lattices keep the same interpretation.

Processing symmetric contexts. There are many efficient algorithms for generating a concept lattice from a binary context [74]. The efficiency of these algorithms mainly depends on the density of the formal context (G, M, I), i.e. $|I|/|G \times M|$. In the case of context materializing a tolerance relation, computational complexity is related to the similarity and the range of each attribute. These algorithms may also be used to obtain the partially ordered set of classes of similarity. We propose here two optimizations of FCA algorithms to process symmetric contexts.

Recall that computing classes of similarity for a given attribute can be done either with the upper part or the lower part of the corresponding lattice. Then, a concept is not generated if its dual concept has already been generated. Bottom-up (dually top-down) algorithms are well adapted for this task: concepts (A, B) are generated from bottom to top until the concept verifies $A = B$, i.e. (A, B) belongs to the axis of polarity. Then, interesting concepts are computed by standard FCA algorithms with a modified backtracking condition. This task can

be also achieved using well-known and efficient closed itemset mining algorithms [131, 122]. A second optimization relies on the fact that the set $W_m \subset \mathbb{R}$ is totally ordered. For intervals, given $a, b, c, d \in \mathbb{R}$, we have $[a, b] \leq [c, d]$ when $a \leq c$, and if $a = b$ when $b \leq d$. Then, similarity has a monotony property: given $v_1 < v_2 < v_3$, when $v_1 \not\approx_\theta v_2$ then $v_1 \not\approx_\theta v_3$. Intuitively, monotony means that the corresponding binary table contains lines and columns of consecutive crosses, e.g. Figure 5 (left). Then, the scan of a context by an FCA algorithm can be reduced accordingly. For example, Figure 8 shows how the performances of the bottom-up algorithm CloseByOne [74] are modified in this case (random data with $\theta = 20$ are used here, but other θ give same result).

Projecting and processing a pattern structure. Processing interval pattern structures with adaptation of classical algorithms of FCA [74] has been detailed in the previous chapter. We showed the scalability of concept lattice design from large data, e.g. with thousands objects and dozens attributes. The projection computation is related to the maximal clique problem in graph theory, known to be a hard problem. However, since we are dealing with numerical data, and that attribute values can be totally ordered (see above), the projection algorithm is simple: it consists in, for each data value, (i) looking for similar elements from a totally ordered set and (ii) testing each pair of the resulting set to keep the maximal set of pairwise similar values. Finally, pattern structures are easier to process when projected, as shown in [46] for graph patterns, while preserving subsumption relations.

Concept lattices and information fusion. Several fusion operators were proposed for combining uncertain information [35, 39]. Generally, the fusion operator is applied on all information sources, i.e. considering all sources simultaneously, and for one particular variable or attribute at a time, see e.g. [35]. However, this leads to some problems, since when sources are conflicting, the fusion result if generally not useful. Consider now the convexification controlled by θ as a fusion operator, i.e. the operator $\sqcap_\theta$. Our method accordingly considers maximal subsets of sources with their fusion results and organizes them in a concept lattice. The concept lattice allows to identify which maximal subsets of objects support the most similar results. This opens further research for the use of concept lattices in information fusion. Actually, the next chapter proposes a deeper investigation.

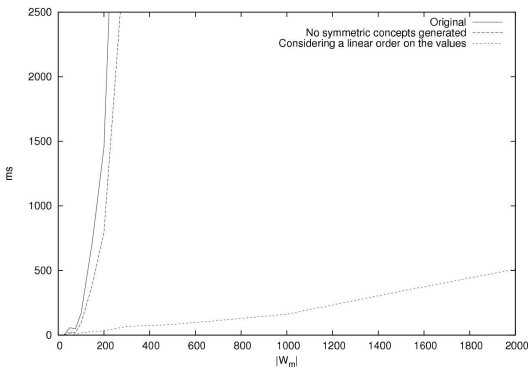


Figure 8: Runtime of modified CloseByOne for computing symmetric contexts

Chapter 6

Enhancing information fusion with concept lattices

The main problem addressed in this chapter is the merging of numerical information provided by several sources (databases, experts...). Merging pieces of information into an interpretable and useful format is a tricky task even when an information fusion method is chosen. Fusion results may not be in suitable form for being used in decision analysis. This is generally due to the fact that information sources are heterogeneous and provide inconsistent information, which may lead to imprecise results. We propose the use of Formal Concept Analysis and more specifically pattern structures for organizing the results of fusion methods. This allows us to associate any subset of sources with its information fusion result. Then once a fusion operator is chosen, a concept lattice is built. With examples throughout this chapter, we show that concept lattices give an interesting classification of fusion results. When the fusion global result is too imprecise, the method enables the users to identify what maximal subset of sources would support a more precise and useful result. Instead of providing a unique fusion result, the method yields a structured view of partial results labelled by subsets of sources. Finally, an experiment on a real-world application has been carried out for decision aid in agricultural practices.

1 Introduction

The problem of information fusion is encountered in various fields of application, e.g sensor fusion, multiple source interrogation systems. Information fusion consists of merging, or exploiting conjointly, several sources of information for answering questions of interest and make proper decisions [19]. A fusion operator is an operation summarizing all information given by sources into an interpretable information, for example the interval intersection for numerical information. The Table 1 gives an example of information given by sources: each object, or source, in line gives a value for an attribute or variable in column. This value intuitively represents the point of view of the source on the quantification of a phenomena, or observation.

Several fusion operators have been proposed for combining uncertain information [38, 37, 39, 14, 31, 99] and no universal method is available [38]. Dubois and Prade [38] overviewed how fuzzy set theory can address the information fusion problem and proposed several fusion operators for numerical information. More recently, a fusion operator based on the notion of Maximal Consistent Subset (MCS) has been proposed for finding a global point of view when no meta-knowledge is available about sources (reliability, conflict) [36, 35]. These works apply the fusion operator on the set of all sources and provide the resulting information. Other approaches

	m_1	m_2
g_1	$[1,5]$	$[1,9]$
g_2	$[2,3]$	$[1,3]$
g_3	$[4,7]$	$[6,7]$
g_4	$[6,10]$	$[8,9]$

Table 1: Information dataset given by sources

define their proper fusion operator in a lattice structure to combine symbolic information [31, 99].

In this work, we use FCA to study all subsets of sources and their information fusion results. The main ability of FCA is to produce formal concepts corresponding to maximal sets of sources associated with a same fused information. Therefore, one has not to study the 2^n possible subsets of sources, but only the closed sets of sources that are concept extents. The concepts are ordered and form a structure called concept lattice. We show that this lattice contains the information fusion result considering all sources proposed by [38, 36, 35]. Moreover, the lattice is meaningful for organizing information fusion results of different subsets of sources and allows more flexibility for the user. Moreover, the lattice keeps a track of the origin of the information such as presented in [37] for the fusion of symbolic information.

This work can be used in many applications where it is necessary to find a suitable value summarizing several values coming from multiple sources. Here, we use an experiment in agronomy for decision helping in agricultural practices.

2 Fusion operators

According to previous works, there are three kinds of behaviours for the fusion operators: conjunctive, disjunctive and trade-off operators [19, 38, 39].

Before introducing these operators, we introduce the following notations: n is the number of sources. $\mathbb{I}_m$ is the set of all values given for the variable m. f_m denotes a fusion operator returning the fusion result for variable m.

2.1 Basic operators

The *conjunctive operator* is the counterpart to a set intersection. The imprecision and the uncertainty in the information associated with the result of a conjunction is less than the imprecision or the uncertainty of each source alone. A conjunctive operator makes the assumption that all the sources are reliable, and usually results in a precise information. If there is some conflict in the information (i.e. at least one source is wrong), then the result of the conjunction can be empty. The conjunctive operator for a variable m is defined by $f_m(\mathbb{I}_m) = \bigcap_{i=1,\ldots,n} I_i$, e.g., in Table 1, $f_{m_1}(I_1, \ldots, I_4) = \emptyset$ represents the intersection of intervals of m_1 with $I_1 = [1,5], I_2 = [2,3], I_3 = [4,7]$ and $I_4 = [6,10]$.

The *disjunctive operator* is the counterpart to a set union. The uncertainty (or the imprecision) resulting from a disjunction is higher than the uncertainty (or the imprecision) of all sources together. A disjunctive operator makes the assumption that at least one source is reliable. The result of a disjunctive operator can be considered as reliable, but is also often (too) weakly informative. The disjunctive operator for the variable m, is defined by $f_m(\mathbb{I}_m) = \bigcup_{i=1,\ldots,n} I_i$, e.g. , $f_{m_1}(I_1, \ldots, I_4) = [1,10]$ that represents the union of the intervals of m_1.

The *trade-off operators* lie between conjunctive and disjunctive behaviors, and are typically

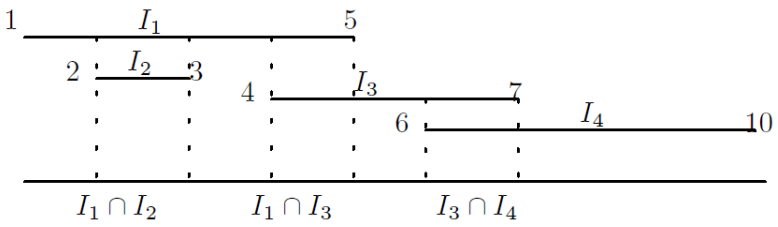


Figure 1: MCS computed from Table 1 for the variable m_1.

used when sources are partly conflicting. They try to achieve a good balance between informativeness and reliability [38]. The fusion based on MCS is an example of trade-off operators.

2.2 Maximal consistent subset fusion method

When no information is available about sources, like conflict between sources, or reliability of sources, a reasonable fusion method should take into account the information provided by all sources. At the same time, it should try to keep a maximum of informativeness. The notion of MCS is a natural way to achieve these two goals.

The idea of MCS goes back to Rescher and Manor [106]. This notion is currently used in the fusion of logical formulas [14] but also of numerical data [36, 35]. Given a set of n intervals $\mathbb{I} = \{I_1, I_2, \ldots, I_n\}$, a subset $\mathbb{K} \subseteq \mathbb{I}$ is *consistent* if $\bigcap_{i=1}^{|\mathbb{K}|} \mathbb{K}_i \neq \emptyset$ with $\mathbb{K}_i \in \mathbb{K}$ and *maximal* if it does not exist a proper super-set $\mathbb{K}' \supseteq \mathbb{K}$ that is also consistent. In Table 1, the set $\mathbb{K}_1 = \{I_1, I_2\}$ is a MCS of the set $\mathbb{I}_{m_1}$, since $I_1 \cap I_2 \neq \emptyset$ and is maximal w.r.t. intersection property.

The fusion operator of n sources based on MCS consists in applying a disjunctive operator on their MCS. Nevertheless, there exists cases where finding MCS is easy, especially when sets are intervals in $\mathbb{R}$. $I_i = [a_i, b_i], (i = 1, \ldots, n)$. The algorithm is based on an increasing sorting of the lower and upper bounds of intervals into a sequence $(c_j)_{j=1,\ldots,2n}$. Each time, an element c_j of type b (i.e. an upper bound of an interval in I_i) and an element c_{j+1} of type a (i.e. a lower bound of an interval in I_i) meet, then a maximal consistent subset is obtained. For example, in Table 1, the MCS for the variable m_1 of the set $\{I_1, I_2, I_3, I_4\}$ are $I_1 \cap I_2 = [2, 3]$, $I_1 \cap I_3 = [4, 5]$ and $I_3 \cap I_4 = [6, 7]$ when $I_1 = [1, 5], I_2 = [2, 3], I_3 = [4, 7]$ and $I_4 = [6, 7]$ (see Figure 1).

For example, the MCS fusion result for m_1 in Table 1 is $f_{m_1}(I_1, \ldots, I_4) = [2, 3] \cup [4, 5] \cup [6, 7]$, as illustrated in Figure 1. The MCS notion appears as a natural way to conciliate the two objectives of gaining information and of remaining in agreement with all sources in information fusion problem. Generally, finding MCS is a problem having exponential complexity [82]. Dubois et al. [36] introduce a linear algorithm to compute the MCS of n intervals.

2.3 Properties of fusion operators

Generally, all fusion operators are commutative and idempotent. The conjunctive and disjunctive operators are associative but not the trade-off fusion operators (more details in [35]). If the final result of the fusion is not convex, it is always possible to take its convex hull (loosing some information in the process but gaining computational tractability). Conjunctive fusion result is convex but this is not the case for the others operators in general.

3 Organization fusion results within a concept lattice

For merging numerical information, a common fusion operator has to be used. This is specially important in case of heterogeneous sources. Fusion operators are often based on assumptions or on meta-knowledge available about the sources (reliability, conflict) and the domain. Sometimes, it happens that the fusion result is not directly useful for decision. For example, in [4] the fused information must be convex for being used in a further decision process, and the convexification of MCS leads to an imprecise result. Here, we propose to identify and characterize interesting subsets of sources, providing more useful fused information. Accordingly, we show how a fusion operator can be embedded in the framework of Formal Concept Analysis (FCA) to build a concept lattice yielding a structured view of partial results labelled by subsets of sources, instead of providing a unique fusion result. As facing here complex data (precisely numerical data), we use the formalism of pattern structures. It requires to define a meet operator on object descriptions, inducing their partial order. We discuss how a fusion operator can be seen as a meet operator. First, we define the notion of information fusion space.

Definition 6.3.1 (Information fusion space) *An information fusion space D_m is composed of the information available for a variable m and all their possible fusion results, w.r.t a fusion operator f_m.*

For example, with the variable m_1 in Table 1 and f_m as the interval intersection, $D_m = \{[1,5],[4,7],[6,10],[2,3],\ [4,5],[6,7],\emptyset\}$.

3.1 A fusion operator in a pattern structure

Let us consider a single variable $m \in M$, its fusion space D_m corresponding to a chosen fusion operator f_m. When f_m is idempotent, commutative and associative, (D_m, f_m) is a meet-semi-lattice, since f_m behaves as a meet operator. This is the case for any conjunctive or disjunctive fusion operator, and we have $c \sqcap d = f_m(c,d), \forall c,d \in D_m$, meaning that the meet of two elements of D_m corresponds to their fusion.

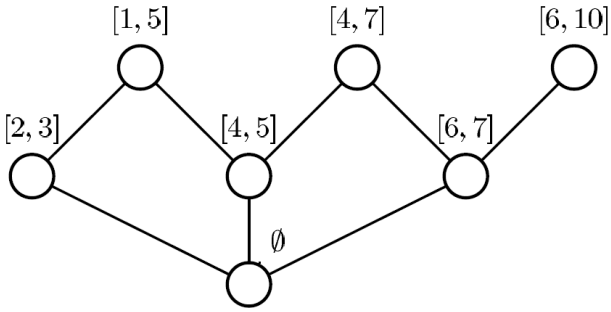


Figure 2: A meet-semi-lattice of intervals

For example, let us consider the numerical variable m_1 in Table 1, and the conjunctive fusion operator f_{m_1} that corresponds to the interval intersection $\cap$. Figure 2 shows the meet-semi-lattice (D_{m_1}, f_{m_1}). The interval labelling a node is the meet of all intervals labelling its ascending nodes, i.e. the resulting information fusion w.r.t f_{m_1} of the sources given the intervals labelling its ascending nodes. In the example, $f_{m_1}([4,7],[6,10]) = [6,7]$ is the fusion of objects g_3 and g_4 for the variable m_1, and $f_{m_1}([2,3],[1,5]) = [2,3]$ for objects g_1 and g_2. Therefore, we have partially ordered the fusion space D_{m_1} with $c \cap d = c \Leftrightarrow c \subseteq d, \forall c,d \in D_{m_1}$. This order

is a particular instance of the pattern subsumption relation defined in pattern structures. It means, in this example, that an interval is subsumed by any larger one, e.g. $[2,3] \sqsubseteq [1,5]$ since $[2,3] \subseteq [1,5]$. For example, we have $[2,3] \sqcap [1,5] = [2,3] \Leftrightarrow [2,3] \sqsubseteq [1,5]$ in terms of semi-lattice, corresponding to $[2,3] \cap [1,5] = [2,3] \Leftrightarrow [2,3] \subseteq [1,5]$ in interval inclusion terms. Note that a disjunctive fusion operator is handled similarly.

3.2 Building and interpreting the concept lattice

Given G a set of sources, $m \in M$ a single variable, (D_m, f_m) the meet-semi-lattice of fusion results, and δ a mapping that gives to any object its information for the variable m, then $(G, (D_m, f_m), \delta)$ is a pattern structure. On the example, we have $(G, (D_{m_1}, f_{m_1}), \delta)$. (D_{m_1}, f_{m_1}) is described in the previous subsection. Descriptions of sources g_1 and g_2 are respectively $\delta(g_1) = [1,5]$ and $\delta(g_2) = [2,3]$. Then, the general Galois connection can be used to compute and order concepts:

$$\begin{array}{rcl} \{g_1, g_2\}^{\square} & = & [1,5] \sqcap [2,3] \\ & = & f_{m_1}([1,5],[2,3]) \\ & = & [2,3] \end{array} \qquad \begin{array}{rcl} [2,3]^{\square} & = & \{g \in G \mid [2,3] \sqsubseteq \delta(g)\} \\ & = & \{g \in G \mid [2,3] \subseteq \delta(g)\} \\ & = & \{g_1, g_2\}. \end{array}$$

Since $\{g_1, g_2\} = [2,3]$ and $[2,3]^{\square} = \{g_1, g_2\}$, the pair $(\{g_1, g_2\}, [2,3])$ is a concept. Efficient FCA algorithms can extract the set of all formal concepts and order them within a concept lattice [74]. They can be easily adapted to compute in pattern structures [46, 59]. The lattice of our example is given in Figure 3.

A concept (A, d) of $(G, (D_{m_1}, f_{m_1}), \delta)$, is interesting from many points of view, as illustrated with the concept $(\{g_1, g_2\}, [2,3])$.

- Its intent d provides the fusion resulting from objects in A, e.g. $[2,3]$ is the conjunctive fusion f_{m_1} of the information from sources g_1 and g_2.
- No other object can be added to A without changing d, e.g. $\{g_1, g_2\}$ is the maximal set of sources whose conjunctive information fusion is $[2,3]$.
- The extent A keeps the track of the origin of the information, e.g. it is known that the new information $[2,3]$ comes from the information of g_1 and g_2.

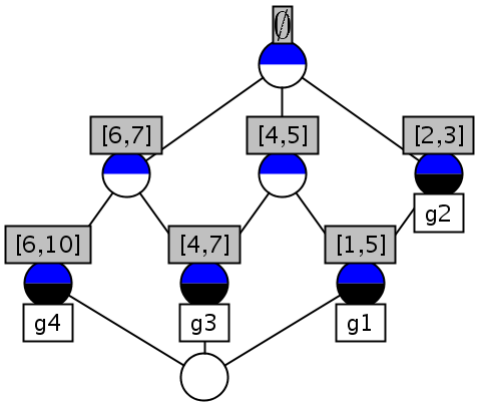


Figure 3: A concept lattice raised from Table 1 for the variable m_1.

The resulting concept lattice provides a suitable classification of information sources and their information fusion results. In Figure 3, a concept extent is read with reduced labelling. However, for sake of readability, intents are given for each concept (not reduced). For example, the node labelled with $[6,7]$ represents the concept $(\{g_3,g_4\},[6,7])$. Due to concept ordering, a concept provides the fusion result of a subset of the extent of its super-concepts (generalization/specialization). Then, the navigation in the lattice gives interesting insights into the fusion results. This allows more flexibility for decision making. For example, in related works, only the fusion of information of all objects is considered which corresponds to the most general concept ($\top$) in the lattice. This result does not always allow to make a decision, e.g. an empty intersection in our example. Then it is interesting to observe subsets of objects, by navigating in the lattice.

3.3 A particular case with a non associative fusion operator

The fusion operator f_m based on the notion of MCS is idempotent and commutative, but not associative. For example in Table 1,

$$f_{m_1}(f_{m_1}([1,5],[2,3]),[4,7]) = [2,3] \cup [4,7]$$

and

$$f_{m_1}(f_{m_1}([1,5],[4,7]),[2,3]) = [2,3] \cup [4,5].$$

Then, the fusion operator cannot be directly used as a meet operator to build a concept lattice.

However, since this operator returns the union of all MCS, we can firstly compute all MCS for a given variable, denoted by the set K and then use the disjunctive operator on the MCS as a meet operator to define a meet-semi-lattice $(K,\cup)$. Formally, we consider $(\mathcal{O},(K,\cup),\delta)$ as a pattern structure where $\mathcal{O}$ is a multi-set of sources, each element is set of sources of one MCS $\mathbb{K} \in K$, i.e. $\delta(o) \in K, \forall o \in \mathcal{O}$. For example, the MCS of intervals for m_1 are $[2,3]$, $[4,5]$ and $[6,7]$ given respectively by $\{g_1,g_2\}$, $\{g_1,g_3\}$ and $\{g_3,g_4\}$. Then, $\mathcal{O}$ represents the multi-set $\{\{g_1,g_2\}, \{g_1,g_3\},\{g_3,g_4\}\}$ with $\delta(\{g_1,g_2\}) = [2,3]$ (meaning that the interval of values $[2,3]$ is related to the sources g_1 and g_2), $\delta(\{g_1,g_3\}) = [4,5]$ and $\delta(\{g_3,g_4\}) = [6,7]$. Then, we use an interval union as a meet operator. The resulting concept lattice is given in Figure 4.

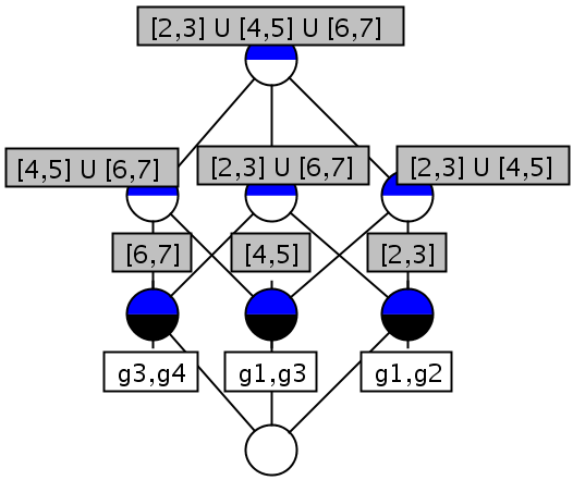


Figure 4: Concept lattice with MCS

A concept extent is read with reduced labelling. A concept intent is given here for each concept. For example, in Figure 4, the right concept in the second line is $(\{\{g_1, g_2\}, \{g_1, g_3\}\}, [2, 3] \cup [4, 5])$ giving the values of m_1 w.r.t. the sources $\{g_1, g_2\}$ and $\{g_1, g_3\}$. Moreover, these values represent the MCS fusion result of the subset $\{g_1, g_2, g_3\}$. The concept $\top$ corresponds to the union of all MCS that is the MCS fusion result of all sources.

The method used here to obtain the lattice based on MCS does not consider all subsets of objects with their MCS fusion results. This is due to the non-associativity of the MCS fusion operator. Thus, the concept lattice does not contain all subsets of G with their MCS fusion results since the interval union is used on the MCS of data and not directly on the data given by sources. Nevertheless, the concept lattice helps us to keep the origin of the information and gives more flexibility for the users in the choice of a maximal consistent subset of sources in many application fields.

3.4 Handling several variables simultaneously

Sources can provide values for different variables. For example, Table 1 involves objects described by vectors of intervals, where each dimension, i.e. column, corresponds to a unique variable, e.g. the description of the object g_1 is denoted by $\delta(g_1) = \langle [1, 5], [1, 9] \rangle$. It can be interesting to compute the fusion information for all variables simultaneously.

To formalize a pattern structure in this case, one defines a meet operator, i.e. fusion operator in our settings, for each dimension, or variable. Assuming that there is a canonical order on vector dimensions, the meet of two vectors is defined as the meet on each dimension. This induces a partial order of object descriptions [59]. Thus, we consider the pattern structure $(G, (D, \sqcap), \delta)$, where G is a set of sources, $(D, \sqcap)$ is a meet-semi-lattice of vectors, and each vector dimension is provided with the fusion operator f_m corresponding to the variable m.

Going back to Table 1, descriptions of objects g_1 and g_2 are respectively the vectors $\langle [1, 5], [1, 9] \rangle$ and $\langle [2, 3], [1, 3] \rangle$. When the fusion operator for both dimension is the interval intersection, the meet of these two vectors is $\langle [1, 5], [1, 9] \rangle \sqcap \langle [2, 3], [1, 3] \rangle = \langle [2, 3], [1, 3] \rangle$. The subsumption relation for vectors is defined similarly: $\langle [2, 3], [1, 3] \rangle \sqsubseteq \langle [1, 5], [1, 9] \rangle$ as $[2, 3] \subseteq [1, 5]$ and $[1, 3] \subseteq [1, 9]$. Then, the general Galois connection can be used to compute and order concepts:

$$\begin{array}{rclcrcl} \{g_1, g_2\}^{\square} & = & \langle [1, 5], [1, 9] \rangle \sqcap \langle [2, 3], [1, 3] \rangle & \quad & \langle [2, 3], [1, 3] \rangle^{\square} & = & \{g | \langle [2, 3], [1, 3] \rangle \sqsubseteq \delta(g)\} \\ & = & \langle [2, 3], [1, 3] \rangle & & & = & \{g_1, g_2\} \end{array}$$

In this way, a concept represents a set of sources and their fusion w.r.t. all variables, such as no other source can be added without changing the fusion result for any variable. The variables can be either symbolic or numerical since a fusion operator is chosen for each variable.

When the fusion operator is based on MCS, we follow the pre-processing introduced above for each variable (see Section 6.3.3). Then, we consider the set of all MCS for all variables. Thus, we consider the pattern structure $(\mathcal{O}, (K, \sqcap), \delta)$, where $\mathcal{O}$ is the set of subsets of sources providing the MCS for all variables, $(K, \sqcap)$ is a meet-semi-lattice of vectors. Each subset in $\mathcal{O}$ is described for each dimension by a maximal interval of values if the subset represents a MCS for the corresponding dimension, otherwise the dimension description is empty. In the example, recalling that an object denotes a set of sources giving a MCS, the description of the object $\{g_1, g_2\}$ is $\delta(\{g_1, g_2\}) = \langle [2, 3], [1, 3] \rangle$ where $[2, 3]$ and $[1, 3]$ are respectively a MCS for m_1 and m_2. By contrast, the description of the object $\{g_3, g_4\}$ is $\delta(\{g_3, g_4\}) = \langle [6, 7], \emptyset \rangle$ since the subset $\{g_3, g_4\}$ does not represent a MCS for the variable m_2.

4 Application

In this section, we show the usefulness of our framework on fusion operators on real-world data. We first recall the problem of indicator computation presented in the last chapter.

4.1 Data and problem settings

Agronomists compute indicators for evaluating the impact of agricultural practices on the environment. Questions such as the following are of importance: what are the consequences of the application of a pesticide given its characteristic, the period of application, and the characteristics of the field? The risk level for a pesticide to reach groundwater is computed by the indicator I_{gro} in [127]. Agronomists try to make a diagnosis w.r.t. the value of I_{gro}. A value below 7 indicates that the farmer has to change its practices (pesticide, soil, date, etc.). By contrast, a value above 7 indicates that the practices of the farmer are environmental friendly [21]. Pesticide characteristics depend on the chemical characteristics of the product while pesticide period application and field characteristics depend on domain knowledge [21]. This knowledge lies in information sources among which books, databases, and expert knowledge in agronomy. Then values for some characteristics vary w.r.t. sources.

	DT50 day	koc L/kg
BUS	[2,74]	?
PM11	[15,72]	?
PM12	?	[44,940]
PM13	?	[44,940]
INRA	?	[1.08,8.98]
Com98	[2,6]	[17,160]
AGXf	[2,6]	[1.08,160]
AGXl	[15,74]	[1.08,160]

Table 2: Characteristics of *Sulcotrione*

Here, we are interested in the use of pesticide *sulcotrione* and its influence on the groundwater. Sulcotrione is a herbicide marketed since 1993. It is used to control a wide range of grasses weeds in maize crops. Sulcotrione is generally weakly absorbed by soils [9]. Three characteristics of *Sulcotrione* are needed to compute the indicator I_{gro}, namely $DT50$, koc, and ADI (more details on these characteristics can be found in [127], and are not crucial for the understanding of this chapter). Table 2 (simplified data) gives the values of the characteristics $DT50$ and koc according to 9 different information sources. The symbol "?" represents the case when the information source does not give data for the characteristic. The value of ADI for the *sulcotrione* is 0.00005. Agronomists look to find a suitable value for each characteristic to be considered for computing the I_{gro} indicator, hence facing an information fusion problem.

4.2 Method

To combine the different pieces of information, a common fusion operator has to be defined. In this application, (1) the sources are heterogeneous (2) no *a priori* knowledge about sources and characteristics is available. Therefore, an appropriate fusion operator is the MCS fusion operator. The MCS for the variable $DT50$ are $\mathbb{K}_1$ and $\mathbb{K}_2$, resp. $\mathbb{K}_3$ and $\mathbb{K}_4$ for koc (see Table 3). Table 4

$\mathbb{K}_1$	$\{BUS, Com98, AGXf\}$
$\mathbb{K}_2$	$\{BUS, PM11, AGXl\}$
$\mathbb{K}_3$	$\{INRA, Com98, AGXf, AGXl\}$
$\mathbb{K}_4$	$\{PM12, PM13, Com98, AGXf, AGXl\}$

Table 3: Label of all MCS

	DT50 (days)	koc (L/kg)
$\mathbb{K}_1$	[2,6]	$\emptyset$
$\mathbb{K}_2$	[15,72]	$\emptyset$
$\mathbb{K}_3$	$\emptyset$	[1.08,8.98]
$\mathbb{K}_4$	$\emptyset$	[44,160]

Table 4: Table 2 pre-processed

results from the pre-processing of Table 2, detailed in Section 6.3.3. The resulting concept lattice is given in Figure 5 with 16 concepts. A concept extent is read with reduced labelling. A concept intent is not given in vectorial form for sake of readability: it is read from the intents of sub-concepts, for example, the intent of the concept C_1 is $\{(DT50, [15, 72]), (koc, [44, 160])\}$. But, if two sub-concepts intents give different values for a same attribute, then the union of values is considered. For example, the intent of the concept C_2 is $\{(DT50, [2, 6] \cup [15, 72]), (koc, [44, 160])\}$ and its sub-concepts intents are $\{(DT50, [2, 6])\}$, $\{(DT50, [15, 72])\}$ and $\{(koc, [44, 160])\}$. Moreover, each concept intent in the lattice represents the MCS fusion result of the subset of sources in the extent. The highest concept in the lattice corresponds to the MCS fusion result of all sources for all characteristics. For example, the "most right-down" concept is $(\{\mathbb{K}_1\}, \{(DT50, [2, 6])\})$ where $[2, 6]$ is the MCS fusion result of the subset $\mathbb{K}_1 = \{BUS, Com98, AGXf\}$ and its "most right" super-concept is $(\{\mathbb{K}_1, \mathbb{K}_2\}, \{(DT50, [2, 6] \cup [15, 72])\}$ where $[2, 6] \cup [15, 72]$ is the fusion result of the set $\mathbb{K}_1 \cup \mathbb{K}_2 = \{BUS, PM11, AGXl, Com98, AGXf\}$.

4.3 Results and discussion

The computing of a lower and higher bound for the indicator and the consequences of the results on agronomic practices and pollution are detailed and discussed in [4], but will not be detailed here as this is not necessary. It is required to consider the convex hull of the fusion result for computing the indicator. The concept lattice allows the users of I_{gro} and experts to give several diagnosis for the farmer. For example, let us consider the concept $\top$ that represents the fusion

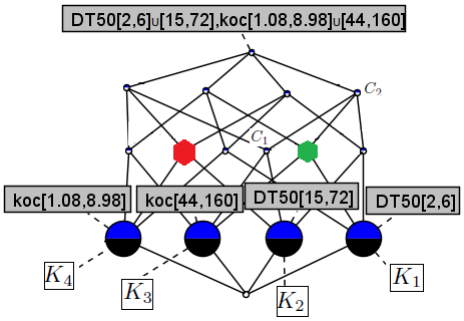


Figure 5: Concept lattice built from Table 4

result of all sources for all characteristics. Then, $DT50$ and koc lie respectively in $[2, 72]$ and $[1.08, 160]$. With these values, the computed value for I_{gro} is $[4, 10]$. This interval is not useful since all values in $[4, 10]$ are neither smaller than 7 nor greater than 7 and the expert cannot make a decision on the practices of the farmer.

Now the indicator I_{gro} can be also computed choosing either intervals of values in higher or lower level concepts. For instance, if we consider the values of $DT50$ in $[2, 6]$, koc in $[44, 160]$ then we obtain the interval $[9.97, 10]$ for I_{gro} and the practices of the farmer are environmental friendly since the I_{gro} value is greater than 7 (see the green concept on Figure 5). However, if $DT50 = [15, 72]$ and $koc = [1.08, 8.98]$, the resulting interval for I_{gro} is $[4.32, 4.32]$ indicating that the farmer must change its practices since values of I_{gro} are smaller than 7 (see the red concept in Figure 5). All other concepts either do not allow indicator computation (since having only one variable in their intent) or do not allow a decision, i.e. the indicator returns a value neither greater or smaller than 7.

Therefore, the two concepts (green and red) give precise results of I_{gro}, which its not the case of the Top concept, as usually used in the literature [4]. The concept lattice allows to identify what maximal subsets of sources support the most interesting results. It allows to characterize the "community of sources" in the dataset that are in agreement w.r.t. a common decision. The final decision stating that the agricultural practice is risky or not for the environment remains to the expert in agronomy. His choice is made easier, since he can study only the two communities (from the green and red concept extents) and use his own knowledge for the final decision.

5 Conclusion

This chapters claimed that Formal Concept Analysis has the capability of supporting a decision making process in the presence of information fusion problems, even when information are complex, e.g. patterns of numbers, thanks to the formalism of pattern structures. A real-world experiment in agronomy shows that when a fusion result does not allow to make a decision, the concept lattice helps the expert by considering an ordered hierarchy of concepts, given the fusion from different maximal sets of sources. Some fusion operators can directly be used to build a concept lattice, e.g. conjunctive and disjunctive operators. To deal with the operator based on maximal coherent subsets (MCS), we proposed to transform the data since MCS is not an associative operator, and the resulting concept lattice entails fusion results of interest. We argue that the concept lattice enhances the expert decision since he cannot (i) either consider all sources simultaneously, (i) or choosing a particular source for each variable, or (iii) study all the 2^n subsets of sources. Moreover, the whole process is automatic, i.e. it does not require human interaction before final decision.

We have considered the case when information are represented by fuzzy intervals and possibility distributions in [6], but do not detail this work in the present thesis (details can also be found in [3]). As a perspective, it is interesting to study how other fusion operators can be embedded in a concept lattice, as well as meta-information on sources (when available).

This work should be extended with Relational Concept Analysis (RCA) [51]. This extension of FCA to relational binary data allows to consider binary relations between sources for the lattice construction, e.g. the relation "works with" when sources are domain experts. This leads to the perspective of adapting RCA for pattern structures.

Another perspective can be expressed as follows. The concept lattice helps to select maximal subsets of sources that agree on a decision. Then, once these subsets are found, the expert has to choose which community he trusts to make his final decision. Now consider that statement in an

opposite way: the expert wants to take a particular decision and needs other sources to support him. The lattice helps him to the find the appropriate community, for each different variable. This is relevant in different domains, such as politic, economics or even social networks. Indeed, the community is not defined on the properties or attributes values the members share, but on a resulting indicator computed from these properties or values.

Chapter 7

A study on closed interval patterns and their generators

This chapter is a deeper investigation of Chapter 4. The aim is to properly define, characterize and extract with efficient algorithms frequent closed interval patterns and their generators. Indeed, pattern structures can be efficiently used to characterize and extract those patterns. We design and experiment two original and efficient algorithms for extracting frequent closed patterns and their generators in numerical data. We conclude showing the usefulness of such patterns, e.g. in classification problems and privacy preserving data-mining.

1 Motivations

In Chapter 4, we showed, in the context of gene expression data mining, how to introduce pattern structures for numerical data, and how to extract closed interval patterns. Intuitively, an interval pattern is a vector of intervals, each dimension corresponding to a range of values of a given attribute. An interval pattern d is closed if no interval pattern e exists with same image ($d^{\square} = e^{\square}$) and smaller intervals ($d \sqsubseteq e$). Since $(.)^{\square\square}$ is a closure operator, it should exist so called classes of equivalence of interval patterns (with same image), each class having a maximal element (closed) and one or more minimal elements (generators), w.r.t. a subsumption relation $\sqsubseteq$ defined on patterns.

Accordingly, we propose in this chapter to define, characterize and extract with efficient algorithms frequent closed interval patterns and their generators. After formalizing the problem from an itemset-mining point of view, we provide a semantic to interval patterns in the Euclidean space. This will help to properly define the notion of closed patterns and their generators. After, we argue that extracting generators from interordinal scaled contexts is still possible, but as for closed patterns, it is not efficient at all. Therefore, we introduce and experiment two algorithms for extracting these patterns directly from numerical data and show their efficiency. Finally, a discussion ends the chapter and highlights several perspectives and usages of such patterns.

Stated in this way, the problem of itemset-like pattern patterns in numerical data is usually referred as quantitative itemset/association rule mining [116]. Generally, an appropriate discretization splits attribute ranges into intervals maximizing some interest functions, e.g. support, confidence. However, none of these works discusses the notion of equivalence classes, closed patterns, and generators, and this is one of the originality of this work.

2 Problem definition

We propose a definition of interval patterns for numerical data following ideas of Chapter 4. Intuitively, each object of a numerical dataset is a vector of numbers, where each dimension corresponds to an attribute. Accordingly, an interval pattern is a vector of intervals, where each dimension describes the range of possible values for a given numerical attributes associated with some objects. We only consider finite intervals.

Definition 7.2.1 (Numerical dataset) *A numerical dataset is given by a set of objects G, a set of numerical attributes M, each attribute $m \in M$ having for range a set of real numbers W_m. We denote by $m(g) = w$ the fact that w is the value of attribute m for object g.*

	m_1	m_2	m_3
g_1	5	7	6
g_2	6	8	4
g_3	4	8	5
g_4	4	9	8
g_5	5	8	5

Table 1: A numerical dataset.

Definition 7.2.2 (Interval pattern and support) *In a numerical dataset, an interval pattern is a vector of intervals $d = \langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$ where $a_i, b_i \in W_{m_i}$, and each component corresponds to an attribute following a canonical order on vector dimensions, and $|M|$ denotes the number of attributes. An object g is in the image of an interval pattern $\langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$ when $m_i(g) \in [a_i, b_i]$, $\forall i \in \{1, ..., |M|\}$. The support $sup(d)$ of d is the cardinality of the image of d.*

Running example. Table 1 is a numerical dataset with objects in $G = \{g_1, ..., g_5\}$, attributes in $M = \{m_1, m_2, m_3\}$. The range of m_1 is $W_{m_1} = \{4, 5, 6\}$, and we have $m_1(g_1) = 5$. Here, we do not consider either missing values or multiple values for an attribute. $\langle [5, 6], [7, 8], [4, 6] \rangle$ is an interval pattern in Table 1, where a vector dimension i corresponds to an attribute m_i. Its image is $\{g_1, g_2, g_5\}$ and its support is 3.

Definition 7.2.3 (Interval pattern search space) *Given a set of attributes $M = \{m_i\}_{i \in \{1,|M|\}}$, the search space of interval patterns is the set D of all interval vectors $\langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$, with $a_i, b_i \in W_{m_i}$ and $a_i \leq b_i$. The size of the search space is given by*

$$|D| = \prod_{i \in \{1,...,|M|\}} \frac{|W_{m_i}| \times (|W_{m_i}| + 1)}{2}$$

where $\frac{|W_{m_i}| \times (|W_{m_i}|+1)}{2}$ is the number of possible intervals for the attribute m_i.

For example, all possible intervals for m_1 are in $\{[4, 4], [5, 5], [6, 6], [4, 5], [5, 6], [4, 6]\}$. Considering also attributes m_2 and m_3, the interval pattern search space is naturally larger, composed of $6 \times 6 \times 10 = 360$ interval patterns in our example. Among well-known solutions to deal with "pattern flooding" in data-mining, one is to efficiently mine frequent patterns, i.e. patterns having support greater than a given threshold, while a second is to define condensed representations

of patterns [115], e.g. closed patterns, (minimal) generators (also called key-sets, free-sets), etc. While generators can be preferable to closed patterns following the minimum descriptions length principle [77], closed patterns and their generators are known to be crucial for extracting valid and interesting association rules [10]. Therefore, we discuss and solve the following problems.

Problem 1: *Mining frequent closed interval patterns.* Whereas an algorithm was proposed for mining closed interval patterns in Chapter 4, it addressed the dual problem of un-frequent interval patterns mining, i.e. with support smaller than given threshold. We propose the algorithm *MinIntChange* for efficiently mining frequent closed interval patterns. Most importantly, this algorithm is useful for considering the two next problems.

Problem 2: *Mining interval pattern generators.* Closed patterns determine equivalence classes. One should expect that these classes have minimal elements w.r.t. a subsumption relation on patterns, called *interval pattern generators*. We propose to characterize these notions and to design an algorithm to efficiently mine frequent generators, called *MinIntChangeG*.

Problem 3: *Associating generators to their closure. MinIntChangeG* can provide each generator with its closure, allowing to produce valid and confident association rules.

Problem 4: *Mining equivalent binary data.* In Chapter 4, we showed that numerical data can be turned into binary with a so-called interordinal scaling, and that resulting binary data (i) can be mined with existing itemset mining algorithms, and (ii) there is a one-to-one correspondence between closed interval patterns and closed itemsets. However, we showed that closed interval patterns have better representation, avoid a local redundancy, and are much more efficient to mine directly in numerical data. Therefore, we should ensure that the same holds for generators, and than our algorithms are more efficient that classical algorithms in these particular binary data.

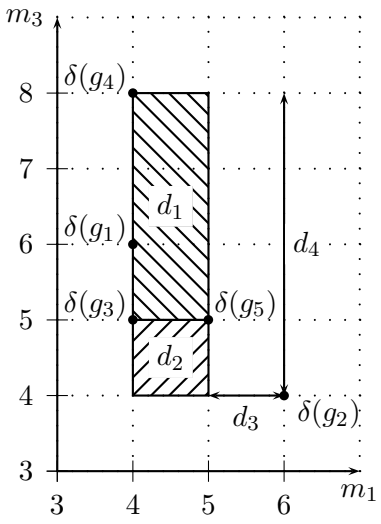
Before solving these problems, we properly define (frequent)(closed) interval patterns (and generators) and their semantics in $\mathbb{R}^{|\mathbb{M}|}$.

3 Interval patterns: semantics and definitions

Consider a numerical dataset with objects in G and numerical attributes in M. An interval pattern d is a $|M|$-dimensional vector of intervals, and can represented by a hyperrectangle (or rectangle for short) in Euclidean space $\mathbb{R}^{|M|}$, whose sides are parallel to the coordinate axes. This geometrical representation will be considered as the semantics of interval patterns. Formally, an interpretation is given by $\mathcal{I} = (\mathbb{R}^{|M|}, (.)^{\mathcal{I}})$ with $\mathbb{R}^{|M|}$ the interpretation domain, and $(.)^{\mathcal{I}} : D \to \mathbb{R}^{|M|}$ the interpretation function.

Example. When illustrating patterns in $\mathbb{R}^{|\mathbb{M}|}$, we consider the numerical dataset of Table 1 with attributes m_1 and m_3 only (it is more convenient here to work on two dimensions). The Figure 1 (left) gives four interval patterns d_1, d_2, d_3, d_4 and their representation in $\mathbb{R}^2$. In two dimensions, a pattern with two intervals with same left and right borders is a point, while a pattern having only one interval with same borders is a segment, e.g. d_3 and d_4. Otherwise, a pattern is represented by a rectangle, e.g. d_1 and d_2.

A basic idea in pattern mining is to define an intersection on patterns allowing to build more general patterns, i.e. shared by more objects. As stated in [46], the set-theoric intersection has the properties of an infimum $\sqcap$ in a semi-lattice $(D, \sqcap)$, i.e. idempotent, commutative, and associative. Accordingly, we introduced an infimum operation on interval patterns [66]:



$d_1 = \langle [4,5], [5,8] \rangle$
$d_1^{\square} = \{g_1, g_3, g_4, g_5\}$
$d_2 = \langle [4,5], [4,5] \rangle$
$d_2^{\square} = \{g_3, g_5\}$
$d_3 = \langle [5,6], [4,4] \rangle$
$d_3^{\square} = \{g_2\}$
$d_4 = \langle [6,6], [4,8] \rangle$
$d_4^{\square} = \{g_2\}$

Figure 1: Interval patterns in the Euclidean space.

Definition 7.3.1 (Infimum of Interval patterns) *The infimum of two interval patterns* $c = \langle [a_i, b_i] \rangle_{i \in \{1,\ldots,|M|\}}$ *and* $d = \langle [e_i, f_i] \rangle_{i \in \{1,\ldots,|M|\}}$ *is given by*

$$c \sqcap d = \langle [min(a_i, e_i), max(b_i, f_i)] \rangle_{i \in \{1,\ldots,|M|\}}$$

The infimum of several patterns is interpreted as the convex hull of their hyperrectangles in $\mathbb{R}^{|\mathbb{M}|}$, e.g. $d_1 \sqcap d_2 = \langle [4,5], [4,8] \rangle$ in Figure 1. This definition induces partial order, or subsomption relation $\sqsubseteq$ on interval patterns, knowing that $c \sqcap d = c \Leftrightarrow c \sqsubseteq d$.

Definition 7.3.2 (Subsumption relation) *Given two interval patterns* c *and* d, $c \sqsubseteq d$ *holds if* $d^{\mathcal{I}} \subseteq c^{\mathcal{I}}$.

This means that two interval patterns c and d are comparable whenever $c^{\mathcal{I}} \subseteq d^{\mathcal{I}}$ or $d^{\mathcal{I}} \subseteq c^{\mathcal{I}}$ and that patterns with "larger" intervals are subsumed by patterns with "smaller" intervals. For example, $\langle [4,5], [4,8] \rangle \sqsubseteq \langle [4,5], [4,5] \rangle$ but $\langle [4,5], [4,5] \rangle$ and $\langle [4,5], [5,8] \rangle$ are not comparable.

Example. We consider in this example one-dimensional interval patterns. Choosing attribute m_1 from Table 1, the set of all possible interval patterns is $D_{m_1} = \{[4,4], [5,5], [6,6], [4,5], [5,6], [4,6]\}$. The semi-lattice $(D, \sqcap)$, or equivalently $(D, \sqsubseteq)$ is given in Figure 2. The interval labelling a node is the infimum of all intervals labelling its descending nodes, e.g. $[4,5] = [4,4] \sqcap [5,5]$, and is also subsumed by these intervals, e.g. $[4,5] \sqsubseteq [5,5]$ and $[4,5] \sqsubseteq [4,4]$.

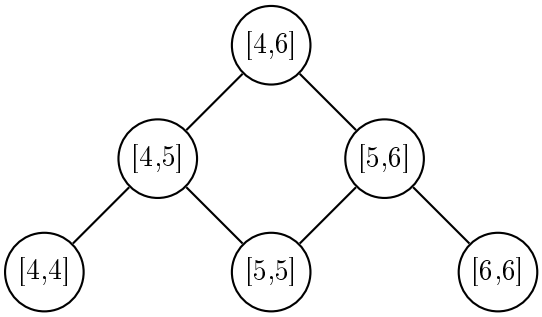


Figure 2: Diagram of $(D_{m_1}, \sqcap)$ or equivalently$(D_{m_1}, \sqsubseteq)$.

Finally, the support of an interval pattern d is interpreted as the the number of objects described by a rectangle included in $d^{\mathcal{I}}$, e.g. support of d_1 is four in Figure 1, with $\delta(g)$ represents the rectangle describing object $g \in G$.

The following definitions formally define pattern structures, involving a closure operator on patterns, based on a Galois connection. Pattern structure is an extension of well-know *formal contexts* (binary tables) to complex data in FCA [47, 46].

Definition 7.3.3 (Pattern structure) *Let G be a set of objects, let $(D, \sqcap)$ be a meet-semi-lattice of object descriptions, called patterns, and let $\delta : G \longrightarrow D$ be a mapping: $(G, (D, \sqcap), \delta)$ is called a pattern structure.*

Definition 7.3.4 *Let the two following operators $(.)^{\square}$ defined as follows.*

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

These operators form a Galois connection between $(\mathfrak{P}(G), \subseteq)$ and $(D, \sqsubseteq)$. The operator $(.)^{\square\square}$ is a closure operator.

Example. Considering the example of Table 1. $(D, \sqsubseteq)$ is the finite ordered set of all interval patterns. $\delta(g) \in D$ is the pattern associated to an object $g \in G$. Then:

$$\begin{aligned} \langle [5,6],[7,8],[4,8]\rangle^{\square} &= \{g \in G | \langle [5,6],[7,8],[4,8]\rangle \sqsubseteq \delta(g)\} \\ &= \{g_1, g_2, g_5\} \\ \{g_1, g_2, g_5\}^{\square} &= \delta(g_1) \sqcap \delta(g_2) \sqcap \delta(g_3) \\ &= \langle [5,6],[7,8],[4,6]\rangle \end{aligned}$$

This means that $\langle [5,6],[7,8],[4,8]\rangle$ is not a closed interval pattern, its closure being $\langle [5,6],[7,8],[4,6]\rangle$. The first operator applies to an arbitrary description $d \in (D, \sqcap)$ and returns the set of objects described by rectangles included in $d^{\mathcal{I}}$. Dually, the second operator applies to a of objects $A \subseteq G$ and returns the convex hull of their interpretation, i.e. a rectangle.

Based on these definitions, we now define the notions of (frequent) closed interval pattern ((F)CIP), equivalence classes of patterns and (frequent) interval patterns generators ((F)IPG), adapted from the classical binary case [96]. We illustrate these definitions with two dimensional interval patterns, and their representation in Figure 1, i.e. considering attributes m_1 and m_3 only.

Definition 7.3.5 (Equivalence class) *Let image(d) be the function that assigns to each interval pattern the set of objects supporting d, i.e. image$(d) = d^{\square}$. Two interval patterns c and d are said equivalent iff they have the same image and we write $c \cong d$. The set of patterns that are equivalent to a pattern d is denoted by $[d] = \{c | c \cong d\}$ and is called the equivalence class of d.*

Example. $\langle [4,5],[6,8]\rangle \cong \langle [4,6],[6,8]\rangle$ as they have the same image $\{g_1, g_4\}$.

Definition 7.3.6 (Closed interval pattern) *A pattern d is closed if there does not exist any pattern e such as $d \sqsubseteq e$ with $d \cong e$.*

Example. $\langle [4,6],[6,8]\rangle$ is not closed as $\langle [4,6],[6,8]\rangle \sqsubseteq \langle [4,5],[6,8]\rangle$, these two patterns having same image, i.e. $\{g_1, g_3, g_4, g_5\}$. $\langle [4,6],[6,8]\rangle$ is closed.

Definition 7.3.7 (Interval pattern generator) *A pattern d is a generator if there does not exist a pattern e such as $e \sqsubseteq d$ with $d \cong e$.*

Example. $\langle[4,6],[5,8]\rangle$ and $\langle[4,5],[4,8]\rangle$ are the generators of the closed interval pattern $d_1 = \langle[4,5],[5,8]\rangle$ with image $\{g_1, g_3, g_4, g_5\}$.

Definition 7.3.8 (Frequent Interval pattern) *A pattern d is frequent if its image has a higher cardinality than a given minimal support threshold minSup, i.e $|d^{\square}| \geq minSup$. Otherwise, d is not frequent.*

Example. Among the four patterns in Figure 1, d_1 is the only frequent interval pattern with $minSup = 3$.

An equivalence class is a set of interval patterns having the same image. According to the defined closure operator, each class is provided with a unique CIP. The interpretation of this closed pattern is the rectangle with smallest area, while generators are rectangles with largest area.

We dedicate a particular attention to interval patterns with null support. In Figure 1, such patterns correspond to rectangles, segments or points containing no object description from the dataset, e.g. $c_1 = \langle[6,6],[5,8]\rangle$, $c_2 = \langle[5,6],[6,8]\rangle$, $c_3 = \langle[4,4],[4,4]\rangle$. Such patterns would not exist if each point in the rectangle $\langle[4,6],[4,8]\rangle$ were covered by some object of the dataset (since the search space is finite). If interval patterns with null support exist, their equivalence class should have a closed element with one or more generators. However, the closed pattern of null support does not exist, since it should subsume any closed pattern of support 1. Any CIP with support 1 is defined by $g^{\square}$ for some $g \in G$. Since dealing with numerical attributes with domains values in $\mathbb{R}$, intervals of $g^{\square}$ are degenerate (same left and right borders), e.g. $\delta(g1) = \langle[5,5],[7,7],[6,6]\rangle$. Therefore, we cannot find a subsumer of this pattern: it is not defined (any degenerate interval has no subintervals). When existing, the generators of null support provide a meaningful information: it characterizes the largest subspaces of the data covered by no objects.

4 Interval patterns in binary data

In this section, we recall how numerical data can be turned into binary with a so-called interordinal scaling. This data transformation is defined in the framework of formal concept analysis (FCA) [47], and allows to produce binary data from which interval patterns can be extracted (see Chapter 4). Most importantly, we show that, in these particular binary data, collections of closed itemsets and generators highlight two forms of redundancy, leading to design efficient algorithms working directly on numerical data in the next section.

4.1 Interordinal Scaling

Conceptual scaling is often used for discretizing numerical data and obtaining a (binary) formal context [47]. Given a numerical attribute, the search space of interval patterns can be expressed in terms of binary attributes, or items, thanks to *interordinal scaling*. We recall here a basic definition while more details lie in [47, 66, 73].

In FCA, a numerical dataset is described by a many-valued context (G, M, W, J) where G is a set of objects, M a set of numerical attributes, W a set of real numbers, and J a ternary

$m_3 \geq 8$				×	
$m_3 \geq 6$	×			×	
$m_3 \geq 5$	×		×	×	×
$m_3 \geq 4$	×	×	×	×	×
$m_3 \leq 8$	×	×	×	×	×
$m_3 \leq 6$	×	×	×		×
$m_3 \leq 5$		×	×		×
$m_3 \leq 4$		×			
$m_2 \geq 9$				×	
$m_2 \geq 8$		×	×	×	×
$m_2 \geq 7$	×	×	×	×	×
$m_2 \leq 9$	×	×	×	×	×
$m_2 \leq 8$	×	×	×		×
$m_2 \leq 7$	×				
$m_1 \geq 6$		×			
$m_1 \geq 5$	×	×			×
$m_1 \geq 4$	×	×	×	×	×
$m_1 \leq 6$	×	×	×	×	×
$m_1 \leq 5$	×		×	×	×
$m_1 \leq 4$			×	×	
	g_1	g_2	g_3	g_4	g_5

Table 2: Interordinally scaled context encoding the dataset from Table 1.

relation defined on the Cartesian product $G \times M \times W$. $(g, m, w) \in J$ or simply $m(g) = w$ means that the object g takes the value w for the attribute m.

Definition 7.4.1 (Interordinal scaling) *Given a numerical attribute m with value domain the set W_m of real numbers, interordinal scaling builds $2 \times |W_m|$ binary attributes, denoted by "$m \leq w$" and "$m \geq w$", $\forall w \in W_m$, called "interordinal scale attributes" or IS-items for short.*

Definition 7.4.2 (Interordinal scaled context) *A formal context (G, N, I) is an interordinal scaled context when it results from the application of interordinal scaling to numerical context (G, M, W, J). N is the set of all IS-items of the form "$m \leq w$" or "$m \geq w$" for each numerical attribute $m \in M$ and value $w \in W_m$. An object g has an IS-item "$m \leq w$" (resp. "$m \geq w$") iff $m(g) \leq w$ (resp. $m(g) \geq w$).*

Example. Table 2 gives the tabular representation of the interordinally scaled formal context built from Table 1. Object g_1 owns the IS-item $m_1 \leq 5$ (denoted by a cross $\times$) but not $m_1 \leq 4$ since $m_1(g_1) = 5$.

4.2 Interval Patterns and IS-Itemsets

It is possible to apply classical mining algorithms to process the binary table for extracting itemsets composed of IS-items. These itemsets are called IS-itemsets in the following, and are linked with interval patterns as follows [66].

An IS-itemset as an interval pattern. An IS-itemset P is composed of IS-items of the forms $m_i \leq w$ and $m_i \geq w$ for some $w \in W_{m_i}$. It is represented by the interval pattern $d = \langle [a_i, b_i] \rangle_{i \in \{1,\ldots,|M|\}}$, where

- a_i is the maximum of the values w in IS-items $m_i \geq w$, and $min(W_{m_i})$ if $m_i \geq w \notin P$.
- b_i is the minimum of the values w in IS-items $m_i \leq w$, and $max(W_{m_i})$ if $m_i \leq w \notin P$.

For example, $\{m_1 \leq 5, m_1 \leq 6, m_1 \geq 4, m_2 \leq 9, m_2 \geq 7\}$ corresponds to $\langle [4, 5], [7, 9], [4, 8] \rangle$, i.e. the smallest interval pattern w.r.t. $\sqsubseteq$ with same image.

An interval pattern as an IS-itemset. Let $d = \langle [a_i, b_i] \rangle_{i \in \{1,\ldots,|M|\}}$ be an interval pattern. An IS-itemset representing d is a set of IS-attributes, $\forall i \in [1, |M|]$.

- $m_i \leq b_i$ if $a_i = min(W_{m_i})$
- $m_i \geq a_i$ if $b_i = max(W_{m_i})$
- $m_i \geq a_i$ and $m_i \leq b_i$ otherwise.

For example, the IS-itemset corresponding to $\langle [4, 5][7, 9][4, 8] \rangle$ is $\{m_1 \leq 5\}$, i.e. the smallest set of IS-items with same image.

We detail in the following some problem when mining IS-itemsets. First, we show that closed IS-itemsets involve a local redundancy making them hard to mine. Secondly, we show that IS-itemsets generators do not behave in the same way, but involve another kind of redundancy that alter their mining.

4.3 Local redundancy problem

Extracting all IS-itemsets in our example returns 31,487 IS-itemsets. This is surprising since there are only 360 possible interval patterns. In fact, a lot of IS-itemsets are locally redundant. For example, $\{m_1 \leq 5\}$ and $\{m_1 \leq 5, m_1 \leq 6\}$ both correspond to the interval pattern $\langle [4,5],[7,9],[4,8]\rangle$. Indeed, the constraint $m_1 \leq 6$ is weaker than $m_1 \leq 5$ on the set of values W_{m_1}.

Definition 7.4.3 *Given two IS-items $n_1, n_2 \in N$, with same sign $\leq$ or $\geq$ and numerical attribute, n_1 characterizes a weaker constraint than n_2 if $n_2' \subseteq n_1'$. n_1 is a redundant condition with respect to n_2.*

Proposition 7.4.1 *An arbitrary IS-itemset $N_1 \subseteq N$ is locally redundant iff it contains two IS-items such as one is a redundant condition with respect to the other one.*

Example. $\{m_1 \leq 5, m_1 \leq 6\}$ and $\{m_1 \leq 4, m_1 \leq 5, m_1 \leq 6\}$ are both locally redundant while $\{m_1 \leq 5\}$ and $\{m_1 \leq 5, m_3 \geq 5\}$ are not. Intuitively, in $\{m_1 \leq 5, m_1 \leq 6\}$ the item $m_1 \leq 6$ brings no new information on the description of the itemset image.

Proposition 7.4.2 *Except G', any closed IS-itemset $P \subseteq N$ is locally redundant and $|P| > 2|M|$.*

Proof. By definition of interordinal scaling, we have $G' = \{m_i \leq max(W_{m_i}), m_i \geq min(W_{m_i})\}_{\forall m_i \in M}$, hence $|G'| = 2|M|$. Any other closed itemset P is such that $G' \subset P$: it is locally redundant.

Proposition 7.4.3 *If $P \subseteq N$ is an IS-itemset generator, then $|P| \leq 2|M|$, and P is not locally redundant.*

Proof. Suppose that P is a generator with $|P| > 2|M|$. Since IS-items are of the form, either "$m \leq w$" or "$m \geq w$" for $m \in M$ and $w \in W_m$, P contains at least two itemsets of one of these form. Therefore, one characterizes a redundant condition and removing it from P does not change its image, leading to a contradiction. Moreover, if P_1 is redundant, $P_1 \subset P_2$ implies that P_2 is also redundant.

4.4 Global redundancy of generators

Due to local redundancy, we showed in Chapter 4 that closed IS-itemsets are hard to mine with classical closed itemset mining algorithms. It seems that IS-itemset generators have a good property to be mined, since not affected by local redundancy. But we remark here another kind of redundancy, called global redundancy: it happens that two different and incomparable IS-itemsets generators correspond to two different interval pattern generators, but one subsuming the other, i.e. one is not an interval pattern generator according to the semantic in $\mathbb{R}$. For example, taking the binary table 2, both IS-itemsets $N_1 = \{m_1 \leq 4, m_3 \leq 5\}$ and $N_2 = \{m_1 \leq 4, m_3 \leq 6\}$, with same image $\{g_3\}$ are generators, i.e. there does not exist a subset of these itemsets with same image. However, their corresponding interval pattern are respectively $c = \langle [4,4],[7,9],[4,5]\rangle$ and $d = \langle [4,4],[7,9],[4,6]\rangle$ and we have $d \sqsubseteq c$, while $c^{\square} = d^{\square}$, hence c is not an interval pattern generator. This is due to the fact $m_3 \leq 6$ is a redundant condition with respect to $m_3 \leq 5$, the only IS-items that differ from N_2 to N_1.

Due to this redundancy problem, it should be not only more efficient to directly explore the search-space of interval patterns but also provide correctness. This is the aim of the next section.

5 Algorithms

In this section, we first detail a depth-first enumeration of interval patterns, starting with the most frequent one. Based on this enumeration, we design the algorithm *MinIntChange* for extracting frequent closed interval patterns (FCIP). This algorithm needs slight modifications to compute frequent interval pattern generators (FIPG), giving the algorithm *MinIntChangeG*.

5.1 Greedy enumeration

Consider firstly one numerical attribute of the example, say m_1. Its semi-lattice of intervals $(D_{m_1}, \sqcap)$ is composed of all possible intervals with borders in W_{m_1} and is ordered by the subsumption relation given in Section 7.3. The unique smallest element w.r.t. $\sqsubseteq$ is the interval with maximal size, i.e. $[4,6] = [min(W_{m_1}), max(W_{m_1})]$ and maximal frequency (here 5). The basic idea of pattern generation lies in *minimal changes* for generating the direct subsumers of a given pattern. For example, two minimal changes can be applied to $[4,6]$. The first consists in replacing the right border with the value of W_{m_1} immediately lower that 6, i.e. 5, for generating the interval $[4,5]$. The second consists in repeating the same operation for the left border, generating the interval $[5,6]$. Repeating these two operations allows to enumerate all elements of $(D_{m_1}, \sqcap)$. A right minimal change is defined formally as, given $a, b, v \in W_m$, $a \neq b$,

$$minChangeR([a,b]) = [a,v] \mid v < b,\ \nexists w \in W_m \text{ s.t. } v < w < b$$

while a left minimal change $minChangeL([a,b])$ is formally defined similarly. Minimal changes give direct next subsumers and implies a monotonicity property of frequency, i.e. support of $[a,v]$ is less or equal than support of $[a,b]$.

The generalization to several attributes is straightforward: for each pattern there are $2.|M|$ minimal changes for modifying the left and the right border for each attribute.

5.2 Lexicographical enumeration

The greedy enumeration is based on minimal changes but does not prevent redundancy since a pattern can be generated several times. For example, considering the attribute m_1, interval $[5,5]$ is generated two times: from $[4,6]$ applying a right then a left minimal change, or applying a left then a right minimal change (indeed, we can see in Figure 2 that $[5,5]$ subsumes two different patterns having a common subsumee $[4,6]$).

To avoid redundancy, a lectic order on changes, or equivalently on patterns, is defined: after a right change, one can apply either a right or left change; after a left change one can apply only a left change. Figure 3 shows the depth-first traversal (numbered arrows) of diagram of $(D_{m_1}, \sqcap)$. Backtracks occur when an interval of the form $[w,w]$ is reached ($w \in W_{m_1}$), or no more change can be applied. Therefore, generated elements form a tree traversed depth first.

This pattern generation can be seen as a classical enumeration used by depth-first algorithms in data-mining. Indeed, each minimal change is the interpretation of an IS-item. Recall that IS-items are of the form "$m \leq w$" or "$m \geq w$". Applying a change $minChangeR([a,b]) = [a,v]$ to a interval pattern is equivalent to add the IS-item "$m \leq v$" in a corresponding IS-itemset. Dually, $minChangeL([a,b]) = [v,b]$ consists in the IS-item "$m \geq v$". These IS-items characterizing minimal changes are drawn on Figure 3. This figure accordingly represents a prefix-tree, factoring out the effort to process common prefixes or minimal changes.

Therefore, the lectic order can be also expressed in terms of IS-items. Any IS-item containing the symbol $\leq$ precedes any IS-item containing $\geq$. Secondly, if both IS-items contains $\leq$, the

one with the largest value w precedes the other one. Dually, if both IS-items contains $\geq$, the one with the smallest value w precedes the other one. Notice that IS-items having the form "$m \leq max(W_m)$" or "$m_1 \geq min(W_m)$" are not considered since they do not characterize minimal changes.

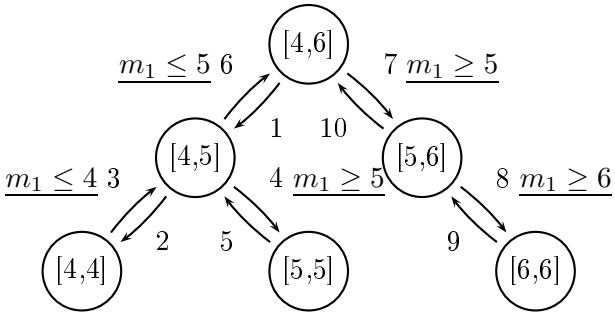


Figure 3: Depth-first traversal of $(D_{m_1}, \sqcap)$.

The generalization to several attribute is again straightforward. A lectic order is classically defined on numerical attributes as a lexicographic order, e.g. $m_1 < m_2 < m_3$. Then changes are applied as explained above for all attributes respecting this order. For example, after applying a change to attribute m_2, one cannot apply a change to attribute m_1 since $m_1 < m_2$. On the example of Table 1, considering that $\langle [4,5], [8,9], [5,8] \rangle$ was previously generated from a left minimal change of a pattern for attribute m_2, only three patterns can be generated in the next step, namely, $\langle [4,5], [9,9], [5,8] \rangle$ (change on m_2 left), $\langle [4,5], [8,9], [5,6] \rangle$ (change on m_3 right) and $\langle [4,5], [8,9], [6,8] \rangle$ (change on m_3 left).

5.3 Extracting closed interval patterns

The pattern enumeration starts with the minimal pattern w.r.t $\sqsubseteq$ and generates its direct subsumers with lower or equal support. The next problem now is that minimal changes do not necessarily generate patterns with strictly smaller support. Therefore, we should apply changes until a pattern with different support is generated to identify a closed interval pattern (FCIP) but this would not be efficient.

However, applying a minimal change does not mandatory implies that resulting pattern has strictly smaller support. Therefore, we should apply changes until the support changes to flag a FCIP. This would be not efficient as it required to generate the whole set of frequent interval patterns. We adopt the idea of the algorithm *CloseByOne* [74]: before applying a minimal change, the closure operator $(.)^{\square\square}$ is applied to the current pattern, allowing to skip all equivalent patterns. Indeed, the minimal pattern d w.r.t. $\sqsubseteq$ is closed as it is given by $d = G^{\square}$. Applying a minimal change returns a pattern c with strictly smaller support, since $d \sqsubseteq c$ and d is closed. If c is frequent, we can continue, apply the closure operator and next changes in lectic order, allowing to completely enumerate all FCIP.

Example. We start from the minimal pattern $c = \langle [4,6], [7,9], [4,8] \rangle$. The first minimal change in lectic order is a right change on attribute m_1. We obtain pattern $d = \langle [4,5], [7,9], [4,8] \rangle$, and obviously $c \sqsubseteq d$. However, $d^{\square\square} = \langle [4,5], [7,9], [5,8] \rangle$, hence d is not closed. Next change will be applied to $d^{\square\square}$.

Since a FCIP may have several different associated generators, it can be generated several times. Still following the idea of *CloseByOne*, a canonicity test can be defined according to lectic order minimal changes: if a pattern d has been generated by a change at attribute $m_j \in M$, it

is canonically generated iff d and $d^{\square\square}$ do not differ for any attribute $m_h \in M$ such as $m_h < m_j$. This test avoids lookup in memory (e.g. using an hashtable of FCIP).

Example. Given the minimal pattern $\langle[4,6],[7,9],[4,8]\rangle$ and the pattern obtained by minimal change on left border for attribute m_3, i.e. $d = \langle[4,6],[7,9],[5,8]\rangle$. We have $d^{\square\square} = \langle[4,5],[7,9],[5,8]\rangle$. We observe that d and $d^{\square\square}$ present a difference for attribute m_1, but d has been generated from a change on m_3. Since $m_1 < m_3$, $d^{\square\square}$ is not canonical and has already been generated (see previous example): it is no more necessary to apply minimal changes to $d^{\square\square}$. Since this FCIP has already been generated, the algorithm backtracks, indicated by $d^{\square\square} <_D d$ in the algorithm given below.

MinIntChange. The algorithm is initialized as follow. G is the set of objects. $G^{\square}$ is the most frequent pattern and minimal w.r.t $\sqsubseteq$. Two integers are used to indicate the current minimal change (attribute and border). A minimal frequency min_{supp} is also given.

Alg. 3 MinIntChange()

1: $FCIP = \emptyset$; // *the FCIP set*
2: process($G^{\square}$,0,0,G,$G^{\square}$);

Given a generated closed pattern d, the main procedure firstly checks whether d is frequent and tests canonicity. If one of these test fails, the algorithm backtracks. Otherwise the current pattern d is stored as being a FCIP not previously generated. Then, the algorithm applies minimal changes to d following the lectic order (from attribute n and border p), computes closure and the procedure is called again. The procedure backtracks when no more minimal changes to current FCIP can be applied. The notation $\delta_{n,l}(d)$ returns the left border of the interval describing attribute n in d while $\delta_{n,r}(d)$ returns its right border. The peusdo code of the procedures $minChangeRight(d,n)$ and $minChangeLeft(d,n)$ is not given for sake of simplicity. It consists in applying the minimal change as previously defined (see *minChangeR([a,b])*) but for a given attribute, namely n. Accordingly, 18 FCIP are extracted from Table 1 with $min_{supp} = 1$. Note that the CIP of null support cannot be extracted if the user specifies $min_{supp} = 0$. The algorithm

Alg. 4 process(c, m, p, A, d), c was generated at previous step with a minimal change on attribute m and border p (p=0 means right, p=1 means left), $A = c^{\square}$ and $d = c^{\square\square}$

if ($|A| \leq min_{sup}$ **or** $d <_D c$) **then**
2: **return**;
end if
4: $FCIP \leftarrow FCIP \cup d$
for $i = m$ **to** $|M|$ **step** 1 **do**
6: **if** ($\delta_{i,l}(d) = \delta_{i,r}(d)$) **then**
continue;
8: **end if**
if ($i = m$ **and** $p = 1$) = **false then**
10: $patR \leftarrow minChangeRight(d,i)$
process(patR, i, 0, $patR^{\square}$, $patR^{\square\square}$);
12: **end if**
$patL \leftarrow minChangeLeft(d,i)$;
14: process(patL, i, 1, $patL^{\square}$, $patL^{\square\square}$);
end for

operates a bounded number of $2|M| \times |FCIP|$ minimal changes. Complexity of minimal change procedure is $log(W_m)$, i.e. getting the next value in a previously sorted set. For each change, closure is computed. First operator $(.)^{\square}$ returns the image of d and requires to scan objects in G and test if their description subsumes d. Actually, it its not needed to scan the whole set of objects, but only those in the image of the previously generated closed pattern. The second operator $(.)^{\square}$ applies to a set of objects, and returns the convex hull of their description in $\mathbb{R}^{|M|}$, requiring only computations of minima and maxima on each dimension separately.

5.4 Extracting interval pattern generators

We now adapt *MinIntChange* to extract FIPG. Indeed, applying the closure operator to a generated pattern is still important: for any FCIP d, a minimal change implies that the support of the resulting pattern c is strictly smaller than the support of d. Therefore, c is a good generator candidate of the next FCIP. However, when applying the closure to this candidate, "equivalent changes can be added" and are not necessary to store for the next generator. This is made clearer with an example.

Example. Consider the pattern $\langle [4,5],[7,9],[4,8]\rangle$ obtained with a right minimal change on the smallest pattern w.r.t $\sqsubseteq$, and characterized by the IS-items "$m_1 \leq 5$". Now consider its closure, i.e. $\langle [4,5],[7,9],[4,8]\rangle^{\square\square} = \langle [4,5],[7,9],[5,8]\rangle$. The closure adds one change, namely "$m_3 \geq 5$". Actually, it can be shown that the changes "$m_1 \leq 5$" and "$m_3 \geq 5$" are equivalent as they characterizing the same image.

Since a generator is characterized by a smallest set of minimal changes as possible (having largest intervals in its equivalence class), we should not consider the changes "added" by the closure. This can also be understood with Propositions 7.4.2 and 7.4.3.

At each step of the depth-first enumeration is generated a FGIP candidate. We know that it has no subsumers in its branch with same support. However, it could exist a branch with another FGIP with same image and resulting from less changes. Regarding to the lectic order on minimal changes, and already suggested in the binary case in [27], we should use a reverse traversal of the tree, see Figure 4. Therefore, if such pattern exists, i.e. the current candidate is not a generator, it has already been generated with few minimal changes. In this case, the algorithm backtracks: these two patterns have the same closure, hence the same minimal change will be used to build next candidate.

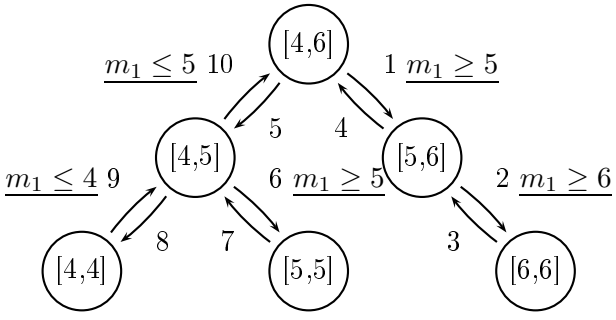


Figure 4: Reverse pre-order traversal of $(D_{m_1}, \sqcap)$.

MinIntChangeG. At the initialization step, we start from the minimal pattern d. This pattern d is both closed and generator, i.e. $d = G^{\square}$ while any change would also change its support. d is stored as FCIP and FGIP. At a given step, if the generator candidate is actually a

generator (see details after) and is frequent, the FCIP is used to characterize the next change. This change is applied to the FGIP to obtain the new candidate, the closure operator is applied to obtain its closure. Next step is called with resulting FCIP and the new FGIP. This means that a FGIP is characterized by a minimal set of changes (branches in the tree), while the FCIP is characterized by the maximal set of changes (branches plus changes added by successive closures). Notice that the canonicity test cannot be used anymore, since a FCIP may have several generators, characterized by different minimal sets of changes.

Alg. 5 MinIntChangeG

$FIPG = \emptyset$;
processGen(0,0,G,$G^{\square}$,$G^{\square}$);

Alg. 6 processGen(m, p, A, d, cand): *cand* is the current candidate, $cand^{\square} = A$, $A^{\square} = d$

if $|A| \leq min_{sup}$ **or** addCandidate(cand) = false **then**
  return;
end if
$FIPG = FIPG \cup cand$;
for i=$|M|$ **to** m **step - 1 do**
  if $\delta_{i,l}(d) = \delta_{i,r}(d)$ **then**
    continue;
  end if
  $clone \leftarrow cand$
  $\delta_{i,l}(clone) \leftarrow \delta_{i,l}(minChangeLeft(d,i))$;
  process(i, 1, $clone^{\square}$, $clone^{\square\square}$,$clone$);
  if $(i = m$ **and** $p = 1)$ = **false then**
    $clone \leftarrow cand$
    $\delta_{i,r}(clone) \leftarrow \delta_{i,r}(minChangeRight(d,i))$;
    process(i, 0, $clone^{\square}$, $clone^{\square\square}$,$clone$);
  end if
end for

Fast subsumption checking with hastable. To test whether a candidate is a generator, we use the same technique as in the algorithm *Charm* [132]. *MinIntChange* hashes the FIPG upon their image. In the testing of a candidate d, the entire list corresponding to its hash value $h(d)$ is retrieved. If there is a FGIP c in the list with same support and such that $c \sqsubseteq d$, d is discarded, otherwise d is declared a FIPG and hashed.

Fast subsumption checking with a trie. A second possibility uses the trie structure (see e.g. [22] for more details). Each word of the trie is the image of a FCIP, and a list of its generators its attached. When testing whether a candidate is a generator or not, we look in the trie for its corresponding image (word) and only test the generators associated to this word. If one of them is subsumed by the candidate, the candidate is discarded, otherwise added to the list. Whereas this solution may be more efficient, it requires more storage space. Most importantly, it allows to associate any FIPG to its closure, answering to the problem 3.

6 Computer experiments

We evaluate the performances of the algorithms designed in Java, namely *MinIntChange*, *MinIntChangeG-h* with auxiliary hashtable and *MinIntChangeG-t* with auxiliary trie. Recalling that closed IS-itemsets and CIP are in 1-1-correspondence, we compare the performance for mining interordinal scaled data with the closed-itemset-mining algorithm *LCMv2*[18]. For studying the global redundancy effect of IS-itemset generators, we use the generator-mining-algorithm *GrGrowth*[19]. Both implementations in C++ are available from the authors. All experiments are conducted on a 2.50Ghz machine with 16GB RAM running under Linux 2.6.18-92.e15. We choose dataset from the Bilkent repository[20], namely Bolts (BL), Basketball (BK) and Airport (AP), AP being worst case where each attribute value is different.

First experiments compare *MinIntChange* for extracting FCIP and *LCMv2* for extracting equivalent frequent closed IS-itemsets in Table 3. Second experiments consist in extracting frequent interval pattern generators (FIPG) with *MinIntChange-h* and *MinIntChange-t*. We also extract frequent itemset generators (FISG) in corresponding binary data after interordinal scaling with *GrGrowth* for studying the global redundancy effect in Table 4.

Dataset	minSupp	MinIntChange	LCMv2	$\|FCIP\|$
BL	80%	< 50	< 50	1,130
	50%	252	**100**	32,107
	25%	1,215	**1,060**	171,192
	10%	**1,821**	1,950	268975
	1	**1,905**	2,090	272,223
AP	80%	4,595	**1,470**	346,741
	50%	**143,939**	149,580	16,214,345
	25%	**413,805**	899,180	58,373,631
	10%	**506,985**	6,810,125	80,504,566
	1	**517,548**	6,813,591	82,467,124

Table 3: Execution time for extracting FCIP (in ms).

In both cases, using binary data is better when the minimal support is high (e.g. 90%). For low supports, a critical issue, our algorithms deliver better execution times. Most importantly, the global redundancy effect discards the use of binary data, e.g. only 1.6% of all FISG are actually FIPG in dataset BL. Finally, the algorithm *MinIntChangeG-t* outperforms *MinIntchangeG-h*. *MinIntChangeG-t* however needs more memory since storing each closed set of objects as a word in the trie, and to each word the list of associated FIPG.

It is very interesting to analyse the compression ability of closed interval patterns and generators. For that, we compare in each dataset the number of those patterns w.r.t. to all possible interval patterns. It gives the ratio of closed (generators) in the whole search space. In both cases, ratio varies between 10^{-7} and 10^{-9}. This means that the volume of useful interval patterns, either closed or generators, is very low w.r.t. the set of all possible interval patterns. Thus, this demonstrates that our interest in equivalence classes for interval patterns is really justified.

7 Discussion

We presented a study on the characterization and the extraction of frequent closed interval patterns and their associated generators from numerical data. For this task, we designed the

[18]Winner of the FIMI'04 – http://fimi.ua.ac.be/src/

[19]http://www.comp.nus.edu.sg/ wongls/projects/pattern-spaces/

[20]http://funapp.cs.bilkent.edu.tr/

Dataset	minSupp	GrGrowth	MinIntChangeG-h	MinIntChangeG-t	$\|FIPG\|$	$\|FISG\|$	$\frac{\|FIPG\|}{\|FISG\|}$	$\|FCIP\|$	$\frac{\|FIPG\|}{\|FCIP\|}$
BL	90%	< 50	< 50	< 50	176	194	90%	112	1.57
	80%	< 50	< 50	< 50	1,952	2,823	69%	1,130	1.73
	50%	**150**	1,212	529	66,350	222,088	29%	32,107	2
	25%	**3,432**	27,988	3,893	411,442	3,559,419	11%	171,192	2.4
	1	123,564	438,214	**24,141**	1,165,824	69,646,301	**1.6**%	272,223	4.3
BK	90%	< 50	1,268	1,207	67,737	75,058	84%	48,847	1.3
	85%	**4,565**	26,154	12,139	554,956	799,574	69%	403,562	1.37
	80%	?	512,126	**107,700**	2,730,812	NA	NA	1,938,984	1.40

(?) means more than 20x execution time of MinIntChangeG-t.

Table 4: Execution time for extracting FIPG and global redundancy evaluation.

algorithms *MinIntChange* and *MinIntChangeG*, our main contribution. These algorithms are reusable for other kind of data, for which a closure operator is defined (e.g. graphs in pattern structure [46]) and a minimal change operation is defined (e.g. adding an edge to a graph pattern). The main drawback of the algorithms lies in their poor scalability when the number of different attribute values is large compared to the number of objects. However, as stated in Chapter 4 for unfrequent closed interval pattern extraction, one can easily embed monotone constraints on the lattice structure of these patterns (e.g. minimal/maximal size of one or several intervals). Indeed, intervals with too large size tend to be frequent but not interesting, whereas small intervals may have too small support [116]. We dedicated this problem in Chapter 5, in the field of information fusion, by introducing a similarity relation between interval patterns. A second solution explored in Chapter 4 with effective results in gene expression data analysis, is to reduce the number of different attribute values before the mining task, e.g. rounding values. For example, the last attribute of the basket ball dataset (BK) describes the points per minutes of a player: a double value with four digits after the comma, e.g. 0.5885. One can round this value to two digits after the coma considering that this loss of information is not significant, making the mining possible with large datasets.

We also showed that mining equivalent binary data (encoding all possible intervals) is not efficient since these data suffers of redundancy. Indeed, classical itemset mining algorithms generally do not consider a semantic associated to binary attribute labels. That was also a contribution to show that pattern structures and associated closure operator provide a simple and elegant framework to consider numerical data. The semantics associated to interval patterns may extend their use to other domains.

Taking into account missing values is a perspective of research, while fault-tolerant interval patterns should be studied, possibly strongly reducing their number (see e.g. [15] for the binary case). This chapter ends with potential use of interval pattern generators and perspectives of research.

Generators are preferable to closed patterns. According to the version of minimum description length principle (MDL) of [50], the best hypothesis to explain a dataset is the one minimizing the sum of (i) the length in bit of the description of the hypothesis, and (ii) the length of the data description when encoded with the help of the hypothesis. The authors of [77] recalled how the MDL principle favors generators. Consider an equivalence class of itemset in binary data. The maximal element, i.e. closed itemset, has higher cardinality, while generators have smallest cardinality. Therefore, the generators with minimal cardinality are best hypothesis to describe the same set of objects. The same holds for interval patterns, modulo the notion of minimality: best patterns are those minimal w.r.t. the subsumption relation on patterns, i.e. patterns with largest interval describing a same set of objects. According to [77], interval pattern generators provide better hypothesis, and seem useful for numerical classification problems, i.e. explaining the resulting cluster description, since usually, the bounding box of object descriptions (a closed interval pattern), is considered.

Interval patterns for k-anonymity. To preserve privacy in a dataset, object identifiers can be removed, e.g. names. However, some combinations of attributes such as birth date and ZIP code possibly allow to identify a unique individual. An important method for de-identification is the method of k-anonymity [1]. A basic idea is to reduce the granularity of data descriptions in such a way that a unique individual cannot be distinguished among at least $(k-1)$ individuals. For numerical attributes, a solution is to "generalize" the attribute values to a range, reducing the granularity, e.g. replacing the age 23 by an interval $[21, 24]$, see e.g. [112]. Now consider an individual $g \in G$ in a numerical dataset as described in this chapter. The descrip-

tion $\delta(g) \in (D, \sqsubseteq)$ is composed of degenerate intervals (i.e. same left and right borders), and is closed. The information brought by one of its generators (with larger intervals) is as follows: this generalization is not sufficient enough to not uniquely identify the individual. One should therefore consider a smaller generator w.r.t. $\sqsubseteq$ depending on the cardinality of its image, and can replace the individual description this generator. This operation is a projection of the pattern searchspace.

Generating association rules. It is known that association rules involving closed itemsets and generators are of high interest in data-mining [10]. Indeed, the confidence of such rules is of 100% and the whole collection of such rules is compact. It is therefore an interesting perspective of research to mine exact and partial association rules within the framework of pattern structures, and to compare with other association rule mining methods from the literature, see e.g. [116, 111, 8].

Generators for information fusion. In the previous chapter we presented how pattern structures can enhance information fusion, by proposing a synthetic view of partial fusion results. We showed how a fusion operator can be embedded in a pattern structure to rise a concept lattice. Each partial fusion result can be interpreted as a closed pattern. Therefore, the question that automatically comes after this chapter is the following. Given partial fusion result, that is a closed pattern, what information can brought its generator(s) and how is it useful for information fusion tasks? Indeed, if the fused interval is considered for decision purposes, its generators may give a useful information, i.e. the largest intervals for which a same set of sources are in agreement. This interpretation relies on the choice of the fusion operator (here convexification), and is different with other operators. Each case should be studied.

Chapter 8

Towards biclustering numerical data with formal concept analysis

This last contribution chapter introduces our main perspective of research. We relate here a preliminary work on how FCA can help the problem of biclustering. Indeed, recall that a bicluster is informally defined as a subrectangle of a numerical table checking a given constraint. In many cases, best rectangles are the largest ones that check this constraint [81]. The parallel with FCA is natural since formal concepts are subrectangles of "1" values in a binary tables such that no super rectangle of "1" values exists. Accordingly, in many cases, a bicluster definition involves implicitly a closure operator. This is the goal of this chapter to give a preliminary outlook on how FCA can help existing biclustering techniques, by considering two particular cases of biclusters. Moreover, this chapter gives answer to questions raised in [16].

1 Introduction

We consider the problem of biclustering numerical data [52, 32, 81] using techniques of Formal Concept Analysis (FCA) [47, 46]. A numerical dataset is given by sets of objects, attributes, and attribute values for objects (many-valued contexts in terms of FCA). The description of an object is a tuple of values, each component corresponding to an attribute value. An example of numerical dataset is given in Table 1 where lines denote objects, while columns denote attributes.

To analyze such a dataset, a major data-mining task is clustering, a data analysis technique used in several domains, e.g. gene expression data analysis. It allows one to group objects into clusters according to some similarity criteria between their description, the similarity being defined according to an adequate distance, following given characteristics [55]. However, clusters are *global* patterns since similarity between objects is computed w.r.t. all attributes simultaneously (possibly weighted). In many applications, and especially in gene expression data analysis, *local* patterns are preferred [23, 81] and consist in pairs (A, B) where A is a subset of objects related to a subset of attributes B. Indeed, it is known that a set of genes is activated (e.g. translated into proteins for enabling a biological process) under some conditions only, i.e. only for some attributes. Accordingly, a bicluster is generally represented by a rectangle of values in a numerical data table, see e.g. a bicluster in Table 2. In Table 1, one can see that both biclusters $(\{g_1, g_2\}, \{m_1, m_2, m_3, m_4\})$ and $(\{g_1, g_2\}, \{m_5\})$ give more meaningful information than cluster $\{g_1, g_2\}$ being described by all attributes, since the values taken by objects in A for attributes in B are more similar.

There are many definitions of a bicluster, depending on the relation between subsets of

Table 1: A numerical dataset

	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

Table 2: $(\{g_2, g_3, g_4\}, \{m_3, m_4\})$

	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

objects and subsets of attributes, as discussed in [81]. In this chapter, we consider two types of biclusters: firstly, constant biclusters that can be represented as rectangle of equal values (see Table 3), and secondly, biclusters of similar values, that can be represented by rectangle of similar values (see Table 4). In general case, extracting all biclusters is an intractable problem [81], so in practice heuristics are used. Obviously, even best heuristics may result in the loss of "interesting" biclusters.

The purpose of this chapter is to show that an approach based on Formal Concept Analysis (FCA [47]) can be used for biclustering numerical data, leading to a complete, correct and non-redundant enumeration of all maximal biclusters (either of constant or similar values). Such non-heuristic based enumeration has not been deeply considered in the literature due to the very important number of possible biclusters. Whereas a first study is given in [16], we propose here two equivalent FCA-based methods, whose underlying closure operator enables a natural enumeration of maximal biclusters. The first one relies on conceptual scaling (discretization) of numerical data giving rise to several binary tables from which biclusters can be extracted as formal concepts. A second method avoids *a priori* scaling and is based on interval pattern structures.

2 Problem setting

Here a numerical dataset is realized by a many-valued context (G, M, W, I) where W is a set of values that objects $g \in G$ can take for attributes $m \in M$. Such many-valued contexts are usually represented by a numerical table where a table-entry gives the value $m(g) \in W$, i.e. the value taken by attribute m in column for object g in line. The Table 1 gives an example (taken from [16]) that we consider throughout this chapter, with objects $G = \{g_1, ..., g_4\}$, attributes $M = \{m_1, ..., m_5\}$, and e.g. $m_2(g_4) = 9$.

A bicluster is given by a pair (A, B) with $A \subseteq G$ and $B \subseteq M$. Intuitively, a bicluster is represented by a rectangle of values, or sub-table (modulo line and column permutations), see e.g. the bicluster $(\{g_2, g_3, g_4\}, \{m_3, m_4\})$ highlighted grey in Table 2.

Definition 8.1 (Bicluster) *Given a numerical dataset (G, M, W, I), a bicluster is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$.*

In [81], several types of biclusters are introduced. The type of a bicluster (A, B) depends on the relation between the values taken by attributes in B for objects in A. In this chapter, we consider constant biclusters (equality relation) and biclusters of similar values (similarity relation) as defined in the next paragraphs.

A constant bicluster can be interpreted as a rectangle of identical values, and is defined as follows.

Definition 8.2 (Constant bicluster) *Given a numerical dataset (G, M, W, I), a constant bicluster is a bicluster (A, B) such that $m_i(g_j) = m_k(g_l), \forall g_j, g_l \in A, \forall m_i, m_k \in B$.*

Table 3: A constant bicluster

	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

Table 4: A bicluster of similar values

	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

Since the number of possible biclusters in a numerical dataset can be very large, the notion of maximality gives naturally rise to maximal constant biclusters, i.e. "largest rectangles of identical values".

Definition 8.3 (Maximal constant biclusters) *Given a numerical dataset (G, M, W, I), a constant bicluster (A, B) is maximal if In other terms, (A, B) is a maximal constant bicluster iff*

- $(A \cup \{g\}, B)$ *is not a constant bicluster* $\forall g \in G \backslash A$
- $(A, B \cup \{m\})$ *is not a constant bicluster* $\forall m \in M \backslash B$

Table 3 shows an example of maximal constant bicluster $(\{g_1, g_2, g_3\}, \{m_5\})$. One should remark that $(\{g_1, g_2\}, \{m_5\})$ is constant but not maximal. Note that maximal constant biclusters taking values 1 in a 1/0 table are formal concepts.

The fact that constant biclusters correspond to sets of objects taking equal values for same attributes is a too strong condition in real-world data. This may lead to the well-known problem of pattern overwhelming. Instead of considering equality, one may relax this condition and consider a similarity relation between values. This idea was introduced in [16] for handling noise in a numerical dataset. Two values $w_1, w_2 \in W$ are said to be similar if their difference does not exceed a user-defined parameter θ. A similarity relation denoted by $\simeq_\theta$ is formally defined by: $w_1 \simeq_\theta w_2 \iff |w_1 - w_2| \leq \theta$. According to this formalization of similarity, a bicluster of similar values can be defined as a "generalization" of constant biclusters.

Definition 8.4 (Bicluster of similar values) *A bicluster (A, B) is a bicluster of similar values if $m_i(g_j) \simeq_\theta m_k(g_l), \forall g_j, g_l \in A, \forall m_i, m_k \in B$.*

How to define a maximal bicluster of similar values is similar with maximal biclusters of equal values.

Table 4 shows an example of maximal bicluster of similar values $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ with $\theta = 1$. Note that bicluster $(\{g_1, g_2\}, \{m_1, m_2\})$ fulfils the conditions of similarity but is not maximal. Obviously, constant biclusters are biclusters of similar values when $\theta = 0$.

In this chapter we consider the problem of mining all maximal (i) constant biclusters and (ii) biclusters of similar values from a numerical dataset. The novelty here lies in the use of Formal Concept Analysis for a correct, complete and non-redundant enumeration (without heuristics). Indeed, we show in the following sections how to define a scaling to build formal contexts whose concepts exactly correspond to the two types of biclusters. However, this leads to the definition of several contexts whose preparation and mining may be inefficient. Then, based on so-called interval pattern structures, we show how binarization can be avoided, which results in reducing practical computational complexity.

3 Mining biclusters by means of conceptual scaling

In this section, we present two scaling procedures allowing to build formal contexts from which (i) constant biclusters and (ii) biclusters of similar values, can be extracted within the existing FCA framework. Intuitively, scaling allows to express bicluster searchspace under the form of binary tables, while the Galois connection allows to extract maximal biclusters represented as concepts.

3.1 Constant biclusters

A maximal constant bicluster can be interpreted as a maximal rectangle of identical values. Recall that formal concepts correspond to maximal rectangles of 1 values in binary tables. Accordingly, a maximal constant bicluster containing values $w \in W$ from a numerical dataset (G, M, W, I) corresponds to a concept in a context $\mathbb{K}_w = (G, M, I_w)$ where $(g, m) \in I_w \iff m(g) = w$. One should naturally consider one formal context for each value $w \in W$, which results in a context family $\mathbb{K}_W$ defined as follows:

$$\mathbb{K}_W = \{\mathbb{K}_w = (G, M, I_w) \mid w \in W \ (m, g) \in I_w \iff m(g) = w\}$$

The procedure building the family $\mathbb{K}_W$ from (G, M, W, I) involves one conceptual scaling for each $w \in W$ (actually nominal scalings related to each value w [47]). Figure 1 gives $\mathbb{K}_w = (G, M, I_w)$ for $w = 1$ and $w = 6$. The collection of concepts of each context $\mathbb{K}_w = (G, M, I_w)$ is denoted by $\mathfrak{B}(G, M, I_w)$, or simply $\mathfrak{B}_w$. Examples are given in Figure 1.

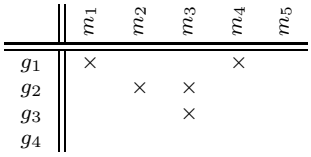
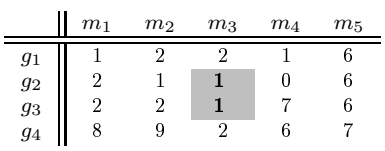
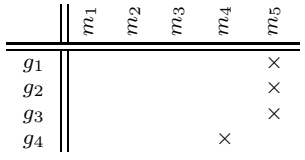
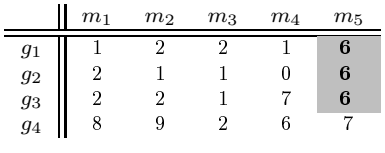
$w \in W$	$\mathbb{K}_w$	$\mathfrak{B}_w$	Bicluster corresponding to first concept on left list
...	...	...	...
1		$(\{g_2, g_3\}, \{m_3\})$ $(\{g_2\}, \{m_2, m_3\})$ $(\{g_1\}, \{m_1, m_4\})$	
...	...	...	...
6		$(\{g_1, g_2, g_3\}, \{m_5\})$ $(\{g_4\}, \{m_4\})$	
...	...	...	...

Figure 1: Extracting constant biclusters from the dataset of Table 1

The two obvious propositions hold.

Proposition 8.1 *Given a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$, a concept (A, B) of $\mathbb{K}_w$ corresponds to a maximal constant bicluster (A, B) of values w from numerical dataset (G, M, W, I).*

Proposition 8.2 *There is a one-to-one correspondence between the set of concepts $\bigcup_{w \in W} \mathfrak{B}_w$ and the set of all maximal biclusters.*

Hence, an algorithm that constructs the set of concepts $\bigcup_{w \in W} \mathfrak{B}_w$ gives a correct, complete and non redundant enumeration of all maximal constant biclusters.

Figure 1 gives two examples of concepts and their corresponding bicluster representation in the original numerical table.

3.2 Biclusters of similar values

The number of constant biclusters can be very large in real-world data, where numerical attribute domains contain many different values. Moreover, it leads to a huge number of artifacts, e.g. the maximal constant bicluster $(A, B) = (\{g_4\}, \{m_4\})$ is a rectangle of area 1, i.e. the product $|A| \times |B|$. One should therefore relax the equality constraint on numerical values when performing scaling with similarity relation $\simeq_\theta$ defined in the introduction. Intuitively, with $\theta = 1$, the previous example is not maximal anymore, whereas $(\{g_3, g_4\}, \{m_4, m_5\})$ is maximal with area equal to 4. For that matter, one should extract rectangles with pairwise similar values w.r.t $\simeq_\theta$. However, this relation is reflexive and symmetric but not transitive, hence a *tolerance relation*.

As related in [71], a tolerance relation T over an arbitrary set G, i.e. $T \subseteq G \times G$, can be represented by a formal context (G, G, T). A formal concept of (G, G, T) where intent is equal to extent corresponds to a *class of tolerance*, i.e., a maximal subset of G such that all pairs of its elements are in relation T.

Going back to the tolerance relation $\simeq_\theta$ on a set of values W, tolerance classes are maximal sets of pairwise similar values, corresponding to concepts (A, B) of $(W, W, \simeq_\theta)$ such that $A = B$. This is exactly what we need to characterize maximal biclusters of similar values. More details on this process were given in Chapter 5, while we show below initial context $(W, W, \simeq_\theta)$ and corresponding classes of tolerance from the numerical dataset of Table 1.

$\simeq_1$	0	1	2	6	7	8	9
0	×	×					
1	×	×	×				
2		×	×				
6				×	×		
7				×	×	×	
8					×	×	×
9						×	×

Classes of tolerance
$\{0, 1\}$
$\{1, 2\}$
$\{6, 7\}$
$\{7, 8\}$
$\{8, 9\}$

Renamed classes
$[0, 1]$
$[1, 2]$
$[6, 7]$
$[7, 8]$
$[8, 9]$

Now that classes of tolerance, or maximal sets of pairwise similar values, are characterized and computed, we can rename them for sake of readability and use them for scaling the initial dataset from which maximal biclusters of similar values can be extracted.

We choose to rename a class $K \subseteq W$ as the convex hull of its elements, i.e. the interval $[k_i, k_j]$ s.t. k_i and k_j are respectively smallest and largest values of K w.r.t. natural order $\leq$ on numbers. Indeed, when $|K|$ becomes large for certain data, this new name is more concise. Moreover, any $k \in [k_i, k_j]$ respects $k \simeq_\theta k_i \simeq_\theta k_j$.

Biclusters of similar values are a generalization of constant ones, i.e. with all values included in interval $[k_i, k_j]$ for a given class of tolerance. We should now also consider one formal context for each class of tolerance, hence a family of contexts. Consider a numerical dataset (G, M, W, I), and a class of tolerance from W which corresponds to the interval $[k_i, k_j]$. The associated formal context is given by:

$$(G, M, I_{[k_i,k_j]}) \text{ s.t. } (g,m) \in I \quad \Leftrightarrow m(g) \in [k_i, k_j] \text{ and all values n(h) from } \{h \in G | h(m) \simeq_\theta m(g)\} \text{ and} \{n \in M | n(g) \simeq_\theta m(g)\} \text{ are similar.}$$

First condition $m(g) \in [k_i, k_j]$ means that $m(g)$ should be similar with all elements of the current class of tolerance. The second condition come from the fact that classes of tolerance are computed from the set W: since a bicluster is represented by a rectangle in the numerical table, we should consider only similar values in column and lines to test whether a value belongs to a class of tolerance.

Consider the formal context $\mathbb{K}_{[k_i,k_j]}$ which corresponds to the class of tolerance $[k_i, k_j]$ and a concept (A, B) from this context. The following propositions hold.

Proposition 8.3 *(A, B) is a maximal bicluster of similar values.*

Proposition 8.4 *There is a one-to-one correspondence between the set of concepts from all formal contexts $\mathbb{K}_{[k_i,k_j]}$ and the set of all maximal biclusters of similar values.*

Thus, an algorithm computing the set of concepts from all formal contexts $\mathbb{K}_{[k_i,k_j]}$ gives a correct, complete and non redundant enumeration of maximal biclusters of similar values.

Figure 2 gives the formal context $\mathbb{K}_{[k_i,k_j]}$ for each class of tolerance $[k_i, k_j]$, their respective concepts and bicluster representation in the initial numerical Table 1.

4 Mining biclusters from pattern concept lattice

Until now, we presented how (constant) biclusters (of similar) values can be extracted using standard FCA tools such as scaling and concept extraction algorithms. Since resulting binary tables may be numerous and large (i.e. one for each class of tolerance), we present in this section an approach based on interval pattern structures, introduced in Chapter 4 that we firstly briefly recall with our current example. We consider in this section only biclusters of similar values, since being more general than constant ones and more useful for real-world applications.

4.1 Interval pattern structures

In Chapter 4, a numerical dataset (G, M, W, I) is represented by a so-called interval pattern structure $(G, (D, \sqcap), \delta)$ where D is a set of interval vectors, the i^{th} dimension giving an interval of values from W for attribute $m_i \in M$. We denote such vectors as *interval patterns*. In Table 1, the description of object g_1 is the interval pattern $\delta(g_1) = \langle [1,1], [2,2], [2,2], [1,1], [6,6] \rangle$. Interval patterns can be represented as $|M|$-hyperrectangles in Euclidean space $\mathbb{R}^{|M|}$, whose sides are parallel to the coordinate axes.

Now we recall how interval patterns are ordered. Consider firstly a single attribute $m \in M$, with value domain $W_m \subseteq W$. Elements of W_m can be ordered within a meet-semi-lattice making them potential object descriptions. Recalling that any $w \in W_m$ can be written as interval $[w, w]$, the infimum $\sqcap$ of two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ is:

Class of tolerance	Formal context[a]	Concepts	Bicluster corresponding to first concept on left list
$[0,1]$		$(\{g_1,g_2\},\{m_4\})$ $(\{g_2\},\{m_2,m_3,m_4\})$	
$[1,2]$		$(\{g_1,g_2,g_3\},\{m_1,m_2,m_3\})$ $(\{g_1\},\{m_1,m_2,m_3,m_4\})$ $(\{g_1,g_2,g_3,g_4\},\{m_3\})$	
$[6,7]$		$(\{g_3,g_4\},\{m_4,m_5\})$ $(\{g_1,g_2,g_3,g_4\},\{m_5\})$	
$[7,8]$		$(\{g_4\},\{m_1,m_5\})$	
$[8,9]$		$(\{g_4\},\{m_1,m_2\})$	

[a]Empty lines and columns are omitted.

Figure 2: Extracting all maximal biclusters of similar values from Table 1

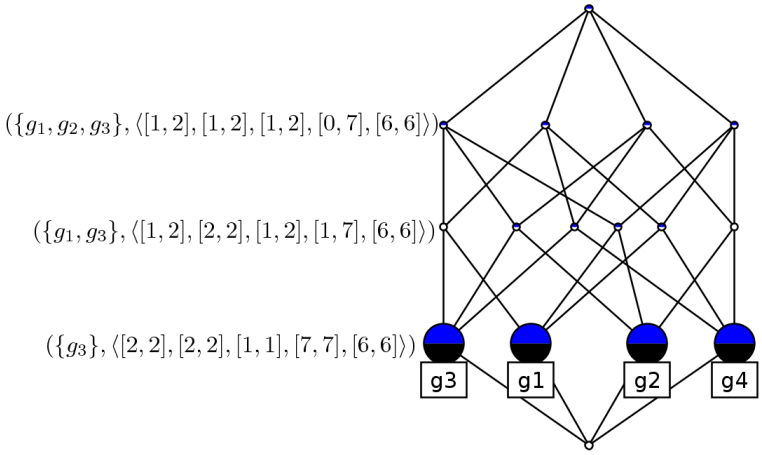


Figure 3: Pattern concept lattice of pattern structure from Table 1.

$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$, i.e. the largest interval containing them. Indeed, when c and d are intervals, $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$ holds:

$$\begin{array}{rcrcl} [a_1, b_1] \sqsubseteq [a_2, b_2] & \Leftrightarrow & [a_1, b_1] \sqcap [a_2, b_2] & = & [a_1, b_1] \\ & \Leftrightarrow & [min(a_1, a_2), max(b_1, b_2)] & = & [a_1, b_1] \\ & \Leftrightarrow & a_1 \leq a_2 \text{ and } & & b_1 \geq b_2 \\ & \Leftrightarrow & [a_1, b_1] & \supseteq & [a_2, b_2]. \end{array}$$

As objects are described by several intervals, each one standing for a given attribute, *interval patterns* have been introduced as p-dimensional vector of intervals, with $p = |M|$. Given two interval patterns $e = \langle [a_i, b_i] \rangle_{i \in [1,p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1,p]}$ their infimum $\sqcap$ and induced ordering relation $\sqsubseteq$ are given by:

$$\begin{array}{rcl} e \sqcap f & = & \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & = & \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1,p]} \end{array} \qquad \begin{array}{rcl} e \sqsubseteq f & \Leftrightarrow & \langle [a_i, b_i] \rangle_{i \in [1,p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1,p]} \\ & \Leftrightarrow & [a_i, b_i] \sqsubseteq [c_i, d_i], \ \forall i \in [1, p] \end{array}$$

This means that patterns with larger intervals are subsumed by patterns with smaller ones. Hence, one can define a pattern structure $(G, (D, \sqcap), \delta)$ from a numerical dataset (G, M, W, I), where $(D, \sqcap)$ is a meet-semi-lattice of interval patterns. This is deeply detailed in Chapter 4. We illustrate here the Galois connection.

$$\begin{array}{rcl} \{g_2, g_3\}^\square & = & \delta(g_2) \sqcap \delta(g_3) \\ & = & \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle \end{array}$$

$$\begin{array}{rcl} \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle^\square & = & \{g \in G | \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle \sqsubseteq \delta(g)\} \\ & = & \{g_2, g_3\} \end{array}$$

Hence $(\{g_2, g_3\}, \langle [2,2], [1,2], [1,1], [0,7], [6,6] \rangle)$ is a pattern concept. The set of all pattern concepts gives rise to a pattern concept lattice, see Figure 3 for our example. In this figure, three concepts are fully described with respective pattern extent and intent. Intuitively, $(A_1, d_1) \leq (A_2, d_2)$ means that corresponding hyperrectangle of (A_1, d_1) is included in corresponding hyperrectangle of (A_2, d_2).

Table 5: Interval pattern as bicluster

	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

Table 6: Introducing $\theta = 1$

	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

4.2 Biclusters of similar values in pattern concepts

A pattern concept (A, d) of a numerical dataset (G, W, M, I) can be seen as a bicluster (A, M) since it gives a range of value for each attribute $m \in M$. Bicluster representation of $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],[0,7],[6,6]\rangle)$ is given in Table 5.

However, a pattern concept (A, d) is not necessarily a bicluster of similar values, for three reasons. First, d may contain intervals larger than θ, i.e. all values in columns are not necessarily similar. Secondly, d may contain different intervals whose values are not similar, i.e. all values in lines may not be similar. Finally, if those conditions are respected, it is not sure that maximality of biclusters holds. We show how to control these statements to extract maximal biclusters of similar values from the pattern concept lattice.

First statement. Avoiding intervals of size larger than θ in a pattern intent d means that a pattern concept will correspond to a rectangle for which each column has similar values. For that matter, consider a modification $(G, (D*, \sqcap), \delta)$ of the interval pattern structure defined in the previous subsection: the set $D*$ consists of tuples, whose components are either intervals or the null element $*$. For two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ their infimum $\sqcap$ is defined as follows: $[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max(b_1, b_2)]$ if $|max(b_1, b_2) - min(a_1, a_2)| \leq \theta$ and $*$ otherwise. Moreover, $* \sqcap [a, b] = *$ for any $a, b \in \mathbb{R}$. Consider that for $d \in D$, d_m denotes the interval given for attribute $m \in M$. Now, given two interval vectors $c = \langle c_i \rangle$ and $d = \langle d_i \rangle$ their infimum is computed componentwise: $c \sqcap d = \langle c_i \sqcap d_i \rangle$. Applying operators of the Galois connection on set $\{g_2, g_3\}$ derives the concept $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],*,[6,6]\rangle)$, while starting with set $\{g_1, g_4\}$ allows to derive concept $(\{g_1, g_2, g_3, g_4\}, \langle*, *, [1,2], *, [6,6]\rangle)$. The resulting pattern concept lattice is given in Figure 4 and contains only 11 concepts compared to 16 when the operation $\sqcap$ is not constrained with θ. Table 6 shows the bicluster representation of $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],*,[6,6]\rangle)$, i.e. a rectangle for which values in each column are similar w.r.t. $\theta = 1$. Note that one should ignore attributes that take the value $*$ in pattern intent.

Second statement. From a pattern structure $(G, (D*, \sqcap), \delta)$, we are able to build a pattern concept lattice whose concepts corresponds to rectangles having similar values in columns. We should therefore also consider similar values in lines. Going back to concept $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],*,[6,6]\rangle)$, we remark that $(\{g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_2, g_3\}, \{m_5\})$ are biclusters of similar values that can be built from the initial pattern concept. Indeed, the intervals describing attributes m_1, m_2, and m_3 and pairwise similar ($[2,2] \simeq_\theta [1,2] \simeq_\theta [2,2]$ with $\theta = 1$), while interval describing attribute m_5 is similar with no others. We should accordingly consider classes of tolerance between attribute descriptions to extract biclusters of similar values. The similarity relation $\simeq_\theta$ is adapted for intervals as follows: $[a_1, b_1] \simeq_\theta [a_2, b_2] \iff max(b_1, b_2) - min(a_1, a_2) \leq \theta$.

Proposition 8.5 *Given a pattern concept (A, d), any pair (A, B) with $B \subseteq M$ is a bicluster of similar values iff $\{d_m\}_{\forall m \in B}$ is a class of tolerance w.r.t. relation $\simeq_\theta$ over the set $\{d_m\}_{\forall m \in M}$.*

Proof 8.1 *Consider that (A, B) is not a bicluster of similar values: $\exists g_1, g_2 \in A$, and $\exists m_1, m_2 \in B$ such that $m_1(g_1) \not\simeq_\theta m_2(g_2)$, a contradiction.*

Third statement. By controlling the two first statements, we are able to extract biclusters of similar values from the pattern concept lattice of $(G, (D*, \sqcap), \delta)$. By the properties of classes of tolerance making a class a maximal set of similar values, we know that biclusters are maximal in colums, i.e. no columns can be added without violating the similarity relation. However, we are not sure that biclusters are maximal in lines. Going back to previous example, i.e. $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],*,[6,6]\rangle)$, the extracted biclusters $(\{g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_2, g_3\}, \{m_5\})$ are not maximal. Indeed, we have $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ and $(\{g_1, g_2, g_3\}, \{m_5\})$ that are also biclusters of similar values. If such biclusters are not maximal, this means that objects can be added in the extent A while B remains the same set. Due to the generalization/specialization property of concept lattices, such larger bicluster can be found in the direct upper neighbours of concept $(\{g_2, g_3\}, \langle[2,2],[1,2],[1,1],*,[6,6]\rangle)$, i.e. concept $(\{g_1, g_2, g_3\}, \langle[1,2],[1,2],[1,2],*,[6,6]\rangle)$

Example. The Figure 4 gives the pattern concept lattice of $(G, (D*, \sqcap), \delta)$ with $\theta = 1$[21]. For each pattern intent, elements of each class of tolerance are either underlined, crossed-off, or in bold. For a pattern concept (A, d), when a class is underlined, or in bold, it means that (A, B), B being the set of attribute corresponding to this class, is a maximal bicluster of similar values. If element of the class are crossed-off, this means that (A, B) is not maximal, i.e (C, B) with $A \subset C$ can be characterized also in a direct upper concept. For example, take concept $(\{g_1, g_2\}, \langle[1,2],[1,2],[1,2],[0,1],[6,6]\rangle)$. From this concept, according to classes of tolerance, one can characterize the following biclusters of similar values $(\{g_1, g_2\}, \{m_1, m_2, m_3\})$, $(\{g_1, g_2\}, \{m_4\})$ and $(\{g_1, g_2\}, \{m_5\})$. However, $(\{g_1, g_2\}, \{m_4\})$ is the one only that is maximal, i.e. that cannot be characterized from upper pattern concepts with larger extents.

Hence, all biclusters of similar values can be computed from pattern concepts by standard algorithms. These considerations lead to two dual ways of constructing maximal biclusters of similar values as pattern concepts: bottom-up and top-down.

5 Discussion and conclusion

This chapter focused on the problem of biclustering numerical data with formal concept analysis. The goal was not to propose a new kind of bicluster, but rather to argue that two existing types of biclusters can be extracted using FCA techniques. For that matter, we proposed two methods producing equivalent results. The first is based on conceptual scaling, while the second on interval pattern structures. It is now expected to experiment these approaches, compare them with other biclustering algorithms (e.g. from [16]) and investigate how to handle other types of biclusters defined in [81]. We should also study the impact of the variation of θ on the concept lattice granularity, or dually on the number of formal contexts/concepts. Finally, we should examine how formal concept analysis in fuzzy seetings can contribute to biclustering problems. Indeed, similarity and tolerance relations are widely studied in such settings [13].

We discuss now our both methods.

Consider the method based on scaling. The strength of such approach is to produce binary tables. Any FCA algorithm (discussed and compared in [74]), or closed itemset algorithm (e.g.

[21] When an interval from a pattern intent has same left and right borders, a value is given instead for sake of readability

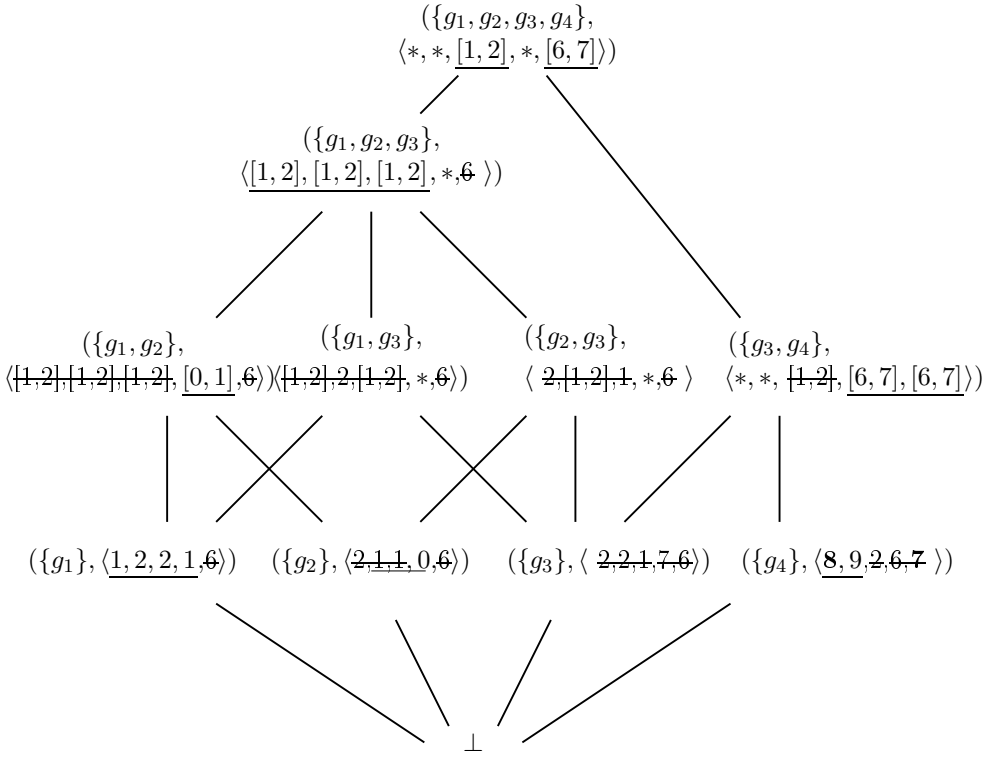


Figure 4: Pattern concept lattice of pattern structure from Table 1 with $\theta = 1$.

Charm [53]) can be used for extracting biclusters. Moreover, since each context of the produced family is independent from the others, a distributed computation is naturally possible: one core can be assigned for each formal context. It also allows to mine other kinds of binary patterns. For example, one can mine fault-tolerant patterns that would correspond to quasi biclusters of similar values, i.e. accepting some exceptions, see e.g. [98]. Meanwhile, searching for frequent biclusters (i.e. involving a number of objects higher than a user-defined threshold [119]) is straightforward. It rises also interesting questions: what is the meaning of an association rule? of a minimal generator?

The second method proposes to extract biclusters from a concept lattice, providing an interesting ordered hierarchy of biclusters. Computing the pattern concept lattice by adapting standard FCA algorithms such as CloseByOne is efficient as experimented in Chapter 4, while this algorithm can be parallelized [67]. In Chapter 7, CloseByOne was adapted to mine frequent closed interval patterns and their minimal generators. How this algorithm can be adapted for mining frequent biclusters is an interesting perspective of research. The fact that biclusters can be extracted from an ordered hierarchy of concepts make the pattern concept lattice a good structure for user queries. For example, a biologist may be interested in a particular set of genes for a given study. Accordingly, navigating in the concept lattice helps him discovering the different biclusters in which those genes occurs with other good candidates. We can describe such query as extensional since it starts by given a set of objects. On another hand, the approach based on scaling is more useful for so called intentional queries: the biologist is interested in all biclusters with values in a given interval (or class of tolerance) and accordingly only selects the formal context associated to this class.

Chapter 9

Conclusion and perspectives

1 Summary

Starting from large volumes of data, *knowledge discovery in databases* (KDD) consists in deriving knowledge units that can be further used for solving real-world problems. A major step of this process is data-mining and aims at automatically extracting patterns from a large search-space while the step of knowledge derivation is facilitated when formalizing knowledge as concepts. Indeed, knowledge units represented in an adequate representation formalism and may be integrated in ontologies to be re-used for solving problems in application domains.

Formal concept analysis is a mathematical framework that both allows a comprehensive formalization of concepts and provides patterns of choice, namely formal concepts, that are natural (bi)clusters of the input data. The set of ordered concepts form a concept lattice that expresses a generalization/specialization relation of concepts. The concept lattice supports many applications in information and knowledge processing including visualization, data analysis (mining) and knowledge management.

However, FCA applies to binary relations in standard settings. In real-world applications, e.g. in biology or chemistry, one rarely obtains binary data directly, *complex* and *heterogeneous* data involving numbers, graphs, intervals, etc., are more typical. Before applying FCA on complex data, a transformation named conceptual scaling has to be achieved, e.g. discretization of numerical values. Although this transformation allows FCA to be applied, it comes either with loss of information (e.g. numerical data), or hard computational properties (e.g. graph data). In best cases, a KDD process should always consider the same representation formalism of data and patterns.

For that matter, we proposed a new approach based on FCA to consider numerical data by adapting pattern structures to numerical data. This approach does not involve discretization, and is defined as a natural extension of FCA. The data are represented by *interval pattern structures* from which so called interval patterns can be extracted. An interval pattern is a vector of intervals, each dimension corresponding to a range of values some objects can take for a given attribute. An interval pattern can be represented in Euclidean space as a hyper-rectangle providing a semantic of formal definitions and models.

A major problem with real-world data is *pattern overwhelming*: the size of the result (i.e. the number of extracted concepts) is exponentially larger than the input (i.e. the number of objects, or dually attributes). Since pattern overwhelming is even worst in numerical data, we proposed several algorithms to extract closed patterns (and their generators). We also designed constraints that should be respected by extracted patterns and studied how a similarity relation

between numerical values can be embedded in interval pattern structures. These methods were successfully applied to both biological and agronomic applications, such as gene expression data analysis, and decision helping for crop sanity.

2 Perspectives

Interval pattern structures establish the basis of a new point of view for mining numerical data from which many perspectives arise. In the following, we divide them into several sections. Firstly, the approach can be used for the extraction of bi-clusters, widely used for applications in biology and recommender systems, with efficient algorithms and adequate semantics lacking in the literature. Secondly, thanks to a closure operator defined on numerical patterns, the definition of closed patterns and generators provides an interesting starting point for generating association rules, the latter being used for supervised classification tasks. Thirdly, we believe that closed patterns and their generators can be used for the k-anonymization of datasets for preserving privacy, a critical issue with the intensive publication of datasets on the web. Finally, we discuss computational issues and extension of our work on information fusion with pattern structures.

2.1 Biclustering of numerical data

Whereas the basic form of an interval pattern is very general, we remarked that it can be adapted to many types of biclusters. Firstly, we gave in Chapter 5 means to extract maximal sets of objects having similar values for each attribute from a maximal set of attributes. In the last chapter, we presented how to extract maximal rectangles of similar values. Other bi-cluster types can be handled similarly, e.g. the so-called δ-valid k-s biclusters [28].

Accordingly, a natural plan of research aims at surveying the different types of biclusters and their respective methods of extraction. This implies the design of efficient and scalable algorithms, and their comparison with state-of-the-art algorithms. Indeed, as related in [16] "*very few researchers have investigated the non heuristic, say complete, search of well-specified local patterns from numerical data*".

Our investigation is motivated from two points of view. Firstly defining appropriate scaling seems possible for several cases, and comes with very efficient algorithms from closed itemset mining community, and tools for handling noise [98]. Secondly, pattern structures allow a direct and ordered enumeration of biclusters since being probably the most general patterns in numerical data (this explains their huge number that we initially reduced thanks to tolerance relations). Embedding constraints upon pattern order could also be helpful for reducing the set of patterns to those of interest. Naturally, the notion of interestingness of biclusters has also to be investigated, a lot of effort has been done in this area [81].

Furthermore, we remark that the method designed for extracting maximal rectangles of similar values can be easily extended to multi-dimensional dataset. Consider a gene expression dataset $genes \times situations \times timestamps$. In these settings, a pattern corresponds to a maximal cube of similar values interpreted as a maximal set of genes having similar expression values in certain biological situations for given times. Interval pattern structures can be easily adapted, scaling as well. Furthemore, scaling would lead to n-ary relations, whose mining has been recently considered from an algorithmic point of view with the efficient algorithm *data-peeler* [30], and from a noise tolerance point of view [29].

2.2 Numerical pattern-based classifier

This perspective deals with supervised classification. Given a set of objects, their description and their target class, the aim is to build a model able to discover the target class of a new individual. A new trend of research relies on so called "pattern based classifiers". Given the group of objects with same target class, the goal is to discover the best patterns that characterize the class, and use them for the classification of a new individual. According to the version of minimum description length principle (MDL) of [50], the best hypothesis to explain a dataset is the one minimizing the sum of (i) the length in bits of the description of the hypothesis, and (ii) the length of the data description when encoded with the help of the hypothesis. The authors of [77] recalled how the MDL principle favors itemset generators as follows. Consider an equivalence class of itemsets in binary data, i.e. set of itemsets with same image, being shared by the same set of objects. The maximal element, i.e. closed itemset, has higher cardinality, while generators have smallest cardinality. Therefore, the generators with minimal cardinality are best hypothesis to describe the same set of objects. The same holds for interval patterns, modulo the notion of minimality: best patterns are those minimal w.r.t. a subsumption relation on patterns, i.e. patterns with largest intervals describing a same set of objects. According to [77], interval pattern generators provide better hypothesis, and seem useful for numerical classification problems, i.e. explaining the resulting cluster descriptions, since usually, the bounding boxes of object descriptions are considered (corresponding to closed interval patterns).

2.3 k-anonymity by means of projections

Most of the datasets are published on the Web, but they can contain private information about individuals. To preserve privacy in a dataset, object identifiers can be removed, e.g. individual names. However, some combinations of attributes such as birth date and ZIP code possibly allow to identify a unique individual. An important method for de-identification is the method of k-anonymity [1]. A basic idea is to reduce the granularity of data descriptions in such a way that a unique individual cannot be distinguished among at least $(k-1)$ individuals. For numerical attributes, a solution is to "generalize" the attribute values to a range, reducing the granularity, e.g. replacing the age 53 by an interval $[50, 60]$, see e.g. [112].

In interval pattern structure settings, the description of an individual is a closed pattern. The information brought by one of its generators (with larger intervals) is as follows: this generalization is not sufficient enough to not uniquely identify the individual. One should therefore consider a smaller generator w.r.t. a subsumption relation on patterns, depending on the cardinality of its image, and can replace the individual description by this generator, i.e. operate a projection. We plan to investigate such projection, and to not restrict only to numerical attributes.

2.4 Computational issues

Another crucial point for interval pattern structures concerns algorithmic issues. We showed that interval pattern structures can be reduced to formal contexts in many different ways depending on the exact formulation of output patterns. It follows that efficient algorithms of closed-itemset algorithms can be used, FCA algorithms as well. However, it happens that the resulting binary table is completely inefficient to process, especially with interordinal scaling. The second way on processing pattern structures is to adapt FCA algorithms. For example we pay a lot of attention in adapting the algorithm Close By One for mining interval patterns. A drawback of several FCA algorithms is that they rely on closure computations that involve an important number of database scan. Closed itemset mining algorithms generally scan the database only one time. How

these algorithms can be shifted to consider numerical data directly is a important perspective of research, coming with the design of adapted data structures for storing interval patterns and computing/estimating their frequency efficiently.

2.5 Information fusion

In Chapter 6 we argued that Formal Concept Analysis has the capacity of supporting a decision making process in the presence of information fusion problems, even when information are complex, e.g. patterns of numbers, thanks to the formalism of pattern structures. We showed how a (pattern) concept lattice enhances the expert decision: instead of providing a unique fusion result which can be problematic (usually the case in the literature), resulting pattern concept lattice yields a structured view of partial results labelled by subsets of sources. This work lies in basic information fusion settings: no knowledge on sources (reliability, preference order, etc.) were available and we considered basic fusion operators (union, intersection, convexification controlled by a similarity relation, and the method based on maximal coherent subsets). As a perspective, it is interesting to study how other fusion operators can be embedded in a concept lattice, as well as meta-information on sources (when available). This is closely related lattice-based argument structures and possibility theory [75].

Bibliography

[1] C. C. Aggarwal and P. S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms.* Springer, 2008.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *SIGMOD Conference*, pages 207–216. ACM Press, 1993.

[3] Z. Assaghir. *Analyse formelle de concepts et fusion d'informations : application à l'estimation et au contrôle d'incertitude des indicateurs agri-environnementaux.* PhD thesis, Institut National Polytechnique de Lorraine, 2010.

[4] Z. Assaghir, P. Girardin, and A. Napoli. Fuzzy logic approach to represent and propagate imprecision in agri-environmental indicator assessment. In *IFSA/EUSFLAT Conf.*, pages 707–712, 2009.

[5] Z. Assaghir, M. Kaytoue, A. Napoli, and H. Prade. Managing Information Fusion with Formal Concept Analysis. In V. Torra, Y. Narukawa, and M. Daumas, editors, *Modeling Decisions for Artificial Intelligence, 6th International Conference (MDAI)*, volume 6408 of *Lecture Notes in Computer Science*, pages 104–115. Springer, 2010.

[6] Z. Assaghir, M. Kaytoue, and H. Prade. A possibility theory-oriented discussion of conceptual pattern structures. In *Int. Conf. on Scalable Uncertainty Management (SUM)*, LNCS. Springer, 2010.

[7] Z. Assaghir, M. Kaytoue, H. Prade, and A. Napoli. Fusion de données imparfaites en utilisant l'analyse formelle de concepts. *Fouille de données complexes – Complexité liée aux données multiples, Numéro spécial de la Revue des Nouvelles Technologies de l'Information (RNTI) – Cépaduès-Éditions*, 2011.

[8] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *KDD*, pages 261–270, 1999.

[9] U. Baer and R. Calvet. Fate of Soil Applied Herbicides: Experimental Data and Prediction of Dissipation Kinetics. *J Environ Qual*, 28(6):1765–1777, 1999.

[10] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic*, pages 972–986, 2000.

[11] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explor. Newsl.*, 2(2):66–75, 2000.

[12] R. Belohlávek. *Fuzzy Relational Systems: Foundations and Principles.* Kluwer Academic Publishers, 2002.

[13] R. Belohlávek and T. Funioková. Similarity and fuzzy tolerance spaces. *J. Log. Comput.*, 14(6):827–855, 2004.

[14] S. Benferhat, D. Dubois, and H. Prade. Reasoning in inconsistent stratified knowledge bases. In *Int. Symp. on Multiple-Valued Logic*, pages 184–189, 1996.

[15] J. Besson, C. Robardet, and J.-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *ICCS*, LNCS 4068, pages 144–157. Springer, 2006.

[16] J. Besson, C. Robardet, L. D. Raedt, and J.-F. Boulicaut. Mining bi-sets in numerical data. In *KDID*, LNCS 4747, pages 11–23. Springer, 2006.

[17] L. Billard and E. Diday. *Symbolic Data Analysis: Conceptual Statistics and Data Mining.* Wiley Series in Computational Statistics. Wiley, Jan. 2007.

[18] S. Blachon, R. Pensa, J. Besson, C. Robardet, J.-F. Boulicaut, and O. Gandrillon. Clustering Formal Concepts to Discover Biologically Relevant Knowledge from Gene Expression Data. *In Silico Biology*, 7(4–5):467–483, 2007.

[19] I. Bloch, A. Hunter, A. Ayoun, S. Benferhat, P. Besnard, L. Cholvy, R. Cooke, D. Dubois, and H. Fargier. Fusion: general concepts and characteristics. *International Journal of Intelligent Systems*, 16:1107–1134, 2001.

[20] H.-H. Bock and E. Diday, editors. *Analysis of Symbolic Data – Exploratory Methods for Extracting Statistical Information from Complex Data.* Springer, 2000.

[21] C. Bockstaller, P. Girardin, and H. Van der Werf. Use of agro-ecological indicators for the evaluation of farming systems. *European journal of agronomy*, 7:261–270, 1997.

[22] F. Bodon and L. Rónyai. Trie: An alternative data structure for data mining algorithms. *Mathematical and Computer Modelling*, 38(7-9):739 – 751, 2003.

[23] J.-F. Boulicaut and J. Besson. Actionability and formal concepts: A data mining perspective. In Medina and Obiedkov [85], pages 14–31.

[24] R. Brachman and H. Levesque. *Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[25] P. Brito. Hierarchical and Pyramidal Clustering with Complete Symbolic Objects. In Bock and Diday [20], pages 312–324.

[26] P. Brito and G. Polaillon. Structuring probabilistic data by Galois lattices. *Mathématiques et sciences humaines*, 169, 2005.

[27] T. Calders and B. Goethals. Depth-first non-derivable itemset mining. In *SIAM International Conference on Data Mining*, 2005.

[28] A. Califano. Splash: structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16(4):341–357, 2000.

[29] L. Cerf, J. Besson, and J.-F. Boulicaut. Extraction de motifs fermés dans des relations n-aires bruitées. In J.-G. Ganascia and P. Gançarski, editors, *EGC*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 163–168. Cépaduès-Éditions, 2009.

[30] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Data peeler: Contraint-based closed pattern mining in n-ary relations. In *SDM*, pages 37–48. SIAM, 2008.

[31] L. Chaudron and N. Maille. Generalized Formal Concept Analysis. In B. Ganter and G. W. Mineau, editors, *ICCS*, volume 1867 of *LNCS*, pages 357–370. Springer, 2000.

[32] Y. Cheng and G. Church. Biclustering of Expression Data. In *Proc. 8th International Conference on Intelligent Systems for Molecular Biology (ISBM)*, pages 93–103, 2000.

[33] H. Ching-Jui and M. Zaki. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):462–478, 2005.

[34] V. Choi, Y. Huang, V. Lam, D. Potter, R. Laubenbacher, and K. Duca. Using formal concept analysis for microarray data comparison. *J Bioinform Comput Biol*, 6(1):65–75, 2008.

[35] S. Destercke, D. Dubois, and E. Chojnacki. Possibilistic information fusion using maximal coherent subsets. *IEEE Transactions on Fuzzy Systems*, 17:79–92, 2009.

[36] D. Dubois, H. Fargier, and H. Prade. Multiple source information fusion: a practical inconsistency tolerant approach. In *IPMU*, pages 1047–1054, 2000.

[37] D. Dubois, J. Lang, and H. Prade. Dealing with multi-source information in possibilistic logic. In *European Conference on Artificial Intelligence*, pages 38–42, 1992.

[38] D. Dubois and H. Prade. Possibility theory and data fusion in poorly informed environments. *Control Eng. Practice*, 2:811–823, 1994.

[39] D. Dubois and H. Prade. Possibility theory in information fusion. In *Data fusion and Perception*, pages 53–76, 2001.

[40] M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice-Hall, 2002.

[41] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–14868, 1998.

[42] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[43] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: an overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 1–34. American Association for Artificial Intelligence, 1996.

[44] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, 1996.

[45] S. Ferré and O. Ridoux. A Logical Generalization of Formal Concept Analysis. In B. Ganter and G. W. Mineau, editors, *ICCS*, volume 1867 of *LNCS*, pages 371–384. Springer, 2000.

[46] B. Ganter and S. O. Kuznetsov. Pattern structures and their projections. In H. S. Delugach and G. Stumme, editors, *ICCS*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2001.

[47] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, mathematical foundations edition, 1999.

[48] A. P. Gasch and M. B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 2002.

[49] A. Gély, R. Medina, and L. Nourine. Representing Lattices Using Many-Valued Relations. *Information Sciences*, 179(16):2729 – 2739, 2009.

[50] P. D. Grünwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications*. The MIT Press, 2005.

[51] M. R. Hacene, M. Huchard, A. Napoli, and P. Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In S. O. Kuznetsov and S. Schmidt, editors, *ICFCA*, volume 4390 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2007.

[52] J. Hartigan. Direct clustering of a data matrix. *J. Am. Statistical Assoc.*, 67(337):123–129, 1972.

[53] C.-J. Hsiao and M. J. Zaki. Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):462–478, 2005.

[54] I. Lacourt and S. Duplessis and S. Abba and P. Bonfante and F. Martin. Isolation and Characterization of Differentially Expressed Genes in the Mycelium and Fruit Body of Tuber Borchii. *Applied and Environmental Microbiology*, 68:4574–4582, 2002.

[55] D. Jiang, C. Tang, and A. Zhang. Cluster Analysis for Gene Expression Data: a Survey. *IEEE Trans. on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

[56] M. Kaytoue, Z. Assaghir, N. Messai, and A. Napoli. Complémentarité de deux méthodes de classification pour la construction de treillis de concepts à partir de données numériques. In *17ème conférence en Reconnaissance des Formes et Intelligence Artificielle*, 2010.

[57] M. Kaytoue, Z. Assaghir, N. Messai, and A. Napoli. Two complementary classification methods for designing a concept lattice from interval data. In *Foundations of Information and Knowledge Systems*, LNCS 5956, pages 345–362. Springer, 2010.

[58] M. Kaytoue, Z. Assaghir, A. Napoli, and S. O. Kuznetsov. Embedding tolerance relations in formal concept analysis: an application in information fusion. In J. Huang, N. Koudas, G. Jones, X. Wu, K. Collins-Thompson, and A. An, editors, *CIKM*, pages 1689–1692. ACM, 2010.

[59] M. Kaytoue, S. Duplessis, S. O. Kuznetsov, and A. Napoli. Two FCA-Based Methods for Mining Gene Expression Data. In S. Ferré and S. Rudolph, editors, *ICFCA*, volume 5548 of *LNCS*, pages 251–266. Springer, 2009.

[60] M. Kaytoue, S. Duplessis, and A. Napoli. Using Formal Concept Analysis for the Extraction of Groups of Co-expressed Genes. In L. T. H. An, P. Bouvry, and P. D. Tao, editors, *MCO*, volume 14 of *CCIS*, pages 439–449. Springer, 2008.

[61] M. Kaytoue, S. Duplessis, and A. Napoli. L'analyse formelle de concepts pour l'extraction de connaissances dans les données d'expression de gènes. In J.-G. Ganascia and P. Gançarski, editors, *Extraction et gestion des connaissances*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 439–440. Cépaduès-Éditions, 2009.

[62] M. Kaytoue, S. Duplessis, and A. Napoli. Toward the Discovery of Itemsets with Significant Variations in Gene Expression Matrices. In B. Fichet et al., editor, *Classification and Multivariate Analysis for Complex Data Structures*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 463–471. Springer, 2011.

[63] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Pattern Mining in Numerical Data: Extracting Closed Patterns and their Generators. Research Report RR-7416, INRIA, 2010.

[64] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Biclustering Numerical Data in Formal Concept Analysis. In *International Conference on Formal Concept Analysis (ICFCA)*, Lecture Notes in Artificial Intelligence. Springer, 2011.

[65] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Revisiting numerical pattern mining with formal concept analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[66] M. Kaytoue, S. O. Kuznetsov, A. Napoli, and S. Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Information Sciences*, In Press, Corrected Proof, 2010.

[67] P. Krajca, J. Outrata, and V. Vychodil. Advances in algorithms based on cbo. In M. Kryszkiewicz and S. Obiedkov, editors, *International Conference on Concept Lattices and Their Applications*, 2010.

[68] S. Kuznetsov. A Fast Algorithm for Computing all Intersections of Objects in a Finite Semi-lattice. *Automatic Documentation and Mathematical Linguistics*, 27(5):11–21, 1993.

[69] S. O. Kuznetsov. JSM-Method as a Machine Learning Method. *Itogi Nauki i Tekhniki, ser. Informatika*, 15:17–50, 1991.

[70] S. O. Kuznetsov. Learning of Simple Conceptual Graphs from Positive and Negative Examples. In J. M. Zytkow and J. Rauch, editors, *PKDD*, volume 1704 of *LNCS*, pages 384–391. Springer, 1999.

[71] S. O. Kuznetsov. Galois Connections in Data Analysis: Contributions from the Soviet Era and Modern Russian Research. In B. Ganter, G. Stumme, and R. Wille, editors, *ICFCA*, volume 3626 of *Lecture Notes in Computer Science*, pages 196–225. Springer, 2005.

[72] S. O. Kuznetsov. On Stability of a Formal Concept. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):101–115, 2007.

[73] S. O. Kuznetsov. Pattern structures for analyzing complex data. In *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, LNCS (5908), pages 33–44. Springer, 2009.

[74] S. O. Kuznetsov and S. A. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.

[75] C. Lafage, J. Lang, and R. Sabbadin. A logic of supporters. In B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, editors, *Information, uncertainty and fusion*, pages 381–392. Kluwer Academic Publishers, 1999.

[76] T. Lee, N. Rinaldi, F. Robert, D. Odom, Z. Bar-Joseph, G. Gerber, N. Hannett, C. Harbison, C. Thompson, I. Simon, J. Zeitlinger, E. Jennings, H. Murray, D. Gordon, B. Ren, J. Wyrick, J.-B. Tagne, T. Volkert, E. Fraenkel, D. Gifford, and R. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, 2002.

[77] J. Li, H. Li, L. Wong, J. Pei, and G. Dong. Minimum description length principle: Generators are preferable to closed patterns. In *Innovative Applications of Artificial Intelligence Conference*. AAAI Press, 2006.

[78] D. Lin. An information-theoretic definition of similarity. In *International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1998.

[79] M. Liquiere. Some Links Between Formal Concept Analysis and Graph Mining. In D. J. Cook and L. B. Holder, editors, *Mining Graph Data*, pages 227–252. John Wiley & Sons, 12 2006.

[80] G. Liu, J. Li, L. Wong, and W. Hsu. Positive borders or negative borders: How to make lossless generator based representations concise. In *SIAM Int. Conf. on Data Mining*, 2006.

[81] S. Madeira and A. Oliveira. Biclustering Algorithms for Biological Data Analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[82] R. Malouf. Maximal consistent subsets. *Comput. Linguist.*, 33(2):153–160, 2007.

[83] F. Martin. The Genome of Laccaria Bicolor Provides Insights into Mycorrhizal Symbiosis. *Nature*, 452(7183):88–92, March 2008. 69 co-authors wrote this paper.

[84] F. Martin, A. Aerts, D. Ahren, A. Brun, E. G. J. Danchin, F. Duchaussoy, J. Gibon, A. Kohler, E. Lindquist, V. Pereda, A. Salamov, H. J. Shapiro, J. Wuyts, D. Blaudez, M. Buee, P. Brokstein, B. Canback, D. Cohen, P. E. Courty, P. M. Coutinho, C. Delaruelle, J. C. Detter, A. Deveau, S. Difazio, S. Duplessis, L. Fraissinet-Tachet, E. Lucic, P. Frey-Klett, C. Fourrey, I. Feussner, G. Gay, J. Grimwood, P. J. Hoegger, P. Jain, S. Kilaru, J. Labbe, Y. C. Lin, V. Legue, F. Le Tacon, R. Marmeisse, D. Melayah, B. Montanini, M. Muratet, U. Nehls, H. Niculita-Hirzel, O.-L. M. P. Secq, M. Peter, H. Quesneville, B. Rajashekar, M. Reich, N. Rouhier, J. Schmutz, T. Yin, M. Chalot, B. Henrissat, U. Kues, S. Lucas, Y. Van de Peer, G. K. Podila, A. Polle, P. J. Pukkila, P. M. Richardson, P. Rouze, I. R. Sanders, J. E. Stajich, A. Tunlid, G. Tuskan, and I. V. Grigoriev. The genome of laccaria bicolor provides insights into mycorrhizal symbiosis. *Nature*, 452(7183):88–92, March 2008.

[85] R. Medina and S. A. Obiedkov, editors. *Formal Concept Analysis, 6th International Conference, Proceedings*, volume 4933 of *LNCS*. Springer, 2008.

[86] N. Messai. *Formal Concept Analysis guided by Domain Knowledge: Application to genomic resources discovery on the Web (in French)*. PhD Thesis in Computer Science, University Henri Poincaré – Nancy 1, France, March 2009.

[87] N. Messai, M.-D. Devignes, A. Napoli, and M. Smail-Tabbone. Many-valued concept lattices for conceptual clustering and information retrieval. In M. G. et al., editor, *Proc. of 18th European Conference on Artificial Intelligence*, pages 127–131, 2008.

[88] N. Messai, M.-D. Devignes, A. Napoli, and M. Smaïl-Tabbone. Using domain knowledge to guide lattice-based complex data exploration. In H. Coelho, R. Studer, and M. Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 847–852. IOS Press, 2010.

[89] R. S. Michalski, I. Bratko, and A. Bratko, editors. *Machine Learning and Data Mining; Methods and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[90] M. Molla, P. Andreae, J. Glasner, F. Blattner, and J. Shavlik. Interpreting Microarray Expression Data Using Text Annotating the Genes. *Information Sciences*, 146(1-4):75 – 88, 2002.

[91] S. Motameny, B. Versmold, and R. Schmutzler. Formal concept analysis for the identification of combinatorial biomarkers in breast cancer. In R. Medina and S. Obiedkov, editors, *Proc. 6th International Conference on Formal Concept Analysis ICFCA'08*, LNAI, pages 229–240. Springer, Feb. 2008.

[92] S. Motameny, B. Versmold, and R. Schmutzler. Formal Concept Analysis for the Identification of Combinatorial Biomarkers in Breast Cancer. In Medina and Obiedkov [85], pages 229–240.

[93] A. Napoli. A smooth introduction to symbolic methods for knowledge discovery. In H. Cohen and C. Lefebvre, editors, *Handbook of Categorization in Cognitive Science*, pages 913–933. Elsevier, Amsterdam, 2005.

[94] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In C. Beeri and P. Buneman, editors, *Database Theory - ICDT'99 Proceedings, 7th International Conference, Jerusalem, Israel*, Lecture Notes in Computer Science 1540, pages 398–416. Springer, 1999.

[95] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25–46, 1999.

[96] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Pruning closed itemset lattices for association rules. *International Journal of Information Systems*, 24(1):25–46, 1999.

[97] R. Pensa, C. Leschi, J. Besson, and J.-F. Boulicaut. Assessment of discretization techniques for relevant pattern discovery from gene expression data. In *Proceedings of the 4th ACM SIGKDD Workshop on Data Mining in Bioinformatics BIOKDD'04*, pages 24–30, 2004.

[98] R. G. Pensa and J.-F. Boulicaut. Towards fault-tolerant formal concept analysis. In *AI*IA*, pages 212–223, 2005.

[99] V. Phan-Luong. A framework for integrating information sources under lattice structure. *Information Fusion*, 9(2):278 – 292, 2008.

[100] J. Poelmans, P. Elzinga, S. Viaene, and G. Dedene. Formal concept analysis in knowledge discovery: A survey. In M. Croitoru, S. Ferré, and D. Lukose, editors, *ICCS*, volume 6208 of *Lecture Notes in Computer Science*, pages 139–153. Springer, 2010.

[101] G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme (in French).* PhD thesis, Université Paris IX-Dauphine, 1998.

[102] G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme.* PhD thesis, Université Paris IX Dauphine, 1998.

[103] G. Polaillon. Pyramidal Classification for Interval Data using Galois Lattice Redutcion. In Bock and Diday [20], pages 324–341.

[104] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data. *Bioinformatics*, 22(9):1122–1129, 2006.

[105] L. D. Raedt. *Logical and Relational Learning*, chapter 9: Kernels and Distances for Structured Data, pages 289–324. Springer, 2008.

[106] N. Rescher and R. Manor. On inference from inconsistent premisses. *Theory and Decision*, 1:179–217, 1970.

[107] F. Rioult, C. Robardet, S. Blachon, B. Crémilleux, O. Gandrillon, and J.-F. Boulicaut. Mining concepts from large sage gene expression matrices. In *KDID*, pages 107–118, 2003.

[108] S. Busch and G. H. Braus. How to Build a Fungal Fruit Body: from Uniform Cells to Specialized Tissue. *Molecular Microbiology*, 64:873–876, 2007.

[109] S. Duplessis, P. E. Courty, D. Tagu and F. Martin. Transcript Patterns Associated with Ectomycorrhiza Development in Eucalyptus Globulus and Pisolithus Microcarpus. *New Phytologist*, 165:599–611, 2005.

[110] S. Gabella, S. Abba, S. Duplessis, B. Montanini, F. Martin, and P. Bonfante. Transcript Profiling Reveals Novel Marker Genes Involved in Fruiting Body Formation in Tuber Borchii. *Eukaryotic Cell*, 4:1599–1602, 2005.

[111] A. Salleb-Aouissi, C. Vrain, and C. Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In M. M. Veloso, editor, *IJCAI*, pages 1035–1040, 2007.

[112] P. Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010 –1027, 2001.

[113] A. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde, and B. J. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press, 2000.

[114] J. Serra. *Image Analysis and Mathematical Morphology.* Boston: Academic Press, 1982.

[115] A. Soulet and B. Crémilleux. Adequate condensed representations of patterns. *Data Min. Knowl. Discov.*, 17(1):94–110, 2008.

[116] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In H. V. Jagadish and I. S. Mumick, editors, *SIGMOD Conference*, pages 1–12. ACM Press, 1996.

[117] R. B. Stoughton. Applications of DNA Microarrays in Biology. *Annual Review of Biochemistry*, 74(1):53–82, 2005.

[118] J. Stuart, E. Segal, D. Koller, and S. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–255, 2003.

[119] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with t. *Data Knowl. Eng.*, 42(2):189–222, 2002.

[120] L. Szathmary. *Symbolic Data Mining Methods with the Coron Platform.* PhD Thesis in Computer Science, Univ. Henri Poincaré – France, 2006.

[121] L. Szathmary and A. Napoli. CORON: A Platform for Itemset Mining Algorithms. In *Fourth International Conference on Formal Concept Analysis – ICFCA '06, Dresden, Germany*, Feb 2006. (oral communication and demo, without paper).

[122] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin. Efficient Vertical Mining of Frequent Closures and Generators. In *International Symposium on Intelligent Data Analysis*, LNCS 5772, pages 393–404, 2009.

[123] A. Tversky. Features of Similarity. *Psychological Review*, 84(4):327–352, 1977.

[124] T. Uno, M. Kiyomi, and H. Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2004.

[125] P. Valtchev, R. Missaoui, and R. Godin. Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges. In P. W. Eklund, editor, *ICFCA*, volume 2961 of *LNCS*, pages 352–371. Springer, 2004.

[126] F. J. Valverde-Albacete and C. Pelez-Moreno. Extending conceptualisation modes for generalised formal concept analysis. *Information Sciences*, In Press, 2010.

[127] H. Van der Werf and C. Zimmer. An indicator of pesticide environmental impact based on a fuzzy expert system. *Chemosphere*, 36(10):2225–2249, 1998.

[128] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In *Rival, I. (ed.): Ordered Sets*, pages 445–470. Boston, 1982.

[129] R. Wille. Why Can Concept Lattices Support Knowledge Discovery in Databases? *J. Exp. Theor. Artif. Intell.*, 14(2-3):81–92, 2002.

[130] Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In *The Data Mining and Knowledge Discovery Handbook*, pages 113–130. Springer, 2005.

[131] M. Zaki. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):462–478, 2005.

[132] M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *SDM*. SIAM, 2002.

Résumé

Le sujet principal de cette thèse porte sur la fouille de données numériques et plus particulièrement de données d'expression de gènes. Ces données caractérisent le comportement de gènes dans diverses situations biologiques (temps, cellule, etc.). Un problème important consiste à établir des groupes de gènes partageant un même comportement biologique. Cela permet d'identifier les gènes actifs lors d'un processus biologique, comme par exemple les gènes actifs lors de la défense d'un organisme face à une attaque. Le cadre de la thèse s'inscrit donc dans celui de l'extraction de connaissances à partir de données biologiques. Nous nous proposons d'étudier comment la méthode de classification conceptuelle qu'est l'analyse formelle de concepts (AFC) peut répondre au problème d'extraction de familles de gènes. Pour cela, nous avons développé et expérimenté diverses méthodes originales en nous appuyant sur une extension peu explorée de l'AFC : les structures de patrons. Plus précisément, nous montrons comment construire un treillis de concepts synthétisant des familles de gènes à comportement similaire. L'originalité de ce travail est (i) de construire un treillis de concepts sans discrétisation préalable des données de manière efficace, (ii) d'introduire une relation de similarité entres les gènes et (iii) de proposer des ensembles minimaux de conditions nécessaires et suffisantes expliquant les regroupements formés. Les résultats de ces travaux nous amènent également à montrer comment les structures de patrons peuvent améliorer la prise de d écision quant à la dangerosité de pratiques agricoles dans le vaste domaine de la fusion d'information.

Mots-clés : Découverte de connaissances, analyse formelle de concepts, extraction de motifs numériques, bi-clustering, fusion d'information

Abstract

The main topic of this thesis addresses the important problem of mining numerical data, and especially gene expression data. These data characterize the behaviour of thousand of genes in various biological situations (time, cell, etc.). A difficult task consists in clustering genes to obtain classes of genes with similar behaviour, supposed to be involved together within a biological process. Accordingly, we are interested in designing and comparing methods in the field of knowledge discovery from biological data. We propose to study how the conceptual classification method called Formal Concept Analysis (FCA) can handle the problem of extracting interesting classes of genes. For this purpose, we have designed and experimented several original methods based on an extension of FCA called pattern structures. Furthermore, we show that these methods can enhance decision making in agronomy and crop sanity in the vast formal domain of information fusion.

Keywords: Knowledge discovery in databases, formal concept analysis, numerical pattern mining, biclustering, information fusion